Optimization Letters (2024) 18:1131–1149
https://doi.org/10.1007/s11590-023-02040-6

ORIGINAL PAPER

A sequential reduction algorithm for the large-scale fixed-charge network flow problems

Lu Yang[1] **· Zhouwang Yang**[1]

Received: 2 March 2023 / Accepted: 29 June 2023 / Published online: 13 July 2023

Abstract
The fixed-charge network flow problem (FCNFP) is widely used in industrial production and can be exactly solved by converting to mixed-integer linear programming (MILP). However, the long solving time of MILP solvers limits their applicability to large-scale problems. This paper proposes a sequential reduction algorithm called adaptive dynamic slope scaling procedure (ADSSP). ADSSP introduces an adaptive edge deletion strategy to improve solution quality and efficiency. Theoretical analysis proves the algorithm convergence and provides the best-case and worst-case time upper bounds of ADSSP. Numerical experiments show that ADSSP outperforms the previous linear programming-based iterative algorithms on different scales. In instances with hundreds of thousands of variables, ADSSP achieves a 2% improvement in the objective and takes only 3% solving time compared to the Cplex MILP Solver. The results demonstrate that ADSSP can significantly improve the solution quality with high efficiency for large-scale FCNFP. As a widely applicable optimization method, ADSSP can be a valuable tool for optimizing FCNFP and other similar problems.

Keywords Fixed-charge network flow · Heuristic algorithm · Edge deletion · Transportation problem

1 Introduction

The fixed-charge network flow problem (FCNFP) appears widely in the field of network optimization, in areas such as transportation science [1–4], integrated scheduling of production [5] and sources supply networks [6]. FCNFP is a subtype of the

✉ Zhouwang Yang
yangzw@ustc.edu.cn

Lu Yang
yl0501@mail.ustc.edu.cn

[1] University of Science and Technology of China, Hefei, Anhui, P. R. China

fixed cost linear programming problem, a nondeterministic polynomial-time hard (NP-hard) problem, introduced in [7].

Let $G = (V, E)$ be a directed graph with n nodes and m edges, where V is the set of nodes and E is the set of edges. Each node $i \in V$ is associated with a supply b_i. Each edge $e \in E$ is associated with a flow x_e, a capacity u_e, a unit cost c_e, and a fixed cost s_e. The cost function $C_e(x_e)$ of edge e is discontinuous and can be expressed as:

$$C_e(x_e) = \begin{cases} 0 & x_e = 0 \\ s_e + c_e x_e & x_e \in (0, u_e] \end{cases} \tag{1}$$

Let $x = (x_1, x_2, \ldots, x_m)$ be the vector of flow, $u = (u_1, u_2, \ldots, u_m)$ be the vector of capacity, A be the node-edge incidence matrix of G, and $b = (b_1, b_2, \ldots, b_n)$ be the node supply vector. The FCNFP can be formulated as the following problem:

$$\begin{aligned} \min_{x} \quad & C(x) = \sum_{e \in E} C_e(x_e) \\ \text{s.t.} \quad & Ax = b \\ & 0 \leqslant x \leqslant u \end{aligned} \tag{2}$$

where $x, u \in \mathbb{R}^m, b \in \mathbb{R}^n,$ and $A \in \mathbb{R}^{n \times m}$.

The algorithms for solving FCNFP mainly include integer programming algorithms, linear programming-based (LP-based) iterative algorithms, and meta-heuristic algorithms. FCNFP can be formulated as a mixed-integer linear programming (MILP) problem by adding a 0–1 integer variable for each edge to indicate whether the flow is greater than zero. MILP can be directly solved by algorithms based on techniques such as cutting plane [8], Benders decomposition [9], column generation [10], lagrangian relaxation [11], and Dantzig-Wolfe decomposition [12]. LP-based iterative algorithms can quickly find suboptimal solutions by approximating discontinuous cost functions using (piecewise) linear functions [13–15]. Meta-heuristic algorithms, such as genetic algorithms [16] and population algorithms [1], are also used to solve FCNFP. While MILP is exact, its solving time can be uncertain or computationally infeasible, especially on large-scale problems. In addition, meta-heuristic algorithms lack theoretical guarantees of optimality and efficiency, and their long solving time makes them impractical for use in industrial production. Despite the advantages in efficiency for problem-solving, the performance of LP-based iterative algorithms still requires improvement.

This paper presents a novel algorithm called adaptive dynamic slope scaling procedure (ADSSP) for solving the fixed-charge network flow problem. ADSSP improves upon existing LP-based iterative algorithms by incorporating the dynamic slope scaling procedure (DSSP) and an edge deletion strategy. DSSP can explore the feasible domain by iteratively adjusting the slope scaling factors to guide the search toward promising solutions. The edge deletion strategy can reduce the interference of fixed cost from invalid edges on DSSP by eliminating the edges with zero flow. The alternating iteration framework of DSSP and edge deletion sequentially reduce

the feasible domain of FCNFP, enabling ADSSP to converge efficiently to a good solution. Through extensive experiments on large-scale instances, we demonstrate that ADSSP outperforms previous LP-based iterative algorithms and the commercial solver Cplex in both solution quality and computational efficiency. Our research contributes to the development of more effective and scalable optimization methods for real-world problems.

The remainder of this paper is organized as follows. Section 2 lists related work on LP-based iterative algorithms. Section 3 outlines the details of the proposed ADSSP algorithm, while Sect. 4 describes and discusses simulation results. This study concludes in Sect. 5 with a summary of the results.

2 Related work

The development of LP-based iterative algorithms for the FCNFP can be roughly divided into three stages. The first LP-based iterative framework for solving FCNFP called DSSP was proposed in [13]. This pioneering approach approximates the FCNFP as a series of linear programming problems. The linear objective through the origin allows the approximate problems to be solved quickly. Based on the DSSP framework, the meta-heuristic technique called Tabu Scheme was explored to produce promising search neighborhoods for high-quality solutions by iteratively adjusting the linear factors [17]. However, the lack of descent and convergence guarantee makes the solving time of DSSP uncertain, and the simple structure of the single-layer iterative framework makes it far from the optimal solution.

An approximate model using a piecewise linear function to approximate the discontinuous objective of FCNFP was proposed in [14] called the concave piecewise linear network flow (CPLNF) model. The complex piecewise linear objective is closer to the objective of FCNFP than the linear objective, but it also makes the CPLNF model hard to solve. That paper transformed the CPLNF model into a mixed-integer linear programming model and linearly relaxed the integer variables, and proposed an algorithm called the dynamic cost updating procedure to solve. An improved algorithm called adaptive dynamic cost updating procedure (ADCUP) dynamically modified the objective of CPLNF to approach the original objective function in iterations [18].

The complex objective function challenges the efficiency of problem-solving. The objective function of CPLNF can be simplified to a new approximate model called the continuous bilinear network flow (CBLNF), which can also be solved by ADCUP [15]. On this basis, ADCUP can be modified by transferring the two-layer iteration into a single-layer iteration to improve the efficiency, which is introduced in an algorithm called the dynamic problem updating procedure (DPUP) [19]. However, these algorithms yield solutions that might be far from optimal, with objectives up to ten percent larger than the optimal one.

3 Algorithm

3.1 Adaptive dynamic slope scaling procedure

This section introduces the heuristic algorithm ADSSP, an iterative framework combining an edge deletion strategy with DSSP. The edge deletion strategy aims to delete the edges unlikely to appear in the optimal solution, which improves the solution optimality and decreases the problem scale for the next iteration. Therefore, ADSSP can quickly find a good solution by solving FCNFP on sequentially reduced feasible domains. Fig. 1 displays the whole solving process of ADSSP. The initialized graph, $G^{(0)}(V, E^{(0)})$, generates an initial FCNFP to trig the solving process of ADSSP. In each iteration, DSSP solves the FCNFP induced by the current graph, $G^{(k)}(V, E^{(k)})$, to provide the solution $\{x_e^{(k)}\}$. Based on the selected deletable edge set $\Delta E_D^{(k)}$ generated from $\{x_e^{(k)}\}$, edge deletion can update $E^{(k)}$ by deleting edges in $\Delta E_D^{(k)}$. This iteration terminates when the generated $\Delta E_D^{(k)}$ is empty.

DSSP approximates FCNFP by a series of linear programming problems, where each edge's cost is redefined as a linear cost through the origin. The slope can be expressed as:

$$\bar{c}_e = c_e + s_e / x_e \tag{3}$$

where $x_e > 0$ is a given flow of edge e. An optional initialization of the slope is $\bar{c}_e^{(0)} = c_e + s_e / u_e$. The dynamic slope update formula is

$$\bar{c}_e^{(k+1)} = \begin{cases} \max\limits_{1 \le l \le k} \{\bar{c}_e^{(l)} : x_e^{(l)} > 0\} & x_e^{(k)} = 0 \\ c_e + s_e / x_e^{(k)} & x_e^{(k)} > 0 \end{cases} \tag{4}$$

DSSP stops when the solution in current iteration satisfies $x_e^{(k-1)} = x_e^{(k)}$ for all edge e. Because the stopping criterion lacks the guarantee of finite-step termination, a maximum number of iterations is set to avoid excessively long solving time, denoted as M_d. For accelerating the computation, we apply an efficient algorithm [20] to solve

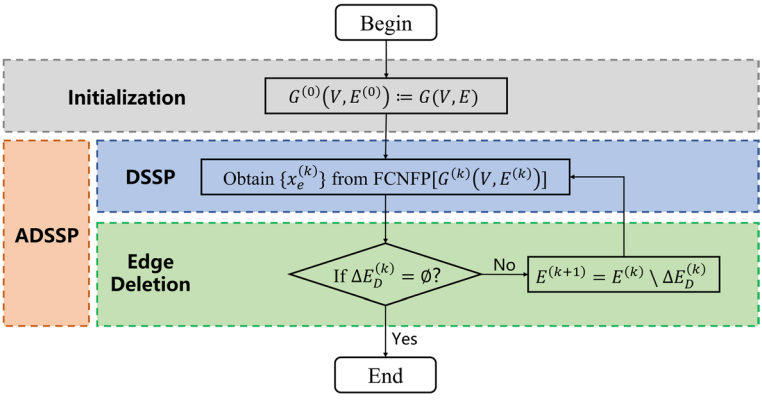


Fig. 1 The flowchart of ADSSP

the minimum-cost flow problem in DSSP. The detail of DSSP is shown in Algorithm 1, where $MCF\left(\bar{c}_e^{(k)}\right)$ represents the solution of a minimum-cost flow problem.

Algorithm 1 Dynamic slope scaling procedure (DSSP)

Require: $G(V,E)$; M_d
Ensure: $(x_{e_1}, x_{e_2}, \ldots, x_{e_m})$
1: **Initialization:** $\bar{c}_e^{(0)} = c_e + s_e/u_e$
2: **for** $k = 1$ to M_d **do**
3: $\quad x_e^{(k)} = MCF(\bar{c}_e^{(k-1)})$
4: $\quad$ **if** $x_e^{(k-1)} = x_e^{(k)}$ **then**
5: $\quad\quad$ break
6: $\quad$ **end if**
7: $\quad \bar{c}_e^{(k)} = \begin{cases} \max\limits_{1 \leqslant l < k} \{\bar{c}_e^{(l)} : x_e^{(l)} > 0\} & x_e^{(k)} = 0 \\ c_e + s_e/x_e^{(k)} & x_e^{(k)} > 0 \end{cases}$
8: **end for**
9: **return** $\mathbf{x}^*$

In the k-th iteration, the set of deletable edges is defined as $\Delta E^{(k)} = \{e \in E^{(k)}\ s_e > 0, x_e^{(k)} = 0\}$, where $x_e^{(k)}$ is obtained from the solution of DSSP. Since DSSP is not an exact algorithm, deleting all edges in $\Delta E^{(k)}$ may cause the algorithm to converge to a poor solution. An alternative approach is to delete a certain ratio of edges from $\Delta E^{(k)}$. The selection on the deleting edges in $\Delta E^{(k)}$ relies on the value ϕ_e calculated as follows.

- TYPE 1: Randomly select a proportion of edges from $\Delta E^{(k)}$ to delete.
- TYPE 2: Delete the edges with large fixed costs, that is $\phi_e = s_e$.
- TYPE 3: Delete the edges with large values $\phi_e = c_e + s_e/u_e$.
- TYPE 4: Delete the edges with large values $\phi_e = c_e + s_e$.

Among these four optional calculations, the first type is not affected by the costs on edges, which can be applied to any graph structure. The second type prioritizes the removal of the zero-flow edges with higher fixed costs. However, it is ineffective when the unit cost is significantly larger than the fixed cost. The third and fourth types rely on the average cost on edges with maximum flow and unit flow, respectively, which are more versatile than the previous two. The set of deleted edges is denoted as $\Delta E_D^{(k)}$, which can be expressed as

$$\Delta E_D^{(k)} = \{e \in \Delta E^{(k)}\ \phi_e > Q_{1-\alpha}(\boldsymbol{\phi})\}$$

where α is the deleting ratio and $Q_{1-\alpha}(\boldsymbol{\phi})$ is the $(1-\alpha)$ quantile of the set $\{\phi_e : e \in \Delta E\}$. After updating the edge set by $E^{(k+1)} = E^{(k)} - \Delta E_D^{(k)}$, the subproblem on the graph $G^{(k+1)}(V, E^{(k+1)})$ is also an FCNFP. The stopping criteria is defined as $\Delta E_D^{(k)} = \emptyset$. Without loss of generality, the number of forced termination steps is set to m. The detail of ADSSP is shown in Algorithm 2.

Algorithm 2 Adaptive dynamic slope scaling procedure (ADSSP)

Require: $G(V, E)$; M_d
Ensure: $(x_{e_1}, x_{e_2}, \ldots, x_{e_m})$
1: **Initialization:** $E^{(0)} = E$
2: **for** $k = 1$ to m **do**
3: $x_e^{(k)} = DSSP(G(V, E^{(k-1)}), M_d)$
4: $\Delta E^{(k)} = \{e \in E^{(k-1)} \ : \ s_e > 0, x_e^{(k)} = 0\}$
5: **if** $\Delta E^{(k)} = \emptyset$ **then**
6: break
7: **end if**
8: $\Delta E_D^{(k)} = \{e \in \Delta E^{(k)} \ : \ \phi_e > Q_{1-\alpha}(\phi)\}$
9: $E^{(k)} = E^{(k-1)} - \Delta E_D^{(k)}$
10: **end for**
11: **return** $\mathbf{x}^*$

3.2 Algorithm convergence and time complexity

DSSP lacks time complexity analysis because its termination condition does not provide a definite number of iterations. However, ADSSP sets the iteration upper limit of DSSP as M_d, which makes Algorithm 2 terminates in a finite number of iterations. Thus, theoretical analysis of time complexity is allowed for ADSSP.

Proposition 1 *The ADSSP solves the FCNFP in a finite number of iterations. The worst-case scenario is that the iteration number is* $m - 1$, *while the best-case scenario is that the iteration number is* $\lceil log_{\frac{1}{1-\alpha}}(m-1) \rceil + 1$, *where* m *is the number of edges in the graph and* α *is the deleting ratio.*

Proof Assuming that the optimal solution consists of only one edge with the nonzero flow in the graph. The worst-case scenario means that only one edge can be deleted per iteration, resulting in a total of $m - 1$ iterations. In the best case, all edges except the nonzero flow edge can be deleted in each iteration. The maximum number of iterations K satisfies $(1-\alpha)^{(K-1)}(m-1) = 1$, where α is the proportion of edges to be deleted. The upper bound on the number of iterations for ADSSP in the best case is $K = \lceil log_{\frac{1}{1-\alpha}}(m-1) \rceil + 1$. □

Theorem 1 *The time bound of ADSSP is* $O(nm \log(n^2) \log(nC) \Gamma(m, \alpha))$, *where* $\Gamma(m, \alpha) = m$ *in the worst case and* $\Gamma(m, \alpha) = \frac{1-(1-\alpha)^K}{\alpha}$ *in the best case.*

Proof The time bound of $O(nm \log(n^2/m) \log(nC))$ on a graph with m edges for minimum-cost circulation problems is given in [20], which is the time complexity

of the function $MCF(\cdot)$ in Algorithm 1. Since the other operations in loop is $O(m)$ and the number of iteration M_d is a constant, the time bound of Algorithm 1 is also $O(nm\log(n^2/m)\log(nC))$.

The time bound of the function $DSSP(\cdot)$ in Algorithm 2 is $O(nm_k\log(n^2/m_k)\log(nC))$. In the worst case, the time bound of ADSSP is

$$\begin{aligned} &O\left(\sum_{i=2}^{m} ni\log(n^2/i)\log(nC)\right) \\ &= O\left(n\log(n^2)\log(nC)\sum_{i=2}^{m} i - n\log(nC)\sum_{i=2}^{m} i\log(i)\right) \\ &= O\left(nm^2\log(n^2)\log(nC)\right) \end{aligned}$$

In the best case, the time bound of ADSSP is[1]

$$\begin{aligned} &O\left(\sum_{i=0}^{K-1} n(1-\alpha)^i m\log(\frac{n^2}{(1-\alpha)^i m})\log(nC)\right) \\ &= O\left(nm\log(n^2)\log(nC)\sum_{i=0}^{K-1}(1-\alpha)^i - nm\log(nC)\sum_{i=0}^{K-1}(i\log(1-\alpha)+\log m)(1-\alpha)^i\right) \\ &= O\left(nm\log(n^2)\log(nC)\frac{1-(1-\alpha)^K}{\alpha}\right)^1 \end{aligned}$$

□

4 Numerical experiment

This section consists of a data illustration and three experiments. First, we introduce the construction of the synthetic data and provide the topological information and generated distributions of parameters for instances. Second, we give the range of choices for algorithm hyperparameters and the selection of the edge-deleting rule through comparative experiments. Based on the selected hyperparameters and rule, we demonstrate the advantages of ADSSP from two perspectives: problem scale and combinatoriality. The combinatoriality of the problem is defined in terms of the order of magnitude difference between unit cost and fixed cost.

By solving the following MILP model that is equivalent to FCNFP, the solution provides a good benchmark for comparing algorithms.

[1] A more accurate upper bound can be obtained by considering the second item, but it is not used here as a conclusion since the complicated form.

$$\begin{aligned} \min_{x,y} \quad & C_I(x) = \sum_{e \in E} c_e x_e + s_e y_e \\ \text{s.t.} \quad & Ax = b \\ & 0 \le x_e \le u_e y_e, \forall e \in E \\ & y_e \in \{0,1\}, \forall e \in E \end{aligned} \tag{5}$$

Cplex is an advanced commercial solver used for solving MILP, capable of finding the optimal solution. However, as the problem scale increase and combinatoriality enhance, the solving time of Cplex becomes unpredictable. By setting a time limit for solving, Cplex can provide the optimal solution for small-scale problems and suboptimal solutions for large-scale problems.

In this paper, numerical experiments are conducted in C++ 17 on a Windows 10 platform with an Intel Core i7 2.50 GHz processor and 32.0 GB of RAM. The version of Cplex MILP Solver is 12.10.0, and the time limit for solving MILP is 3600 s.

4.1 Data

The experimental instances are constructed on the graph with given parameters, including small-scale and large-scale ones. The scale of FCNFP relies on its number of nodes and edges. Referring to [19], we set nine scales for small-scale instances and four for large-scale instances. The instance with the largest scale in this paper contains ten thousand nodes and one million edges. In graph generation, nodes are selected randomly from a two-dimensional plane and labeled in order. Each node is associated with a supply, the type of which is determined by its relationship with zero. Edges are generated between nodes whose distance is smaller than a specified threshold. The structure data of the graph are shown in Table 1.

In each instance, we need to generate the supply on nodes, and the capacity, unit cost, and fixed cost on edges. The experimental data construction in the related work does not specify how these parameters generating. To construct instances more realistic, we sample these parameters according to distributions in Table 2. In addition, the combinatoriality coefficient γ aims to adjust the magnitude difference between the unit and fixed cost, making the instances with different combinatoriality. The instance feasibility is verified by a minimum-cost flow algorithm by setting the fixed cost to zero.

4.2 Hyperparameters and deleting rule in ADSSP

The hyperparameters M_d and α are pre-given in ADSSP. The parameter selection is guided by comparing the relative error between the optimal value and the objective of ADSSP on a grid of parameter values. We choose M_d from the set $\{i \in \mathbb{Z} : 1 \le i \le 30\}$ and select α from the set $\{0.05 * k : k = 1, 2, \ldots, 20\}$. The experiments are conducted on multiple groups of small-scale instances with strong combinatoriality to make the results more illustrative. Figure 2 shows the relative errors on gridded parameters. The darker area, i.e., the area with smaller relative

Table 1 The topological information of the graph for each instance

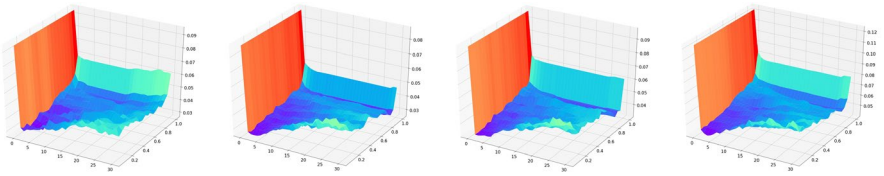
	Node	Edge	Source	Intermediate	Sink
Small scales	20	100	1.3	16.9	1.8
	60	400	5.2	49.5	5.3
	120	1500	7.3	104.5	8.2
	150	2500	8.3	132.5	9.1
	200	4000	10.6	178.2	11.2
	250	7500	9.8	230.1	10.0
	300	9000	9.9	279.4	10.7
	500	10000	10.4	479.1	10.5
	1000	20000	10.6	978.1	11.4
Large scales	6000	250000	91.7	5816.7	91.6
	7500	500000	101.4	7296.7	101.9
	9000	750000	137.1	8726.1	136.8
	10000	1000000	153.6	9693.6	152.8

The values of Source, Intermediate, and Sink are the average number of corresponding nodes in each scale, where the supply of source is over zero, that of the intermediate node is zero, and that of sink is below zero

Table 2 The distributions of parameters on the graph

Item	Name	Illustration
1	Supply[a]	$b_i \sim \pm\mathrm{U}(3, 30)$
2	Capacity	$u_e \sim \mathrm{U}(3, 30)$
3	Unit cost	$c_e \sim \mathrm{U}(0, 10) + 1$
4	Fixed cost	$s_e \sim \gamma \cdot \mathrm{U}(0, 10) + 1$
5	Combinatoriality coefficient	$\gamma \in \{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$

[a]The supply of node i obeys distribution $\mathrm{U}(3, 30)$ when i is a source, and obeys distribution $\mathrm{U}(-30, -3)$ when i is a sink



(a) Scale: (200, 4000) (b) Scale: (250, 7500) (c) Scale: (300, 9000) (d) Scale: (500, 10000)

Fig. 2 Relative error between ADSSP and the Cplex MILP Solver under gridded parameters

error, is mainly concentrated in $\{(M_d, \alpha) : 0 < M_d \leqslant 5; 0 < \alpha \leqslant 0.5\}$, which is verified across these four scales. When M_d is set to one, the relative error between the objective of ADSSP and the optimal value is more than 10%, which causes the red, steep slope in Fig. 2. However, as M_d increases, the relative error also increases

Table 3 Performance of ADSSP under four deleting rules for small-scale instances

Scale	Average solving time (ms)				Average relative error (%)			
	TYPE 1	TYPE 2	TYPE 3	TYPE 4	TYPE 1	TYPE 2	TYPE 3	TYPE 4
(20, 100)	4.80	4.83	**4.27**	4.50	0.55	0.45	0.52	**0.44**
(60, 400)	**22.77**	23.10	24.17	24.3	2.17	**1.71**	2.17	1.81
(120, 1500)	**95.83**	99.03	101.3	101.2	2.38	1.83	2.21	**1.50**
(150, 2500)	**155.2**	157.5	181.0	175.6	2.55	**1.70**	2.48	1.91
(200, 4000)	276.3	**265.7**	296.2	292.8	2.23	**1.47**	1.88	1.53
(250, 7500)	491.5	**489.3**	505.0	510.5	1.94	1.58	1.80	**1.47**
(300, 9000)	**600.9**	605.1	625.7	604.7	2.01	1.22	1.84	**1.19**
(500, 10000)	773.5	**754.0**	802.8	779.7	2.90	**1.94**	2.84	2.05
(1000, 20000)	1519	**1453**	1619	1622	2.54	**1.99**	2.47	2.05
Mean					2.13	**1.54**	2.02	1.55
Std.					0.64	0.45	0.63	0.48

Bolded the minimal average solving time and minimal average relative error in each line to emphasize the advantages of ADSSP

relatively. Therefore, a few steps of dynamic slope scaling in ADSSP can improve the correctness of edge deletion. Also, the smaller value of α can reduce the impact of incorrect judgments in edge deletion on the solution.

The edge deleting rules for ADSSP cannot be directly specified. We compare them through necessary experiments on small-scale instances. Regarding the solution obtained by Cplex as a benchmark, we calculate the relative error between it and the solutions of ADSSP with different deleting rules. Table 3 shows the average solving time and relative error of ADSSP with different deleting rules on multi-scale instances. The relative error is calculated as

$$\text{Relative Error } (\%) = \frac{C_A - C_I}{C_I} \times 100\% \tag{6}$$

where C_A is the objective of FCNFP solved by ADSSP and C_I is the objective of MILP solved by the Cplex MILP Solver. Different edge-deleting rules do not significantly affect the efficiency of problem-solving. However, TYPE 2 and 4 help the algorithm find better solutions, which indicates that fixed costs have a major impact on the effectiveness of edge deletion.

In the following experiments, we set the number of iterations in DSSP as three and the deleting ratio as 0.5. The performance of ADSSP under four deleting rules shows that TYPE 2 and TYPE 4 are good choices for edge deletion. Considering that the TYPE 2 rule can not handle the case when the unit cost is much larger than the fixed cost, we apply the TYPE 4 rule on ADSSP to delete edges.

4.3 Simulation on instances with different scale

This section discusses the simulation results on problems with different scales. In small-scale problems, we calculate the relative error between the solutions of

Table 4 Performance of Cplex on small-scale instances

Scale	Solving time (s)	Success	Optimality gap (%)
(20, 100)	0.012 (0.01)	30/30	0.00 (0.00)
(60, 400)	0.331 (0.75)	30/30	0.00 (0.00)
(120, 1500)	426.362 (1073.15)	27/30	0.04 (0.00)
(150, 2500)	1079.326 (1484.90)	23/30	0.15 (0.00)
(200, 4000)	2302.578 (1707.13)	11/30	0.31 (0.00)
(250, 7500)	2441.885 (1638.82)	10/30	0.41 (0.00)
(300, 9000)	2517.281 (1551.13)	10/30	0.43 (0.00)
(500, 10000)	2626.966 (1430.04)	10/30	0.70 (0.01)
(1000, 20000)	Limit	0/30	1.29 (0.01)

*The column Solving Time shows the average solving time and the standard deviation. The column Success shows the counting number of instances that Cplex has found its optimum on 30 instances. The column Optimality Gap shows the average optimality gap and its standard deviation

ADSSP and Cplex solver and compare it with the previous LP-based iterative algorithms DSSP, ADCUP, and DPUP. In large-scale problems, we compare the performance of DSSP, ADSSP, and Cplex to explore the impact of problem combinatoriality on ADSSP.

4.3.1 Small-scale instances

Small-scale instances are divided into nine groups with the topological information listed in Table 1, and each group contains 30 instances. Table 4 indicates that Cplex can find the optimal solution or a suboptimal solution with low optimality gaps[2] for the small-scale instance. By regarding the solution of Cplex as the benchmark, we compare the relative errors of solutions solved by DSSP, ADCUP, DPUP, and ADSSP. Table 5 shows the results of the algorithms on small-scale problems, with the minimum value of indicators highlighted in bold. The first column states the problem scale of this group. The second to fifth columns list the average solving times for the four algorithms. The sixth to ninth columns list the average relative error between the benchmark and the solutions of these four algorithms.

The results indicate that ADSSP far exceeds other LP-based iterative algorithms in performance and stability. For small-scale problems, the average solving time of ADSSP can be less than 1.7 s, and the average relative error of the objective can be controlled within 2%. Although the solving time of ADSSP is slightly longer than DSSP in the instances where the number of edges is less than 1000, ADSSP finds a better solution through the edge deletion strategy. However, judging from optimality alone, the solution of ADSSP cannot reach the optimal solution. ADSSP can find a

[2] The introduction and calculation of Optimality Gap was shown on IBM website that https://www.ibm.com/docs/en/icos/12.9.0?topic=mip-progress-reports-interpreting-node-log.

Table 5 Performance of DSSP, ADCUP, DPUP, and ADSSP on small-scale instances

Scale	Average solving time[a] (seconds)				Average relative error (%)			
	DSSP	ADCUP	DPUP	ADSSP	DSSP	ADCUP	DPUP	ADSSP
(20, 100)	**0.002**	0.051	0.004	0.005	1.19	5.50	3.38	**0.44**
(60, 400)	**0.016**	0.294	0.028	0.024	2.54	10.22	5.68	**1.81**
(120, 1500)	0.159	2.118	0.190	**0.101**	4.03	12.52	5.89	**1.50**
(150, 2500)	0.333	4.691	0.422	**0.176**	3.50	13.89	6.65	**1.91**
(200, 4000)	0.643	11.13	1.048	**0.293**	3.47	14.16	7.22	**1.53**
(250, 7500)	1.623	27.97	3.092	**0.511**	3.95	15.17	7.98	**1.47**
(300, 9000)	2.478	30.91	3.821	**0.605**	3.30	14.41	7.45	**1.19**
(500, 10000)	2.241	67.02	6.956	**0.780**	4.73	19.07	9.03	**2.05**
(1000, 20000)	5.123	287.5	30.88	**1.622**	5.75	21.30	9.55	**2.05**
Mean					3.61	14.03	6.98	**1.55**
Std.					1.21	4.34	1.77	**0.48**

Bolded the minimal average solving time and minimal average relative error in each line to emphasize the advantages of ADSSP

[a]The solving time is reported with four significant digits

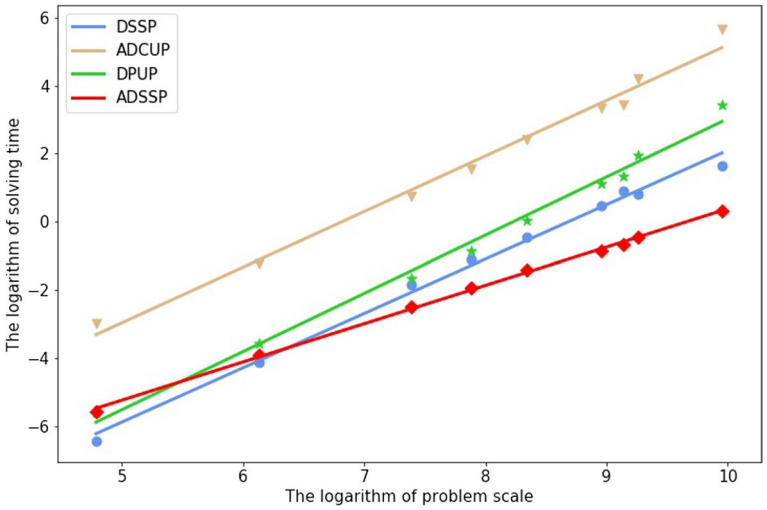


Fig. 3 Scatter plot of solving time and fitting line of problem scale. The solving time of each algorithm is from Table 5. The problem scale is equal to the total number of nodes and edges. To facilitate the display and comparison of algorithms, the solving time and problem scale are logarithmic

solution within 2% relative error on average in less than 1.7 s, while the Cplex MILP Solver takes much longer, even reaching the time limit.

The complexity of FCNFP is related to the number of variables and constraints. Using the total number of nodes and edges as the problem scale, Fig. 3 shows the trend of solving time with the problem scale. The solving time of ADSSP is not only significantly smaller than that of the other three algorithms, but the slope of

Table 6 Optimality gap of Cplex on large-scale instances

γ	Average Optimality Gap (%)			
	(6000, 250000)	(7500, 500000)	(9000, 750000)	(10000, 1000000)
10^{-4}	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
10^{-3}	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
10^{-2}	0.06 (0.00)	0.07 (0.00)	0.05 (0.02)	0.04 (0.02)
0.1	0.51 (0.06)	0.66 (0.13)	0.55 (0.21)	0.57 (0.18)
1	4.33 (0.48)	4.85 (1.25)	3.87 (1.22)	3.20 (0.97)
10	12.21 (0.98)	12.02 (1.35)	11.11 (2.40)	9.05 (1.91)
10^2	19.93 (2.25)	17.20 (1.10)	16.58 (3.92)	14.50 (2.46)
10^3	23.54 (2.50)	24.12[a] (2.79)	21.52 (4.37)	17.79 (3.66)
10^4	24.08 (2.60)	25.03 (2.77)	20.05 (4.08)	19.95 (4.23)

[a]There is an instance in this group that Cplex has not solved out, which is not included in the calculation of the metrics

the fitting line is also noticeably narrower than the others. ADSSP outperforms the currently available LP-based iterative algorithms in terms of both the solution quality and the solving efficiency. The better result with less solving time indicates that ADSSP is more advantageous in solving large-scale problems.

4.3.2 Large-scale instances

We construct four groups of instances on large scales, and the number of nodes and edges are (6000, 250000), (7500, 500000), (9000, 750000), and (10000, 1000000). Each group includes nine classes with different combinatoriality coefficients, and each class contains ten instances. We compare the performance of DSSP, ADSSP, and Cplex in these instances. Table 6 shows the optimality gap of Cplex with varying scales and combinatoriality coefficients, which increases with respect to the combinatoriality coefficient. Table 7 shows the relative error calculated by Eq. (6). Regarding the solution of Cplex as the benchmark, the relative error of DSSP on average is greater than zero, increasing with γ. On the other hand, the relative error of ADSSP on average is less than zero with a decreasing trend as γ increases. Table 8 shows the solving time of DSSP, ADSSP, and Cplex on the instances. The solving time of FCNFP significantly increases with the increasing magnitude of γ. In Table 8, ADSSP shows the slowest growth trend in solving time, with a maximum value of 91.6 s in large-scale instances, which is significantly smaller than Cplex and only one-tenth of DSSP.

ADSSP outperforms the Cplex MILP Solver when the problem scale reaches at least 250,000, finding a better solution faster with a relative error that decreases by 2%, while DSSP performs vastly worse than Cplex. Also, the solving time of ADSSP in large-scale instances is further short than DSSP and Cplex, and it is less affected by the combinatoriality of the problem. The results demonstrate that the edge deletion strategy brings a breakthrough improvement in the effectiveness and

Table 7 Performance of DSSP and ADSSP on large-scale instances

γ	Relative Error to CPLEX on varying scales (%)							
	(6000, 250000)		(7500, 500000)		(9000, 750000)		(10000, 1000000)	
	DSSP	ADSSP	DSSP	ADSSP	DSSP	ADSSP	DSSP	ADSSP
10^{-4}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10^{-3}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10^{-2}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.1	0.07	0.00	0.13	0.02	0.06	−0.03	0.10	0.01
1	3.18	−1.09	6.70	−0.47	4.29	−0.86	2.66	−0.86
10	10.89	−3.07	32.02	−1.38	18.07	−2.95	10.85	−3.14
10^2	32.09	−5.16	51.78	−2.87	42.74	−5.08	32.13	−4.62
10^3	45.45	−4.89	76.69	−4.24	62.88	−6.58	44.46	−5.99
10^4	46.17	−3.99	76.70	−5.57	56.77	−6.39	48.23	−7.27
Min	0.00	−11.11	−0.03	−11.18	0.00	−11.40	0.00	−10.49
Mean	15.32	−2.02	26.56	−1.58	20.53	−2.43	15.38	−2.43
Max	65.73	0.27	103.37	4.89	108.93	0.04	114.89	0.13

efficiency of problem-solving in the large-scale FCNFP. This strategy helps ADSSP quickly find better solutions than the commercial solver in a short time. Therefore, ADSSP is a practical and effective algorithm for solving large-scale FCNFP.

4.4 Simulation on instances with different combinatoriality

The impact of combinatoriality on the solution of ADSSP is evident. Table 7 clearly shows a noticeable difference in relative errors between groups with different γ. Figure 4 illustrates that ADSSP performs similarly to the Cplex MILP Solver for weakly combinatorial problems while finding better solutions than the Cplex MILP Solver for strongly combinatorial problems. The relative error of the objectives decreases as the combinatoriality enhances.

Table 8 does not reveal a clear relationship between the combinatoriality coefficient and the solving time. To shed light on the impact of combinatoriality on the solving process of ADSSP, Table 9 provides the iterative number in Algorithm 2 for different values of γ, with the best-case iterative number as a reference mentioned in Proposition 1. The findings indicate that the combinatoriality coefficient has a effect on the iterative number, as the difference between the average iterative number and the theoretical best case increases with γ. Although this suggests that combinatoriality will lengthen the solving time, the increase is slight. When the unit cost is much higher than the fixed cost, the iterative number for all instances equals the best-case scenario. Even when the fixed cost is significantly greater than the unit cost, the iterative process of ADSSP only requires an additional iteration for solving FCNFP. Overall, the experimental results demonstrate that ADSSP typically solves FCNFP in a best-case or near-best-case iterative process.

Table 8 Solving time of DSSP, ADSSP, and CPLEX on large-scale instances

γ	Solving time on varying scales (s)											
	(6000, 250000)			(7500, 500000)			(9000, 750000)			(10000, 1000000)		
	DSSP	ADSSP	CPLEX	DSSP	ADSSP	CPLEX	DSSP	ADSSP	CPLEX	DSSP	ADSSP	CPLEX
10^{-4}	3.8	8.7	7.4	9.0	20.1	10.9	15.7	30.5	22.8	44.9	63.6	29.3
10^{-3}	6.5	12.2	18.3	16.1	25.9	26.9	42.4	49.7	367.2	110.8	73.7	387.5
10^{-2}	14.5	12.6	Limit[a]	37.1	23.1	Limit	110.8	45.2	Limit	210.6	72.0	Limit
0.1	46.6	11.6	Limit	121.5	21.7	Limit	369.5	43.4	Limit	813.2	54.8	Limit
1	139.5	11.3	Limit	274.8	19.9	Limit	806.5	40.3	Limit	1584.7	64.6	Limit
10	145.8	12.3	Limit	341.8	21.7	Limit	792.7	45.2	Limit	1269.8	69.6	Limit
10^2	152.5	13.7	Limit	401.6	27.3	Limit	772.8	51.6	Limit	1241.9	77.8	Limit
10^3	152.6	15.6	Limit	427.5	29.2	Limit	674.9	52.6	Limit	1145.1	89.3	Limit
10^4	148.6	16.4	Limit	447.4	29.9	Limit	586.2	60.4	Limit	905.2	91.6	Limit
Mean	90.1	12.7	2803.5	230.8	27.6	2805.1	463.5	46.5	2845.1	812.9	73.0	2848.7
Std.	65.8	2.5	–	174.8	5.8	–	317.8	14.7	–	553.2	18.2	–

[a]Limit means that the solving time of all instances in this group reaches 3600 s

Table 9 The average number of iterations for ADSSP

Scale	Average iterative number K									K^{*a}
	$\gamma = 10^{-4}$	$\gamma = 10^{-3}$	$\gamma = 10^{-2}$	$\gamma = 0.1$	$\gamma = 1$	$\gamma = 10$	$\gamma = 10^{2}$	$\gamma = 10^{3}$	$\gamma = 10^{4}$	
(6000, 250000)	19	19	19	19	19	19.2	19.3	19.8	19.9	19
(7500, 500000)	20	20	20	20.1	20	20.2	20.6	20.9	20.9	20
(9000, 750000)	21	21	21	21	21	21	21	21	21.2	21
(10000, 1000000)	21	21	21	21	21.6	21.8	21.9	22	22	21
$\bar{\Delta}_K^{b}$	0	0	0	0.025	0.15	0.3	0.45	0.675	0.75	–

[a] K^* is the best-case iterative number of ADSSP, which satisfies $K = \lceil -log_{1-\alpha}(m-1) \rceil + 1$

[b] $\bar{\Delta}_K$ is the mean of $K - K^*$ for all scales

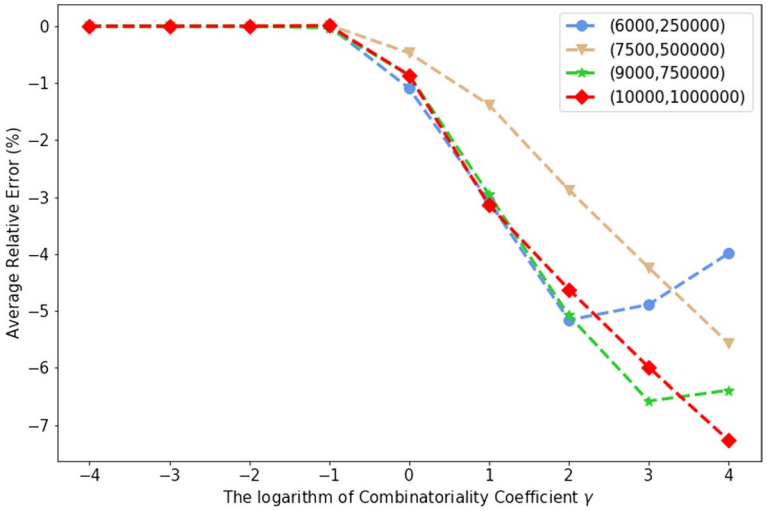


Fig. 4 The curve of relative error between objectives of ADSSP and Cplex MILP Solver on large-scale problems

5 Conclusion

The FCNFP has a wide range of applications in real-world scenarios. Although FCNFP can be transformed into an integer programming problem and solved by commercial solvers, finding the exact solution in a finite time or a sufficiently good suboptimal solution in a short time is challenging, especially for problems with large scales or strong combinatoriality. This paper proposes an LP-based iterative algorithm called ADSSP to reduce the scale of the subproblem in iterations by deleting edges to converge quickly to a suboptimal solution. ADSSP outperforms the commercial solver Cplex on large-scale problems and achieves a controllable relative error with the optimal solution on small-scale problems. Furthermore, this paper analyzes the convergence and time complexity of ADSSP from theoretical and experimental perspectives. Finally, the hyperparameter selection experiments guide the optimal selection interval for the parameters, allowing the algorithm to be better employed.

Acknowledgments The work is supported by Anhui Center for Applied Mathematics, the NSF of China (No. 92270205), and the Major Project of Science & Technology of Anhui Province (Nos. 202203a05020050, 202103a07020011).

Data availibility The datasets generated and analyzed during the current study are available in our repository https://github.com/lufizyang/Data-and-Method-for-FCNFP. And the C++ code of ADSSP is also in there.

Declarations

Conflict of interest The authors declared that there is no confict of interest.

References

1. Sadeghi-Moghaddam, S., Hajiaghaei-Keshteli, M., Mahmoodjanloo, M.: New approaches in metaheuristics to solve the fixed charge transportation problem in a fuzzy environment. Neural Comput. Appl. **31**(1), 477–497 (2019). https://doi.org/10.1007/s00521-017-3027-3
2. Hajiaghaei-Keshteli, M., Aminnayeri, M., Ghomi, S.F.: Integrated scheduling of production and rail transportation. Comput. Ind. Eng. **74**, 240–256 (2014). https://doi.org/10.1016/j.cie.2014.05.026
3. Molla-Alizadeh-Zavardehi, S., Nezhad, S.S., Tavakkoli-Moghaddam, R., Yazdani, M.: Solving a fuzzy fixed charge solid transportation problem by metaheuristics. Math. Comput. Model. **57**(5–6), 1543–1558 (2013). https://doi.org/10.1016/j.endm.2017.03.019
4. Golmohamadi, S., Tavakkoli-Moghaddam, R., Hajiaghaei-Keshteli, M.: Solving a fuzzy fixed charge solid transportation problem using batch transferring by new approaches in meta-heuristic. Electron. Notes Discret. Math. **58**, 143–150 (2017). https://doi.org/10.1016/j.endm.2017.03.019
5. Moccia, L., Cordeau, J.-F., Laporte, G., Ropke, S., Valentini, M.P.: Modeling and solving a multimodal transportation problem with flexible-time and scheduled services. Networks **57**(1), 53–68 (2011). https://doi.org/10.1002/net.20383
6. Willet, J., Wetser, K., Dykstra, J.E., Bianchi, A.B., Essink, G.H.O., Rijnaarts, H.H.: Waterroute: a model for cost optimization of industrial water supply networks when using water resources with varying salinity. Water Res. **202**, 117390 (2021). https://doi.org/10.1016/j.watres.2021.117390
7. Hirsch, W.M., Dantzig, G.B.: The fixed charge problem. Naval Res. Logist. Q. **15**(3), 413–424 (1968). https://doi.org/10.1002/nav.3800150306
8. Jünger, M., Reinelt, G., Thienel, S.: Cutting plane algorithms. Comb. Optim. Papers DIMACS Special Year **20**, 111 (1995)
9. Costa, A.M.: A survey on benders decomposition applied to fixed-charge network design problems. Comput. Oper. Res. **32**(6), 1429–1450 (2005). https://doi.org/10.1016/j.cor.2003.11.012
10. Zhao, Y., Larsson, T., Rönnberg, E., Pardalos, P.M.: The fixed charge transportation problem: a strong formulation based on lagrangian decomposition and column generation. J. Glob. Optim. **72**(3), 517–538 (2018). https://doi.org/10.1007/s10898-018-0661-y
11. Sáez Aguado, J.: Fixed charge transportation problems: a new heuristic approach based on lagrangean relaxation and the solving of core problems. Ann. Oper. Res. **172**(1), 45–69 (2009). https://doi.org/10.1007/s10479-008-0483-2
12. Vanderbeck, F., Savelsbergh, M.W.: A generic view of Dantzig–Wolfe decomposition in mixed integer programming. Oper. Res. Lett. **34**(3), 296–306 (2006). https://doi.org/10.1016/j.orl.2005.05.009
13. Kim, D., Pardalos, P.M.: A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. Oper. Res. Lett. **24**(4), 195–203 (1999). https://doi.org/10.1016/S0167-6377(99)00004-8
14. Nahapetyan, A., Pardalos, P.M.: A bilinear relaxation based algorithm for concave piecewise linear network flow problems. J. Ind. Manag. Optim. **3**(1), 71 (2007). https://doi.org/10.3934/jimo.2007.3.71
15. Rebennack, S., Nahapetyan, A., Pardalos, P.M.: Bilinear modeling solution approach for fixed charge network flow problems. Optim. Lett. **3**(3), 347–355 (2009). https://doi.org/10.1007/s11590-009-0114-0
16. Adlakha, V., Kowalski, K.: A simple heuristic for solving small fixed-charge transportation problems. Omega **31**(3), 205–211 (2003). https://doi.org/10.1016/S0305-0483(03)00025-2
17. Kim, D., Pan, X., Pardalos, P.M.: An enhanced dynamic slope scaling procedure with tabu scheme for fixed charge network flow problems. Comput. Econ. **27**(2), 273–293 (2006). https://doi.org/10.1007/s10614-006-9028-4
18. Nahapetyan, A., Pardalos, P.: Adaptive dynamic cost updating procedure for solving fixed charge network flow problems. Comput. Optim. Appl. **39**(1), 37–50 (2008). https://doi.org/10.1007/s10589-007-9060-x
19. Nie, Z., Wang, S.: A dynamic method to solve the fixed charge network flow problem. IFAC-PapersOnLine **53**(2), 11231–11236 (2020). https://doi.org/10.1016/j.ifacol.2020.12.344
20. Goldberg, A.V., Tarjan, R.E.: Finding minimum-cost circulations by successive approximation. Math. Oper. Res. **15**(3), 430–466 (1990). https://doi.org/10.1287/moor.15.3.430