NSGA-II for Solving Multiobjective Integer Minimum Cost Flow Problem with Probabilistic Tree-Based Representation

Ghasemishabankareh, Behrooz; Ozlen, Melih; Li, Xiaodong

Ghasemishabankareh, B., Ozlen, M., & Li, X. (2019). NSGA-II for Solving Multiobjective Integer Minimum Cost Flow Problem with Probabilistic Tree-Based Representation. In: Deb K. et Al. (eds) Evolutionary Multi-Criterion Optimization. EMO 2019. Lecture Notes in Computer Science, Vol 11411., 541–552. https://doi.org/10.1007/978-3-030-12598-1_43
Document Version: Accepted Manuscript

Published Version: https://doi.org/10.1007/978-3-030-12598-1_43

NSGA-II for Solving Multiobjective Integer Minimum Cost Flow Problem with Probabilistic Tree-Based Representation

Behrooz Ghasemishabankareh *, Melih Ozlen, and Xiaodong Li

School of Science, RMIT University, Melbourne, Australia
{behrooz.ghasemishabankareh,melih.ozlen,xiaodong.li}@rmit.edu.au

Abstract. Network flow optimisation has many real-world applications. The minimum cost flow problem (MCFP) is the most common network flow problem, which can also be formulated as a multiobjective optimisation problem, with multiple criteria such as time, cost, and distance being considered simultaneously. Although there exist several multiobjective mathematical programming techniques, they often assume linearity or convexity of the cost functions, which are unrealistic in many real-world situations. In this paper, we propose to use the non-dominated sorting genetic algorithm, NSGA-II, to solve this sort of Multiobjective MCFPs (MOMCFPs), because of its robustness in dealing with optimisation problems of linear as well as nonlinear properties. We adopt a probabilistic tree-based representation scheme, and apply NSGA-II to solve the multiobjective integer minimum cost flow problem (MOIMCFP). Our experimental results demonstrate that the proposed method has superior performance compared to those of the mathematical programming methods in terms of the quality as well as the diversity of solutions approximating the Pareto front. In particular, the proposed method is robust in handling linear as well as nonlinear cost functions.

Keywords: Multiobjective optimisation · Minimum cost flow problem · Genetic algorithm

1 Introduction

Minimum cost flow problem (MCFP) is the most general case of a network optimisation problem where a commodity is transferred through the network to satisfy a demand and minimise/maximise objective function(s). There are different applications of the MCFP such as distribution and manufacturing problems, optimal loading of a Hopping aeroplane, or human resource management [1].

For MCFP, sometimes it is necessary to consider multiple criteria such as time, cost, and distance. In this case, we can formulate the MCFP as a multiobjective MCFP (MOMCFP) [14]. This will allow the decision maker to consider various conflicting objective criteria and select the most appropriate solution from a set of Pareto-optimal solutions.

* Corresponding author.

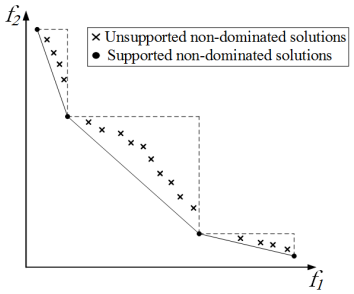


Fig. 1: Supported and unsupported non-dominated solutions in the objective space, for a bi-objective MCFP.

There are various types of MOMCFP: linear MOMCFP, integer multiobjective MCFP (MOIMCFP), and nonlinear integer multiobjective MCFP [14]. The linear MOMCFP has continuous decision variables with linear cost functions and constraints, where the solution set consists of only supported non-dominated points [14]. In contrast, the MOIMCFP has integer decision variables with linear cost functions and constraints, and the solution set consists of the supported and a large number of unsupported non-dominated points [5]. Supported non-dominated solutions can be found by applying weighted sum methods, but finding an unsupported non-dominated solution is a challenging task for mathematical programming methods [3, 17]. The supported non-dominated solutions are located on the convex hull of the feasible region, while the unsupported non-dominated solutions are found inside the feasible region [4]. The supported and unsupported non-dominated solutions for a bi-objective MCFP are shown in Fig. 1 (See Section 2 for the definition of supported and unsupported points). Finally, the nonlinear integer multiobjective MCFP has integer decision variables and employs nonlinear cost functions. The solution set for nonlinear integer multiobjective MCFP generally consists of both supported and unsupported points.

The approaches for solving MOMCFP and MOIMCFP can be categorised into exact and approximation methods [9]. A pseudo polynomial approximation algorithm was proposed in [15] to solve MOMCFP by approximating the optimal value function. In [6], a piecewise linear convex curve of a MOMCFP was approximated by following the trade-off curve. Other approximation methods make use of upper and lower bounds which "sandwich" the Pareto-front [9].

To solve MOMCFP using exact methods, a generalisation of the *out-of-kilter* method can be used [12]. However, this algorithm dealt with only small-sized networks, and the extreme non-dominated solution in the objective space could not be found. A modified out-of-kilter algorithm was developed in [10] to overcome the drawbacks of the generalised out-of-kilter method. In [16], a method was used to identify all the efficient extreme points in the objective space [16]. On the other hand, a branch-and-bound method [18] can be used to produce a representative set of Pareto-optimal solutions for MOIMCFP. All the above

mentioned methods considered only small to medium-sized linear MOMCFPs or MOIMCFPs.

The main challenge in mathematical programming for solving MOIMCFPs is to find the unsupported non-dominated points and ultimately the entire set of non-dominated solutions efficiently, without violating the MCFP's constraints [5]. Although algorithms exist to generate the entire set of non-dominated solutions, e.g., using the ϵ-constraint and branch-and-bound method to generate a set of non-dominated solutions for bi-objective integer MCFP [3], or using a two-phase parametric simplex algorithm [13], these algorithms can only solve small and medium-sized bi-objective MCFPs with reasonable efficiency. However, when dealing with a much larger MCFP, e.g., a network consisting of 50 nodes and 870 arcs, the computational time can be as high as 14,000 seconds.

Mathematical programming methods make a strong assumption that the cost function is linear or convex. However, most real-world problems are nonlinear and non-convex in their more realistic settings. It would be desirable to have a method that does not make these assumptions and can handle both linear and nonlinear cost functions effectively.

The above identified limitations motivated us to employ NSGA-II to solve MOIMCFP with a probabilistic tree-based representation scheme. NSGA-II is able to handle MOIMCFP using either linear or nonlinear cost functions. The algorithm can also control the number of non-dominated solutions to be generated. We compare the NSGA-II method with the state-of-the-art mathematical programming method Bensolve [11] on a set of small to medium-sized MOIMCFP instances. Our results suggest the superiority of NSGA-II over Bensolve in at least two aspects: 1) NSGA-II can find a set of non-dominated solutions (both supported and unsupported) with control on its size, but Bensolve can only generate one solution depending on its previous solution in a sequential manner, with no control over the total number of solutions. Furthermore, it produces only supported non-dominated points; 2) NSGA-II can handle both linear and nonlinear cost functions, but Bensolve is confined to handling just linear cost functions.

The rest of the paper is structured as follows: Section 2 provides the preliminaries and the definition of MOIMCFP. The NSGA-II method using the probabilistic tree-based representation is presented in Section 3. Section 4 presents the computational results, and finally Section 5 concludes the paper.

2 Problem formulation

Let $G = (\mathcal{N}, A)$ be a set $\mathcal{N}$ which consists of n nodes and a set A of m arcs. Each arc (i, j) has a capacity of u_{ij} and lower bound of l_{ij} which denote the maximum and minimum amount that can be sent on the arc (i, j), respectively. Each node $i \in \mathcal{N}$ is associated with an integer value $b(i)$. If $b(i)$ is positive, it shows that node i is a supply node, if $b(i)$ is negative, node i is a demand node with demand of $\mid b(i) \mid$ and finally $b(i) = 0$ shows the transshipment node i. A decision variable in MOIMCFP is an *integer* flow and denoted by x_{ij}. $F(\mathbf{x}) = (f_1(x), \ldots, f_N(x))$

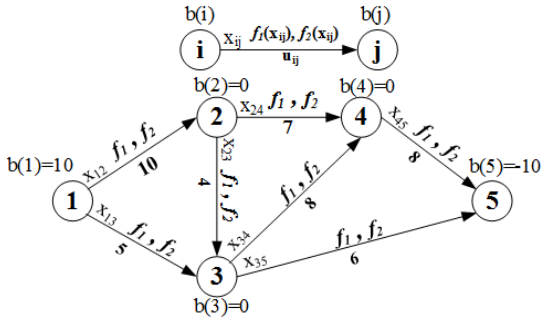


Fig. 2: An example of a bi-objective MCFP (time and cost) (n=5, m=7).

defines N objective functions for the MOIMCFP. Fig. 2 shows an example of the bi-objective MCFP (with n=5 nodes and m=7 arcs), which has a supplier node ($b(1) = 10$) and a demand node ($b(5) = -10$). There are two different costs associated to each arc: $f_1(x_{ij})$ denotes the time that takes to send a flow from node i to node j; $f_2(x_{ij})$ denotes the cost of sending a flow on the arc (i, j).

Generally in MOIMCFP, we aim to send flows through the network to satisfy all demands by minimising the objective functions. The formulation of the MOIMCFP is as follows [1]:

$$Minimise : F(\mathbf{x}) = \big(f_1(\mathbf{x}), \ldots, f_N(\mathbf{x})\big) \tag{1}$$

$$\text{s.t.} \quad \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \forall \quad i \in \mathcal{N}, \tag{2}$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall \quad (i,j) \in A, \tag{3}$$

$$x_{ij} \in \mathbb{Z} \quad \forall \quad (i,j) \in A, \tag{4}$$

where Eq.1 minimises multiple (N) objective functions through the network. Eq. 2 is a flow balance constraint which states the difference between the total outflow (first term) and the total inflow (second term). The flow on each arc should be between an upper bound and zero (Eq. 3), and finally all the flow values are integer numbers (Eq. 4). In this paper we consider the following assumptions for the MCFP: 1) the network is directed; 2) there are no two or more arcs with the same tail and head in the network; 3) the total demands and supplies in the network are equal, i.e., $\sum_{i=1}^{n} b(i) = 0$.

Since we are dealing with the multiobjective optimisation problem (MOIMCFP), it is necessary to declare the following definitions. The feasible region in the variable space is defined by $X^I = \{\mathbf{x} = (x_{i_1 j_1}, \ldots, x_{i_n j_n}) \in \mathbb{Z}^n : \mathbf{x}$ satisfies Eqs. 2-4$\}$. The feasible region in the objective space is defined by $Y^I = F(X^I) = \{\mathbf{y} = (y_1, \ldots, y_N) : y_1 = f_1(\mathbf{x}), \ldots, y_N = f_N(\mathbf{x}), \mathbf{x} \in X^I\}$. Let $X = conv(X^I)$ and $Y = conv(Y^I)$ be the convex hull of the sets X^I and Y^I, respectively. Let $y', y'' \in \mathbb{R}^N$, the following notation $y' \leqq y''$ denotes that $y'_q \leq y''_q, \forall q = 1, \ldots, N$.

Definition 2.1. Consider two feasible vectors y' and y'' in $Y^I(Y)$, y' *dominates* y'' if $y' \leqq y''$ and $y' \neq y''$ ($y'_q \leq y''_q$) with at least one strict inequality. The vector y' is said to be a *non-dominated* solution if there does not exist another y in Y^I which satisfies $y \leqq y'$ and $y \neq y'$. The set of all non-dominated points in Y^I is denoted by $ND(Y^I)$.

Definition 2.2. There are two categories of non-dominated solutions in MOIMCFP called supported and unsupported non-dominated solutions. Let $Y^{\geqq} = conv(ND(Y^I) + \mathbb{R}^N_{\geqq})$ where $\mathbb{R}^N_{\geqq} = \{y \in \mathbb{R}^N | y \geqq 0\}$ and $ND(Y^I) + \mathbb{R}^N_{\geqq} = \{y \in \mathbb{R}^N : y = y' + y'', y' \in ND(Y^I) \text{ and } y'' \in \mathbb{R}^N_{\geqq}\}$. If y is on the boundary of the $Y^{\geqq}$, y is referred to as a *supported* non-dominated solution, otherwise y is an *unsupported* non-dominated solution. Note that y is always denoted here as a non-dominated solution in the objective space.

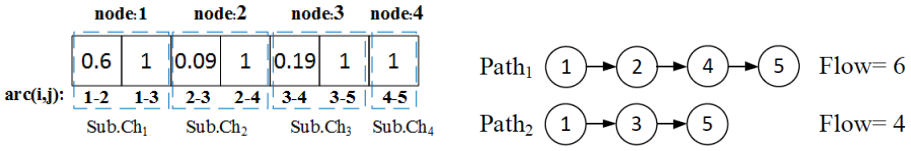
Definition 2.3. A solution $x' \in X^I$ (in the variable space) is called *efficient* if it is not possible to find another solution ($x \in X^I$) with a better objective function value without deteriorating the value of at least another objective value.

3 NSGA-II and the probabilistic tree-based representation for MCFP

As aforementioned, when dealing with MOIMCFP, there are supported and unsupported non-dominated solutions in the objective space. Supported non-dominated solutions can be found by applying weighted sum methods [3, 17], while finding an unsupported non-dominated solution is a challenging task by exact methods. Several such exact methods rely on a mechanism that must find a large number of non-dominated solutions one by one across the entire Pareto-front. However, there is no control of the number of non-dominated solutions to be produced. The excessive number of solutions is unnecessary to the decision maker (DM) and may also require a very high computational cost [5]. In this paper, we propose to use NSGA-II to solve the MCFP with a probabilistic tree-based representation, in order to find a controllable set of non-dominated solutions including both supported and unsupported points.

3.1 Representation

The most popular representation method for solving MCFP using genetic algorithm (GA) is priority-based representation (PbR) [7]. However, the PbR method has serious drawbacks. To counteract the limitations of PbR, the PTbR (Probabilistic Tree based Representation) is introduced in [8]. The PTbR chromosome has n-1 sub-chromosomes (Sub.Ch) and the value of each gene is a random number between 0 and 1 which is then accumulated to 1 in each sub-chromosome. In order to obtain a feasible solution from PTbR, in phase I, a path is first constructed, and then a feasible flow is sent through the constructed path in phase II. An example of PTbR and its feasible solution (for the network in Fig. 2) are shown in Fig. 3.



(a) The PTbR chromosome. (b) A feasible path and flows.

Fig. 3: PTbR and its corresponding feasible solution for the network in Fig. 2.

3.2 NSGA-II for solving MOIMCFP

After describing the PTbR, we now present how to adopt PTbR and apply NSGA-II to solve MOIMCFP.

Initialisation: First a population with *pop_size* individuals (chromosomes) is generated (P_0). To initialise the population, we can either generate the whole population randomly or use a heuristic technique to seed the initial population. As aforementioned, the mathematical programming techniques can efficiently generate the supported non-dominated points for MOIMCFP with linear objective function. One state-of-the-art mathematical solver that can create the supported non-dominated points is Bensolve [11].

To initialise the population, we apply two different approaches. The first approach is to generate individuals (chromosomes) randomly, namely NSGA-II(R). The second approach is to initialise $\alpha\%$ of the individuals using the supported point obtained from a mathematical programming method (e.g., Bensolve) and the remaining individuals are generated randomly (namely NSGA-II(H)). After initialising the population the binary tournament and genetic operators are applied to create the population Q_0.

NSGA-II procedure: In each iteration It, we combine the P_{It} with Q_{It} to form the R_{It} population, which is then sorted based on the non-dominance definition. We then calculate the crowding distance, and by sorting the population in each front we select the best-fit *pop_size* individuals in the population as P_{It+1} [2].

Crossover and mutation: The new population P_{It+1} is now used for binary tournament selection, crossover, and mutation to form a new population Q_{It+1}. A two-point crossover operation is applied, where two blocks (sub-chromosomes) of the selected chromosome (parents) are first randomly selected. Then, two parents swap the selected sub-chromosomes to generate new offspring. To perform mutation, a random parent is selected and the randomly chosen sub-chromosome is regenerated to create a new offspring [8].

Termination criteria: After genetic operators are applied, we first check the stopping criteria: 1) if the maximum number of function evaluations (NFEs) is reached; or 2) no improvement in the average of the objective function values on the Pareto-front for β successive iterations. If any of the criteria is satisfied the algorithm is terminated; otherwise we go back to the NSGA-II step.

4 Computational results

In the following subsection, we first describe the cost functions, followed by the network instances which we have adopted for MOIMCFP. We tackle the test instances using two variants of NSGA-II, i.e., NSGA-II(R) and NSGA-II(H), as well as the state-of-the-art mathematical solver Bensolve. Since Bensolve is not able to solve the MOIMCFP using nonlinear cost functions, we only present results in comparing NSGA-II(H) and NSGA-II(R) for MOIMCFP using nonlinear cost functions. Note that for NSGA-II(H) using nonlinear cost functions, the *fmincons()* in MATLAB is employed to generate the initial solutions.

4.1 Test instances

This paper considers MOIMCFP using linear and nonlinear cost functions. Note that, two objectives considered here are time (f_1) and cost (f_2). Our aim is to compare the performance of NSGA-II and Bensolve on bi-objective MCFPs. Eq. 1 can be rewritten as follows by employing the linear cost functions [9]:

$$Minimise : F(\mathbf{x}) = \big\{ f_1(\mathbf{x}) = \sum_{(i,j)\in A} c_{ij}^1 x_{ij},\ f_2(\mathbf{x}) = \sum_{(i,j)\in A} c_{ij}^2 x_{ij} \big\}, \tag{5}$$

where c_{ij}^1 and c_{ij}^2 are non-negative integer time and cost associated with one unit of flow on arc (i,j) respectively. To consider nonlinearity, the following concave nonlinear cost functions are adopted [20]:

$$Minimise : F(\mathbf{x}) = \big\{ f_1(\mathbf{x}) = \sum_{(i,j)\in A} c_{ij}^1 \sqrt{x_{ij}},\ f_2(\mathbf{x}) = \sum_{(i,j)\in A} c_{ij}^2 \sqrt{x_{ij}} \big\}. \tag{6}$$

For our experiments, a set of 30 MOIMCFP instances with different number of nodes ($n = \{5, 10, 20, 40, 60, 80\}$) is randomly generated and presented in Table 1 (*No.* denotes the instance number, and each instance has n nodes and m arcs). Note that, for each node size (n), five different networks are randomly generated. The number of supply/demand for nodes $1/n$ are set to q=20/-20 in the test instances up to 20 nodes and for all other test problems supply/demand are set to q=30/-30 [8].

Table 1: A set of 30 randomly generated MOIMCFP instances.

No.	n	m	*No.*	n	m	*No.*	n	m	*No.*	n	m	*No.*	n	m	*No.*	n	m
1		6	6		25	11		86	16		287	21		697	26		1322
2		7	7		28	12		81	17		336	22		721	27		1298
3	**5**	8	8	**10**	25	13	**20**	87	18	**40**	370	23	**60**	635	28	**80**	1356
4		8	9		28	14		74	19		334	24		693	29		1250
5		6	10		27	15		92	20		358	25		695	30		1140

4.2 Results and analysis

NSGA-II and PTbR are implemented in MATLAB on a PC with Intel(R) Core(TM) i7-6500U 2.50 GHz processor with 8 GB RAM, and we run 30 times for each problem instance. To solve MOIMCFP instances using a mathematical solver, we use the MATLAB version of Bensolve[1].

The parameter settings for NSGA-II are as follows: maximum number of iterations (It_{max}=200), population size ($pop\_size = min\{n \times 10, 300\}$), crossover rate ($P_c$=0.95), mutation rate ($P_m$=0.3), maximum number of function evaluations ($NFEs$=100,000) and the termination criterion $\beta = 30$ [8]. For NSGA-II(H) only $\alpha = 10\%$ of the initial individuals are generated using the heuristic method explained in Section 3.2 and the rest are generated randomly.

To evaluate the performance of a multiobjective optimisation algorithm, two aspects need to be measured: convergence and distribution of the solutions approaching the Pareto front [2]. We adopt Hypervolume (HV) (or S metric), a widely-used metric for evaluating the performance of a multiobjective optimisation algorithm [21]. Hypervolume computes how close the solutions are to the Pareto-front as well as the spread of the solutions across the Pareto-front [19].

We compare the performance of NSGA-II(R), NSGA-II(H) with Bensolve on all network instances using linear cost functions (Eq. 5). The results are presented in Table 2 (where t, nPF and HV denote the average running time in second, average number of solutions on the Pareto-front and average of the hypervolume metric, respectively over 30 runs). Since Bensolve cannot solve the nonlinear MOIMCFP, we only present the results of the NSGA-II(R) and NSGA-II(H) for solving the network instances using concave nonlinear cost functions (Eq.6) in Table 3, including the average time (t) and average hypervolume value (HV). Note that for NSGA-II(H) using nonlinear cost functions (Table 3), we employ *fmincons()* in MATLAB to seed the initial population by converting the bi-objective problem to the single objective problem using a weighted sum method (i.e., considering equal weights for all objective functions). *fmincons()* uses an interior point method by default to solve constrained nonlinear single objective problems.

As shown in Table 2, NSGA-II(H) has greater or equal HV value than those of the Bensolve and NSGA-II(R). Note that, in Tables 2 and 3 the algorithm with the best HV value for each instance is highlighted in boldface. It is noticeable

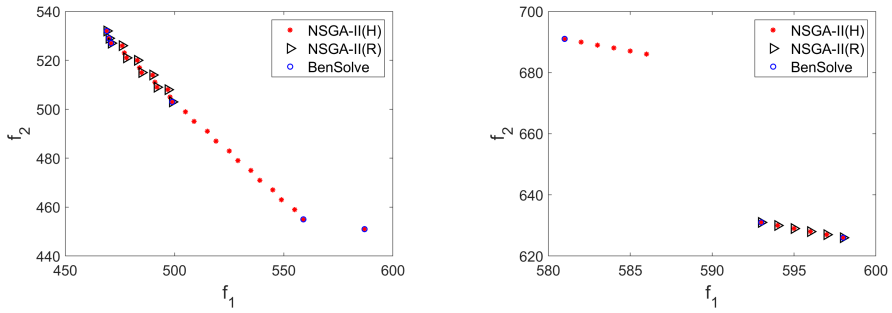
[1] Bensolve MATLAB version is available on: http://bensolve.org/.

Table 2: Results for solving MOIMCFP using linear objective functions.

No.	n	m	Bensolve			NSGA-II(R)			NSGA-II(H)		
			t	nPF	HV	t	nPF	HV	t	nPF	HV
1		6	1	2	4248	16	3.0	**4274.4**	17	3.0	**4274.4**
2		7	1	3	2326	14	11.0	**2385.7**	14	11.0	**2385.7**
3	5	8	1	2	2910	13	6.0	**3030.0**	14	6.0	**3030.0**
4		8	1	2	1570	16	3.0	**1605.0**	16	3.0	**1605.0**
5		6	1	2	2114	13	6.0	**2153.6**	14	6.0	**2153.6**
6		25	1	3	7373	59	17.9	7624.9	66	18.0	**7643.4**
7		28	1	3	10155	59	2.9	9058.7	66	6.3	**11047.8**
8	10	25	1	2	4368	66	4.0	**4394.6**	83	4.0	**4394.6**
9		28	1	2	4347	53	5.0	**4509.3**	58	5.0	**4509.3**
10		27	1	3	5872	55	5.8	5629.6	63	8.7	**5994.7**
11		86	1	2	3866	105	9.8	4712.1	102	10.1	**4760.3**
12		81	1	2	2722	113	12.3	3003.3	105	12.8	**3010.0**
13	20	88	1	3	2894	96	4.0	2850.8	106	4.4	**2913.6**
14		74	1	2	2700	103	3.3	2821.0	94	5.7	**2999.5**
15		92	1	3	9872	98	7.3	10345.8	109	7.9	**10408.7**
16		287	1	1	**1093**	108	1.0	**1093.0**	106	1.0	**1093.0**
17		336	1	2	3168	107	2.1	2669.9	108	2.2	**3189.1**
18	40	370	1	3	8269	111	2.2	7915.6	111	3.1	**8280.4**
19		334	1	3	4423	105	2.1	3685.8	105	3.0	**4426.8**
20		358	1	1	**1215**	103	1.0	**1215.0**	105	1.0	**1215.0**
21		697	1	5	**5898**	192	2.0	5057.3	223	5.7	**5898.0**
22		721	1	6	18194	217	10.5	14139.5	213	27.4	**19755.7**
23	60	635	1	5	16731	216	4.0	14576.1	214	10.5	**16920.3**
24		693	1	2	7203	231	5.5	7730.7	224	5.9	**7795.5**
25		695	1	3	7643	228	3.6	6807.9	226	5.5	**7889.3**
26		1322	1	3	6484	269	19.0	**6723.6**	277	19.0	**6723.6**
27		1298	1	4	8692	227	13.0	8714.5	217	14.0	**8961.7**
28	80	1356	1	5	25518	317	22.2	25141.1	293	28.0	**26632.6**
29		1250	1	4	11130	271	6.0	10838.6	262	8.0	**11178.7**
30		1140	1	3	9494	243	6.0	8674.7	258	12.0	**9548.9**

Table 3: Results for solving MOIMCFP using nonlinear objective functions.

Algorithms	No.	*1*	2	3	4	5	6	7	8	9	10
NSGA-II(R)	t	20	17	16	18	15	71	63	75	64	72
	HV	21970.0	7975.4	14794.0	10521.8	12478.0	28017.9	38407.0	28573.1	28265.2	37078.8
NSGA-II(H)	t	20	18	14	17	15	69	62	69	57	64
	HV	**21970.0**	**7975.4**	**14794.0**	**10521.8**	**12478.0**	**28017.9**	**38653.6**	**28573.1**	**28616.9**	**38624.5**
Algorithms	*No.*	11	12	13	14	15	16	17	18	19	20
NSGA-II(R)	t	129	113	128	129	131	139	147	145	139	141
	HV	**9801.6**	**8184.8**	8947.6	**7944.3**	**14962.6**	5096.2	10519.8	12607.4	10436.2	**4791.2**
NSGA-II(H)	t	111	111	99	101	103	104	104	107	109	112
	HV	**9801.6**	**8184.8**	**8973.3**	**7944.3**	**14962.6**	**5130.8**	**10804.1**	**13023.9**	**10982.4**	**4791.2**
Algorithms	*No.*	21	22	23	24	25	26	27	28	29	30
NSGA-II(R)	t	254	275	264	264	264	304	264	289	337	324
	HV	4471.0	8195.8	19618.1	25460.3	30509.0	283.3	6072.9	13885.3	3122.5	10808.9
NSGA-II(H)	t	238	235	222	244	248	333	232	309	312	322
	HV	**13799.6**	**27278.5**	**30252.9**	**31030.9**	**34056.3**	**24882.2**	**17830.5**	**35182.3**	**21993.3**	**31489.4**



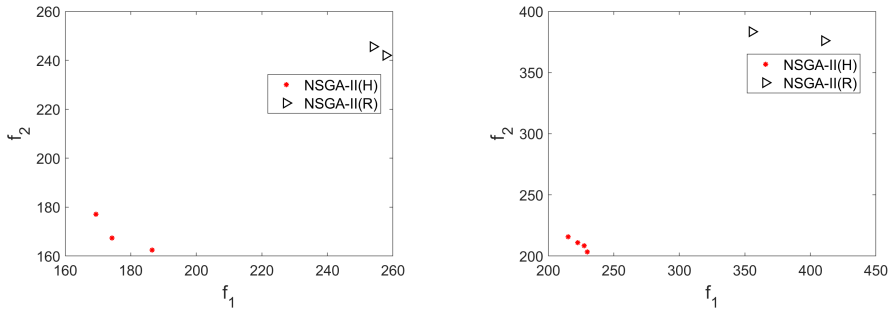
(a) Instance No.22. (b) Instance No.30.

Fig. 4: Results for solving MOIMCFP using linear objective functions.

that Bensolve terminates after 1 second on all instances since they cannot find any other solutions. Although our NSGA-II variants took longer to generate the non-dominated solutions, it can converge to a better set of non-dominated solutions with a better diversity as indicated by the HV metric. As can be seen in Fig. 4, Bensolve is only capable of finding supported non-dominated points, and NSGA-II(R) is able to find just one part of the Pareto-front, however NSGA-II(H) is able to find a set of non-dominated solutions with better diversity and convergence. This shows the superiority of NSGA-II(H) over NSGA-II(R) and Bensolve.

As shown in Fig. 4a, for network instance No.22, Bensolve can find only 6 points on the Pareto-front, which are all supported non-dominated points with HV=18,194, while NSGA-II(R) can find on average 10.5 non-dominated points with HV=14,139.5 and NSGA-II(H) can obtain on average 27.4 non-dominated points with HV=19,755. It shows that NSGA-II(H) provided not only a better quality of non-dominated solutions, but also better solution diversity (Fig. 4b). This pattern is observed on all instances in Table 2, indicating that NSGA-II(H) has better performance than Bensolve and NSGA-II(R).

Table 3 shows the results on the MOIMCFP instances in Table 1 using concave nonlinear cost functions (Eq. 6) using NSGA-II(H) and NSGA-II(R). On all instances NSGA-II(H) has equal or better performance than the NSGA-II(R). For example, Fig. 5 shows that NSGA-II(H) can converge to a better non-dominated solution set as compared to NSGA-II(R) on instances No.26 and 27. It is consistent with the results of MOIMCFP using linear cost functions and it suggests that using heuristic initialisation (or seeding) can dramatically improve the performance of NSGA-II in dealing with MOIMCFPs.



(a) Instance No.26. (b) Instance No.27.

Fig. 5: Results for solving MOIMCFP using nonlinear objective functions (Bensolve is not included since it cannot handle nonlinear objective functions).

5 Conclusion

In this paper we have adopted the probabilistic tree-based representation for handling the MCFPs, and apply NSGA-II to solve the MOIMCFP using linear and nonlinear cost functions. Unlike the mathematical solvers which are unable to handle nonlinear cost functions, NSGA-II is more robust in dealing with various types of cost functions. The performance of the two variants of NSGA-II (i.e., NSGA-II(H) and NSGA-II(R)) algorithms are evaluated on a set of 30 MOIMCFP instances and compared with that of the state-of-the-art mathematical solver Bensolve. The experimental results demonstrate that NSGA-II(H) has superior performance than that of the Bensolve and NSGA-II(R) in terms of the quality of solutions as well as the diversity of solutions in the objective space. As can be seen in Fig. 4, Bensolve only managed to generate a limited number of solutions (i.e., supported non-dominated solutions) and it cannot find the unsupported non-dominated solutions. However, NSGA-II does not have such a limitation and is able to generate a controllable set of non-dominated solutions. Furthermore, Bensolve cannot handle nonlinearity. It is also worth noting that using a heuristic initialisation procedure (i.e., seeding with solutions found by an exact method) can improve the performance of NSGA-II for solving MOIMCFP.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows: theory, algorithms, and applications. Prentice hall , pp.4–6 (1993)
2. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation **6**(2), 182–197 (2002)

3. Eusébio, A., Figueira, J.R.: Finding non-dominated solutions in bi-objective integer network flow problems. Computers & Operations Research **36**(9), 2554–2564 (2009)
4. Eusébio, A., Figueira, J.R.: On the computation of all supported efficient solutions in multi-objective integer network flow problems. European Journal of Operational Research **199**(1), 68–76 (2009)
5. Eusébio, A., Figueira, J.R., Ehrgott, M.: On finding representative non-dominated points for bi-objective integer network flow problems. Computers & Operations Research **48**, 1–10 (2014)
6. Fruhwirth, B., Bukkard, R., Rote, G.: Approximation of convex curves with application to the bicriterial minimum cost flow problem. European Journal of Operational Research **42**(3), 326–338 (1989)
7. Gen, M., Cheng, R., Lin, L.: Network models and optimization: Multiobjective genetic algorithm approach. Springer Science & Business Media (2008)
8. Ghasemishabankareh, B., Ozlen, M., Neumann, F., Li, X.: A probabilistic tree-based representation for non-convex minimum cost flow problems. In: International Conference on Parallel Problem Solving from Nature. pp. 69–81. Springer (2018)
9. Hamacher, H.W., Pedersen, C.R., Ruzika, S.: Multiple objective minimum cost flow problems: A review. European Journal of Operational Research **176**(3), 1404–1422 (2007)
10. Lee, H., Pulat, P.S.: Bicriteria network flow problems: Continuous case. European Journal of Operational Research **51**(1), 119–126 (1991)
11. Löhne, A., Weißing, B.: The vector linear program solver bensolve–notes on theoretical background. European Journal of Operational Research **260**(3), 807–813 (2017)
12. MALHOTRA, R., Puri, M.: Bi-criteria network problem. Cahiers du Centre d'études de recherche opérationnelle **26**(1-2), 95–102 (1984)
13. Raith, A., Ehrgott, M.: A two-phase algorithm for the biobjective integer minimum cost flow problem. Computers & Operations Research **36**(6), 1945–1954 (2009)
14. Raith, A., Sedeño-Noda, A.: Finding extreme supported solutions of biobjective network flow problems: An enhanced parametric programming approach. Computers & Operations Research **82**, 153–166 (2017)
15. Ruhe, G.: Algorithmic aspects of flows in networks, vol. 69. Springer Science & Business Media (2012)
16. Sedeño-Noda, A., González-Martın, C.: The biobjective minimum cost flow problem. European Journal of Operational Research **124**(3), 591–600 (2000)
17. Steuer, R.: Multiple criteria optimization: theory, computation, and application. 1986. Willey, New York
18. Sun, M.: A branch-and-bound algorithm for representative integer efficient solutions in multiple objective network programming problems. Networks **62**(1), 56–71 (2013)
19. While, L., Hingston, P., Barone, L., Huband, S.: A faster algorithm for calculating hypervolume. IEEE transactions on evolutionary computation **10**(1), 29–38 (2006)
20. Yan, S., Shih, Y., Wang, C.: An ant colony system-based hybrid algorithm for square root concave cost transhipment problems. Engineering Optimization **42**(11), 983–1001 (2010)
21. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: Methods and applications, vol. 63. Citeseer (1999)