Research Article

Journal of Algorithms & Computational Technology

A hybrid algorithm for large-scale non-separable nonlinear multicommodity flow problems

Journal of Algorithms & Computational Technology
Volume 17: 1–25

DOI: 10.1177/17483026231157214
journals.sagepub.com/home/act
SAGE

Trung Hieu Tran[1], ThuBa T Nguyen[2] and Yirui Jiang[1]

Abstract

We propose an approach for large-scale non-separable nonlinear multicommodity flow problems by solving a sequence of subproblems which can be addressed by commercial solvers. Using a combination of solution methods such as modified gradient projection, shortest path algorithm and golden section search, the approach can handle general problem instances, including those with (i) non-separable cost, (ii) objective function not available analytically as polynomial but are evaluated using black-boxes, and (iii) additional side constraints not of network flow types. Implemented as a toolbox in commercial solvers, it allows researchers and practitioners, currently conversant with linear instances, to easily manage large-scale convex instances as well. In this article, we compared the proposed algorithm with alternative approaches in the literature, covering both theory and large test cases. New test cases with non-separable convex costs and non-network flow side constraints are also presented and evaluated. The toolbox is available free for academic use upon request.

Keywords

Large-scale optimization, hybrid algorithm, multicommodity flows, nonlinear cost, non-separable cost

Received 18 September 2022; Revised received 9 December 2022; accepted 30 January 2023

Introduction

The importance of network flow problems has been demonstrated in many successful real-world applications.[1] Underlying network structures are also found in a great variety of large-scale optimization problems ranging from transportation, power grids, communications, supply chains to social networks.

While earlier works in network flow problems focus on linear instances,[2] it has been recognized that real-world situations can easily lead to nonlinearities due to physical phenomena and economic considerations.[3,4] Indeed, as more applications look more complicated planning problems such as those dealing with uncertainties, disruption, and sabotage, convex instances arise. For example, separable convex cost functions are considered by Dembo and Klincewicz,[5] Bertsekas et al.,[6] Babonneau and Vial,[7] and Bertsekas et al.[8] However, it has also been reported that separable costs are not realistic representations of real traffic networks,[9] and possibly non-separable, non-convex functions.[3] Moreover, separable convex network flow problems are not much harder than linear optimization problems[10] while general non-separable optimization problems have been shown to be considerably more difficult than separable problems.[11]

Although such research works of non-separable costs are not much in the literature, their applications are often encountered in many transportation networks. For example, considering a two-way road[12] travel cost depends not only on traffic volume traveling in its direction but also on traffic volume in the opposite direction. Other example in urban transportation networks, travel cost on a link depends on congestion occurring at intersections since traffic volumes increase at the intersecting links. Akcelik[13] introduced non-separable travel cost functions to estimate the capacity of a shared lane at a signalized intersection. Larsson et al.[14] refined conventional non-separable and typically asymmetric strategy on a traffic equilibrium assignment model. Agustin et al.[15] proposed an air traffic flow management model for flight cancelation and rerouting with separable and non-separable ground

[1]Centre for Design Engineering, Cranfield University, Cranfield, UK
[2]Through-life Engineering Services Institute, Cranfield University, Cranfield, UK

Corresponding author:
Trung Hieu Tran, Centre for Design Engineering, Cranfield University, Cranfield MK43 0AL, UK.
Email: T.H.Tran@cranfield.ac.uk

holding and air delay costs. Gabriel and Bernstein[16] introduced some non-separable nonlinear route costs in (i) nonlinear valuation of travel time: the small amount of time has relatively low value whereas the large amount of time varies valuable, (ii) non-separable tolls and fares: toll roads and transit systems have non-separable toll/fare structure, and (iii) emission fares: emissions of hydrocarbons and carbon monoxide are a nonlinear function of travel times. Hence, non-separable convex cost networks as well as their solution methods are very important and need to be investigated carefully. Recently, Meselhi et al.[17] has investigated large-scale optimization problems with the objective function of non-separable costs, and proposed three different strategies for handling overlapping variables to reduce overlapping among sub-problems. However, this model did not consider multicommodity and a set of constraints in the optimization problems. Hence, their approach could not handle general problem instances and not be easily applied for industrial applications. Consequently, there is little insight into the behavior of network flow optimal solutions in the presence of such non-separable costs.

In addition, the open-source codes of the existing solution methods for network flow problems with separable/non-separable convex cost are not readily available, or specialized codes[2] are restricted to a small group of researchers. Those involved in large-scale network problems do not have resources for algorithmic developments. We plan to fill the gap. In fact, in theory, these problems can be solved by general nonlinear programing techniques. Nevertheless, optimization methods based on exploring their special structure make it much more attractive and especially get advantages of computation time. We may achieve efficiency by solving sub-problems for which polynomial algorithms, such as shortest path.[2]

In this article, we introduce an efficient hybrid algorithm (namely, CMNET) for solving large-scale non-separable convex network flow problems. This method is a combination of gradient projection, shortest path algorithm, and golden section search with the advantages of convergence property and efficient computation time. In addition to our main contribution of solving such non-separable convex network flow problems, our approach can handle the problems with objective functions that are not available analytically as polynomial but can be evaluated using black-boxes. Moreover, it can solve the network flow problems including additional side constraints (i.e. non-network flow types).

The remainder of this article is organized as follows. A detailed description of network flow problems with generalized non-separable convex costs is provided in section 'Multicommodity flows with non-separable nonlinear costs'. In this section, we also introduce the generalized formulation of the non-separable network flow problems. Section 'CMNET' presents the two-phase projected gradient algorithm proposed, as well as its convergence. Section 'Computational results' describes the computational experiments to illustrate the efficiency of the proposed algorithm. The experiments include a comparison with analytic center cutting plane method (ACCPM),[7] an implementation of a constrained ACCPM to solve nonlinear multicommodity flow problems, on separable instances, and the performance evaluation of our algorithm on non-separable instances are presented. Finally, conclusions and future work are summarized in section 'Conclusions and future work'.

Multicommodity flows with non-separable nonlinear costs

Definition of the problem

Assumption. We only investigate non-separable multicommodity flow problems with continuously differentiable convex objective function.

General formulation. Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a directed graph with the number of nodes $n := |\mathcal{N}|$ and the number of arcs $a := |\mathcal{A}|$. Then, we denote $\mathbf{H} \in \mathrm{R}^{n \times a}$ to be the node-arc incidence matrix of $\mathcal{G}$. Let K be the number of commodities to be transported through the network. For each commodity k, there is an amount of demand d^k required to deliver from source node s_k to destination node t_k. Incorporating multicommodity into the network flow problem increases significantly the size of problem due to the additional number of decision variables and constraints. The general non-separable nonlinear multicommodity flow problem (NNMFP) can be formulated by

[**NNMFP**] :

$\min_x$	$g(x)$
s.t.	$\mathbf{H}x^k = d^k\delta^k, \forall k \in \mathcal{K}$,
	$\sum_{k\in\mathcal{K}} x_{ij}^k \le c_{ij}, \forall (i,j) \in \mathcal{A}$,
	$x_{ij}^k \ge 0, \forall k \in \mathcal{K}, (i,j) \in \mathcal{A}$,

where

$g(x)$	is a continuously differentiable non-separable convex objective function,
x	is a decision vector with the components x_{ij}^k, $\forall k \in \mathcal{K}, (i,j) \in \mathcal{A}$,
x^k	denotes the a-dimension flow vector of commodity k,
c_{ij}	is the flow capacity on arc (i,j),
δ^k	is the n-dimension vector with the components $\delta_j^k, \forall j \in \mathcal{N}$ determined by

$$\delta_j^k = \begin{cases} 1 & \text{if} \quad s_k = j \\ -1 & \text{if} \quad t_k = j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Congestion and propagation functions. Convex objective functions investigated in this article are often encountered in many real-life applications. For example, congestion functions pose generalized convex cost in network flow problems. Various types of such congestion functions can be found:

- Traffic assignment manual[18]
- Link capacity functions: A review[19]
- Improved speed-flow relationships: Application to transportation planning models.[20]

However, propagation functions which pose non-separable cost in network flow problems have not considered appropriately. In the literature, there are a few works relevant to such functions. They arose mainly from traffic assignment problems. For example, Gabriel and Bernstein[16] introduced some non-separable route costs caused by tolls and fares.

Practical applications

Network flow problems with non-separable nonlinear costs are often encountered in logistics and supply chain management (i.e. multicommodity flows). In addition, such problems also arise in the fields of transportation and telecommunications.

Transportation problems. In general, transportation problems with non-separable cost are different from ordinary network flow problems. Caused by tolls and fares, these transportation problems have to be formulated in the route-flow space. Hence, they cannot be solved using the traditional link-based algorithms, such as the Frank-Wolfe algorithm.[21] Also, the diagonalized methods do not work well on the non-additive problems since the diagonalized subproblems are poor approximations of the original problem. Two types of transportation problems include[9]:

- Traffic equilibrium problem is to find the traffic-flow pattern by allocating the origin–destination demands to the traffic network such that all used routes between each origin–destination pair have equal and minimum travel cost, and no unused route has lower travel cost than minimum travel cost.
- Traffic assignment problem is to allocate a given set of trip/route interchanges to a certain transport network. The solution of the assignment problem includes an estimate of the traffic volumes and the corresponding travel times or costs on each arc of the transport network.

Telecommunications. Such network flow problems with non-separable nonlinear costs are also encountered in the field of telecommunications, that is, data networks, where we have to transmit data appropriately to satisfy the demands of hubs in the network.[22] In addition, in telecommunication applications there are usually additional time-delay or reliability requirements on paths, which may cause non-separable cost as well as side constraints on paths.[23]

Importance of non-separable nonlinear costs and side constraints

Node congestion/junction effect/delay propagation. Non-separable cost is often encountered in traffic assignment models where two-way traffic is considered.[12] Then, traffic time/cost depends not only on traffic volume traveling in this direction but also on traffic volume in the opposite direction. Furthermore, in urban transportation networks, travel time/cost on a link depends on delays occurring at intersections or traffic volumes on the intersecting links. As a result, travel time/cost on precedent and/or successive links is also affected, which leads to delay propagation in transport networks.

Side constraints. This problem includes flow capacity and conservation (network) constraints and other additional (non-network) constraints, aka side constraints. There is a few studies devoted to side constrained network models. In the traffic assignment problem, researchers mainly investigate the link capacity as the side constraints.[9] In the traffic equilibrium problem, side constraints may be caused by (i) the effects of a traffic control policy, (ii) the refinement strategies, and (iii) the flow restrictions that a central authority wishes to impose upon the users of the network.[14] In telecommunication applications, side constraints causing time-delay or reliability requirements on paths were investigated by Holmberg and Yuan.[23]

CMNET

CMNET is a toolbox of the commercial solvers (e.g. CPLEX), which is constructed on a two-phase gradient projection algorithm. The proposed algorithm is a combination of gradient projection, shortest path formulation and golden section search for solving large-scale non-separable nonlinear multicommodity flow problems. In this section, after describing the details of the two-phase gradient projection algorithm, we provide the optimal properties of this algorithm and its convergence rate. Next, the importance of a commercial solver add-on is discussed.

Two-phase gradient projection algorithm

At each iteration in the proposed algorithm, we solve the sequence of the following subproblems: linearized problem (LP) and second-order conic programing (SOCP). The linearized multicommodity flow problem can be formulated by

[**LP**]:

$\min_x$	$\sum_{(i,j)\in\mathcal{A}}\sum_{k\in\mathcal{K}} g_{ij}(x_{ij}^k)$
s.t.	$\mathbf{H}x^k = d^k\delta^k, \forall k \in \mathcal{K}$
	$x^k \geq 0, \forall k \in \mathcal{K}$

where $g_{ij}(x_{ij}^k)$ is a linear cost objective function. $g_{ij}(x_{ij}^k) = t_{ij}x_{ij}^k$ is only used at the initialization step, otherwise $g_{ij}(x_{ij}^k) = \frac{\partial g(x)}{\partial x_{ij}}$.

Let $\mathbf{X}$ be the feasible solution set of network flow problem considered. SOCP at iteration $h+1$ in CMNET can be formulated by

[**SOCP**]:

$$\min_{x\in\mathbf{X}} \|z_{h+1} - x\|_2^2$$

where z_{h+1} is a scalar vector, out of feasible solution space, determined by $z_{h+1} := x_h - \beta_h \nabla g(x)|_{x=x_h}$.

In addition, we use the golden section search to find a solution improvement at each iteration. We illustrate the basic implementation of this proposed algorithm at iteration $h+1$ in Figure 1.

Descent direction from the shortest path (SP). Dijkstra's algorithm which was firstly introduced by Dutch computer scientist Edsger Dijkstra[24] is a graph-based search algorithm. In the literature, this algorithm has been applied for searching the SPs from a source to a destination on a graph with non-negative arc costs. In the two-phase gradient projection algorithm, instead of solving the LP by commercial solvers we used Dijkstra's algorithm as a subroutine for determining the transport path with the lowest cost between a supply and a customer. For the multicommodity flow problems where each commodity has a pair of supply–demand nodes, we repeatedly use Dijkstra's algorithm to find the corresponding lowest cost paths. To save the computation time, we solve the commodities with the same supply nodes at the same moment. A parallel computing strategy with a number of CPU cores can be applied to improve the efficiency of the procedure.

A modified gradient projection method with β adjustment strategy. We develop the two-phase gradient projection algorithm based on the gradient projection method introduced by Bertsekas.[25] The optimal properties of this hybrid algorithm are thus still maintained as that of the standard gradient projection method. However, to improve the convergent rate we propose an efficient stepsize adjustment strategy, where value β_h at iteration $h+1$ reduces if solution x_{h+1}^{SOCP} obtained by solving the SOCP does not satisfy one of the following conditions:

- Improvement condition:

$$g(x_{h+1}^{SOCP}) < g(x_h) \tag{2}$$

- Local search condition:

$$\nabla g(x)'|_{x=x_{h+1}^{SOCP}}(x_h - x_{h+1}^{SOCP}) < 0 \tag{3}$$

Then, the SOCP is resolved with an updated value β_h. This procedure terminates if condition (2) or (3) is satisfied.

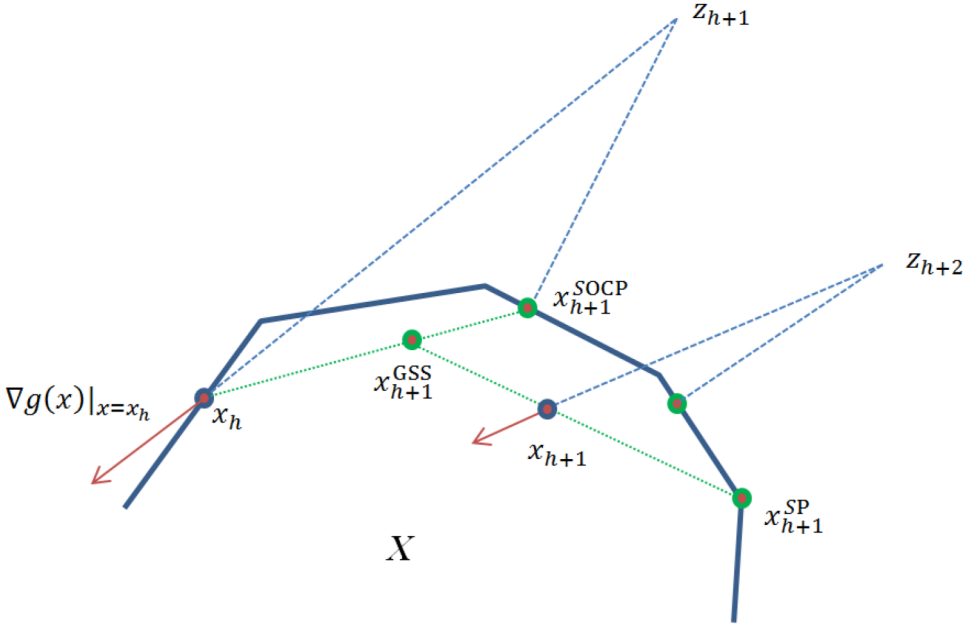

Figure 1. Illustration of one iteration in the two-phase gradient projection algorithm.

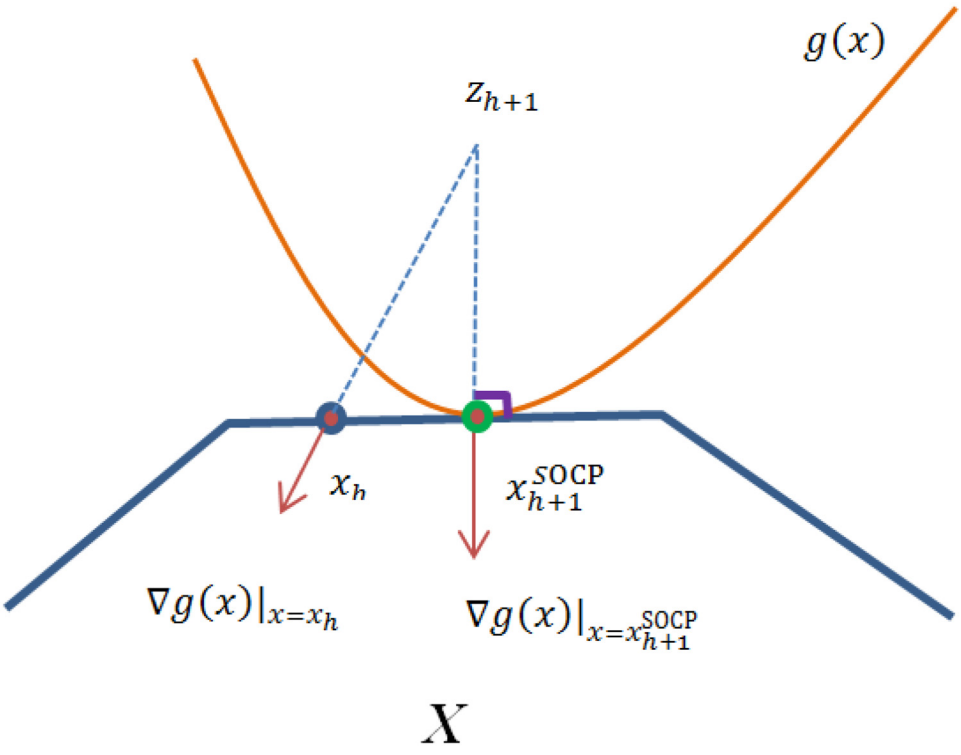

Figure 2. Illustration of termination criterion of the two-phase gradient projection algorithm.

In addition, if value *cosine* made by two vectors $-\nabla g(x)|_{x=x_{h+1}^{SOCP}}$ and $z_{h+1} - x_{h+1}^{SOCP}$ at iteration $h+1$ decreases as compared with that at the previous iteration h, value β is reduced in the next iteration. This may help obtain solutions with larger value *cosine* in the next iterations, and then the convergence rate is improved. In the implementation of the algorithm, β decrement factor is set to be 0.1, while the corresponding factor in decreasing β at the SOCP is 0.5.

Golden section search (GSS). The GSS is a technique for finding the minimal or maximal points of a strictly unimodal function. These points are found by successively limiting the range of values inside that the points are known to exist.[26] In the CMNET, GSS is used to determine the improved solution between the implements of SP, as well as after solving the SOCP where local search condition 3 is satisfied.

Termination criterion. In convex optimization problem, the gradient vector of optimal solution makes an angle 90° with the tangent hyperplane at that solution point. Based on this viewpoint, we propose a new termination criterion for stopping the algorithm. The algorithm stops when value *cosine* made by two vectors $-\nabla g(x)|_{x=x_{h+1}^{SOCP}}$ and $(z_{h+1} - x_{h+1}^{SOCP})$ at iteration $h+1$ is large enough:

$$cosine: = \frac{(-\nabla g(x)|_{x=x_{h+1}^{SOCP}})'(z_{h+1} - x_{h+1}^{SOCP})}{\| -\nabla g(x)|_{x=x_{h+1}^{SOCP}}\|_2 \|z_{h+1} - x_{h+1}^{SOCP}\|_2} \geq (1-\epsilon) \quad (4)$$

where ϵ is a small positive scalar. Figure 2 illustrates the termination.

Pseudo code and flowchart. A pseudo code of the two-phase gradient projection algorithm is shown in Algorithm 1. A flowchart of this algorithm is shown in Figure 3. The algorithm is a hybrid of gradient projection with SP and GSS. Hence, its computational complexity is $O(n^2 + log(1/\varepsilon))$.

Strategies to reduce computation memory and improve CMNET's performance

- For computation time, we used Dijkstra's algorithm to search the lowest cost paths for commodities with the same supply nodes simultaneously.
- WIN64 is used to capture the advantage of unlimited RAM memory.
- Only non-zero values of decision variable during the solution procedure are recorded to save the memory of CPU.
- To save the memory of CPU for solving the SOCP in Step 2 by CPLEX, we solved such subproblem with each commodity k independently:

[**individualSOCP**]:

$\min_{x^k}$	$\|z_{h+1}^k - x^k\|_2^2$
s.t.	$\mathbf{N}x^k = d^k\delta^k$,
	$x^k \geq 0$

- where $z_{h+1}^k := x_h^k - \beta_h \nabla g(x)|_{x=x_h^k}$.
- Similarly, when using the GSS to find the best solution between two vectors x_h and x_{h+1}, we perform the searching process on two corresponding vectors y_h and y_{h+1}, where $\sum_{k\in K} x_h^k = y_h$.

Algorithm 1.

Step 0. **Initialization**:
Set $h := 0$, $\epsilon := 1\times10^{-8}$, and $\beta_0 := 1000$.
Let x_0 be the initial decision vector arbitrarily chosen. In this paper, we used dijkstra's algorithm[24] to solve shortest paths subproblem with linear cost t_{ij} at each arc (i, j) to find x_0.

Step 1. **Shorest path and GSS**:
Solve a sequence of shortest paths subproblems with linear cost $\frac{\partial g(x)}{\partial x_{ij}}|_{x=x_h}$ at each arc (i, j) and apply GSS to find an improved solution between two implementations of solving the shortest path subproblems. The procedure is repeated 50 times to find x_{h+1}, and let $\beta_{h+1} := \beta_h$, $h := h+1$.

Step 2. **Modified gradient projection**:
Solve the **SOCP** with $z_{h+1} := x_h - \beta_h \nabla g(x)|_{x=x_h}$.
If satisfying one of the following conditions:
$g(x_{h+1}^{SOCP}) < g(x_h)$ or $\nabla g(x)'|_{x=x_{h+1}^{SOCP}}(x_h - x_{h+1}^{SOCP}) < 0$,
Then go to Step\, 3,
Else reduce β a half, and go back Step 2.

Step 3. **Check termination condition**:
If satisfying the termination criterion (4),
Then stop and conclude that $x_{h+1}^{SOCP} := \text{argmin}_{x\in\mathbf{X}} g(x)$.
If $\nabla g(x)'|_{x=x_{h+1}^{SOCP}}(x_h - x_{h+1}^{SOCP}) < 0$,
Then implement GSS to find an improved solution x_{h+1}^{GSS} between x_h and x_{h+1}^{SOCP}.
Let $h := h+1$, and go back Step 1.

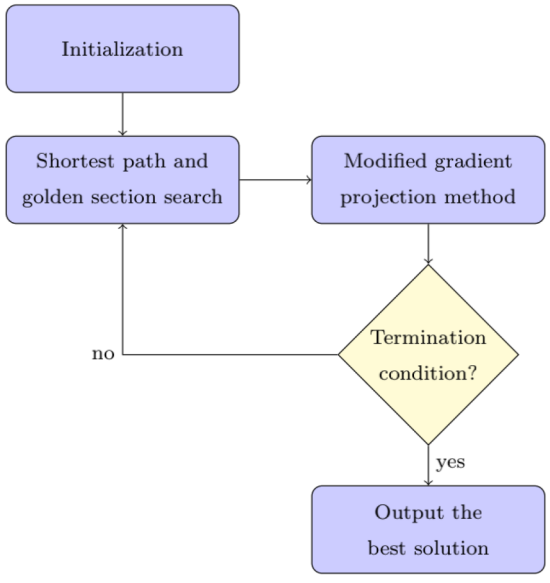

Figure 3. Flowchart of the two-phase gradient projection algorithm.

Optimality properties

Convergence analysis. In section 'Two-phase gradient projection algorithm', if the optimal solution is an extreme point, the CMNET can obtain this solution as satisfying Proposition 1.

Proposition 1: Given that x^* occurs at an extreme point of $\mathbf{X}$, x_h is the global optimal solution of (**NNMFP**) if and only if $x_{h+1} = x_h$.

Proof: If $x_h := \text{argmin}_{x\in\mathbf{X}} g(x)$, then $\nabla g(x)'_{x=x_h}(x - x_h) \geq 0$, $\forall x \in \mathbf{X}$.

We have $x_{h+1} := \text{argmin}_{x\in\mathbf{X}} \nabla g(x)|_{x=x_h}{}' x$.

Because of the unique solution, $x_{h+1} = x_h$.

To obtain a contradiction, assume that
$\exists x \in \mathbf{X}, x \neq x_h : \nabla g(x)|_{x=x_h}{}'(x - x_h) < 0$.

Then, $\nabla g(x)|_{x=x_h}{}' x < \nabla g(x)|_{x=x_h}{}' x_h$.

When (**LP**) is solved, if $x_{h+1} = x_h$, then $\nabla g(x)|_{x=x_h}{}' x < \nabla g(x)|_{x=x_h}{}' x_{h+1}$, which is contradicted since $x_{h+1} := \text{argmin}_{x\in\mathbf{X}} \nabla g(x)|_{x=x_h}{}' x$.

Hence, when $x_{h+1} = x_h$, not $\exists x \in \mathbf{X}, x \neq x_h : \nabla g(x)|_{x=x_h}{}'(x - x_h) < 0$.

In other words, this completed the proof.

In section 'Two-phase gradient projection algorithm', in the case where the optimal solution is on the boundary of feasible solution set, the CMNET can obtain this solution as the termination condition is satisfied according to Proposition 2.

Proposition 2: Given that $z_{h+1} := x_h - \beta_h \nabla g(x)|_{x=x_h}$, where β_h is a positive stepsize corresponding to the β adjustment strategy in the CMNET; and $x_{h+1}^{SOCP} := \text{argmin}_{x\in\mathbf{X}} \|z_{h+1} - x\|_2^2$. If

$$\frac{(-\nabla g(x)|_{x=x_{h+1}^{SOCP}})'(z_{h+1} - x_{h+1}^{SOCP})}{\|-\nabla g(x)|_{x=x_{h+1}^{SOCP}}\|_2 \|z_{h+1} - x_{h+1}^{SOCP}\|_2} \geq (1-\varepsilon)$$

with ε is a sufficient small positive scalar, then $x_{h+1}^{SOCP} := \text{argmin}_{x\in\mathbf{X}} g(x)$.

Proof.: We say that a solution $x^* \in \mathbf{X}$ is the global optimal solution of the convex network problem if and only if it satisfies the first-order condition for optimality

$$\nabla g(x)'|_{x=x^*}(x - x^*) \geq 0, \forall x \in \mathbf{X} \tag{5}$$

In other words, $\nabla g(x)'|_{x=x^*}$ makes an angle 90° with the tangent hyperplane at x^*.

From the definition of x_{h+1}^{SOCP} as the projected solution on $\mathbf{X}$ of z_{h+1}:

$$x_{h+1}^{SOCP} := \text{argmin}_{x\in\mathbf{X}} \|z_{h+1} - x\|_2^2$$

we have

$$\left[z_{h+1} - x_{h+1}^{SOCP}\right]'\left[x - x_{h+1}^{SOCP}\right] \leq 0, \forall x \in \mathbf{X}. \tag{6}$$

Here, $(z_{h+1} - x_{h+1}^{SOCP})$ is orthogonal to the tangent hyperplane at x_{h+1}^{SOCP}.

From (5) and (6), we can say that

$$x_{h+1}^{SOCP} : = \text{argmin}_{x \in \mathbf{X}} g(x)$$

with the condition of

$$\frac{(-\nabla g(x)|_{x=x_{h+1}^{SOCP}})'(z_{h+1} - x_{h+1}^{SOCP})}{\| -\nabla g(x)|_{x=x_{h+1}^{SOCP}} \|_2 \| z_{h+1} - x_{h+1}^{SOCP} \|_2} \geq (1 - \varepsilon)$$

Therefore, this completed the proof.

Under Proposition 2 and that $\{x_k\}$ is a sequence generated by solving the SOCP and SP where β_k is updated according to the β adjustment strategy, we always obtain

$$f(x_{k+1}) \leq f(x_k), \forall k \tag{7}$$

Then, every limit solution of $\{x_k\}$ is a stationary point.

Convergence to local minima of non-convex multicommodity flow problems. In theory, we may divide a non-convex function into a combination of quasi-convex functions considering decision variables a bounded range. Hence, based on the convergence property of the CMNET for solving convex multicommodity flow problems discussed in the section, the CMNET can find the local minima.

Importance of a commercial solver add-on

Commercial solvers (e.g. CPLEX) have been widely used by researchers and practitioners for solving LP problems as well as convex quadratically constrained problems. The performance of such solvers is proven by many successfully practical applications. However, the solvers may not solve generalized convex optimization problems, which are often encountered in real-life. Hence, based on the efficiency of the solvers to develop a solver add-on is very important. This not only makes an inheritable powerful solver add-on, but also the fundamental to build other add-ons.

Computational results

The major goal of our computational experiments is to evaluate the efficiency of the CMNET for solving test cases with (i) non-separable convex costs and (ii) additional side constraints. Commercial solvers for convex or non-convex optimization can only solve the small-scale instances, while our paper investigates the large-scale instances. Hence, we did not make the comparison with such commercial solvers. To compare our algorithm with ACCPM,[7] an algorithm for separable convex network flow problems, we firstly carry out the experiments on test cases with separable convex costs.

We developed the CMNET in Visual C++, and performed the experiments on a PC (Intel Core i5-2500 CPU 3.30 GHz, RAM 16.0 GB) under Windows 7 operating system.

All instances were solved by the CMNET with parameter settings:

- Initial $\beta_0 = 1000$
- Termination condition $\epsilon = 1{\times}10^{-5}$ or $1{\times}10^{-8}$ (to compare the computational time and the solution quality, respectively). In addition, after 1000 iterations, if the termination condition is not met, the program will stop and report the best solution found.
- SOCP termination = $1{\times}10^{-11}$
- β down factor = 0.5
- β down factor cosine = 0.1
- Number of SP iterations per SOCP = 50
- Number of GSS iterations = 20
- CPX_PARAM_BARSTARTALG = 3: Barrier starting point algorithm (i.e. average of primal estimate, dual 0 (zero)) is used
- CPX_PARAM_BARCOLNZ = 3: Number of nonzero entries that make a column dense
- CPX_PARAM_PREIND = 0: Switch off presolving
- CPX_PARAM_DEPIND = 0: Turn off dependency checking

In addition to using the pure solution quality and computation time to compare the performance of the CMNET and the ACCPM, we use *ratio* and *relativegap* as performance measures. These measures are calculated as follows:

$$ratio = \frac{a}{b} \tag{8}$$

$$relativegap = \frac{a - b}{b} * 100\% \tag{9}$$

where a is the objective value or the computation time obtained by the evaluating algorithm (e.g. CMNET), and b represents the objective value or the computation time obtained by the compared algorithm (e.g. ACCPM).

Test cases with separable convex costs

For separable convex costs, the computational experiments were carried out on the benchmark instances of network flows (i.e. planar and grid instances) taken from Babonneau and Vial[7] with two congestion functions: the BPR congestion function and the Kleinrock delay function. In particular, we have all 10 planar instances, which are generated by Di Yuan to simulate telecommunication problems, and 15 grid instances with such grid structure in which each node has four incoming and four outgoing arcs.

Here, the BPR and Kleinrock congestion functions at arc (i, j) are given by

$$g_{ij}(y_{ij}) = \frac{y_{ij}}{c_{ij} - y_{ij}}, y_{ij} \in [0, c_{ij}) \tag{10}$$

$$g_{ij}(y_{ij}) = t_{ij}y_{ij}\left(1 + \frac{\alpha}{\beta+1}\left(\frac{y_{ij}}{c_{ij}}\right)^{\beta}\right),\ y_{ij} \in \mathbf{R}^{+} \qquad (11)$$

where y_{ij} is a sum of all commodity flows at corresponding arc (i, j); t_{ij} and c_{ij} are *free – flowtraveltime* (i.e. linear cost) and *practicalcapacity* (i.e. capacity), respectively; while the parameter values $\alpha = 0.15$ and $\beta = 4$ are used in the experiments.

To be able to use Dijkstra's algorithm for finding shortest path of the test cases with Kleinrock function (i.e. including capacity constraints), we replaced this function by

$$\bar{g}_{ij}(y_{ij}) = \begin{cases} g_{ij}(y_{ij}) = \frac{y_{ij}}{c_{ij}-y_{ij}} & \text{if} \quad y_{ij} \in [0, \zeta c_{ij}] \\ \phi_{ij}(y_{ij}) = \frac{y_{ij}}{c_{ij}(1-\zeta)^2} - \frac{\zeta^2}{(1-\zeta)^2} & \text{if} \quad y_{ij} \in (\zeta c_{ij}, +\infty) \end{cases} \qquad (12)$$

where ζ is a small positive scalar, and set to be 0.99. In fact, $\phi_{ij}(y_{ij})$ is a linearized function at point $y_{ij} = \zeta c_{ij}, \forall (i, j) \in \mathcal{A}$. In particular, $\phi_{ij}(y_{ij}) = \nabla g_{ij}(y_{ij})'|_{y_{ij}=\zeta c_{ij}}(y_{ij} - \zeta c_{ij}) + g_{ij}(\zeta c_{ij})$. Figure 4 illustrates the modified Kleinrock function.

The results obtained by the CMNET on the planar and grid instances are shown in Appendices A to D. In the appendices, we also compare with the results of the ACCPM presented by Babonneau and Vial.[7] Since the authors did not solve planar150, we only put our solution for this instance in the tables, but do not make a comparison.

From the appendices, we see that the CMNET can solve well all test cases with separable convex costs. Even in some instances, the CMNET can obtain better solutions than those of the ACCPM such as:

- Appendix A: planar30 and planar300.
- Appendix B: planar80.
- Appendix C: grid4, grid5, and grid9.
- Appendix D: grid1, grid3, and grid7.

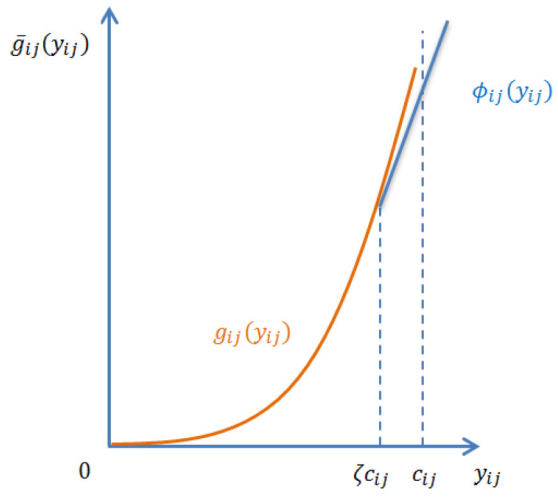

Figure 4. A modified Kleinrock function.[2]

In the rest of these instances, the relative gap is very small. In particular, for termination condition $\epsilon = 1\times10^{-5}$ the maximum value of relative gap is 0.0516% (see Appendix D), while for $\epsilon = 1\times10^{-8}$ this value is only 0.0031% (see Appendix B).

The goal of running experiments with two termination values, $\epsilon = 1\times10^{-5}$ and $\epsilon = 1\times10^{-8}$ is to compare the computation time and the solution quality obtained by the CMNET with those of the ACCPM. The results in the column $\epsilon = 1\times10^{-5}$ from Appendices A to D show that the CMNET can find the near-optimal solutions within a reasonable computation time as compared with the ACCPM. The CMNET's average computation time on all instances with $\epsilon = 1\times10^{-5}$ increases 2.5 times than ACCPM. With $\epsilon = 1\times10^{-5}$, the CMNET can still find a better solution than the ACCPM (i.e. grid 1 in Appendix D). The number of better solutions found by the CMNET increases significantly when the average computation time of the CMNET increases 7.34 times than the ACCPM (see in the column $\epsilon = 1\times10^{-8}$ from Appendices A to D).

Figures 5 to 8 describe intuitively the comparison results between the CMNET and the ACCPM on the separable test cases. From these figures, we can see that the computation time for larger sized instances usually increases linearly or slightly as compared with the ACCPM. The maximum relative gap drops on planar2500 or grid8.

Test cases with non-separable convex costs

According to our best knowledge, there is no benchmark instances of non-separable convex costs in the literature although such problems are often encountered in real-life. Hence, in this section we construct the first test cases with non-separable convex costs, which are based on the planar and grid instances with a modified objective function. The objective function includes the BPR congestion function and a new term of non-separable convex cost (i.e. the junction effect at nodes in the network). We can formulate the impact of junction effect at node j as follows:

$$f_{ij}(y) = \left(\frac{y_{ij}}{\sum_k y_{kj}}\right)\left(\frac{\sum_k y_{kj}}{\theta\sum_k c_{kj} - \sum_k y_{kj}}\right) \qquad (13)$$

where y is a vector of flows at arcs in the network, and θ is a capacity coefficient which is used to determine the congestion level caused by incoming flows/arcs (k, j) to node j. Here, we only consider incoming flows to node j as causes to contribute the congestion at node j. Hence, the impact level of an incoming flow (i, j) to node j is calculated as the product of ratio of flow (i, j) to sum of incoming flows at node j: $(\frac{y_{ij}}{\sum_k y_{kj}})$ and ratio of sum of incoming flows at node j to the capacity of node j: $(\frac{\sum_k y_{kj}}{\theta\sum_k c_{kj} - \sum_k y_{kj}})$.

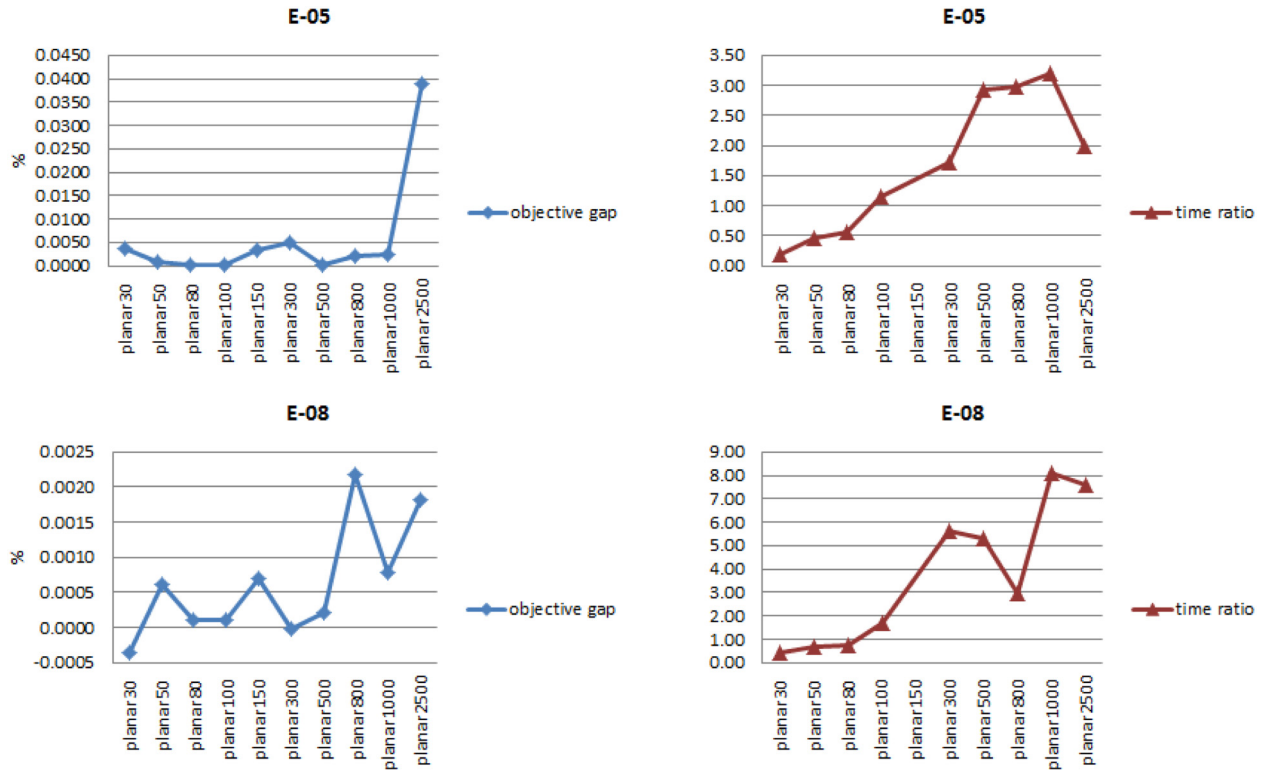

Figure 5. Comparison result of the CMNET and the ACCPM on planar instances (BPR function).

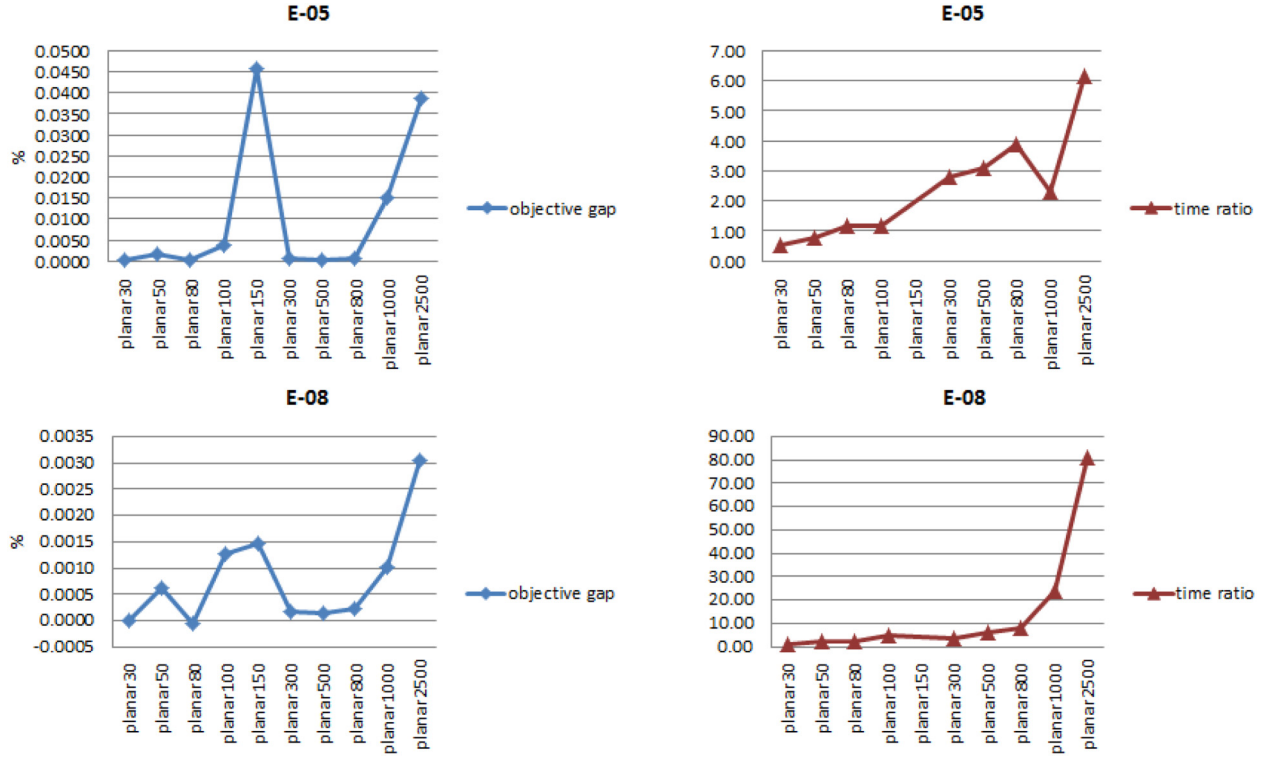

Figure 6. Comparison result of the CMNET and the on planar instances (Kleinrock function).

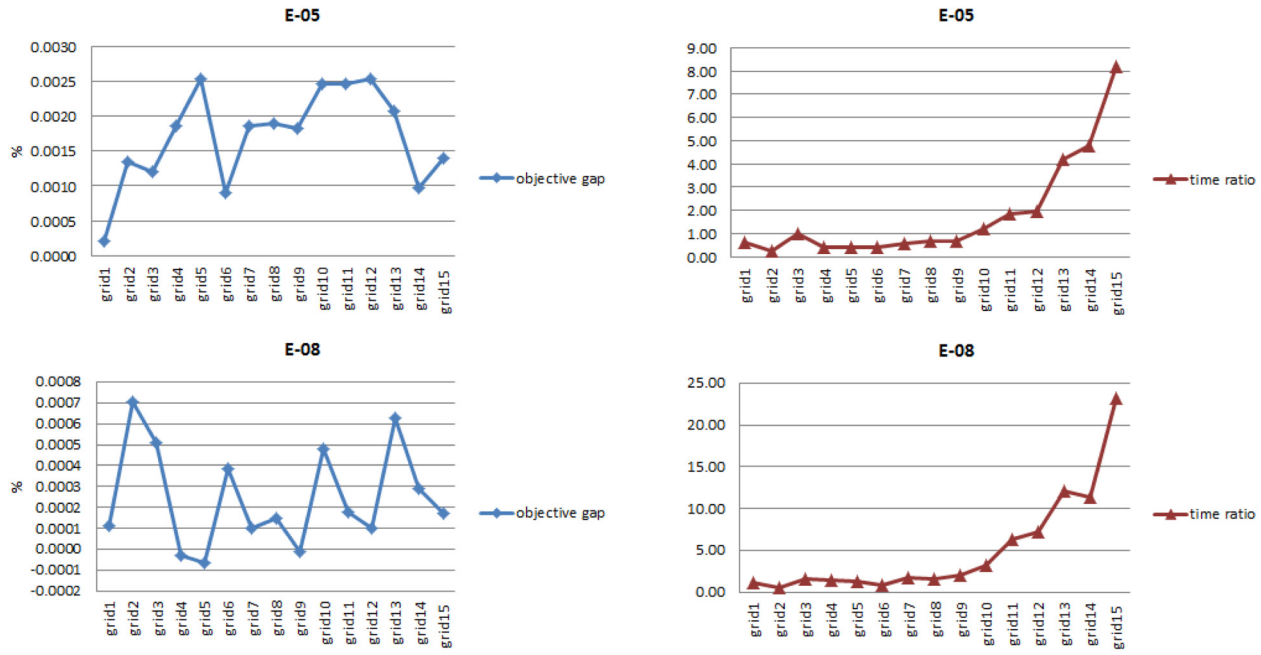

Figure 7. Comparison result of the CMNET and the ACCPM on grid instances (BPR function).

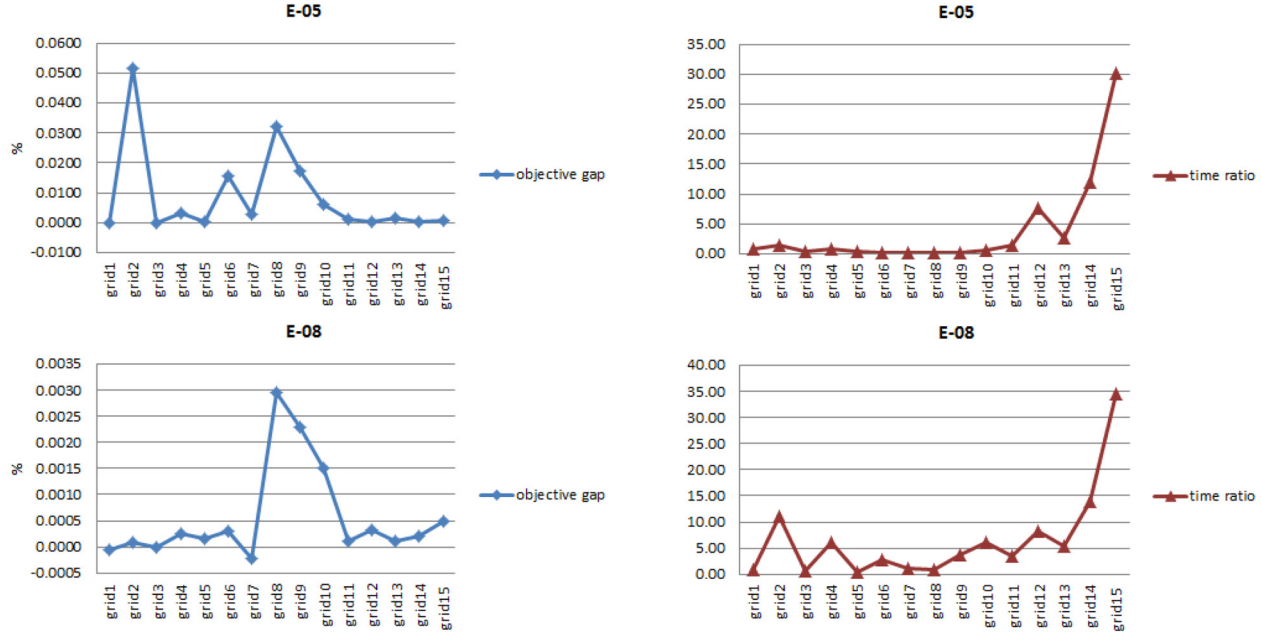

Figure 8. Comparison result of the CMNET and the ACCPM on grid instances (Kleinrock function).

The modified objective function can be rewritten by

$$g_{ij}(y) = t_{ij}y_{ij}\left(1 + \frac{\alpha}{\beta+1}\left(\frac{y_{ij}}{c_{ij}}\right)^{\beta}\right) + \omega_t\left(\frac{y_{ij}}{\theta\sum_k c_{kj} - \sum_k y_{kj}}\right) \quad (14)$$

where ω_t represents the weight of junction effect function in total cost. The function of junction effect used is closely to the Kleinrock delay function. When solving separately the BPR and Kleinrock functions, we see that the objective value of BPR function is about 10 million as large as that of Kleinrock function. Hence, we set $\omega_t = 10$ million when solving the test cases of non-separable convex costs.

We illustrate the impact of non-separable cost term on total cost with planar1000 and grid15. We observe the impact when changing the capacity coefficient θ with values 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 10.0. Here, the smaller θ is, the stronger junction effect is.

Analyze, discuss and conclude. Computational results are shown in Figures 9 to 11 for planar1000, and Figures 12 to 14 for grid15. Figures 9 and 12 demonstrate the impact of junction effect to total cost on planar1000 and grid15, respectively. When decreasing the capacity coefficient θ

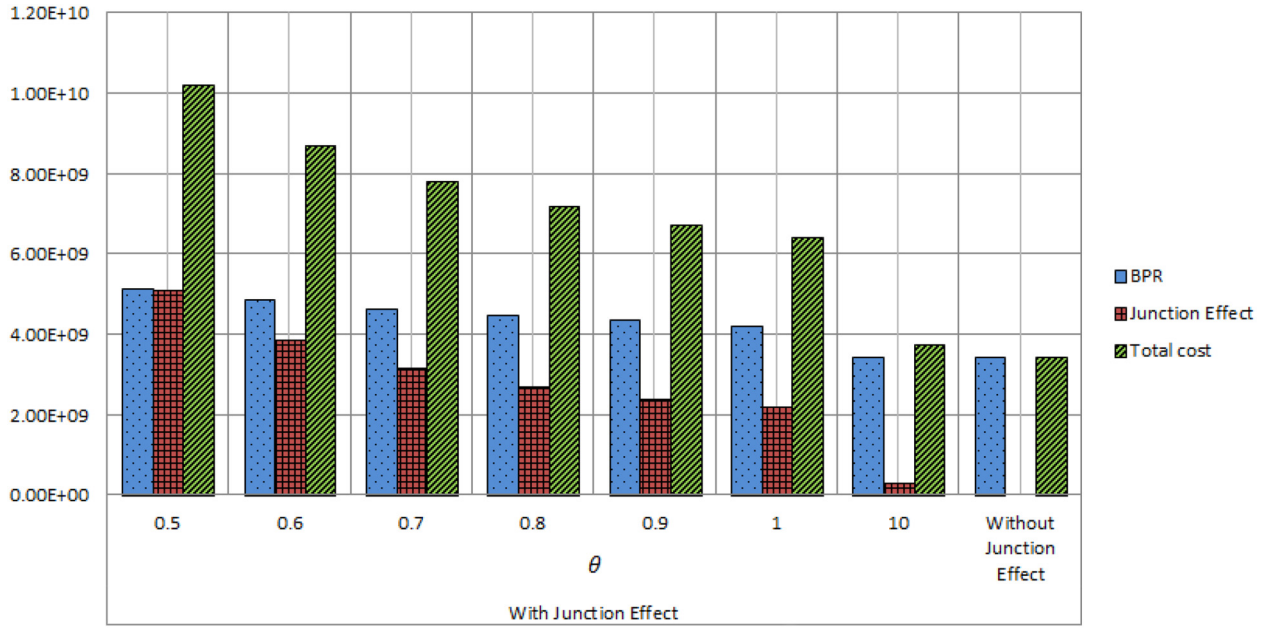

Figure 9. Impact of junction effect to total cost (planar1000).

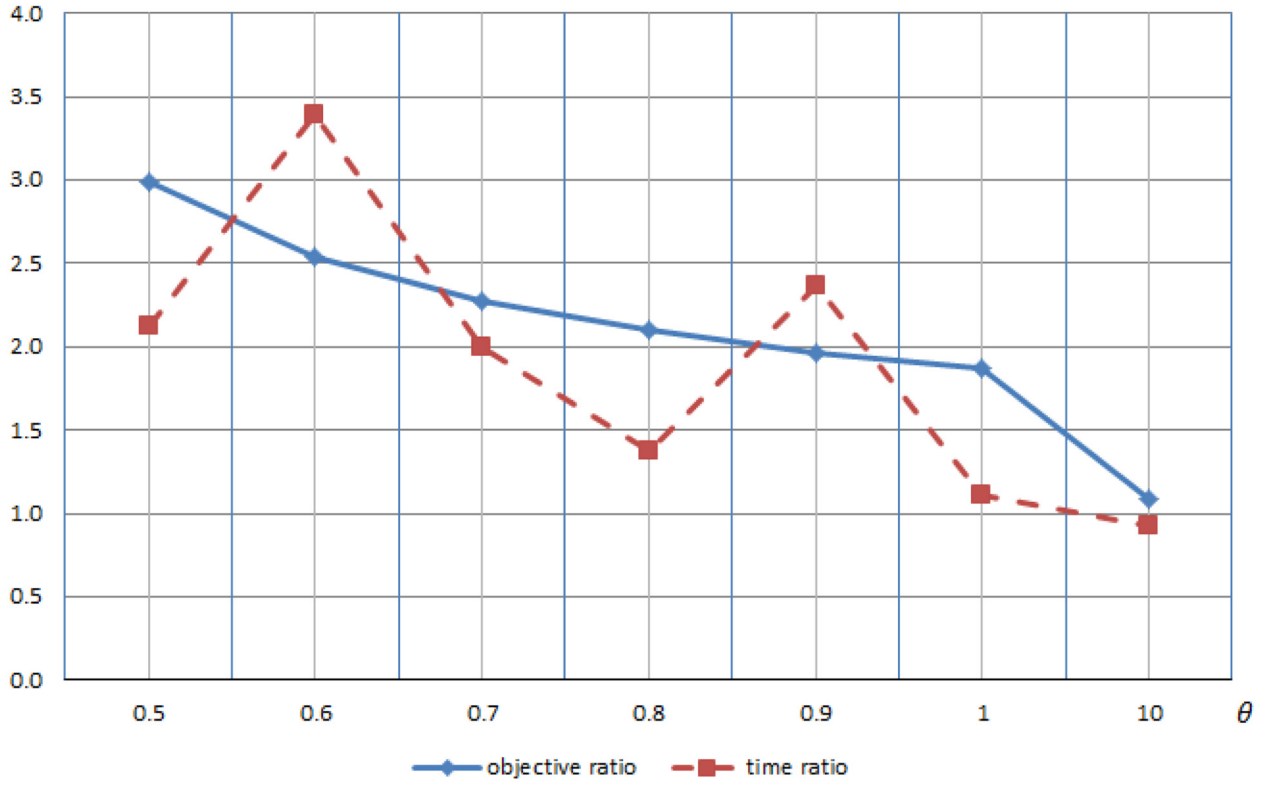

Figure 10. Comparison of objective value and computation time on ratio (planar1000).

at junction nodes (e.g. from 1 to 0.5 with deviation 0.1), the congestion magnitude at the junction nodes increases significantly. This makes total cost raise up. Comparing with the base solution (without junction effect), total cost increases approximately 3 times at the capacity coefficient 0.5 for planar1000, and 4 times for grid15.

While the cost contributed by junction effect increases strongly, the cost of BPR function only increases slightly. This shows clearly when we compare the cost at capacity coefficient 0.5 and 10.

Strong impact of junction effect to total cost also affects to the complexity of non-separable network flow problems,

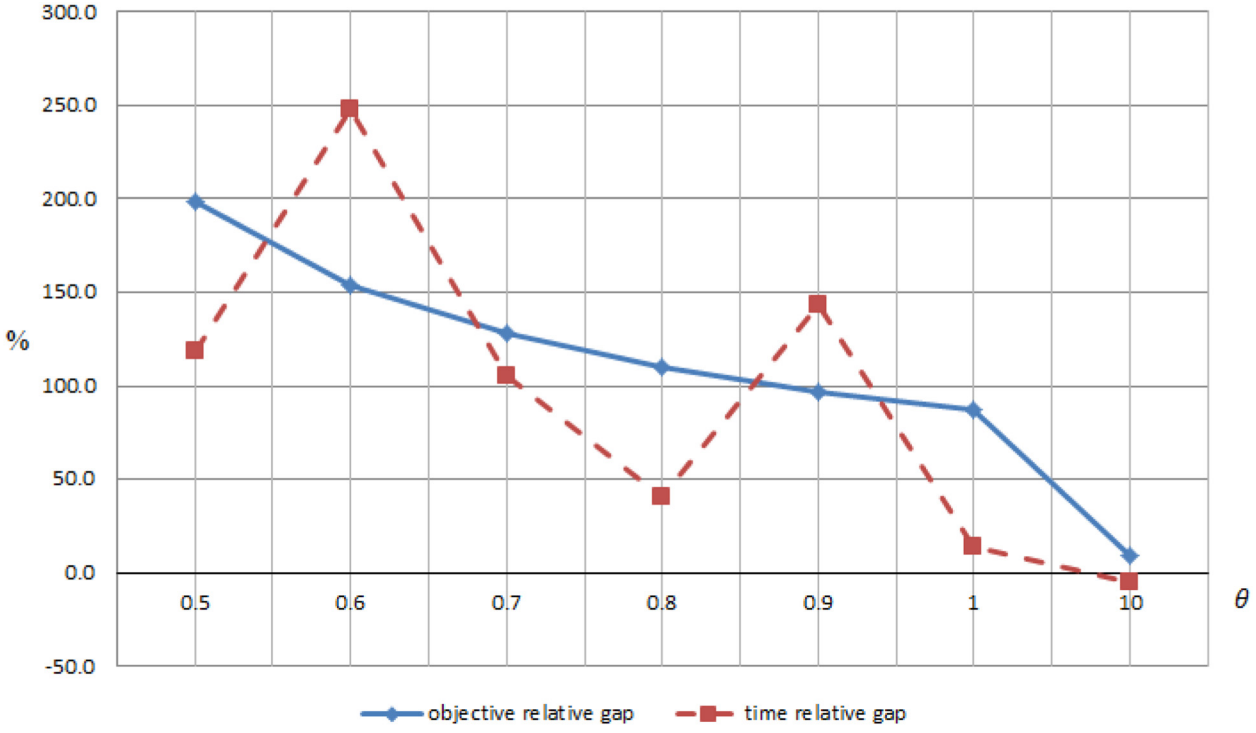

Figure 11. Comparison of objective value and computation time on relative gap (planar1000).

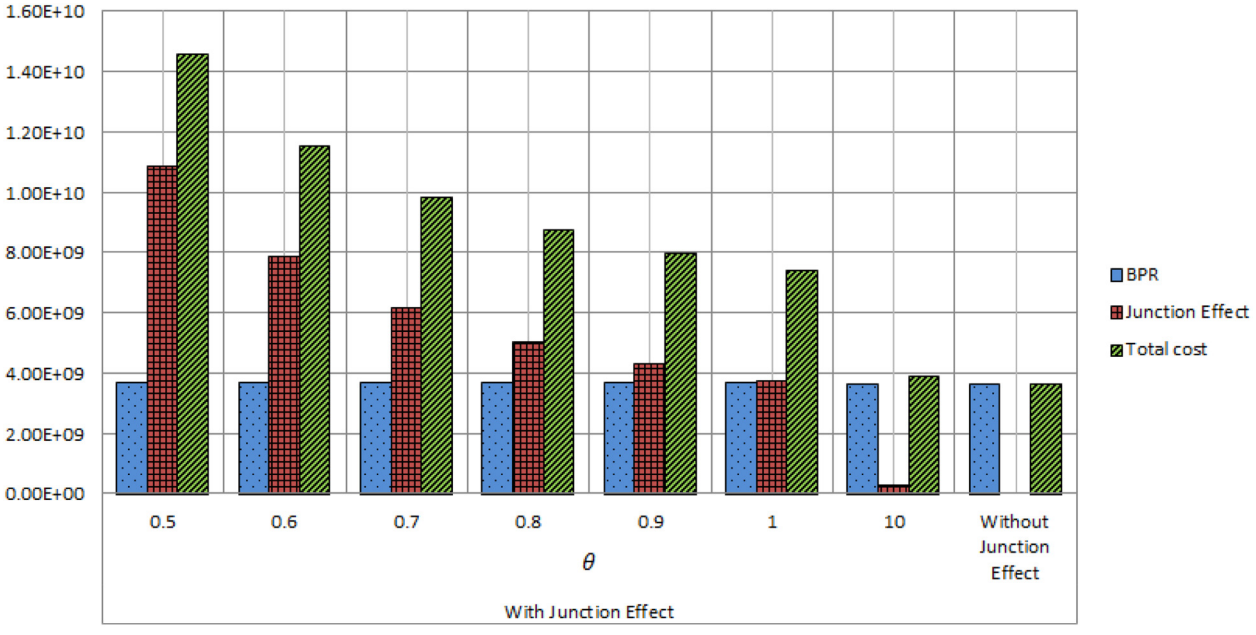

Figure 12. Impact of junction effect to total cost (grid15).

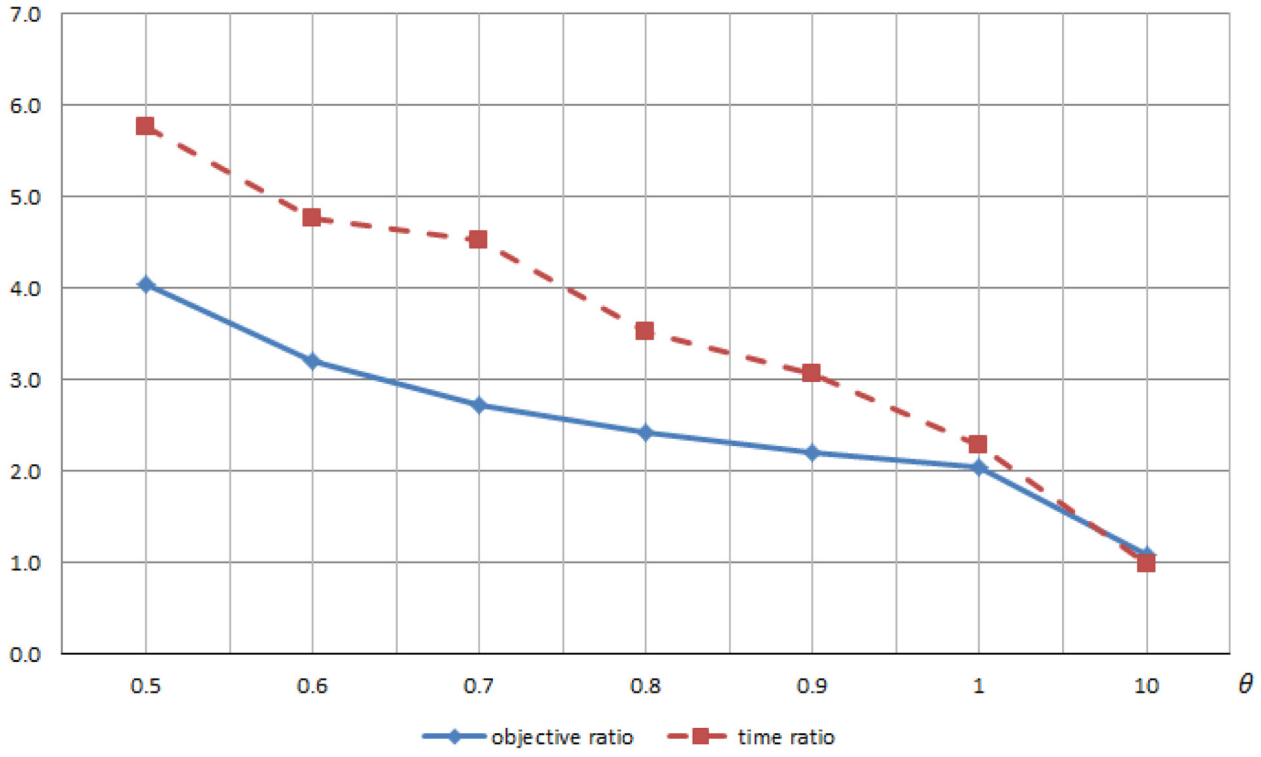

Figure 13. Comparison of objective value and computation time on ratio (grid15).

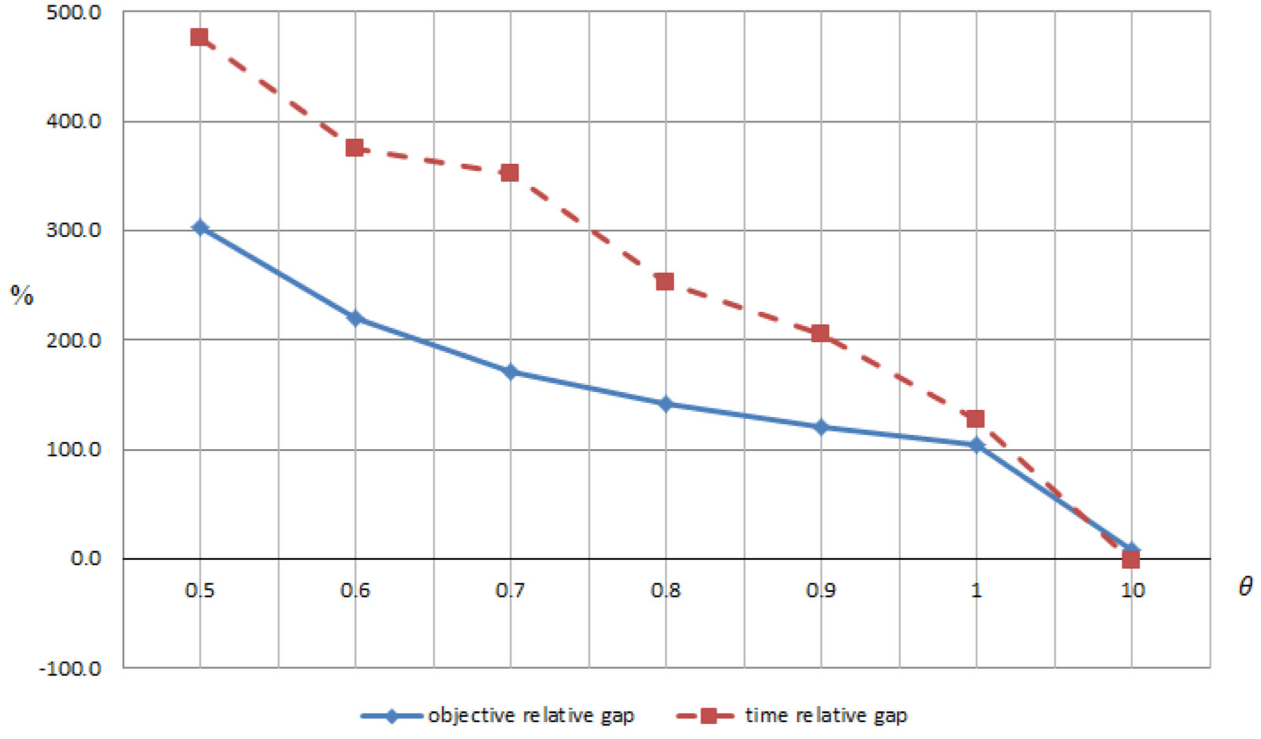

Figure 14. Comparison of objective value and computation time on relative gap (grid15).

Table 1. Results of the CMNET on planar instances with non-separable cost.

		ω_t = 10 millions							
		θ							
Problem	*CMNET*	0.5	0.6	0.7	0.8	0.9	1	10	$\omega_t = 0$
planar30	Objective	2.190750×10^{8}	1.761541×10^{8}	1.508551×10^{8}	1.342751×10^{8}	1.224911×10^{8}	1.137653×10^{8}	5.200142×10^{7}	4.445474×10^{7}
	Time (s)	1.19	1.23	0.63	0.46	0.38	0.38	0.38	0.52
	No. of iterations	10	11	6	4	3	3	3	7
planar50	Objective	9.525005×10^{8}	5.762907×10^{8}	4.539112×10^{8}	3.866600×10^{8}	3.432433×10^{8}	3.126639×10^{8}	1.377090×10^{8}	1.212368×10^{8}
	Time (s)	85.42	6.36	4.82	2.52	3.00	3.62	2.42	1.94
	No. of iterations	263	22	16	8	10	12	8	9
planar80	Objective	1.586457×10^{9}	1.086824×10^{9}	8.572356×10^{8}	7.232352×10^{8}	6.360912×10^{8}	$5.744556\times10^{8}8$	2.178872×10^{8}	1.819062×10^{8}
	Time (s)	25.62	12.09	9.22	11.23	7.39	8.74	8.63	5.52
	No. of iterations	38	17	13	16	10	12	11	9
planar100	Objective	1.465782×10^{9}	1.066978×10^{9}	8.689936×10^{8}	7.505782×10^{8}	$6.71262l\times10^{8}$	6.140928×10^{8}	2.682617×10^{8}	2.291143×10^{8}
	Time (s)	19.19	20.08	15.80	15.24	13.04	11.03	12.32	7.14
	No. Iterations	14	15	12	12	10	8	8	6
planar150	Objective	7.855872×10^{9}	3.797494×10^{9}	2.713782×10^{9}	2.180323×10^{9}	1.861127×10^{9}	1.646278×10^{9}	6.105529×10^{8}	5.279887×10^{8}
	Time (s)	1,580.69	229.62	76.02	55.07	49.83	56.08	50.82	33.26
	No. of iterations	381	68	21	15	14	16	14	11
planar300	Objective	2.302135×10^{9}	1.950056×10^{9}	1.737585×10^{9}	1.593123×10^{9}	1.488024×10^{9}	1.407407×10^{9}	7.709480×10^{8}	6.907479×10^{8}
	Time (s)	200.67	122.72	108.84	66.47	73.52	65.38	87.70	52.96
	No. of iterations	20	12	11	7	8	7	9	7
planar500	Objective	1.468117×10^{9}	1.312565×10^{9}	1.206013×10^{9}	1.127716×10^{9}	1.067252×10^{9}	1.018313×10^{9}	5.569095×10^{8}	4.833101×10^{8}
	Time (s)	130.10	78.94	62.21	45.32	45.13	44.82	26.25	27.15
	No. of iterations	7	4	3	2	2	2	1	2
planar800	Objective	3.142523×10^{9}	2.817143×10^{9}	2.594417×10^{9}	2.430800×10^{9}	2.304926×10^{9}	2.204266×10^{9}	1.298706×10^{9}	1.169546×10^{9}
	Time (s)	348.39	243.91	176.55	172.03	178.85	105.92	105.82	80.56
	No. of iterations	4	3	2	2	2	1	1	1
planar1000	Objective	1.021800×10^{10}	8.690936v	7.786128×10^{9}	7.176674×10^{9}	6.732759×10^{9}	6.389558×10^{9}	3.729196×10^{9}	3.418617×10^{9}
	Time (s)	2,006.47	3,196.44	1,888.65	1,296.14	2,233.48	1,049.99	874.06	918.78
	No. of iterations	15	26	15	10	17	8	6	8
planar2500	Objective	-	2.947262×10^{10}	2.642591×10^{10}	2.436787×10^{10}	2.287121×10^{10}	2.172797×10^{10}	1.327839×10^{10}	1.238293×10^{10}
	Time (s)	-	36,256.03	18,444.81	17,349.80	16,306.18	17,179.85	11,326.20	11666.50
	No. of iterations	-	25	12	12	11	12	7	8

Table 2. Results of the CMNET on grid instances with non-separable cost.

		$\omega_t = 10$ millions							
		θ							
Problem	CMNET	0.5	0.6	0.7	0.8	0.9	1	10	$\omega_t = 0$
grid1	Objective	3.111906×10^{10}	9.243753×10^{9}	4.225905×10^{8}	2.785320×10^{8}	2.142544×10^{8}	1.750493×10^{8}	1.078009×10^{7}	8.335999×10^{5}
	Time (s)	0.96	0.29	0.54	0.82	0.55	0.66	0.22	0.44
	No. of iterations	13	3	7	13	8	10	2	9
grid2	Objective	5.016990×10^{11}	1.286580×10^{11}	2.769531×10^{9}	8.739776×10^{8}	5.451362×10^{8}	3.971996×10^{8}	1.676195×10^{7}	1.726902×10^{6}
	Time (s)	83.70	90.68	96.62	1.57	1.15	1.39	0.78	0.42
	No. Iterations	1000	1000	1000	19	14	17	9	6
grid3	Objective	5.393188×10^{8}	4.020119×10^{8}	3.201661×10^{8}	2.659610×10^{8}	2.274094×10^{8}	1.985624×10^{8}	1.685141×10^{7}	1.532418×10^{6}
	Time (s)	7.11	5.69	4.57	5.30	4.09	5.16	1.12	1.10
	No. of iterations	20	18	13	16	10	16	3	5
grid4	Objective	1.266980×10^{9}	8.581656×10^{8}	6.482507×10^{8}	5.204340×10^{8}	4.344326×10^{8}	3.726059×10^{8}	2.854590×10^{7}	3.055429×10^{6}
	Time (s)	22.18	16.41	10.91	8.25	6.37	6.46	1.51	2.06
	No. Iterations	60	43	28	20	14	15	3	12
grid5	Objective	1.548514×10^{9}	1.140409×10^{9}	9.024649×10^{8}	7.465786×10^{8}	6.364964×10^{8}	5.546081×10^{8}	4.726632×10^{7}	5.079207×10^{6}
	Time (s)	28.24	28.28	21.61	32.57	36.68	31.84	4.87	2.41
	No. of iterations	18	18	14	22	26	24	3	8
grid6	Objective	7.826886×10^{9}	3.959311×10^{9}	2.662175×10^{9}	2.007523×10^{9}	1.611947×10^{9}	1.346873v	9.162650×10^{7}	1.050754×10^{7}
	Time (s)	239.38	145.51	92.05	44.88	29.95	29.51	12.98	5.66
	No. of iterations	175	109	68	28	18	18	8	12
grid7	Objective	7.003848×10^{9}	4.484321×10^{9}	3.302495×10^{9}	2.615316×10^{9}	2.165788×10^{9}	1.848751×10^{9}	1.498749×10^{8}	2.606693×10^{7}
	Time (s)	533.08	362.09	273.72	218.24	157.68	89.16	26.85	12.53
	No. of iterations	131	87	64	46	31	17	5	10
grid8	Objective	1.673972×10^{10}	9.478416×10^{9}	6.624531×10^{9}	5.097262×10^{9}	4.145145×10^{9}	3.494353×10^{9}	2.621074×10^{8}	4.212406×10^{7}
	Time (s)	1,600.67	992.10	973.30	669.70	471.43	308.22	89.34	30.45
	No. of iterations	161	90	96	64	43	26	7	13
grid9	Objective	4.595436×10^{10}	1.668445×10^{10}	1.024978×10^{10}	7.413728×10^{9}	5.815285×10^{9}	4.789839×10^{9}	3.542035×10^{8}	8.363939×10^{7}
	Time (s)	6,926.42	2,165.51	1,292.86	380.92	994.69	792.99	134.55	90.04
	No. of iterations	575	193	107	27	90	69	10	22
grid10	Objective	5.966468×10^{10}	1.700617×10^{10}	1.045019×10^{10}	7.568890×10^{9}	5.948836×10^{9}	4.910370×10^{9}	4.394879×10^{8}	1.660848×10^{8}
	Time (s)	20,987.94	3,205.92	1,643.35	672.76	434.58	1,120.36	229.43	144.54
	No. of iterations	1000	229	105	36	25	78	14	19
grid11	Objective	1.660000×10^{10}	9.660999×10^{9}	6.873306×10^{9}	5.369133×10^{9}	4.428305×10^{9}	3.783809×10^{9}	5.575892×10^{8}	3.324756×10^{8}
	Time (s)	4,449.69	1,932.74	761.69	1,347.42	996.43	726.01	204.58	194.711
	No. of iterations	219	89	32	64	47	33	9	14
grid12	Objective	1.242131×10^{10}	8.616448×10^{9}	6.663868×10^{9}	5.475804×10^{9}	4.676188×10^{9}	4.101014×10^{9}	8.369617×10^{8}	5.814886×10^{8}
	Time (s)	3,788.06	1,309.06	1,983.94	1,184.71	778.52	769.88	251.86	237.027
	No. Iterations	81	27	43	25	16	16	5	8

(continued)

Table 2. Continued.

		$\omega_t = 10$ millions							
		θ							
Problem	CMNET	0.5	0.6	0.7	0.8	0.9	1	10	$\omega_t = 0$
grid13	Objective	1.699347×10^{10}	1.127847×10^{10}	8.598296×10^{9}	7.043901×10^{9}	6.028342×10^{9}	5.312461×10^{9}	1.456647×10^{9}	1.169337×10^{9}
	Time (s)	8,284.60	6,518.27	3,524.50	1,936.57	1,344.97	2,151.53	521.99	622.848
	No. of iterations	111	92	48	26	18	30	7	9
grid14	Objective	1.150807×10^{10}	8.928208×10^{9}	7.435499×10^{9}	6.461531×10^{9}	5.775457×10^{9}	5.265827×10^{9}	2.083582×10^{9}	1.812975×10^{9}
	Time (s)	5,124.82	5,333.12	3,116.24	2,417.35	1,886.12	2,568.77	684.19	629.411
	No. of iterations	37	40	23	18	14	20	5	5
grid15	Objective	1.457560×10^{10}	1.154448×10^{10}	9.832353×10^{9}	8.730471×10^{9}	7.961010×10^{9}	7.392795×10^{9}	3.904739×10^{9}	3.615686×10^{9}
	Time (s)	7,928.48	6,542.36	6,221.99	4,855.01	4,208.44	3,130.27	1,354.24	1376.404
	No. of iterations	33	28	27	21	19	14	6	6

that is, computation time to find the optimal solution. For planar1000, as considering the junction effect, the average of computation time increases 1.95 times. Especially, at the strict capacity coefficient (e.g. 0.6) the average of computation time increases up to 3.48 times as compared with the computation time of seeking the base solution. A similar trend also occurs to grid15.

While total cost increases linearly with respect to decreasing the capacity coefficient at junctions nodes, the computation time fluctuates. However, in general the computation time increases when decreasing the capacity coefficient at junctions nodes.

We also solved all instances of planar and grid for test cases with non-separable convex costs by the CMNET. The results are shown in Tables 1 and 2, respectively. Only for planar2500 with $\theta = 0.5$, the CMNET cannot solve because of limited memory. Most all instances need more computation time to search for the optimal solution when the term of non-separable cost is integrated. The ratio of computation time between the non-separable cost instances and the separable cost instance (i.e. reference solution) is provided in Tables 3 and 4 for planar and grid instances, respectively. We see that the average computation time of the CMNET increases 3 and 16 times for planar and grid instances, respectively. This demonstrates the efficiency of the CMNET for solving the test cases with non-separable convex costs.

Table 3. Impact of non-separable cost to the computation time of the CMNET on planar instances.

	ω_t = 10 millions							
	θ							
Problem	0.5	0.6	0.7	0.8	0.9	1	10	Average ratio
planar30	2.27	2.36	1.21	0.89	0.73	0.72	0.72	1.27
planar50	44.03	3.28	2.48	1.30	1.55	1.87	1.24	7.96
planar80	4.64	2.19	1.67	2.04	1.34	1.58	1.56	2.15
planar100	2.69	2.81	2.21	2.13	1.83	1.55	1.72	2.13
planar150	47.52	6.90	2.29	1.66	1.50	1.69	1.53	9.01
planar300	3.79	2.32	2.06	1.26	1.39	1.23	1.66	1.96
planar500	4.79	2.91	2.29	1.67	1.66	1.65	0.97	2.28
planar800	4.32	3.03	2.19	2.14	2.22	1.31	1.31	2.36
planar1000	2.18	3.48	2.06	1.41	2.43	1.14	0.95	1.95
planar2500	-	3.11	1.58	1.49	1.40	1.47	0.97	1.67

Table 4. Impact of non-separable cost to the computation time of the CMNET on grid instances.

	ω_t = 10 millions							
	θ							
Problem	0.5	0.6	0.7	0.8	0.9	1	10	Average ratio
grid1	2.18	0.66	1.23	1.87	1.26	1.50	0.49	1.31
grid2	200.23	216.94	231.15	3.75	2.76	3.33	1.86	94.29
grid3	6.46	5.17	4.15	4.81	3.71	4.69	1.02	4.29
grid4	10.77	7.97	5.30	4.01	3.09	3.14	0.73	5.00
grid5	11.72	11.74	8.97	13.52	15.22	13.22	2.02	10.92
grid6	42.28	25.70	16.26	7.93	5.29	5.21	2.29	14.99
grid7	42.53	28.89	21.84	17.41	12.58	7.11	2.14	18.93
grid8	52.57	32.58	31.96	21.99	15.48	10.12	2.93	23.95
grid9	76.93	24.05	14.36	4.23	11.05	8.81	1.49	20.13
grid10	145.20	22.18	11.37	4.65	3.01	7.75	1.59	27.96
grid11	22.85	9.93	3.91	6.92	5.12	3.73	1.05	7.64
grid12	15.98	5.52	8.37	5.00	3.28	3.25	1.06	6.07
grid13	13.30	10.47	5.66	3.11	2.16	3.45	0.84	5.57
grid14	8.14	8.47	4.95	3.84	3.00	4.08	1.09	4.80
grid15	5.76	4.75	4.52	3.53	3.06	2.27	0.98	3.55

Test cases with side constraints

As mentioned in section 'Two-phase gradient projection algorithm', the CMNET can also solve non-separable convex multicommodity flow problems with side constraints. However, the CMNET need to be modified slightly in the process of determining feasible solution after each iteration. A Pseudo code of the modified two-phase gradient projection algorithm for non-separable convex multicommodity flow problems with side constraints is shown in Algorithm 2. We illustrate one iteration of this algorithm in Figure 15.

To evaluate the performance of the CMNET for solving such problems, we constructed test cases with the same objective function in Section 'Test cases with non-separable convex costs'. However, we modified a number of side constraints into the instances which are simply defined as capacity constraints at some arcs for some certain commodities. We solved planar1000 and grid12 to illustrate the capability of the CMNET for solving such problems. Here, the percentage of number of commodity which is assigned to be side constraints is 1%, 2%, and 3% of **K**, while the percentage of number of corresponding arcs is 1%, 2%, 3%, 4%, and 5% of **A**.

The results of the CMNET for solving planar1000 and grid12 with different number of side constraints are shown in Figures 16, 17 and 18, 19, respectively. In Figures 16 and 17, we compare the objective values and computation time of test cases with side constraints with those of separable test cases and non-separable test cases, respectively. From the results, we can see that the objective values of

Algorithm 2.

Step 0. **Initialization**:
Set $h := 0$, $\epsilon := 1\times10^{-8}$, and $\beta_0 := 1000$.

Step 1. **Shortest path and GSS**:
Solve a sequence of SP and GSS with 50 times to find x_h.

Step 2. **Check the feasible condition**:
If ($x_h \notin \mathbf{X}_S$) where $\mathbf{X}_S$ is the feasible solution set including side constraints, **Then** solve once **SOCP** with $z_{h+1} := x_h$ to obtain the feasible solution x_{h+1};
Else set $x_{h+1} := x_h$, let $h := h + 1$, and go to Step 3.

Step 3. **Modified gradient projection**:
Solve the **SOCP** with $z_{h+1} := x_h - \beta_h \nabla g(x)|_{x=x_h}$.
If satisfying one of the following conditions:
$g(x_{h+1}^{SOCP}) < g(x_h)$ or $\nabla g(x)'|_{x=x_{h+1}^{SOCP}}(x_h - x_{h+1}^{SOCP}) < 0$,
Then go to Step 4;
Else reduce β a half, and go back Step 3.

Step 4. **Check termination condition**:
If satisfying the termination criterion (4),
Then stop and conclude that $x_{h+1}^{SOCP} := \text{argmin}_{x\in\mathbf{X}} g(x)$.
If $\nabla g(x)'|_{x=x_{h+1}^{SOCP}}(x_h - x_{h+1}^{SOCP}) < 0$,
Then implement GSS to find an improved solution x_{h+1}^{GSS} between x_h and x_{h+1}^{SOCP}.
Let $h := h + 1$, and go back Step 1.

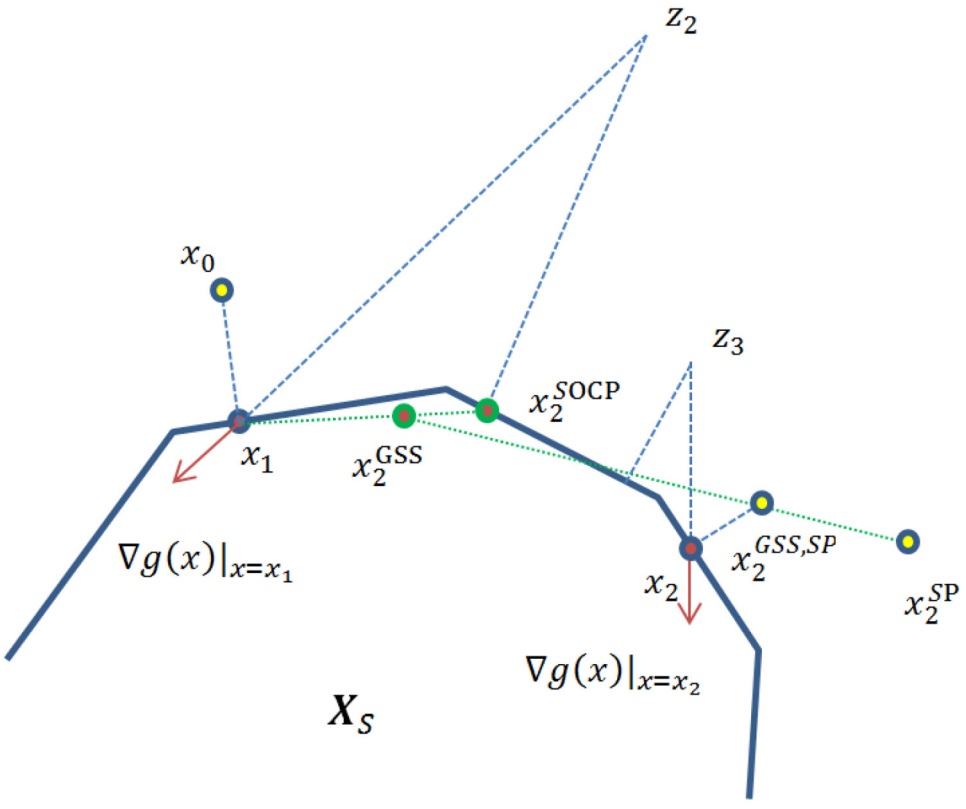

Figure 15. Illustration of one iteration in the two-phase gradient projection algorithm for the problem with side constraints.

test cases with side constraints is not as much different as those of non-separable test cases. However, the computation time increases significantly as compared with non-separable test cases. There is the same trend of computation time for separable test cases.

We also obtain the similar results when applying the CMNET to solve grid12 with side constraints. However, the computation time for this instance increases more significantly than that of solving planar1000. This may be since the structure of grid instances is more difficult than planar instances. In general, the CMNET can handle successfully the non-separable convex network flow problems with side constraints in an increment of reasonable computation time.

In summary, while separable convex optimization is not much harder than linear optimization,[10] nonseparable optimization problems have been shown to be considerably more difficult than separable problems.[11] This shows

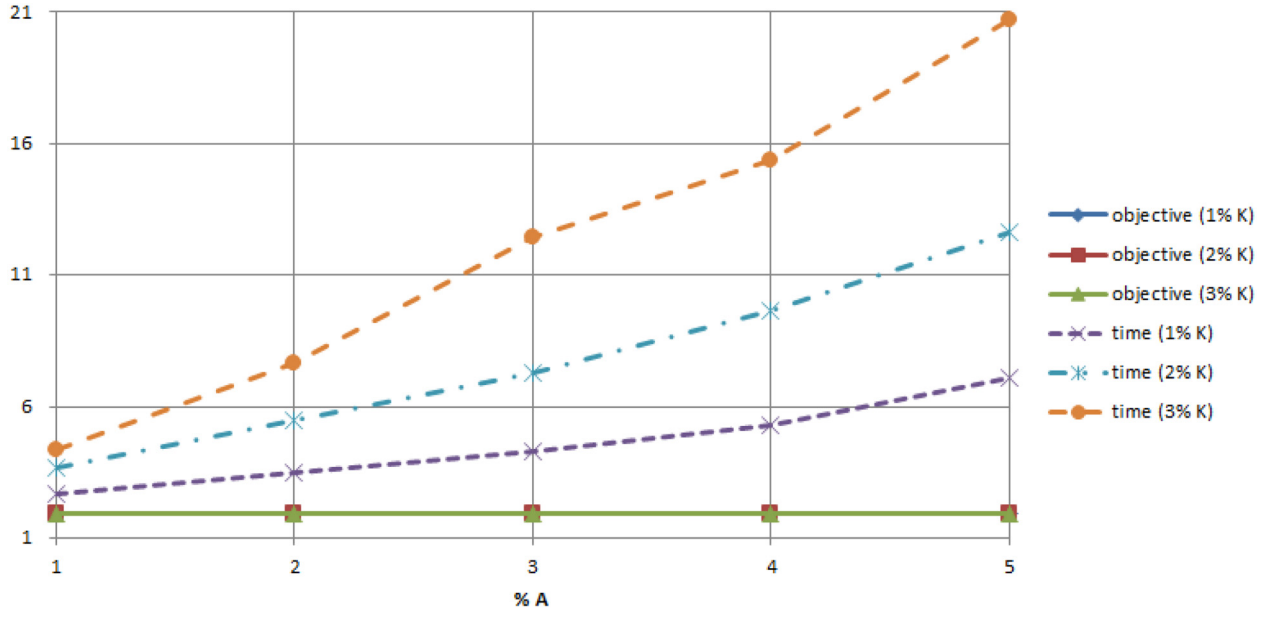

Figure 16. Comparison of solutions in separable test cases and side constraints test cases on planar1000.

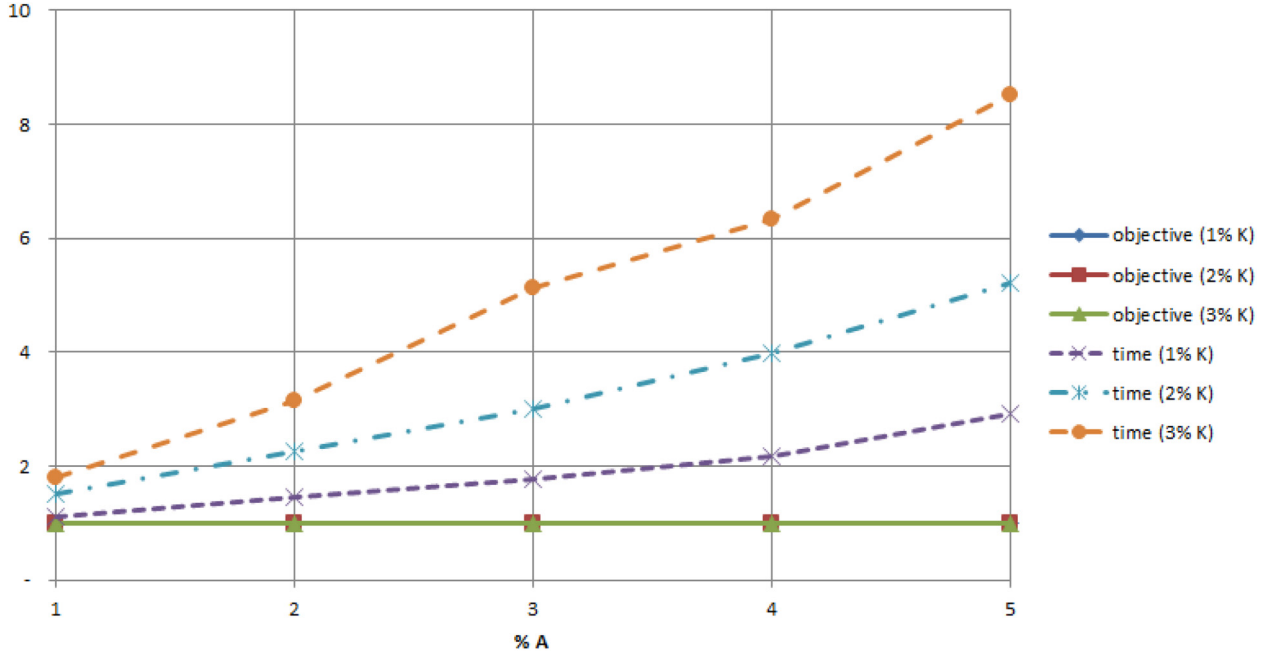

Figure 17. Comparison of solutions in non-separable test cases and side constraints test cases on planar1000.

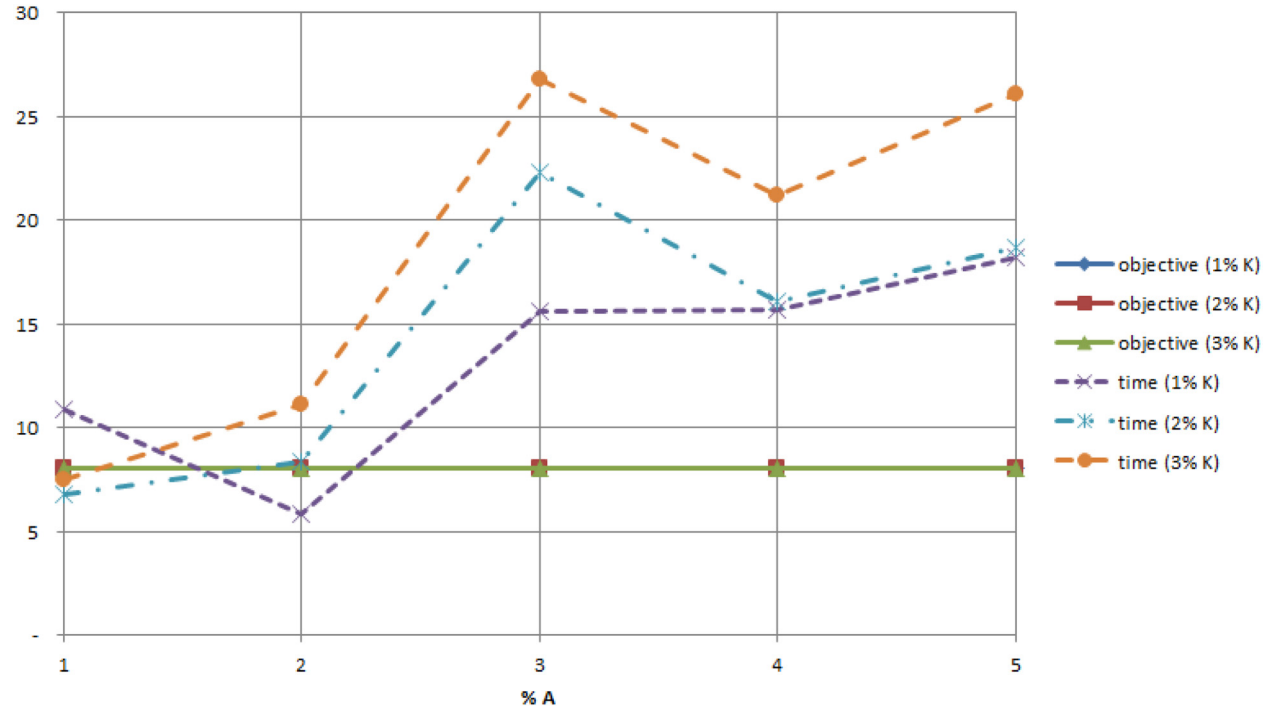

Figure 18. Comparison of solutions in separable test cases and side constraints test cases on grid12.

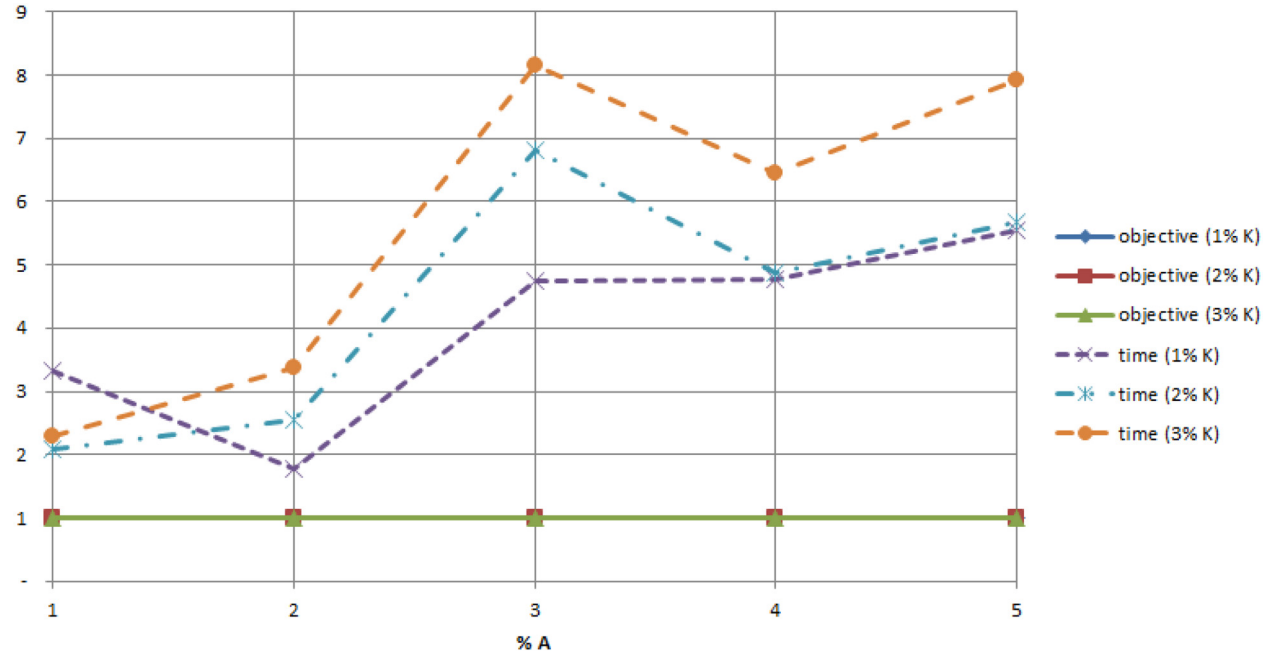

Figure 19. Comparison of solutions in non-separable test cases and side constraints test cases on grid12.

more clearly through the experiments carried out in this section. Hence, with solving successfully the non-separable convex network flow problems with side constraints the CMNET could be a promising toolbox for handling large-scale industrial applications.

Conclusions and future work

In this article, we introduce a CMNET toolbox based on a two-phase gradient projection algorithm for solving large-scale non-separable nonlinear multicommodity flow problems. The CMNET toolbox can solve multicommodity

flow problems with non-separable convex costs, while other current algorithms cannot do. Also, the CMNET is applicable for practical problems where objective function not available analytically as polynomial but are evaluated using black-boxes. In addition, it can handle the problems in which additional side constraints are not of network flow types. As compared with the ACCPM on the test cases with separable convex costs, the experimental results show that the CMNET can find the optimal solutions within a reasonable increasing computation time. For the test cases with non-separable convex costs, while the ACCMP fails, the CMNET can solve them successfully. This demonstrates the promising potential of the CMNET for industrial applications where non-separable convex costs are often encountered. A possible future work is to develop this algorithm for solving the network flow problems under uncertainties or disruptions.

Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The authors received no financial support for the research, authorship and/or publication of this article.

ORCID iD


Trung Hieu Tran https://orcid.org/0000-0002-3989-4502

References

1. Ahuja RK, Magnanti TL and Orlin JB. *Network flows: Theory, algorithms, and applications*. Englewood Cliffs: Prentice Hall, 1993.
2. Ouorou A, Mahey P and Vial J-P. A survey of algorithms for convex multicommodity flow problems. *Manage Sci* 2000; 46: 126–147.
3. Dembo RS, Mulvey JM and Zenios SA. Large-scale nonlinear network models and their application. *Oper Res* 1989; 37: 353–372.
4. Fathi M, Khakifirooz M and Pardalos PM eds. Optimization in large scale problems: Industry 4.0 and Society 5.0 applications. Springer, 2019.
5. Dembo RS and Klincewicz JG. A scaled reduced gradient algorithm for network flow problems with convex separable costs. *Math Program Stud* 1981; 15: 125–147.
6. Bertsekas DP, Polymenakos LC and Tseng P. An epsilon-relaxation method for separable convex cost network flow problems. *IAM J Optim* 1997; 7: 853–870.
7. Babonneau F and Vial J-P. ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. *Math Program Ser B* 2009; 120: 179–210.
8. Bertsekas DP, Polymenakos LC and Tseng P. Separable convex cost network flow. *Network Optim* 2012; 450: 103.
9. Patriksson M. *The traffic assignment problem: Models and methods*. The Netherlands: VSP, Utrecht, 1994.
10. Hochbaum DS and Shanthikumar JG. Convex separable optimization is not much harder than linear optimization. *J ACM* 1990; 37: 843–862.
11. Hochbaum DS. Complexity and algorithms for nonlinear optimization problems. *Ann Oper Res* 2007; 153: 257–296.
12. Dafermos SC. An extended traffic assignment model with applications to two-way traffic. *Transp Sci* 1971; 5: 366–389.
13. Akcelik R. Capacity of a shared lane. *Aust Road Res Board Proc* 1988; 14: 228–241.
14. Larsson T and Patriksson M. Side constrained traffic equilibrium models—analysis, computation and applications. *Transport Res Part B: Meth* 1999; 33: 233–264.
15. Agustin A, Alonso-Ayuso A, Escudero LF, Pizarro C et al. On air traffic flow management with rerouting. Part I: Deterministic case. *Eur J Oper Res* 2012; 219: 156–166.
16. Gabriel SA and Bernstein D. The traffic equilibrium problem with nonadditive path costs. *Transp Sci* 1997; 31: 337–348.
17. Meselhi M, Sarker R, Essam D, Elsayed S et al. A decomposition approach for large-scale non-separable optimization problems. *Appl Soft Comput* 2022; 115: 108168.
18. Road Bureau of Public. *Traffic Assignment Manual*. Washington, DC, US: Department of Commerce, Urban Planning Division, 1964.
19. Branston D. Link capacity functions: a review. *Transp Res* 1976; 10: 223–236.
20. Singh R and Dowling R. Improved speed-flow relationships: Application to transportation planning models. In: *Proceedings of the 7th TRB Conference on Application of Transportation Planning Methods*, Boston, Massachusetts (1999).
21. Chen A, Zhou Z and Xu X. A self-adaptive gradient projection algorithm for the nonadditive traffic equilibrium problem. *Comput Oper Res* 2012; 39: 127–138.
22. Bertsekas DP and Gallager RG. *Data Networks*. 2nd. Englewood Cliffs: Prentice Hall, 1992.
23. Holmberg K and Yuan D. A multicommodity network flow problem with side constraints on paths solved by column generation. *INFORMS J Comput* 2003; 15: 42–57.
24. Dijkstra EW. A note on two problems in connexion with graphs. *Numer Math* 1959; 1: 269–271.
25. Bertsekas DP. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Trans Automat Control* 1976; 21: 174–183.
26. Kiefer J. Sequential minimax search for a maximum. *Proc Am Math Soc* 1953; 4: 502–506.

Appendix A. Comparison result of the ACCMP and the CMNET on planar instances (BPR function).

						$\epsilon = 1\times10^{-}5$			$\epsilon = 1\times10^{-}88$		
				ACCPM		*CMNET*			*CMNET*		
Problem	*N*	*A*	*K*	Objective	Time (s)	Objective	Time (s)	No. of iterates	Objective	Time (s)	No. of iterates
planar30	30	150	92	4.445490×10^{7}	1.2	4.445655×10^{7}	0.24	3	4.445474×10^{7}	0.52	7
planar50	50	250	267	1.212360×10^{8}	2.9	1.212370×10^{8}	1.32	6	1.212368×10^{8}	1.94	9
planar80	80	440	543	1.819060×10^{8}	7.6	1.819063×10^{8}	4.35	7	1.819062×10^{8}	5.52	9
planar100	100	532	1085	2.291140×10^{8}	4.2	2.291145×10^{8}	4.89	4	2.291143×10^{8}	7.14	6
planar150	150	850	2239	5.279850×10^{8}		5.280034×10^{8}	16.27	5	5.279887×10^{8}	33.26	11
planar300	300	1680	3584	6.907480×10^{8}	9.4	6.907823×10^{8}	16.10	2	6.907479×10^{8}	52.96	7
planar500	500	2842	3525	4.833090×10^{8}	5.1	4.833109×10^{8}	14.98	1	4.833101×10^{8}	27.15	2
planar800	800	4388	12,756	1.169520×10^{9}	27.1	1.169546×10^{9}	80.56	1	1.169546×10^{9}	80.56	1
planar1000	1000	5200	20,026	3.418590×10^{9}	113.6	3.418677×10^{9}	364.00	3	3.418617×10^{9}	918.78	8
planar2500	2500	12,990	81,430	1.238270×10^{10}	1540.00	1.238752×10^{10}	3081.61	2	1.238293×10^{10}	11666.50	8
RelativeGap					$\leq$ **0.0389%**			$\leq$ **0.0022%**			

Appendix B. Comparison result of the ACCMP and the CMNET on planar instances (Kleinrock function).

						$\epsilon = 1\times10^{-5}$			$\epsilon = 1\times10^{-88}$		
				ACCPM		*CMNET*			*CMNET*		
Problem	*N*	*A*	*K*	Objective	Time (s)	Objective	Time (s)	No. of iterates	Objective	Time (s)	No. of iterates
planar30	30	150	92	40.56680	1.1	40.56684	0.61	9	40.56680	1.14	17
planar50	50	250	267	109.47800	2.2	109.48001	1.70	8	109.47867	4.88	26
planar80	80	440	543	232.32100	6.5	232.32125	7.56	15	**232.32086**	14.46	30
planar100	100	532	1085	226.29900	6	226.30786	7.02	7	226.30190	28.79	30
planar150	150	850	2239	715.30900		715.63618	52.24	19	715.31953	931.89	336
planar300	300	1680	3584	329.12000	22.2	329.12253	62.44	10	329.12056	86.79	14
planar500	500	2842	3525	196.39400	24.1	196.39469	74.23	8	196.39430	142.23	15
planar800	800	4388	12,756	354.00800	77.1	354.01066	299.01	6	354.00883	640.36	13
planar1000	1000	5200	20,026	1250.92000	303.6	1251.10949	703.79	7	1250.93257	7265.66	72
planar2500	2500	12,990	81,430	3289.05000	2398.50	3290.32364	14767.95	11	3289.15051	194797.74	142
RelativeGap					$\leq$ **0.0457**%			$\leq$ **0.0031**%			

Appendix C. Comparison result of the ACCMP and the CMNET on grid instances (BPR function).

						$\epsilon = 1\times10^{-5}$			$\epsilon = 1\times10^{-8}$		
				ACCPM		*CMNET*			*CMNET*		
Problem	*N*	*A*	*K*	Objective	Time (s)	Objective	Time (s)	No. of iterates	Objective	Time (s)	No. of iterates
grid1	25	80	50	8.335990×10^{5}	0.4	8.336008×10^{5}	0.25	5	8.335999×10^{5}	0.44	9
grid2	25	80	100	1.726890×10^{6}	0.8	1.726913×10^{6}	0.21	3	1.726902×10^{6}	0.42	6
grid3	100	360	50	1.532410×10^{6}	0.7	1.532428×10^{6}	0.69	2	1.532418×10^{6}	1.10	5
grid4	100	360	100	3.055430×10^{6}	1.4	3.055487×10^{6}	0.60	3	3.055429×10^{6}	2.06	12
grid5	225	840	100	5.079210×10^{6}	1.9	5.079339×10^{6}	0.77	2	5.079207×10^{6}	2.41	8
grid6	225	840	200	1.050750×10^{7}	6.4	1.050759×10^{7}	2.51	5	1.050754×10^{7}	5.37	12
grid7	400	1520	400	2.606690×10^{7}	6.7	2.606738×10^{7}	3.93	3	2.606693×10^{7}	11.51	10
grid8	625	2400	500	4.212400×10^{7}	19.1	4.212480×10^{7}	12.63	5	4.212406×10^{7}	30.45	13
grid9	625	2400	1000	8.363940×10^{7}	39	8.364092×10^{7}	26.66	7	8.363939×10^{7}	80.85	22
grid10	625	2400	2000	1.660840×10^{8}	40.9	1.660881×10^{8}	48.84	7	1.660848×10^{8}	129.31	19
grid11	625	2400	4000	3.324750×10^{8}	28.4	3.324832×10^{8}	52.51	4	3.324756×10^{8}	177.95	14
grid12	900	3480	6000	5.814880×10^{8}	29.4	5.815027×10^{8}	57.00	2	5.814886×10^{8}	212.02	8
grid13	900	3480	12,000	1.169330×10^{9}	38.9	1.169354×10^{9}	163.14	3	1.169337×10^{9}	469.93	9
grid14	1225	4760	16,000	1.812970×10^{9}	40.8	1.812988×10^{9}	195.41	2	1.812975×10^{9}	466.50	5
grid15	1225	4760	32,000	3.615680×10^{9}	47.4	3.615731×10^{9}	388.30	2	3.615686×10^{9}	1103.17	6
RelativeGap					$\leq$ **0.0025%**			$\leq$ **0.0007%**			

Appendix D. Comparison result of the ACCMP and the CMNET on grid instances (Kleinrock function).

						$\epsilon = 1\times10^{-5}$			$\epsilon = 1\times10^{-8}$		
				ACCPM		*CMNET*			*CMNET*		
Problem	*N*	*A*	*K*	Objective	Time (s)	Objective	Time (s)	No. of iterates	Objective	Time (s)	No. of iterates
grid1	25	80	50	66.40020	0.5	66.40016	0.35	9	**66.40016**	0.42	10
grid2	25	80	100	194.51200	0.8	194.61232	1.09	19	194.51216	8.86	179
grid3	100	360	50	84.56180	2.3	84.56181	1.03	10	**84.56179**	1.39	14
grid4	100	360	100	171.33100	3.1	171.33636	2.46	17	171.33143	19.15	151
grid5	225	840	100	236.69900	12	236.69960	3.21	12	236.69937	5.05	20
grid6	225	840	200	652.87700	24.3	652.97924	5.70	13	652.87892	68.85	175
grid7	400	1520	400	776.56600	90.9	776.58672	20.52	18	**776.56436**	109.33	100
grid8	625	2400	500	1542.15000	384	1542.64507	27.47	12	1542.19543	338.15	151
grid9	625	2400	1000	2199.83000	305.5	2200.20889	70.68	18	2199.88045	1117.92	292
grid10	625	2400	2000	2212.89000	199.8	2213.02316	131.91	17	2212.92327	1196.17	156
grid11	625	2400	4000	1502.75000	96.6	1502.76715	138.75	10	1502.75176	348.69	26
grid12	900	3480	6000	1478.93000	107.2	1478.93560	808.67	28	1478.93477	892.49	31
grid13	900	3480	12,000	1760.53000	125.8	1760.55635	340.33	6	1760.53203	675.96	12
grid14	1225	4760	16,000	1414.39000	166.5	1414.39429	2002.88	19	1414.39275	2315.94	22
grid15	1225	4760	32,000	1544.15000	161.5	1544.16030	4881.69	21	1544.15756	5589.12	24
RelativeGap					$\leq$ **0.0516%**			$\leq$ **0.0029%**			