TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 09/20 pp753–765
DOI: 10.26599/TST.2023.9010055
Volume 29, Number 3, June 2024

Increasing the Maximum Capacity Path in a Network and Its Application for Improving the Connection Between Two Routers

Adrian M. Deaconu* and Javad Tayyebi

Abstract: This paper addresses the problem of improving the optimal value of the Maximum Capacity Path (MCP) through expansion in a flexible network, and minimizing the involved costs. The only condition applied to the cost functions is to be non-decreasing monotone. This is a non-restrictive condition, reflecting the reality in practice, and is considered for the first time in the literature. Moreover, the total cost of expansion is a combination of max-type cost (e.g., for supervision) and sum-type cost (e.g. for building infrastructures, price of materials, price of labor, etc.). For this purpose, two types of strategies are combined: (I) increasing the capacity of the existing arcs, and (II) adding potential new arcs. Two different problems are introduced and solved. Both the problems have immediate applications in Internet routing infrastructure. The first one is to extend the network, so that the capacity of an MCP in the modified network becomes equal to a prescribed value, therefore the cost of modifications is minimized. A strongly polynomial-time algorithm is deduced to solve this problem. The second problem is a network expansion under a budget constraint, so that the capacity of an MCP is maximized. A weakly polynomial-time algorithm is presented to deal with it. In the special case when all the costs are linear, a Meggido's parametric search technique is used to develop an algorithm for solving the problem in strongly polynomial time. This new approach has a time complexity of $O(n^4)$, which is better than the time complexity of $O(n^4 \log^2(n))$ of the previously known method from literature.

Key words: Maximum Capacity Path (MCP); network expansion; Internet routing; polynomial-time algorithms

1 Introduction

In a network, the Maximum Capacity Path Problem (MCPP), also known as the widest path problem, is to find a path between two specific nodes having the largest capacity. The capacity of a path is given by the minimum capacity of its arcs. There are several algorithms that efficiently solve MCPP in $O(m \log(n))$ time, where m is the number of arcs, n is the number of nodes, and log is the iterated logarithm, i.e., it is the minimum number of times the logarithm function must be iteratively applied to n nodes before the result is less than or equal to 1[1].

MCPP has real practical applications. One of the most important and immediate application is the connection between routers in Internet. The capacity of an arc represents the bandwidth of a connection between two routers. Here, Maximum Capacity Path (MCP) is to find an end-to-end path between two Internet nodes that has the maximum possible bandwidth[2]. The smallest arc capacity on this path is known as the capacity or bandwidth of the path. Other possible applications of MCPP include: the Schulze method for deciding the winner of a multiway election

• Adrian M. Deaconu is with Department of Mathematics and Computer Science, Transilvania University of Brasov, Brasov 500036, Romania. E-mail: a.deaconu@unitbv.ro.
• Javad Tayyebi is with Department of Industrial Engineering, Birjand University of Technology, Birjand 000097, Iran. E-mail: javadtayyebi@birjandut.ac.ir.
* To whom correspondence should be addressed.
Manuscript received: 2023-01-01; revised: 2023-04-25; accepted: 2023-05-26

that has been applied to digital compositing[3, 4], metabolic pathway analysis[5], or in finding the maximum flow in a network[6].

Since the expansion problems belong to the class of inverse/reverse optimization problems[7], let us review the related works in this field. The Inverse Maximum Capacity Path problem (IMCP) is studied in Ref. [8]. This is to modify the capacities of the arcs as little as possible, so that a given path becomes a maximum capacity path in the modified network. Two cases of IMCP are considered: the capacity of the given path is preserved, or not. IMCP is studied and solved in two cases, under any sum-type (e.g., weighted l_k norms and sum-type Hamming distance) and max-type distances (e.g., weighted l_∞ norm or bottleneck Hamming distance). The obtained algorithms for IMCP are applied to solve a real road transportation network optimization problem.

It is very likely that at some moment the network needs to be extended by increasing the arc capacities, and/or by adding new arcs. There are costs involved, and a given budget, and so, not all the new arcs can be added, and not all the capacities on the arcs can be increased to the maximum. Depending on the desired purpose of expansion, it results different problems with different strategies to solve.

In Ref. [9], a network expansion problem to increase the maximum flow in a network of electricity, water, gas, etc., is studied. In this problem, the flow augmentation can be achieved either by increasing the capacities on the existing arcs, or by adding new arcs to the network. Both operations are coming with an expansion cost. It is shown that this problem is reduced to the calculation of a minimum cost flow in an auxiliary network.

In this paper, the problem to efficiently increase the capacity of the MCP through network expansion is addressed. The expansion can be obtained by increasing the capacities of the existing arcs, and by adding new arcs to the network. This paper focuses on a general case for cost functions. Let us explain it in more details. Most network optimization problems consider the cost as a function linearly increasing, which generally does not reflect reality (see problems mentioned above and others, such as minimum cost flow problems[10]). When estimating the amount of money that has to be spent, different kinds of costs may be involved. In this paper, the only condition applied to the cost function on an arc is non-decreasing monotone, which is really non-restrictive from practical viewpoint. Moreover, the total cost may be a hybrid of max-type costs (e.g., for supervision of the project implementation) and sum-type costs (e.g., for building infrastructures, price of materials, labor, etc.). Two kinds of MCP expansion problems are introduced and solved. The first one is to extend the network, so that the capacity of MCP becomes equal to a predetermined value $z > 0$, and the cost of expansion is minimized. For this problem, a strongly polynomial algorithm in $O(n^2)$ time is developed. The second problem is a network expansion under a budget constrained, so that the capacity of MCP is maximized. In this case, a weakly polynomial algorithm is proposed in the general case. Moreover, the concept of Meggido's parametric search is used to introduce a better time complexity algorithm for solving the problem in the case of linear cost functions. To the best of our knowledge, there is only one paper which considers the MCP expansion problem by modifying the arcs capacities under a budget constraint[7]. This problem is studied under two sum-type distances: fixed costs and linear costs. In the former, an algorithm is designed to solve the problem in $O(m \log(n) + n \log^2(n))$ time. In the latter, a two phases polynomial-time approach is developed. The first phase applies the first algorithm as a subroutine to find a small interval containing the optimal value. The second phase converts the reduced problem into a minimum ratio path problem. Then, a Secant-Newton hybrid algorithm is proposed to obtain the exact optimal solution in $O(An^2 \log^2(n))$, where A is the running time of the first problem under linear cost functions (see Theorem 4.3 in Ref. [7] and Theorem 4 in Ref. [11]). So, based on the results of this paper, since we know that the first problem can be solved in $O(n^2)$ time, it follows that the complexity of the algorithm presented in Ref. [7] is $O(n^4 \log^2(n))$, which is greater than time complexity $O(n^4)$ of the algorithm proposed in this paper for the linear-cost case in Section 3.1.

The proposed methods of this paper have applicability for many practical optimization problems, such as to increase the bandwidth between two routers in Internet (e.g., in business or military communications at long distances), or for road network expansion between two locations (cities) by adding lanes or new links, so that after modifications, there is the possibility to have a connection between the two locations with minimum n lanes (e.g., $n = 3$) on each

link (portion of road). In the literature, there are studies that address the Discrete Network Design Problem (DNDP) with multiple capacity levels, which determines the optimal number of lanes to add to each candidate link in a road network. Of course, this problem is different than the proposed one in this paper, but shares the same idea of road widening by adding lanes, and DNDP is solved with different techniques, such as system optimal relaxation or mixed-integer linear programming[12].

2 Minimum Cost MCP Expansion

Let $G=(V,A,s,t,c)$ be a directed st-network, where V is the node set, $A\subseteq V\times V$ is the arc set, $s\in V$ is the source node, $t\in V$ is the sink node, and $c(w)=c(i,j)>0$ is the capacity of arc $w=(i,j)\in A$. For G, n denotes the number of nodes, i.e., $n=|V|$, and m is the number of arcs, i.e., $m=|A|$. An st-path P is a sequence of nodes $(s=i_1, i_2, \ldots, i_k=t)$ $(k\geqslant 1)$, so that $(i_l, i_{l+1})\in A$ for every $l=1, 2, \ldots, k-1$. The capacity of an st-path $P=(s=i_1, i_2, \ldots, i_k=t)$ is denoted by $c(P)$, and is the minimum capacity of its arcs, i.e., $c(P)=\min\{c(i_l, i_{l+1})|l=1, 2, \ldots, k-1\}$. The maximum capacity path is an st-path P^* having the maximum capacity among all st-paths, i.e., $c(P^*)\geqslant c(P)$ for every st-path P in G. So, the maximum capacity path problem in an st-network G can be formulated as follows:

$$\begin{aligned}&\max\ c(P),\\&\text{s.t., } P \text{ is an } st\text{-path in } G\end{aligned}\tag{1}$$

Let z be a given positive value greater than the capacity of the MCP in G, i.e., $z>c(P_G)$, where P_G is an MCP in G. The network G is intended to be extended (by increasing the capacities of the arcs, and by adding new arcs), so that the capacity of an MCP in the modified network is equal to z, and the cost of modification is minimized. We call this problem as Minimum Cost MCP Expansion Problem (MCMCPEP).

Since it is possible that the capacities of some arcs belonging to A are not allowed to be increased to the value z, we denote by $A_z\subseteq A$ the set of arcs having the capacities greater than or equal to z, and the arcs whose capacities are less than z, and however, their capacities are permitted to be increased to the value of z, i.e.,

$$\begin{aligned}A_z=\{&w\in A|c(w)\geqslant z \text{ or}\\&(c(w)<z \text{ and } c(w) \text{ is allowed to become } z)\}\end{aligned}\tag{2}$$

We define the cost of increasing the capacities of the arcs belonging to A_z by a generic cost function,

$$b: A_z\times\mathbf{R}^+\to\mathbf{R}^+\tag{3}$$

The cost function $b(w,\cdot)$ is considered non-decreasing monotone on the interval $[z,+\infty)$ for every arc $w\in A_z$. Obviously, since there is no need to increase the capacity of an arc from A_z when $c(w)\geqslant z$, we may define

$$b(w,z)=0, \text{ if } z\leqslant c(w), \forall w\in A_z\tag{4}$$

For instance, the total cost of expanding an arc $w\in A_z$ can include the following costs:

(1) A fixed cost that includes demounting the exiting infrastructures, the implementation of new infrastructures, the price of labor, etc.

(2) A cost of the materials, which is given by the price of the cable that is used in the implementation of the Ethernet and Internet networks, that depends on the type (Cat5, Cat5e, Cat6, Cat6a, or Cat7), characteristics (shielded or unshielded), section diameter, build quality, etc. Obviously, the higher the price of the materials, the larger bandwidth is obtained. So, the cost increases whenever the maximum desired bandwidth z is increased on arcs.

Let us consider now the case that potential new arcs may be added to the network. We denote by $A^a\subseteq(V\times V)-A$ a set of potential new arcs having the capacity equal to z that can be added to the network G. In order to add a new arc, at least two kinds of costs may be involved:

(1) A fixed cost for preparing new infrastructures, including the price of labor.

(2) A cost of materials which increases with the enlargement of the desired maximum bandwidth on the new arc.

So, we can extend the definition of the cost function b for the arcs of A^a as follows:

$$b: A^a\times\mathbf{R}^+\to\mathbf{R}^+\tag{5}$$

where $b(w,\cdot)$ is also non-decreasing monotone on the interval $[z,+\infty)$ for every arc $w\in A^a$.

We introduce the following notation:

$$A^E=A_z\cup A^a\tag{6}$$

Using Eq. (4) and Formula (5) the cost function b is now defined for $A^E\times\mathbf{R}^+$.

From now on, an arc $w\in A^E$ is called expansion arc if and only if it is an existing arc of which capacity is increased, or it is a new arc added to the network. We denote by $A^e\subseteq A^E$ the set of expansion arcs.

Besides the per arc expansion costs we also consider a supervision cost of the whole network expansion. The supervision cost depends on the whole expansion time (for all arcs). So, in order to minimize this cost, we suppose that the expansion of any arc is performed in parallel by a different team for each arc. Consequently, the supervision cost is given by the longest time needed to finish each arc. It can be formulated as follows:

$$\max_{w\in A^e} u\,(w,\,z) \tag{7}$$

where $u\,(w,\,z)$ is the cost for supervising the arc w to reach z capacity (bandwidth). Since a higher increase of the capacity on an arc w should take longer or the same time as for a lower increase of the capacity of w, we could consider $u\,(w,\,z)=f\,(t\,(w,\,z))$, where $f:\mathbf{R}^+\to\mathbf{R}^+$ is a non decreasing cost function depending on time, and $t\,(w,z)$ is the time needed to increase the capacity of w to the value of z. So, in our model, $u\,(w,\cdot)$ is also non-decreasing monotone on the interval $[z,+\infty)$ for every arc $w\in A_z$.

For A^e and the desired capacity z, the total expansion cost is

$$\max_{w\in A^e} u\,(w,\,z)+\sum_{w\in A^e} b\,(w,\,z) \tag{8}$$

When solving MCMCPEP, the capacities are modified on an st-path in the graph $G^E=(V,A^E)$. This path becomes MCP in the extended network. So, MCMCPEP can be formulated as follows:

$$\begin{aligned}&\min\left\{\max_{w\in P} u\,(w,\,z)+\sum_{w\in P} b\,(w,\,z)\right\},\\ &P \text{ is an } st\text{-path in } G^E=(V,\,A^E)\end{aligned} \tag{9}$$

Starting from Formula (9), Algorithm 1 (AMCMCPEP) is proposed for solving MCMCPEP, where $R\,(i)$ is a binary label used to indicate the status of the node i ($R\,(i)=1$ if i is checked, $R\,(i)=0$ otherwise), $M\,(j)$ and $S\,(j)$ are used to store the maximum and sum costs, respectively, on a path from s to j, $p\,(j)$ keeps the predecessor i of node j on the path found from s to j if the costs of j are updated for an arc $(i,\,j)$.

We shall demonstrate next the correctness of AMCMCPEP. But, first, let us prove some preliminary results (Lemmas 1, 2, and 3).

Lemma 1 For any node k having $R\,(k)=1$, a path P_k can be constructed from s to k, so that every node i from P_k has $R\,(i)=1$.

Proof For a node k with $R\,(k)=1$, a path P_k from s to k can be calculated using Algorithm 2 denoted APF (k): It is easy to see that every node j from P_k has $p\,(j)\neq -1$. When $p\,(j)$ becomes equal to $i\neq -1$, $R\,(i)$ is previously set to 1. So, every node i from P_k has $R\,(i)=1$. ■

Definition 1 For a node k with $R\,(k)=1$, the path P_k constructed by APF (k) is called the path found by AMCMCPEP from s to k.

Let us notice that the path P^* constructed by AMCMCPEP is the path P_t from s to t.

Lemma 2 For any node k having $R\,(k)=1$, we have

$$\begin{aligned}M\,(k)&=\max_{w\in P_k} u\,(w,\,z),\\ S\,(k)&=\sum_{w\in P_k} b\,(w,\,z)\end{aligned} \tag{10}$$

where P_k is the path found by AMCMCPEP from s to k.

Proof We shall prove Eq. (10) through induction by l being the length (number of arcs) of the paths P_k found by AMCMCPEP.

A path with 0-length is a path from s to $k=s$ with no arcs, or a path from s to a node k of which arcs have 0-costs. Of course, for such a path we have $M\,(k)=0$ and $S\,(k)=0$.

For an integer value $l\geqslant 1$, we suppose as induction hypothesis, that for every node h with $R\,(h)=1$ that has the path P_h found by AMCMCPEP with the length less than or equal to $l-1$, we have

$$\begin{aligned}M\,(h)&=\max_{w\in P_h} u\,(w,\,z),\\ S\,(h)&=\sum_{w\in P_h} b\,(w,\,z)\end{aligned} \tag{11}$$

Let k be a node, so that the path P_k found by AMCMCPEP has the length equal to l. Since $l\geqslant 1$, there exists a node h so that $p\,(k)=h$, and

$$\begin{aligned}M\,(k)&=\max\{M(h),\,u\,((h,\,k),\,z)\},\\ S\,(k)&=S\,(h)+b\,((h,\,k),\,z)\end{aligned} \tag{12}$$

If the last arc $(h,\,k)$ is eliminated, the path P_h found by AMCMCPEP from s to h is obtained. This path has the length equal to $l-1$. Then, from the induction hypothesis, Eq. (11) is true for node h.

Using Eqs. (11) and (12), we obtain

$$\begin{aligned}M\,(k)&=\max\,\{M\,(h),\,u\,((h,\,k),\,z)\}=\max_{w\in P_k} u\,(w,\,z),\\ S\,(k)&=b\,((h,\,k),\,z)+S\,(h)=\sum_{w\in P_k} b\,(w,\,z)\end{aligned} \tag{13}$$

■

Algorithm 1 (AMCMCPEP) Algorithm to solve MCMCPEP

Input: G, A_z, A^a, z
Build $G^E = (V, A_z \cup A^a)$;
if there is no path from s to t in G^E **then**
  MCMCPEP is not feasible;
  stop
end if
Find P_G = MCP in G;
if $c(P_G) \geqslant z$ **then**
  There is no need to expand G;
  Stop
end if
for $i \in V - \{s\}$ **do**
  $M(i) = \infty$;
  $S(i) = \infty$;
end for
$M(s) = 0$;
$S(s) = 0$;
for $i \in V$ **do**
  $R(i) = 0$;
  $p(i) = -1$;
end for
while $R(t) = 0$ **do**
  Find $i \in V$, so that $R(i) = 0$ and i has the smallest $M(i) + S(i)$;
  $R(i) = 1$;
  for all $w = (i, j) \in A^E$, so that $R(j) = 0$ **do**
    if $\max\{M(i), u(w, z)\} + S(i) + b(w, z) < M(j) + S(j)$ **then**
      $M(j) = \max\{M(i), u(w, z)\}$;
      $S(j) = S(i) + b(w, z)$;
      $p(j) = i$;
    end if
  end for
end while
Call APF (t) to calculate P^*;
$A^* = A$;
for $w \in A$ **do**
  if $w \in P^*$ **and** $z < c(w)$ **then**
    $c^*(w) = z$;
  else
    $c^*(w) = c(w)$;
  end if
end for
for $w \in P^* \cap A^a$ **do**
  $A^* = A^* \cup \{w\}$;
  $c^*(w) = z$;
end for
expanded network (solution of MCMCPEP) is $G^* = (V, A^*, s, t, c^*)$;
Total cost of expansion is $M(t) + S(t)$;
P^* is MCP in G^* with $c^*(P^*) = z$;
Output: P^* and G^*

Algorithm 2 Algorithm Path Find (APF)

Input: Node k
$P_k = \varnothing$;
$j = k$;
while $j \neq s$ **do**
  Insert the arc $(p(j), j)$ at the beginning of P_k;
  $j = p(j)$;
end while
Output: Path P_k from s to k found by AMCMCPEP

Lemma 3 Let i and j be two nodes having $R(i) = R(j) = 1$. We have

(a) If $R(i)$ is set by Algorithm 1 to the value 1 before $R(j)$ is set to the value 1, then $M(i) + S(i) \leqslant M(j) + S(j)$.

(b) If $M(i) + S(i) < M(j) + S(j)$, then $R(i)$ is set by Algorithm 1 to the value 1 before $R(j)$ is set to the value 1.

Proof The proof of (a) in Lemma 3: Let i and j be two nodes, so that $R(j)$ is the next value set by AMCMCPEP to 1, immediately after $R(i)$ becomes 1. There are two cases in the following:

Case 1: $(i, j) \in A^E$, and in the iteration when $R(i)$ becomes 1, we have

$$\max\{M(i), u((i, j), z)\} + S(i) + b((i, j), z) < M(j) + S(j) \tag{14}$$

Obviously, in this case, $M(i) + S(i) < M(j) + S(j)$.

Case 2: $(i, j) \notin A^E$, or in the iteration when $R(i)$ becomes 1, we have

$$\max\{M(i), u((i, j), z)\} + S(i) + b((i, j), z) \geqslant M(j) + S(j) \tag{15}$$

This means that the values of $M(j)$ and $S(j)$ are not changed in the iteration when $R(i)$ becomes 1, $R(j)$ is 0 when this iteration starts, and since the node i is selected having the smallest $M(i) + S(i)$, it is clear that $M(i) + S(i) \leqslant M(j) + S(j)$.

Since i and j are two nodes arbitrarily chosen for which the values of R are consecutively set to 1, and for these two nodes $M(i) + S(i) \leqslant M(j) + S(j)$, it results that for every two nodes i and j, if $R(i)$ is set by Algorithm 1 to the value 1 before $R(j)$ is set to the value 1, then $M(i) + S(i) \leqslant M(j) + S(j)$.

Proof The proof of (b) in Lemma 3: We suppose that $M(i)+S(i)<M(j)+S(j)$, and $R(j)$ is set by Algorithm 1 to the value 1 before $R(i)$ is set to the value 1. Using (a) in Lemma 3, it follows that $M(j)+S(j)\leqslant M(i)+S(i)$ (contradiction). ■

Theorem 1 If the node t is accessible from s in G^E (there is a path from s to t in G^E), and P^G is MCP in G with $c(P^G)<z$, then AMCMCPEP builds the path P^* which is the solution of MCMCPEP.

Proof We suppose that the node t is accessible from s in G^E, and P^G is MCP in G with $c(P^G)<z$. This means that the execution of AMCMCPEP enters the "while" statement.

Let $P^E=(s=i_1, i_2, \ldots, i_p=t)$ be a path from s to t of length $p-1\geqslant 0$ in G^E. We suppose that in the iterations of "while" loop $R(t)$ never becomes 1. So, its execution never ends. Obviously, in the first iteration, the node s is selected, and $R(s)$ becomes 1.

Let us consider a node i_k from P^E $(1\leqslant k\leqslant p-1)$, so that $R(i_k)=1$ (of which existence is assured by the fact that $R(s)=1$). Since (i_k,i_{k+1}) is an arc of P^E in G^E, in the iteration when $R(i_k)$ becomes 1, if $M(i_{k+1})+S(i_{k+1})=\infty$, then $M(i_{k+1})+S(i_{k+1})$ is set to a value $<\infty$. Since $M(i_{k+1})+S(i_{k+1})<\infty$, in a future iteration, the node i_{k+1} is selected, and $R(i_{k+1})$ becomes 1. By successively applying this property, we obtain that $R(i_p=t)$ is set to 1 (contradiction).

So, $R(t)$ is set to 1 by AMCMCPEP, and the execution of "while" loop ends in a finite number of iterations.

We shall prove now that the path P^* built by AMCMCPEP is the solution of MCMCPEP. Since $P^*=P_t$ is the path found by AMCMCPEP from s to t, the cost of expansion of G on P^* is $M(t)+S(t)$ (see Lemma 2).

We suppose that there is a path P in G^E, so that

$$M(t)+S(t)>\max_{w\in P} u(w,z)+\sum_{w\in P} b(w,z) \tag{16}$$

We have two cases:

Case 1: At the end of the iterations of "while" loop (when $R(t)$ becomes 1), for every node i from P, we have $R(i)=1$.

Let us denote by u^* the previous node of t in P^*. We have

$$M(t)+S(t)=\max\{M(u^*), u((u^*, t), z)\}+ S(u^*)+b((u^*,t), z) \tag{17}$$

Let us consider now the last arc (u, t) of P. Since $R(u)=1$, using Lemma 2, we have

$$M(u)=\max_{w\in P_u} u(w, z),\quad S(u)=\sum_{w\in P_u} b(w, z) \tag{18}$$

From Formula (16) and Eq. (18), it results that

$$M(t)+S(t)>\max_{w\in P_u\cup\{(u,t)\}} u(w, z)+\sum_{w\in P_u\cup\{(u,t)\}} b(w, z)=\max\{M(u), u((u, t), z)\}+S(u)+b((u, t), z) \tag{19}$$

From Formula (19) and (a) in Lemma 3, it results that $R(u)$ becames 1 before $R(t)$ becomes 1, and a lower value of $M(t)+S(t)$ could have been obtained from u than from u^*, and in this case $M(t)+S(t)$ would have been set to the value $\max\{M(u), u((u, t), z)\}+S(u)+b((u, t), z)$ or less, and not changed in the next iterations to the value of $\max\{M(u^*), u((u^*, t), z)\}+S(u^*)+b((u^*, t), z)$ (contradiction).

Case 2: At the end of Algorithm 1, there exists a node i from P, so that $R(i)=0$. Starting from s on P we denote u as the last node in P with $R(u)=1$, and v the next node in P (that has $R(v)=0$). If the sub-path from s to u denoted P'_u of P is different than the path P_u found by AMCMCPEP from s to u, and since the cost of the paths P'_u and P_u have the same costs (this can be proved in a similar way as for case 1, where instead of t, node u is considered), we replace P'_u with P_u in P resulting a new path denoted P' for which $c(P')=c(P)$, where the nodes from P and P' are the same from s to u.

Since $R(u)=1$, and using Lemma 2, we have

$$M(u)=\max_{w\in P_u} u(w, z),\quad S(u)=\sum_{w\in P_u} b(w, z) \tag{20}$$

Since $c(P')=c(P)$, from Formula (16), Eq. (20), and Lemma 2, it results that

$$M(t)+S(t)>\max_{w\in P'} u(w, z)+\sum_{w\in P'} b(w, z)\geqslant \max_{w\in P} u(w, z)+\sum_{w\in P} b(w, z)=\max\{M(u), u((u, v), z)\}+S(u)+b((u, v), z) \tag{21}$$

Since $R(v)=0$ when $R(t)$ becomes 1, we let the iterations of the "while" loop continue after $R(t)$ becomes 1 until $R(v)=1$. Then, we have

$$M(v)+S(v)\leqslant\max\{M(u), u((u, v), z)\}+ S(u)+b((u, v), z) \tag{22}$$

From Formulas (21) and (22) we obtain

$$M(t)+S\ (t) > M(v)+S(v) \tag{23}$$

Using (b) in Lemma 3, it results that for v, $R(v)$ becomes 1 before $R(t)$ becomes 1 (contradiction with the assumption that $R(v)=0$ when $R(t)$ becomes 1).

So, in both the cases a contradiction is obtained. So we have

$$M(t)+S\ (t) = \min\{\max_{w\in P} u(w,\ z)+ \sum_{w\in P} b\ (w,\ z) | P \text{ is } st\text{-path in } G^E\}.$$

Moreover, using Lemma 2, we have

$$M(t)+S\ (t) = \max_{w\in P^*} u\ (w,\ z) + \sum_{w\in P^*} b\ (w,\ z).$$

■

Theorem 2 The time complexity of AMCMCPEP is $O(n^2)$, which is strongly polynomial.

Proof Testing if t is accessible from s takes $O(m+n)$ time by applying a depth search algorithm from s.

Finding MCP in G takes $O(m\log(n))$ time[1].

At each iteration for a new node i, $R(i)$ is set to the value 1. So, the "while" loop has at most n iterations. At each iteration, a minimum is calculated in $O(n)$ time. During all iterations, every arc from $A^E = A\cup A^a$ is considered at most once. We make the notation $m^a = |A^a|$. So, the time complexity of AMCMCPEP is $O(\max\{n^2,\ m+m^a,\ m\log(n)\}) = O(\max\{n^2,\ m\log(n)\})$, since $A\cup A^a \subseteq V\times V$ and $A\cap A^a = \varnothing$. Moreover, if $m\log(n) > n^2$, then the computation of MCP in G can be avoided by a slight modification to the "while" statement. More exactly, the condition can be extended to "$R(t)=0$ and there is node i, so that $R(i)=0$ and $M(i)+S\ (i) < \infty$". If $R(t)$ remains 0 when the iterations of "while" loop end, then it means that G does not need expansion. So, with this adaptation, the initial construction of MCP in G is not needed, and the time complexity of AMCMCPEP is $O(n^2)$. ■

We shall consider now an example to illustrate how Algorithm 1 works. In Fig. 1, a network G is presented that has to be expanded, so that in the modified network there exists an MCP with the value of $z=20$. The dotted arcs are those of which capacities cannot be increased to the value of z. The arcs that can be added to the network are dashed.

We choose the following functions which are the cost of modifying the capacities of the arcs $w=(i,j)$ having the capacities less than $z=20$:

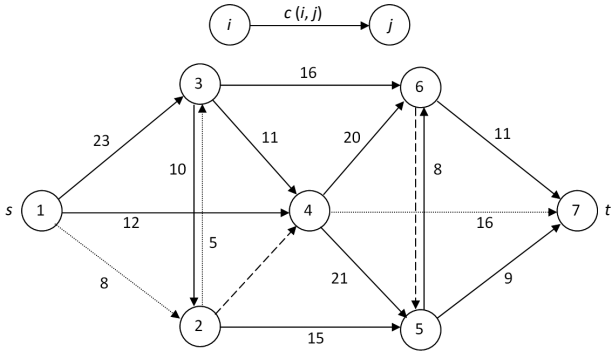

Fig. 1 Network G that has to be expanded.

$$b\ (w,x) = 10\,|j-i|\ (x-c\ (i,j)),$$
$$u\ (w,x) = |j-i|\ (x-c\ (i,j))^2.$$

Also, we consider the following functions for the cost of adding new arcs $w=(i,j)$:

$$b\ (w,x) = 4\,|j-i|\ x,$$
$$u\ (w,x) = |j-i|\ (x-10)^2.$$

Let us observe that the cost functions above are all non-decreasing monotone on the interval $[z,+\infty)$.

In Fig. 2, the network G^E is presented. Since there are paths from $s=1$ to $t=7$ (e.g., (1, 4, 5, 7)) in G^E, the problem is feasible.

Next we shall apply the iterations of Algorithm 1. First, initializations are done,

$$M = (0,\ \infty,\ \infty,\ \infty,\ \infty,\ \infty,\ \infty),$$
$$S = (0,\ \infty,\ \infty,\ \infty,\ \infty,\ \infty,\ \infty),$$
$$R = (0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0),$$
$$p = (-1,\ -1,\ -1,\ -1,\ -1,\ -1,\ -1).$$

The iterations of Algorithm 1 are reported in Table 1. At the end we have

$$M = (0,\ 100,\ 0,\ 100,\ 100,\ 108,\ 121),$$
$$S = (0,\ 100,\ 0,\ 180,\ 180,\ 180,\ 180,\ 290),$$
$$R = (1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1),$$
$$p = (-1,\ 3,\ 1,\ 2,\ 4,\ 3,\ 5).$$

In Fig. 3, the modified network G^* is presented. By applying Algorithm 2 on the vector p starting from node $k=t=7$, $P^*=(1,\ 3,\ 2,\ 4,\ 5,\ 7)$ is found, which is MCP in G^*. Of course, the path P^* has the desired capacity of $z=20$. The total cost of modification of the network is $M(7)+S(7)=411$. Let us observe that the capacities of two arcs have been modified ((3, 2) and (5, 7)), and the arc (2, 4) has been added.

3 MCP Expansion Under Budget Constraint

As mentioned in Section 1, the second problem is the

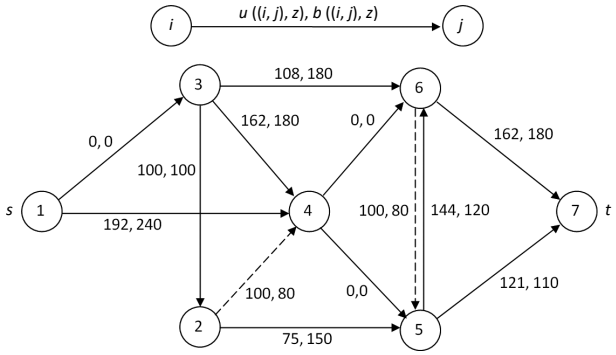

Fig. 2 Network G^E.

Table 1 Iterations of Algorithm 1

Iteration time	i	Minimum of $M(i)+S(i)$	Update		
			$M(j)$	$S(j)$	$p(j)$
1	1	0	$M(3)=0$	$S(3)=0$	$p(3)=1$
			$M(4)=192$	$S(4)=240$	$p(4)=1$
2	3	0	$M(2)=100$	$S(2)=100$	$p(2)=3$
			$M(4)=162$	$S(4)=180$	$p(4)=2$
			$M(6)=108$	$S(6)=180$	$p(6)=2$
3	2	200	$M(4)=100$	$S(4)=180$	$p(4)=3$
			$M(5)=100$	$S(5)=250$	$p(5)=2$
4	4	280	$M(5)=100$	$S(5)=180$	$p(5)=4$
5	6	288	$M(7)=162$	$S(7)=360$	$p(7)=6$
6	5	350	$M(7)=121$	$S(7)=290$	$p(7)=5$
7	7	–	–	–	–

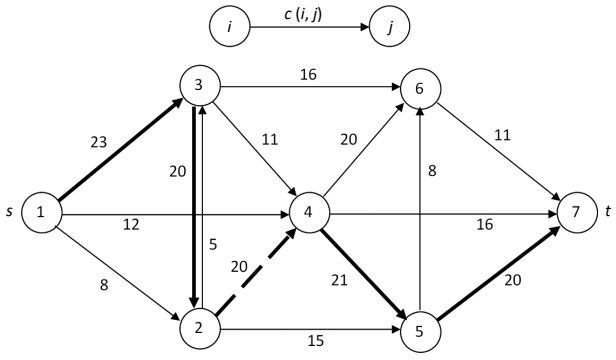

Fig. 3 Modified network G^* having MCP with the capacity of $z = 20$.

network expansion for increasing the capacity of MCP under a budget constraint. The problem consists of expanding the network, so that the capacity of an MCP in the modified network is maximized, and the total cost of expansion does not exceed a given budget. We denote this problem as MCPEBCP. Here, we similarly consider the same costs for increasing the capacities of the existing arcs and for adding new arcs from A^a as for MCMCPEP. For MCPEBCP, we introduce some constraints to the increase of the capacities of the existing arcs and to the maximum capacity allowed on any new added arcs, together with a maximum given budget denoted W for the network expansion.

The increased capacity $c'(w)$ of an arc w from A cannot exceed a given maximum $\hat{c}(w)$, i.e., $c'(w) \leqslant \hat{c}(w)$, where $\hat{c}(w) \geqslant c(w)$. If added to the network, on an arc w from A^a, the capacity cannot exceed a given maximum $\hat{c}(w)$, i.e., $c'(w) \leqslant \hat{c}(w)$, where $\hat{c}(w) > 0$.

So, MCPEBCP can be formulated as follows:

$$\begin{aligned}
&\max\ z,\\
&\max_{w\in P^*} u(w, z) + \sum_{w\in P^*} b(w, z) \leqslant W,\\
&P^* \text{ is MCP in } G^E = (V, A^E = A_z \cup A_z^a),\\
&A_z = \{w \in A | z \leqslant \hat{c}(w)\},\\
&A_z^a = \{w \in A^a | z \leqslant \hat{c}(w)\}
\end{aligned} \tag{24}$$

Let us observe that if an MCP of G has the capacity less than a given value z, and on a st-path in G^E, all the capacities of the arcs become all equal to z, then this path is an MCP in the expanded network. Consequently, MCPEBCP can be rewritten as follows:

$$\begin{aligned}
&\max\ z,\\
&\min\{\max_{w\in P} u(w, z) + \sum_{w\in P} b(w, z)\} \leqslant W,\\
&P \text{ is } st\text{-path in } G^E = (V, A^E = A_z \cup A_z^a),\\
&A_z = \{w \in A | z \leqslant \hat{c}(w)\},\\
&A_z^a = \{w \in A^a | z \leqslant \hat{c}(w)\}
\end{aligned} \tag{25}$$

For each possible value of z an MCMCPEP can be solved, and if the cost of expansion for its solution respects the budget constraint from Formula (25), then this solution is feasible for MCPEBCP. So, using Formula (9), Formula (25) can be farther rewritten as follows:

$$\begin{aligned}
&\max\ z,\\
&C \leqslant W,\\
&C \text{ is the cost of expansion of the solution of}\\
&\text{MCMCPEP for } z \text{ in } G^E = (V, A^E = A_z \cup A_z^a),\\
&A_z = \{(i, j) \in A | z \leqslant \hat{c}(i, j)\},\\
&A_z^a = \{(i, j) \in A^a | z \leqslant \hat{c}(i, j)\}.
\end{aligned} \tag{26}$$

Lemma 4 Let $z_1 > 0$ and $z_2 > 0$ be two values, so that MCMCPEP is feasible for both z_1 and z_2 in $G_1^E = (V, A_1^E = A_{z_1} \cup A_{z_1}^a)$ and $G_2^E = (V, A_2^E = A_{z_2} \cup A_{z_2}^a)$, respectively. Then, if $z_1 < z_2$, we have

$$C_1 \leqslant C_2,$$

where C_1 is the cost of expansion of G to z_1, and C_2 is the cost of expansion of G to z_2.

Proof If MCMCPEP is feasible for $z_i > 0$ in $G_i^E = (V, A_i^E = A_{z_i} \cup A_{z_i}^a)$ $(i = 1, 2)$, then,

$$C_i = \max_{w \in P_i^*} u(w, z_i) + \sum_{w \in P_i^*} b(w, z_i) \tag{27}$$

where P_i^* is the MCP found by solving MCMCPEP for z_i $(i = 1, 2)$.

If $z_1 < z_2$, then we have

$$\begin{aligned} C_1 &= \max_{w \in P_1^*} u(w, z_1) + \sum_{w \in P_1^*} b(w, z_1) \leqslant \\ &\max_{w \in P_2^*} u(w, z_1) + \sum_{w \in P_2^*} b(w, z_1) \leqslant \\ &\max_{w \in P_2^*} u(w, z_2) + \sum_{w \in P_2^*} b(w, z_2) = C_2 \end{aligned} \tag{28}$$

In the first inequality from Formula (28), we use the fact that if a path exists in G_2^E, then it exists in G_1^E, since all the arcs from G_2^E exists in G_1^E. Moreover, the cost on P_1^* is less than or equal to the cost on P_2^* in G_1^E, since P_1 is optimum in G_1^E (it has the lowest cost). The second inequality holds because $u(w, \cdot)$ and $b(w, \cdot)$ are both non-decreasing for every arc w, and $z_1 < z_2$. ■

We denote by $z_{\min}$ the capacity of MCP in G. Of course, the cost of expansion for $z = z_{\min}$ is 0.

Let us consider the network expansion of G to the maximum by setting the capacity of each arc (i, j) from $A \cup A^a$ to the maximum, i.e., $c'(i, j) = \hat{c}(i, j)$. We denote this network as $\hat{G} = (V, \hat{A} = A \cup A^a, \hat{c})$. The capacity denoted $z_{\max}$ of MCP in $\hat{G}$ is the maximum value of z for which MCPEBCP has solution by ignoring the budget constraint. We have the following result:

Theorem 3 The solution of MCPEBCP (Formula (26)) is obtained for the maximum value of z in the interval $[z_{\min}, z_{\max}]$, where the budget constraint is fulfilled.

Proof This result is immediately obtained from Lemma 4. ■

Using Theorem 3 and by considering that the values of capacities and costs are all integers, it is clear that Algorithm 3 calculates (in a divide and conquer manner) the solution of MCPEBCP.

In Algorithm 3, $\lceil \cdot \rceil$ is the ceiling operator, i.e., $\lceil x \rceil$ denotes the least integer greater than or equal to the real number x.

Theorem 4 The time complexity of Algorithm 3 is weakly polynomial,

$$O(\max\{(m + m^a)\log(n), n^2 \log(z_{\max} - z_{\min})\}),$$

where $z_{\min}$ and $z_{\max}$ are the initial values calculated by Algorithm 3, and $m^a = |A^a|$.

Proof MCP can be calculated in $O(m \log(n))$ time for G and in $O((m + m^a)\log(n))$ time for $\hat{G}$.

The "while" loop has $O(\log(z_{\max} - z_{\min}))$ iterations, and for each iteration AMCMCPEP is executed once in $O(n^2)$ time (see Theorem 2).

So, the time complexity of AMCPEBCP is $O(\max\{(m + m^a)\log(n), n^2 \log(z_{\max} - z_{\min})\})$, which is weakly polynomial. ■

Let us observe that since $m + m^a \leqslant n^2$ and $\log(n)$ is negligible, in practice, it is very likely that $(m + m^a)\log(n) \leqslant n^2$, and so, the time complexity of AMCPEBCP becomes $O(n^2 \log(z_{\max} - z_{\min}))$.

3.1 Linear-cost case

Algorithm 3 solves the second problem (Formula (25)) in weakly polynomial time. In the special case that the cost functions are linear, we shall present next an algorithm to solve the problem in strongly polynomial time.

MCPEBCP under linear costs was investigated in

Algorithm 3 (AMCPEBCP) **Algorithm to solve MCPEBCP**

1: **Input:** G and W
2: Compute $z_{\min}$ = capacity of MCP in G;
3: Build network $\hat{G} = (V, \hat{A} = A \cup A^a, \hat{c})$;
4: Compute $z_{\max}$ = capacity of MCP in $\hat{G}$;
5: **if** $z_{\min} = z_{\max}$ **then**
6: Network G cannot be expanded;
7: **stop**;
8: **end if**
9: **while** $z_{\min} < z_{\max}$ **do**
10: $z = \lceil (z_{\min} + z_{\max})/2 \rceil$;
11: Calculate A_z and A_z^a for z;
12: Call AMCMCPEP (G, A_z, A_z^a, z) to obtain P^* and $G^* = (V, A^*, s, t, c^*)$;
13: **if** G^* satisfies the budget constraint on P^* **then**
14: $z_{\min} = z$;
15: **else**
16: $z_{\max} = z$;
17: **end if**
18: **end while**
19: **Output:** Last computed network G^*

Ref. [7]. There, a hybrid Secant-Newton algorithm is proposed to solve the problem in $O(A\,n^2\log^2(n))$, where A is the time complexity of solving MCMCPEP under linear cost functions (see Theorem 4.3 from Ref. [7] and Theorem 4 from Ref. [11]). So, based on Theorem 2, it follows that its complexity is $O(n^4\log^2(n))$. Now, we use the concept of Meggido's parametric search to present an algorithm for solving the problem in $O(n^4)$ time[13]. Hence, our proposed algorithm has a better time complexity than the algorithm given in Ref. [7].

In Formula (25), in the linear case, $u(w, z)=0$ and $b(w, z)=r_w(z-c(w))$, where r_w is the cost of increasing the capacity of the arc $w\in A_z$ by one unit. Without loss of the generality, we hereafter suppose that every $w\in A^a$ is an arc belonging to A_z with $c(w)=0$. To solve MCPEBCP under linear costs, we propose an algorithm that has two phases:

Phase I: This phase finds an interval containing the optimal value. The total cost is a convex piecewise-linear function in terms of the optimal value.

Phase II: This phase uses the idea of Megiddo's parametric search to exactly find the optimal value.

Let us explain both the phases in complete details. Suppose we have arranged the distinct values of the set,

$$\bigcup_{w\in A}\{c(w)\}\cup\bigcup_{w\in A^z}\{\hat{c}(w)\}\cup\{z_{\max}\},$$

in which $z_{\max}$ is computed the same as for Algorithm 3. Let $z_1<z_2<\cdots<z_l$ be the sorted list. Obviously, $l\leqslant O(m)$. The first phase applies a binary search to find an index $k=1, 2, \ldots, l-1$, so that the minimum cost of MCMCPEP for z_k is less than or equal to W while that of MCMCPEP for z_{k+1} is greater than W. For this purpose, Algorithm 1 is performed in every iteration to determine the minimum cost of MCMCPEP for some value z_k. This phase implies that the optimal value belongs to the half-closed interval $[z_k, z_{k+1})$. This fact determines the set of arcs which are allowed to be modified linearly on this interval. Trivially, this set is

$$\bar{A}=\{w\in A_z: c-(w)\leqslant z_k \text{ and } \hat{c}-(w)\geqslant z_{k+1}\} \tag{29}$$

So an arc $w\notin\bar{A}$ has a fixed value in the second phase, while the capacity of an arc $w\in\bar{A}$ can be modified by imposing the linear cost $r_w-(z-c(w))$.

It is easy to see that in the second phase we have to find a value $z^*\in[z_k, z_{k+1})$ to satisfy the budget constraint in the equality form. For this purpose, we define a cost vector $\bar{r}$ as

$$\bar{r}_w=\begin{cases}r_w, & w\in\bar{A};\\ 0, & w\in A-\bar{A}.\end{cases}$$

So, the problem is reduced to finding a value z, so that the shortest path length (minimum cost) with respect to $\bar{r}_w(z-c(w))$ is equal to W. Let us introduce the function

$$\mathrm{tc}(z)=\sum_{w\in P^*}b(w, z)=\sum_{w\in P^*}\bar{r}_w(z-c(w)).$$

At the interval $[z_k, z_{k+1})$, in which P^* is the shortest path with respect to $\bar{r}_w(z-c(w))$. Obviously, $\mathrm{tc}(z)$ is the same total cost function in terms of the optimal value z. By noting the sensitivity analysis for shortest path problems, it is easy to prove the following result.

Lemma 5 $\mathrm{tc}(z)$ is an increasing convex piecewise-linear function at $[z_k, z_{k+1})$.

To find the exact optimal value z, we can apply Megiddo's parametric search. Its vital concept is to use Algorithm 1 without knowing the exact value of z. So, z is assumed to be considered as a parameter whose value is not explicitly given in Algorithm 1. The only challenge is how to evaluate the accuracy of inequality,

$$S(i)+b(w, z)<S(j) \tag{30}$$

Because we update $S(j)=S(i)+b(w, z)$ if the inequality holds, and we do not change it, otherwise. Notice that $S(i)$, $i\in V$, is not a number, but it is a linear function of z. So, $S(j)-S(i)-b(w, z)$ is also a linear function as $u-vz$. So, the Inequality (30) is converted into $u/v>z$. Hence, we can run Algorithm 1 for u/v. Based on Lemma 5, if the total cost $\mathrm{tc}(u/v)$ is greater than W, {it results} that $u/v>z$ (see Fig. 4). A formal description of our proposed algorithm is stated in Algorithm 4.

Theorem 5 The worst-case time complexity of Algorithm 4 is $O(n^4)$.

Proof The complexity of the first phase is $O(n^2\log(m))=O(n^2\log(n))$ since AMCMCPEP runs in $O(n^2)$ and $l\leqslant O(m)$. The complexity analysis of the second phase is $O(n^4)$. So, the whole Algorithm 4 has a time complexity of $O(n^4)$. ■

3.2 Example

Let us now provide an example to illustrate the process of Algorithm 4. Figure 5 depicts an acyclic instance of the problem with linear costs, in which the maximum amount of increased capacities is supposed to be 15 and $W=24$. It is assumed that $s=1$ and $t=5$.

Algorithm 4 Algorithm to solve MCPEBCP under linear cost functions

1: **Input:** G and W
2: **Phase I:**
3: Run Lines 2–8 in Algorithm 3;
4: Let $z_1 < z_2 < \cdots < z_l$ be the sorted list of $\bigcup_{w\in A}\{c(w)\}\cup\bigcup_{w\in A^z}\{\hat{c}(w)\}\cup\{z_{\max}\}$;
5: Obtain $\bar{A}$ using Eq. (29);
6: $L=1$ and $U=l$;
7: **while** $U-L>1$ **do**
8: $k=\left\lfloor\frac{L+U}{2}\right\rfloor$;
9: Calculate A_z and A_z^a for z;
10: Call AMCMCPEP (G, A_z, z) to obtain P^*;
11: **if** G^* satisfies the budget constraint on P^* **then**
12: $L=k$;
13: **else**
14: $U=k$;
15: **end if**
16: **end while**
17: $k=L$;
18: **Phase II:**
19: **for** $i\in V-\{s\}$ **do**
20: $S(i)=\infty$;
21: **end for**
22: $S(s)=0$;
23: **for** $i\in V$ **do**
24: $R(i)=0$;
25: $p(i)=-1$;
26: **end for**
27: **while** $R(t)=0$ **do**
28: Find $i\in V$ so that $R(i)=0$ and i has the smallest $S(i)$;
29: $R(i)=1$;
30: **for** all $w=(i,j)\in A^E$ so that $R(j)=0$ **do**
31: **if** $S(j)\neq\infty$ **then**
32: Let $S(j)$ be in the form $u-vz$;
33: **if** $v\neq 0$ **then**
34: Call AMCMCPEP $(G, A_z, u/v)$ to obtain total cost tc (u/v);
35: **end if**
36: **end if**
37: **if** tc $(u/v)>W$ or $v=0$ or $S(j)=\infty$ **then**
38: $S(j)=S(i)+b(w,z)$;
39: $p(j)=i$;
40: **end if**
41: **end for**
42: **end while**
43: Run Lines 33–44 of Algorithm 1;
44: Expanded network is $G^*=(V, A^*, s, t, c^*)$;
45: **Output:** Network G^*

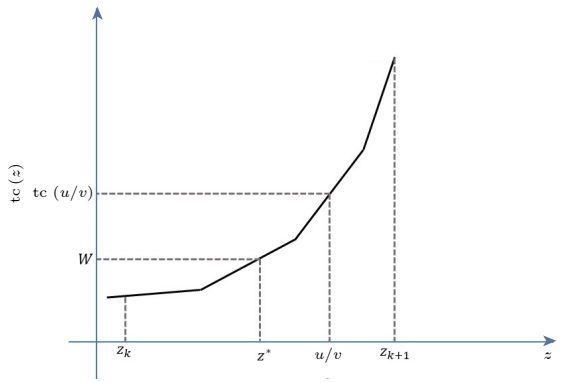

Fig. 4 Graph of tc (z).

Algorithm 4 begins Phase I by running Line 3. So, we have $z_{\min}=10$ and $z_{\max}=15$. Since $z_{\min}\neq z_{\max}$, Algorithm 4 continues the steps of Phase I. The sorted list of Line 4 is $\{6, 8, 10, 12, 14, 15\}$. So, $L=1$ and $U=6$. Table 2 shows the iterations of Phase I. In the end of Phase I, we have $k=4$ and $\bar{A}=A\backslash\{(3, 5)\}$. Table 3 shows the iterations of Phase II. Since $S(t)=9z-100$ is finally equal to 24, it follows that $z^*=13.777\,77$ is the optimal value, and the optimal path is $1-2-3-5$.

4 Conclusion

This paper introduces two problems which concern

Table 2 Phase I in Algorithm 4 of the presented example.

L	U	k	z_k	P^*	Total cost
1	6	3	10	$1-3-5$	$0\ (\leqslant 24)$
3	6	4	12	$1-2-3-5$	$8\ (\leqslant 24)$
4	6	5	14	$1-2-3-5$	$26\ (>24)$
4	5	–	–	–	–

Table 3 Phase II in Algorithm 4 of the presented example.

Iteration time	Node	1	2	3	4	5
1	$S(\cdot)$	0	∞	∞	∞	∞
	$p(\cdot)$	-1	-1	-1	-1	-1
	$R(\cdot)$	1	0	0	0	0
2	$S(\cdot)$	0	$5z-60$	$8z-80$	$4z-32$	∞
	$p(\cdot)$	-1	1	1	1	-1
	$R(\cdot)$	1	1	0	0	0
3	$S(\cdot)$	0	$5z-60$	$9z-100$	$4z-32$	$13z-124$
	$p(\cdot)$	-1	1	2	1	2
	$R(\cdot)$	1	1	1	0	0
4	$S(\cdot)$	0	$5z-60$	$9z-100$	$4z-32$	$9z-100$
	$p(\cdot)$	-1	1	2	1	3
	$R(\cdot)$	1	1	1	1	0
5	$S(\cdot)$	0	$5z-60$	$9z-100$	$4z-32$	$9z-100$
	$p(\cdot)$	-1	1	2	1	3
	$R(\cdot)$	1	1	1	1	1

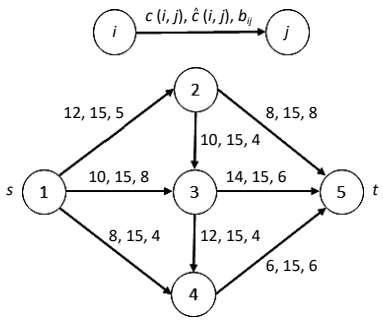

Fig. 5 Instance of the problem with linear costs and W = 24.

MCP expansion. The admissible actions are to increase capacity on existing arcs and to add new arcs. These problems have real applications, e.g., for increasing the bandwidth between two routers in Internet, or road widening between two cities. The first problem expands the network to a desired capacity of MCP by minimizing the cost. The total cost is a combination of max-type and sum-type costs. The only condition applied to the cost function on an arc is to be non-decreasing monotone, which is really non-restrictive from practical viewpoint. Most network problems consider costs as linear functions or fixed, which generally does not reflect reality. In this paper, for the first problem, a quadratic-time algorithm is developed. The second problem consists of increasing the capacity of MCP as much as possible, so that the sum of costs does not exceed a given budget. A weakly polynomial method is presented to solve this problem. Finally, a strongly polynomial-time algorithm is designed to solve the problem with linear costs. This algorithm has a better complexity than the one presented in Ref. [7].

As future extension of this work, the more general cases of one source to multiple sinks, multiple sources to one sink, or multiple sources to multiple sink nodes can be studied.

References

[1] H. N. Gabow and R. E. Tarjan, Algorithms for two bottleneck optimization problems, *J. Algorithm.*, vol. 9, no. 3, pp. 411–417, 1988.

[2] N. Shacham, Multicast routing of hierarchical data, in *Proc. Discovering a New World of Communications*, Chicago, IL, USA, 1992, pp. 1217–1221.

[3] M. Schulze, A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method, *Soc. Choice Welfare*, vol. 36, no. 2, pp. 267–303, 2011.

[4] E. Fernandez, R. Garfinkel, and R. Arbiol, Mosaicking of aerial photographic maps via seams defined by bottleneck shortest paths, *Oper. Res.*, vol. 46, no. 3, pp. 293–304, 1998.

[5] E. Ullah, K. Lee, and S. Hassoun, An algorithm for identifying dominant-edge metabolic pathways, in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design-Digest of Technical Papers*, San Jose, CA, USA, 2009, pp. 144–150.

[6] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*: *Theory*, *Algorithms and Applications*. Englewood Cliffss, NJ, USA: Prentice Hall, 1993.

[7] J. Tayyebi and A. Deaconu, Expanding maximum capacity path under weighted sum-type distances, *AIMS Math.*, vol. 6, no. 4, pp. 3996–4010, 2021.

[8] A. M. Deaconu and J. Tayyebi, Inverse maximum capacity path problems under sum-type and max- type distances and their practical application to transportation networks, *IEEE Access*, vol. 8, pp. 225957–225966, 2020.

[9] A. M. Deaconu and L. Majercsik, Flow increment through network expansion, *Mathematics*, vol. 9, no. 18, p. 2308, 2021.

[10] Y. Hu, X. Zhao, J. Liu, B. Liang, and C. Ma, An efficient algorithm for solving minimum cost flow problem with complementarity slack conditions, *Math. Probl. Eng.*, vol. 2020, p. 2439265, 2020.

[11] J. Zhang and Z. Liu, An oracle strongly polynomial algorithm for bottleneck expansion problems, *Optim. Methods Softw.*, vol. 17, no. 1, pp. 61–75, 2002.

[12] S. Wang, Q. Meng, and H. Yang, Global optimization methods for the discrete network design problem, *Transp. Res. Part B*: *Meth.*, vol. 50, pp. 42–60, 2013.

[13] N. Megiddo, Combinatorial optimization with rational objective functions, *Math. Oper. Res.*, vol. 4, no. 4, pp. 414–424, 1979.

Adrian M. Deaconu received the BS, MS, and PhD degrees in computer science from Transilvania University of Brasov, Romania in 1997, 1998, and 2008, respectively, Starting from 2009, he is an associate professor at Department of Mathematics and Computer Science, Transilvania University of Brasov, Romania. From 2019 to 2020, he was a visiting professor at University College Cork, Ireland. In 2023 he obtained his habilitation in computer science. Since 2018 he has been the coordinator of the research group "Optimization algorithms" inside Transilvania University of Brasov. He is a member of the Romanian Mathematical Society, and a member of the Mathematical Modeling and Software Products Research Centre from the Research Development Institute, Brasov. He is the author of four books, five electronic courses, three book chapters, and more than 50 research articles. His research interests include algorithms, graphs, optimization, inverse network optimization, and heuristics.

Javad Tayyebi received the BS degree in mathematics from University of Birjand, Iran in 2007, the MS degree in applied mathematics from Sharif University of Technology, Iran in 2009, and the PhD degree from University of Birjand, Iran in 2014. The subject of his thesis was "inverse optimization". He is a member of the Iranian Operations Research Society. From 2022 to now, he is an associate professor at Department of Industrial Engineering, Birjand University of Technology, Iran. He is the author of more than 40 articles. His research interests include network optimization, game theory, and fuzzy theory.