Optim Lett (2017) 11:1319–1339
DOI 10.1007/s11590-016-1073-x

ORIGINAL PAPER

The energy-constrained quickest path problem

Herminia I. Calvete[1] · **Lourdes del-Pozo**[2] · **José A. Iranzo**[1]

Received: 20 May 2015 / Accepted: 25 August 2016 / Published online: 8 September 2016

Abstract This paper addresses a variant of the quickest path problem in which each arc has an additional parameter associated to it representing the energy consumed during the transmission along the arc while each node is endowed with a limited power to transmit messages. The aim of the energy-constrained quickest path problem is to obtain a quickest path whose nodes are able to support the transmission of a message of a known size. After introducing the problem and proving the main theoretical results, a polynomial algorithm is proposed to solve the problem based on computing shortest paths in a sequence of subnetworks of the original network. In the second part of the paper, the bi-objective variant of this problem is considered in which the objectives are the transmission time and the total energy used. An exact algorithm is proposed to find a complete set of efficient paths. The computational experiments carried out show the performance of both algorithms.

Keywords Quickest path · Energy constraint · Bi-objective optimization

✉ Herminia I. Calvete
herminia@unizar.es

Lourdes del-Pozo
lpozo@unizar.es

José A. Iranzo
joseani@unizar.es

1 Dpto. de Métodos Estadísticos, IUMA, Universidad de Zaragoza, Pedro Cerbuna 12, 50009 Zaragoza, Spain

2 Dpto. de Métodos Estadísticos, Universidad de Zaragoza, Violante de Hungría 23, 50009 Zaragoza, Spain

1 Introduction

The quickest path problem (QPP) is a path problem in a directed network which aims to minimize the time taken to transmit a given amount of data. The transmission time depends on two parameters, an additive function which represents the traversal time or the delay along the path and a bottleneck function which represents the path capacity.

Let $\mathcal{G} = [\mathcal{N}, \mathcal{A}]$ be a directed network without multiple arcs and self loops, where $\mathcal{N}$ denotes the set of nodes and $\mathcal{A}$ the set of directed arcs. Let n be the number of nodes and m the number of arcs. Let s and t be two distinguished nodes in the network called, respectively, origin and destination and σ the data units to be sent from node s to node t. Each arc $(u, v) \in \mathcal{A}$ is endowed with a capacity $c(u, v) > 0$ and a delay time $l(u, v) > 0$. The capacity represents the amount of data that can be sent through arc (u, v) per time unit. The delay time is the time required for the data units to traverse the arc (u, v).

Let us assume that a message is transmitted as a continuous stream along the arc (u, v) at a constant flow rate $\rho \leqslant c(u, v)$. At this flow rate, a message of σ data units is sent from node u to node v through arc (u, v) in $l(u, v) + \frac{\sigma}{\rho}$ time. This expression takes its minimum value when $\rho = c(u, v)$. Thus, the minimum required transmission time is $l(u, v) + \frac{\sigma}{c(u,v)}$.

A simple path or loopless path P from node s to node t is a sequence of nodes and arcs $P = (s = u_1, u_2, \ldots, u_k = t)$ such that $u_i \in \mathcal{N}, i = 1, \ldots, k, u_i \neq u_j$ if $i \neq j$, and $(u_i, u_{i+1}) \in \mathcal{A}, i = 1, \ldots, k-1$. In the paper, we use the term path in place of simple or loopless path for short as well as the term $s - t$ path in place of a path from s to t. We assume that the set of $s - t$ paths in the network $\mathcal{G}$ is nonempty.

The delay experienced by a message sent via path P depends on the message forwarding mechanism used at the intermediate nodes [18]. If σ data units are sent at a constant rate from s to t along the $s - t$ path P with no buffering at intermediate nodes (circuit switching mode), the minimum transmission time or end-to-end delay of path P is

$$T_\sigma(P) = l(P) + \frac{\sigma}{c(P)} \tag{1}$$

where $l(P) = \sum_{i=1}^{k-1} l(u_i, u_{i+1})$ denotes the delay time of path P and $c(P) = \min_{i=1,\ldots,k-1} c(u_i, u_{i+1})$ denotes its capacity.

Hence, the QPP can be formulated as finding an $s - t$ path so that:

$$\begin{aligned} &\min_P \; T_\sigma(P) \\ &\text{s.t.} \quad P \text{ is an s-t path in the network } \mathcal{G} \end{aligned} \tag{2}$$

A characteristic of the QPP is that, in general, the size of the message has a strong influence on the optimal path. When σ is small with respect to the arc capacities, the transmission time is controlled by the arc delays and a shortest path with respect to the arc delay could be a good solution to the problem. However, when σ is very large, the transmission time is controlled by the arc capacities and the problem could be approached by computing the shortest path with respect to the arc delay among all paths with the largest capacity. It is also worth mentioning that the QPP does not satisfy the property known as ‘the optimality principle’, i.e. an $s' - t'$ subpath of an optimal $s - t$ path is not necessarily an $s' - t'$ optimal path.

The QPP was first proposed by Moore [11] to model flows of convoy-type traffic. Then it was proposed by Chen and Chin [5] in the context of modeling transmission problems in communication networks where nodes represent transmitters/receivers without data memories and arcs represent communication channels. Clímaco et al. [6] applied the model to the routing of data packets in Internet networks. Hamacher and Tijandra [8] proposed the QPP for a special evacuation problem where evacuees may use only a single path or tunnel from their initial position. Martins and Santos [10] and Pelegrín and Fernández [16] approached the QPP as a special minsum-maxmin bi-objective path problem. They proved that any optimal solution of the QPP is a nondominated solution of the bi-objective problem in which the delay time is minimized and the capacity of the path is maximized. Hence, a quickest path can be obtained by solving this bi-objective problem and selecting a nondominated path with the minimum transmission time.

Several polynomial time algorithms have been proposed in the literature, all with the same time complexity. They are based on solving a shortest path problem in an enlarged network [2,5], solving a sequence of shortest path problems with respect to the delay time on networks where the minimum capacity increases [10,11,16,19], using a label-setting algorithm [12] or taking into account that a quickest path is a supported efficient solution of the aforementioned bi-objective problem [20].

Several variants and extensions of the QPP have been addressed in the literature. The problem of finding the first K quickest paths in nondecreasing order of transmission time has been analyzed in [4,6,13,15,19]. The QPP constrained to contain a given subpath has been studied in [3,19]. The problem of determining the transmission process when data are transmitted in batches of variable size but with required limits has been considered in [1]. The problem of computing the quickest path whose reliability is not lower than a given threshold has been analyzed in [2]. Pascoal et al. [14] provide a survey on the subject.

When formulating the QPP, no attention is paid to the characteristics of the transmitters/receivers represented by the nodes. It is implicitly assumed that they have unlimited energy available for transmitting messages. Usually, this can be the case in wired networks. However, for mixed networks which combine wired and radio link connections, some of the nodes can have limited power to transmit messages. This available power must be taken into account when computing the QPP since the energy consumed at node u during the transmission of the message along the arc (u, v) depends on the units of time during which node u is active, i.e. while it is sending data. Hence, it depends on the rate at which data are transmitted. In this paper, we introduce the energy-constrained quickest path problem (EQPP) which aims to obtain a quickest path whose nodes are able to support the transmission of σ data units. We formulate the problem and develop a polynomial time algorithm to solve it based on computing shortest paths with respect to the delay time in a sequence of subnetworks of the original network. Although it is a constrained QPP, the time complexity of the algorithm is the same as that of any of the algorithms developed for solving the QPP. In the second part of the paper we address the minsum–minsum bi-objective variant of this problem (BEQPP) in which the total transmission time and the total consumed energy are minimized. We approach this NP-hard problem by determining a complete set of efficient paths and develop an exact algorithm based on solving bi-objective shortest path problems. The paper is structured as follows. Sections 2 and 3 formally set out the EQPP and prove the main theoretical results which support the algorithm developed for solving it. Section 4 goes on to develop a polynomial algorithm to solve the EQPP and shows its computational complexity. In Sect. 5 the bi-objective EQPP is formulated as well as its properties are proved and the algorithm is developed to find a complete set of efficient paths. Section 6 displays the results of the computational experiment carried out to assess the performance of the proposed algorithms. Finally, our conclusions are presented in Sect. 7.

2 The energy-constrained quickest path problem

Let $\mathcal{G} = [\mathcal{N}, \mathcal{A}]$ be the directed network introduced in Sect. 1. In order to formulate the EQPP, we assume that each arc $(u, v) \in \mathcal{A}$ is endowed with an energy rate $\omega(u, v) > 0$ which measures the energy required at node u to transmit data units along the arc (u, v) per time unit. This energy typically depends on the characteristics of the arc (delay time and capacity).

Each node $u \in \mathcal{N}$ is endowed with a power b_u which represents the limited energy available for transmission at node u. If σ data units are transmitted as a continuous stream from node u to node v along the arc (u, v) at a constant flow rate ρ, then the node u is active, i.e. sending data, during $\frac{\sigma}{\rho}$ time units. Hence the required energy at node u is $\omega(u, v)\,\frac{\sigma}{\rho}$. Without loss of generality, we assume that

$$\omega(u, v)\frac{\sigma}{c(u, v)} \leqslant b_u, \forall (u, v) \in \mathcal{A} \tag{3}$$

If the arc (u, v) does not hold this condition it cannot support the transmission of the σ data units and so can be removed.

Taking into account the message forwarding mechanism, σ data units are sent along the $s-t$ path P at a constant rate $c(P)$. Therefore, the total energy required to transmit σ data units using the path P is:

$$E_\sigma(P) = \sum_{i=1}^{k-1} \omega(u_i, u_{i+1})\,\frac{\sigma}{c(P)} \tag{4}$$

Let us denote $W(P) = \sum_{i=1}^{k-1} \omega(u_i, u_{i+1})$.

The residual energy $b_u(\sigma, P)$ at node u after transmitting σ data units through the path P is

$$b_u(\sigma, P) = \begin{cases} b_u - \omega(u_i, u_{i+1})\frac{\sigma}{c(P)} & \text{if u = u}_\text{i}\text{, i = 1, \dots, k - 1} \\ b_u & \text{otherwise} \end{cases}$$

In order for P to be an $s - t$ feasible path, $b_u(\sigma, P) \geqslant 0$, $\forall u \in P$. That is to say, the feasibility of a path P is measured through the availability of its nodes to transmit the whole data units at a rate $c(P)$. Hence, the energy-constrained quickest path problem (EQPP) can be formulated as finding an $s - t$ path so that:

$$\begin{aligned} &\min_P \; T_\sigma(P) \\ &\text{s.t.} \;\; b_u(\sigma, P) \geqslant 0, u \in \mathcal{N} \\ &\qquad P \text{ is an s-t path in the network } \mathcal{G} \end{aligned} \tag{5}$$

The special characteristics of the side constraint on the residual energy allow us to develop an algorithm which is based on successively solving shortest path problems in subnetworks of the original network $\mathcal{G}$ which guarantee the feasibility of the path with respect to the energy availability at the nodes.

3 Main theoretical results

In what follows, we assume without loss of generality that there are r different capacities $c_1 < c_2 < \cdots < c_r$ in the network $\mathcal{G}$. Let us assign to each arc $(u, v) \in \mathcal{A}$ the label

$$c^{\min}(u, v) = \min_{i=1,\ldots,r} \left\{ c_i : b_u - \omega(u, v) \frac{\sigma}{c_i} \geqslant 0 \right\}$$

This label provides the minimum capacity at which the node u is able to support the transmission of the σ data units along the arc (u, v). Therefore, it gives an idea of the feasible paths in which this arc can be included. Note that (u, v) can be an arc of an $s - t$ feasible path P only if $c(P) \geqslant c^{\min}(u, v)$.

We define $\mathcal{G}_j = [\mathcal{N}, \mathcal{A}_j]$, $j = 1, \ldots, r$, to be a subnetwork of $\mathcal{G}$ where $(u, v) \in \mathcal{A}_j$ if and only if $(u, v) \in \mathcal{A}$, $c(u, v) \geqslant c_j$ and $c^{\min}(u, v) \leqslant c_j$.

It is worth mentioning that, in general, the network $\mathcal{G}_{j+1}$ is not a subnetwork of $\mathcal{G}_j$ and so the number of arcs in the successive networks does not necessarily decrease. For illustration, Fig. 1 displays a network $\mathcal{G}$ with the capacities 1, 2, 3 and 4, and the associated networks $\mathcal{G}_j$.

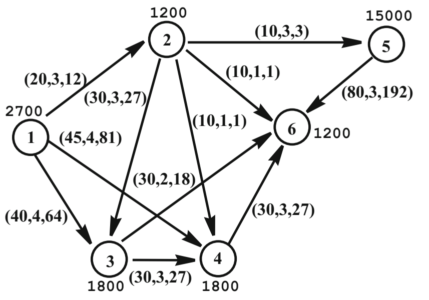
Lemma 1 *Let $P = (s = u_1, u_2, \ldots, u_k = t)$ be an $s - t$ path in the network $\mathcal{G}_j$. Then, P is an $s - t$ feasible path for the EQPP.*

Proof If P is an $s - t$ path in the network $\mathcal{G}_j$, then $c(P) \geqslant c_j \geqslant c^{\min}(u_i, u_{i+1})$, $i = 1, \ldots, k - 1$. Hence

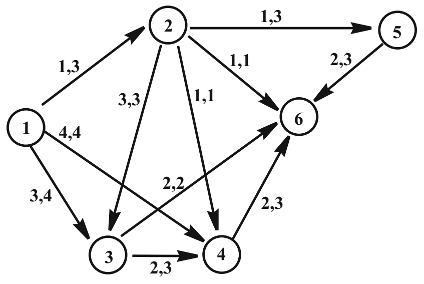
$$b_{u_i}(\sigma, P) = b_{u_i} - \omega(u_i, u_{i+1}) \frac{\sigma}{c(P)} \geqslant b_{u_i} - \omega(u_i, u_{i+1}) \frac{\sigma}{c^{\min}(u_i, u_{i+1})} \geqslant 0$$

For the remaining nodes u of the network, b_u is not modified when the σ data units are transmitted. Therefore, $b_u(\sigma, P) \geqslant 0$ $\forall u \in \mathcal{N}$ and the result follows. □

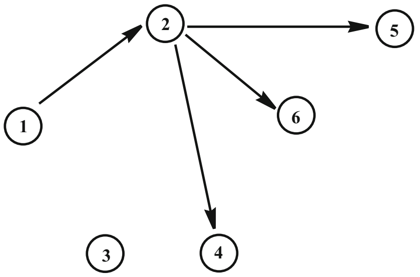
Lemma 2 *Let $P = (s = u_1, u_2, \ldots, u_k = t)$ be an $s - t$ feasible path for the EQPP with capacity $c(P) = c_j$. Then, P is an $s - t$ path in the network $\mathcal{G}_j$.*



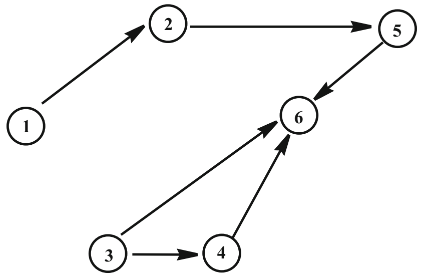
(a) Network $\mathcal{G}$. On the arcs, $(l(u,v), c(u,v), w(u,v))$. On the nodes b_u. Data units $\sigma = 120$.



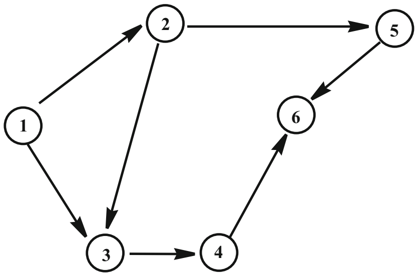
(b) On the arcs, $c^{\min}(u,v), c(u,v)$



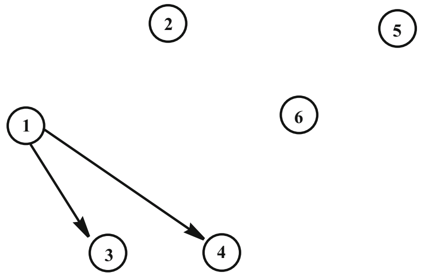
(c) Network $\mathcal{G}_1$



(d) Network $\mathcal{G}_2$



(e) Network $\mathcal{G}_3$



(f) Network $\mathcal{G}_4$

Fig. 1 Networks $\mathcal{G}$ and $\mathcal{G}_j$, $j = 1, \ldots, 4$

Proof Since P is feasible

$$b_{u_i}(\sigma, P) = b_{u_i} - \omega(u_i, u_{i+1}) \frac{\sigma}{c(P)} \geqslant 0, i = 1, \ldots, k-1$$

Hence, $c^{\min}(u_i, u_{i+1}) \leqslant c(P) = c_j \leqslant c(u_i, u_{i+1})$, $i = 1, \ldots, k-1$. Taking into account the definition of the network $\mathcal{G}_j$, we conclude that the arc (u_i, u_{i+1}), $i = 1, \ldots, k-1$, is in $\mathcal{A}_j$, and so P is an $s - t$ path in $\mathcal{G}_j$. $\square$

It is worth pointing out that an $s-t$ feasible path P for the EQPP with capacity $c(P) > c_j$ is not necessarily an $s-t$ path in the network $\mathcal{G}_j$. For instance, in the example displayed in Fig. 1, the capacity of the path $1-3-4-6$ is equal to 3. However, this path is neither in network $\mathcal{G}_1$ nor in network $\mathcal{G}_2$. In other words, the network $\mathcal{G}_j$ contains the paths $P = (s = u_1, u_2, \ldots, u_k = t)$ of $\mathcal{G}$ which are feasible for the EQPP with capacity greater than or equal to c_j, for which

$$b_{u_i}(\sigma, P) = b_{u_i} - \omega(u_i, u_{i+1})\frac{\sigma}{c_j} \geqslant 0, i = 1, \ldots, k-1$$

i.e., whose nodes are able to support the transmission with capacity c_j. In particular, the network $\mathcal{G}_j$ contains all the $s-t$ feasible paths for the EQPP with capacity c_j. Hence, if there is no $s-t$ path in the network $\mathcal{G}_j$, then there will not be an optimal solution of the EQPP with capacity c_j.

Let us consider the following SPP with respect to the delay time in $\mathcal{G}_j$:

$$\begin{aligned} \mathrm{SPP_j}: \quad & \min_{P} \ l(P) \\ & \text{s.t.} \quad P \text{ is an s} - \text{t path in the network } \mathcal{G}_\mathrm{j} \end{aligned} \tag{6}$$

Lemma 3 *Let P be an optimal solution of the* $\mathrm{SPP_j}$ *and $c(P) = c_h > c_j$. Then, there is no optimal solution of the EQPP with capacity c_j.*

Proof Let Q be an $s-t$ feasible path for the EQPP with capacity c_j. Then Q is a path in $\mathcal{G}_j$ and

$$T_\sigma(P) = l(P) + \frac{\sigma}{c_h} < l(Q) + \frac{\sigma}{c_j} = T_\sigma(Q)$$

Thus, Q cannot be an optimal solution of the EQPP. □

Next we prove that any optimal solution to the EQPP can be obtained as a shortest path with respect to the delay time.

Theorem 1 *Let P^* be an optimal solution of the EQPP and $c(P^*) = c_h$. Then, P^* is an optimal solution of the* $\mathrm{SPP_h}$ *and any optimal solution of the* $\mathrm{SPP_h}$ *is an optimal solution of the EQPP.*

Proof Since P^* is an $s-t$ feasible path for the EQPP with capacity c_h, then P^* is an $s-t$ path in $\mathcal{G}_h$. Let Q be an $s-t$ path in the network $\mathcal{G}_h$. Thus, $c(Q) \geqslant c_h$. If $l(Q) < l(P^*)$, then

$$T_\sigma(Q) = l(Q) + \frac{\sigma}{c(Q)} < l(P^*) + \frac{\sigma}{c_h} = T_\sigma(P^*)$$

which contradicts the optimality of P^*. Furthermore, by applying Lemma 3, the capacity of any $s-t$ shortest path $\widetilde{P}$ in $\mathcal{G}_h$ is $c(\widetilde{P}) = c_h$. Hence, $\widetilde{P}$ is an $s-t$ feasible path for the EQPP such that $T(\widetilde{P}) = T(P^*)$ and so is an optimal solution of the EQPP. □

4 EQPA: an algorithm for solving the EQPP

As a consequence of Theorem 1, the optimal solutions of the EQPP can be obtained by computing shortest paths with respect to the delay time in the networks $\mathcal{G}_j$ and determining those which have minimum transmission time.

```
The algorithm EQPA
Step 0.
    Set $j = 1$
Step 1.
    Solve the SPP$_j$.
    If there is no $s - t$ shortest path in $\mathcal{G}_j$ with capacity $c_j$, go to Step 2.
    Otherwise, let $P_j$ be an optimal solution of the SPP$_j$ with $c(P_j) = c_j$.
Step 2.
    If $j = r$, go to Step 3. Otherwise, set $j = j + 1$ and go to Step 1.
Step 3.
    Find the index $h \in \{1, \ldots, r\}$ such that $T_\sigma(P_h) = \min_{j=1,\ldots,r} T_\sigma(P_j)$
    $P_h$ is an optimal solution of the EQPP.
```

It is worth at this point emphasizing the important differences existing between the networks $\mathcal{F}_j$, $j = 1, \ldots, r$ constructed by the algorithms proposed in [10,11,16,19] to solve the QPP and the networks $\mathcal{G}_j$, $j = 1, \ldots, r$.

The network $\mathcal{F}_j = [\mathcal{N}, \tilde{\mathcal{A}}_j]$ is defined to be a subnetwork of $\mathcal{G}$ where $(u, v) \in \tilde{\mathcal{A}}_j$ if and only if $(u, v) \in \mathcal{A}$ and $c(u, v) \geqslant c_j$. Hence $\mathcal{F}_1 \supset \mathcal{F}_2 \supset \cdots \supset \mathcal{F}_r$. This property allows the algorithms in [10,11,16,19] to skip analyzing some of the networks $\mathcal{F}_j$ when solving the QPP. In fact, if the shortest path in the network $\mathcal{F}_j$ has capacity $c'_j > c_j$, the networks from $\mathcal{F}_{j+1}$ to $\mathcal{F}_{j'}$ can be omitted since they cannot provide a better candidate for the optimal solution of the QPP. However, as the networks $\mathcal{G}_j$ do not satisfy that property, when solving the EQPP it is necessary to solve the shortest path problem in each of the networks $\mathcal{G}_j$. No network can be skipped since arcs which are not in the network $\mathcal{G}_j$ can be in the network $\mathcal{G}_{j+1}$ and vice versa.

Notice also that, at the iteration j, the algorithm saves the shortest path Q as a candidate to be the optimal solution of the EQPP only if its capacity equals c_j (Step 1). Otherwise, it is of no interest at this point of the algorithm. Indeed, if $c(Q) = c_{j'} > c_j$, by applying Lemma 3 no path with capacity c_j can be an optimal solution of the EQPP. Moreover, the path Q will be one of the $s - t$ paths in $\mathcal{G}_{j'}$ and only if Q is an $s - t$ shortest path in the network $\mathcal{G}_{j'}$ will it be a candidate to be the optimal solution of the EQPP.

Finally, notice that if there are P^* and Q^* optimal paths with capacities $c_j = c(P^*) \neq c(Q^*) = c_{j'}$, the algorithm is able to provide both paths because they are $s - t$ shortest paths in $\mathcal{G}_j$ and $\mathcal{G}_{j'}$, respectively.

Theorem 2 *The time complexity of the Algorithm EQPA is $O(r(m + n\log(n)))$ and uses $O(n + m)$ space.*

Proof It is enough to realize that the algorithm essentially amounts to solving r times a shortest path problem each running in $O(m + n\log(n))$ time [7]. □

5 The bi-objective energy-constrained quickest path problem

In this section we propose to take into consideration not only the transmission time but also the total energy used and thus to minimize both over the set of feasible paths. The bi-objective energy-constrained quickest path problem (BEQPP) can be stated as:

$$\begin{aligned} \min_P \quad & (T_\sigma(P), \quad E_\sigma(P)) \\ \text{s.t.} \quad & b_u(\sigma, P) \geqslant 0, u \in \mathcal{N} \\ & P \text{ is an s-t path in the network } \mathcal{G} \end{aligned} \tag{7}$$

According to the theory of multi-objective optimization, a feasible solution P is efficient if and only if there is no other feasible solution Q so that $T_\sigma(Q) \leqslant T_\sigma(P)$ and $E_\sigma(Q) \leqslant E_\sigma(P)$ with at least one strict inequality. If P is an efficient solution, it will be called an $s - t$ efficient path. The image $(T_\sigma(P), E_\sigma(P))$ of P is called a non-dominated point. Two feasible solutions P and Q are called equivalent if they have the same image. A complete set of efficient solutions is a set of efficient solutions X_e such that every feasible solution not in X_e is either dominated or equivalent to at least one feasible solution in X_e.

The BEQPP is NP-hard since the bi-objective shortest path problem (BSPP) is also NP-hard [17]. Note that the BSPP can be obtained as a particular case of the BEQPP when $r = 1$ and the b_u is big enough not to constrain the transmission. The following theorem allows us to conclude that the efficient paths of the BEQPP can be obtained by solving bi-objective shortest path problems in $\mathcal{G}_j$.

Theorem 3 *Let $\widetilde{P}$ be an $s - t$ efficient path for the BEQPP and $c(\widetilde{P}) = c_h$. Then, $\widetilde{P}$ is an $s - t$ efficient path with respect to:*

$$\begin{aligned} \mathrm{BSPP_h} : \min_P \ & (l(P), W(P)) \\ \text{s.t.} \ & P \textit{ is an s-t path in the network } \mathcal{G}_h \end{aligned} \tag{8}$$

Proof The path $\widetilde{P}$ is an $s - t$ path in the network $\mathcal{G}_h$, by construction of this network. Let us assume that there is an $s - t$ path Q in $\mathcal{G}_h$ which dominates $\widetilde{P}$ with respect to the bi-objective function (l, W). Then, $c(Q) \geqslant c_h$ and $l(Q) \leqslant l(\widetilde{P})$ and $W(Q) \leqslant W(\widetilde{P})$ with at least one strict inequality. Let us assume for the time being that $l(Q) < l(\widetilde{P})$. Then,

$$T_\sigma(Q) = l(Q) + \frac{\sigma}{c(Q)} \leqslant l(Q) + \frac{\sigma}{c(\widetilde{P})} < l(\widetilde{P}) + \frac{\sigma}{c(\widetilde{P})} = T_\sigma(\widetilde{P})$$

and

$$E_\sigma(Q) = W(Q)\frac{\sigma}{c(Q)} \leqslant W(Q)\frac{\sigma}{c(\widetilde{P})} \leqslant W(\widetilde{P})\frac{\sigma}{c(\widetilde{P})} = E_\sigma(\widetilde{P})$$

which contradicts that $\widetilde{P}$ is an $s - t$ efficient path for the BEQPP. The other case is analogous. □

As a consequence of Theorem 3, the description of the algorithm BEQPA proposed to solve the BEQPP is as follows:

```
The algorithm BEQPA
Step 0.
    Set $j = 1, \mathcal{E} = \emptyset$
Step 1.
    Solve the $\mathrm{BSPP_j}$.
    If there is no $s - t$ path in $\mathcal{G}_j$, go to Step 2.
    Otherwise, let $\mathcal{E}_j$ be a complete set of efficient paths of the $\mathrm{BSPP_j}$.
    For each $P \in \mathcal{E}_j$, compute $(T_\sigma(P), E_\sigma(P))$
    $\mathcal{E} = \mathrm{Merge}(\mathcal{E}, \mathcal{E}_\mathrm{j})$
Step 2.
    If $j = r$, stop. $\mathcal{E}$ solves the BEQPP.
    Otherwise, set $j = j + 1$ and go to Step 1.
```

where the operation Merge is defined as follows:

$$\mathrm{Merge}(\mathcal{E}, \mathcal{E}_\mathrm{j}) = \{\mathrm{P} \in \mathcal{E} \cup \mathcal{E}_\mathrm{j} : \text{There is no Q} \in \mathcal{E} \cup \mathcal{E}_\mathrm{j} \text{ such that Q dominates P with respect to the bi} - \text{objective function } (\mathrm{T}_\sigma, \mathrm{E}_\sigma)\}$$

Note that in Step 1 of the algorithm there is only the need to compute a complete set of efficient paths. Indeed, let P and Q be equivalent efficient paths for the $\mathrm{BSPP_j}$ with capacities $c(P) = c_h$, $c(Q) = c_i$ such that $c_h > c_i \geqslant c_j$. Since they are equivalent, $l(P) = l(Q)$ and $W(P) = W(Q)$. Hence,

$$T_\sigma(P) = l(P) + \frac{\sigma}{c(P)} < l(Q) + \frac{\sigma}{c(Q)} = T_\sigma(Q)$$
$$E_\sigma(P) = W(P)\frac{\sigma}{c(P)} < W(Q)\frac{\sigma}{c(Q)} = E_\sigma(Q)$$

Therefore, P dominates Q with respect to the bi-objective function (T_σ, E_σ) and so the path Q is not relevant. In the case that the algorithm records Q, the path P would be considered for sure when solving the $\mathrm{BSPP_h}$.

6 Computational experience

This section presents the results of the computational experiment carried out to evaluate the performance of the algorithms EQPA and BEQPA proposed in this paper. The numerical experiments have been performed on a PC Intel® Core™ I7-3820 CPU at 3.6 GHz × 8 having 32 GB of RAM under Ubuntu Linux 14.04 LTS. Although we had a multi-processor computer at hand, only one processor was used in our tests. The code has been written in C++, GCC 4.8.2. The algorithm EQPA involves a Dijkstra's algorithm whose implementation is based on a min-priority queue implemented using a binary heap data structure. In the implementation of the algorithm BEQPA, we have used the Biobjective Label Correcting Algorithm as described in [17] to solve the $\mathrm{BSPP_j}$. As mentioned above, the EQPP can be solved in polynomial time whereas the BEQPP is NP-hard. Due to this very distinctive characteristic, we have used different sets of test problems to assess the performance of the algorithms. Notice also that both algorithms heavily rely on the performance of the algorithms to solve the SPP and the BSPP which they have embedded.

6.1 The EQPA performance evaluation

We have considered three different sets of test problems as in [20]. Set 1 uses the network generator NETGEN [9] to provide the skeleton of the network. Set 2 is based on the network generator GRIDGEN, which is able to provide larger networks. It has been obtained from ftp://dimacs.rutgers.edu/pub/netflow/generators/network/gridgen/gridgen.c. Finally, Set 3 is based on seven USA road networks which have been obtained from http://www.dis.uniroma1.it/challenge9/download.shtml.

Table 1 shows the parameters n, m and r of the networks in Sets 1 and 2. There are 60 problem groups defined by the number of nodes n, the number of arcs m and the number of distinct capacities r in Set 1 and 75 in Set 2. For each problem group, we have generated 10 instances. Delay time and capacity coefficients are generated from uniform distributions in the range [10, 10,000]. To generate problems with a fixed number of capacities, first the required number of capacities is generated from the corresponding uniform distribution. Then, each arc is assigned one of the capacities generated with a uniform probability. The energy rate of the arc (u, v) is computed as $\omega(u, v) = 10^{-5}c(u, v)l^2(u, v)$. The power at the nodes has been fixed at 3×10^8. For assessing the effect of the number of items which are sent, we have taken $\sigma_1 = 100$, $\sigma_2 = 10{,}000$ and $\sigma_3 = 1{,}000{,}000$.

Tables 2 and 3 display the results provided by the EQPA for Sets 1 and 2. The first to third columns show the value of the parameters r, n and m. The fourth to sixth columns display the mean of the $s-t$ shortest paths computed by the algorithm in the 10 runs which are candidate to be an optimal solution of the EQPP, depending on the size of σ. The seventh to ninth columns show the mean CPU time in seconds of the 10 runs for the different values of σ. In the algorithm, there are as many networks $\mathcal{G}_j$ as distinct capacities. Hence, in principle, we could expect to have as many candidate $s-t$ shortest paths as distinct capacities. However, with an increasing number of distinct capacities, the number of candidate $s - t$ shortest paths increases more slowly. For instance, when $r = 10$, the mean of the $s-t$ shortest paths computed by the algorithm varies between 4.3 and 8.5 for Set 1 and between 5.1 and 9.1 for Set 2. However, when $r = 1000$, the range is 11.1–36.9 for Set 1 and 14.3–38.3 for Set 2. Regarding the CPU times, these are almost negligible when the number of distinct capacities is $r = 10$. As expected, the CPU time increases as long as the number of capacities and the size of the network increases, but for the largest problems the average CPU time is

Table 1 Parameters of test problems Set 1 and Set 2

	n	m	r
Set 1	10,000, 20,000, 30,000, 40,000	$10n$, $20n$, $30n$, $40n$, $50n$	10, 100, 1000
Set 2	20,000, 40,000, 60,000, 80,000, 100, 000	$10n$, $20n$, $30n$, $40n$, $50n$	10, 100, 1000

Table 2 EQPA test results: Set 1

r	n	m	# Shortest paths			CPU time		
			σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
10	10,000	100,000	6.5	6.5	4.3	0.04	0.04	0.03
		200,000	7.8	7.8	7.1	0.08	0.09	0.08
		300,000	8.4	8.4	7.4	0.13	0.13	0.13
		400,000	7	7	6.6	0.15	0.15	0.15
		500,000	7.4	7.4	7.2	0.20	0.20	0.19
	20,000	200,000	7	7	4.7	0.10	0.10	0.08
		400,000	7.6	7.6	7.1	0.20	0.20	0.19
		600,000	7.3	7.2	6.4	0.27	0.27	0.25
		800,000	8.3	8.3	7.7	0.35	0.35	0.36
		1,000,000	8.5	8.5	8.4	0.44	0.44	0.45
	30,000	300,000	6.4	6.4	5	0.17	0.17	0.15
		600,000	8.2	8.2	7.2	0.34	0.34	0.31
		900,000	8.1	8.1	7.5	0.45	0.46	0.44
		1,200,000	8.3	8.3	8.3	0.61	0.61	0.60
		1,500,000	7.7	7.7	7.3	0.71	0.71	0.71
	40,000	400,000	6.7	6.7	4.6	0.25	0.25	0.21
		800,000	7.8	7.8	6.7	0.47	0.47	0.42
		1,200,000	7.8	7.8	6.8	0.62	0.62	0.60
		1,600,000	7.9	7.9	7.4	0.77	0.77	0.76
		2,000,000	8.1	8.1	8	0.95	0.95	0.95
100	10,000	100,000	13.4	13.4	8.8	0.23	0.23	0.15
		200,000	17.1	17.1	15.5	0.36	0.36	0.31
		300,000	20.1	20.1	18.8	0.57	0.57	0.50
		400,000	16.9	16.9	14.6	0.70	0.70	0.61
		500,000	20.1	20.1	19.1	0.74	0.75	0.67
	20,000	200,000	17	17.1	12	0.61	0.62	0.38
		400,000	18.8	18.8	13.3	1.04	1.05	0.78
		600,000	18	18	14.8	1.38	1.39	1.06
		800,000	19.8	19.8	19.3	1.73	1.74	1.59
		1,000,000	22.5	22.5	19.5	2.18	2.21	1.97
	30,000	300,000	18.1	18.1	11.6	1.07	1.08	0.71
		600,000	18	18	15.1	1.81	1.82	1.50
		900,000	20.9	20.9	16.9	2.45	2.45	2.12
		1,200,000	22.2	22.2	19.6	3.00	3.02	2.66
		1,500,000	24.4	24.4	20.9	4.24	4.28	3.73

Table 2 continued

r	n	m	# Shortest paths			CPU time		
			σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
	40,000	400,000	19.8	19.8	9.5	1.62	1.63	0.95
		800,000	18	17.9	15.9	2.45	2.47	1.83
		1,200,000	18.3	18.3	14.4	3.33	3.35	2.83
		1,600,000	24.3	24.3	23.1	4.60	4.62	3.99
		2,000,000	24.8	24.8	20.8	5.14	5.18	4.37
1000	10,000	100,000	17.1	17.1	11.1	2.15	2.15	1.25
		200,000	24.4	24.4	23	3.64	3.64	2.77
		300,000	24.6	24.6	20.3	5.13	5.14	4.05
		400,000	24.1	24.1	19.4	5.57	5.59	4.12
		500,000	27.5	27.5	23.8	8.18	8.20	6.88
	20,000	200,000	17.2	17.2	15.1	4.76	4.76	2.80
		400,000	23.5	23.5	18.2	8.86	8.87	6.35
		600,000	22.3	22.3	19.6	12.41	12.41	9.13
		800,000	27.7	27.7	23.7	16.44	16.47	14.24
		1,000,000	30.4	30.4	23.3	21.29	21.32	17.81
	30,000	300,000	21.8	21.8	13.9	9.09	9.11	6.64
		600,000	22.4	22.4	21.1	18.79	18.82	15.10
		900,000	27.4	27.4	22.3	18.84	18.89	15.65
		1,200,000	30.2	30.2	23.8	29.03	29.09	24.10
		1,500,000	35.7	35.7	22.8	35.12	35.21	27.44
	40,000	400,000	19.1	19.1	11.6	13.58	13.58	7.24
		800,000	26.7	26.7	20.3	22.26	22.24	16.20
		1,200,000	28.5	28.5	23.1	34.20	34.26	26.50
		1,600,000	29.6	29.6	25.1	33.58	33.74	26.94
		2,000,000	36.9	36.9	25.5	41.43	41.61	33.41

Mean of the number of candidate $s-t$ shortest paths P_j and mean of the computing time (CPU time in seconds)

less than 6 min and usually takes less than 1 min. In order to get an overall picture of the CPU time invested by the algorithm in these sets of instances, Fig. 2 displays the boxplot of the CPU time for each number of capacities and each value of σ, depending on the type of network generator. Every boxplot summarizes the information of 200 problems when using Set 1 and 250 problems when using Set 2. Note that in both groups the variability increases when the number of capacities increases. Networks of Set 2 are larger and so CPU times are longer.

As for Set 3, Table 4 displays the characteristics of these USA road networks: name of the network, number of nodes and arcs, and the destination node t. In all cases, the node origin is $s = 1$. The energy rate of the arcs and the power of the nodes is the same as in Sets 1 and 2. Based on these networks we have constructed two different groups of test problems. In the first group, the delay is taken as the parameter distance of

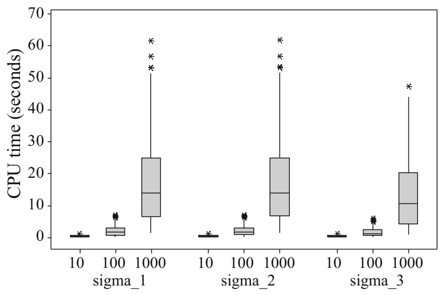
Table 3 EQPA test results: Set 2

r	n	m	# Shortest paths			CPU time		
			σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
10	20,000	200,000	7.4	7.1	5.1	0.14	0.14	0.12
		400,000	8.2	8.0	6.9	0.32	0.30	0.26
		600,000	7.9	8.4	7.3	0.48	0.48	0.46
		800,000	8.2	7.5	7.6	0.65	0.64	0.64
		1,000,000	8.5	8.9	7.6	0.78	0.79	0.76
	40,000	400,000	5.6	7.0	5.1	0.31	0.35	0.28
		800,000	7.3	7.3	7.1	0.63	0.78	0.68
		1,200,000	8.6	7.9	7.9	1.04	0.95	0.96
		1,600,000	8.6	7.8	7.9	1.48	1.14	1.30
		2,000,000	8.3	7.8	7.7	1.57	1.52	1.55
	60,000	600,000	7.7	7.5	6.5	0.53	0.53	0.41
		1,200,000	8.4	7.5	6.8	1.08	1.12	0.93
		1,800,000	8.9	8.5	8.7	1.48	1.55	1.35
		2,400,000	8.0	8.6	8.4	1.99	2.10	1.98
		3,000,000	9.0	8.5	8.6	2.63	2.31	2.39
	80,000	800,000	7.5	7.3	6.2	0.85	0.82	0.65
		1,600,000	8.0	7.9	7.4	1.45	1.48	1.32
		2,400,000	7.8	8.4	7.0	1.90	2.02	1.76
		3,200,000	8.5	8.5	8.9	2.88	2.69	2.67
		4,000,000	8.3	8.4	8.7	2.82	3.21	3.32
	1,00,000	1,000,000	7.1	7.2	4.5	0.97	1.00	0.72
		2,000,000	8.6	8.7	7.6	1.95	1.85	1.67
		3,000,000	8.4	8.2	7.8	2.92	2.74	2.45
		4,000,000	9.0	9.1	8.1	3.64	3.40	3.09
		5,000,000	8.6	8.4	8.4	4.49	4.53	4.52
100	20,000	200,000	17.5	15.1	16.1	0.77	0.88	0.62
		400,000	18.1	20.4	16.1	1.92	2.14	1.82
		600,000	22.1	21.0	19.1	3.38	3.43	2.97
		800,000	23.0	22.1	17.0	4.58	4.53	3.58
		1,000,000	23.0	22.5	21.6	5.50	5.11	5.40
	40,000	400,000	15.4	18.9	14.3	2.69	2.65	2.20
		800,000	21.0	20.7	18.4	5.45	5.74	4.86
		1,200,000	21.4	23.3	19.4	6.34	6.90	6.51
		1,600,000	26.8	23.1	20.8	10.39	9.04	7.88
		2,000,000	21.7	23.9	20.0	11.11	11.33	10.31
	60,000	600,000	19.0	17.4	17.1	3.92	3.53	2.32
		1,200,000	21.9	19.1	19.5	8.09	7.08	6.55
		1,800,000	24.7	25.0	22.3	10.39	10.91	9.30

Table 3 continued

r	n	m	# Shortest paths			CPU time		
			σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
		2,400,000	23.8	23.1	21.5	14.34	15.76	13.05
		3,000,000	24.4	24.6	24.2	16.11	16.96	15.05
	80,000	800,000	19.8	16.2	13.9	5.99	6.32	4.07
		1,600,000	23.7	21.4	20.7	11.20	9.89	9.26
		2,400,000	22.7	23.4	22.7	13.46	13.15	14.53
		3,200,000	25.5	25.6	21.8	18.86	18.47	15.41
		4,000,000	24.6	26.3	21.9	24.07	22.62	20.98
	100,000	1,000,000	17.0	18.2	11.2	7.01	7.69	4.40
		2,000,000	22.9	24.2	21.7	14.74	15.03	11.03
		3,000,000	25.9	23.8	20.8	23.22	22.43	16.20
		4,000,000	24.8	24.5	22.5	24.84	23.00	19.95
		5,000,000	26.5	26.7	23.1	35.49	32.16	27.34
1000	20,000	200,000	16.4	21.2	17.4	7.05	7.64	5.59
		400,000	28.3	24.7	20.3	18.83	16.80	13.81
		600,000	28.4	25.2	20.0	33.84	28.61	26.85
		800,000	31.4	32.5	24.7	40.10	42.63	35.63
		1,000,000	32.4	30.4	26.4	47.70	47.07	45.59
	40,000	400,000	21.6	20.4	14.3	24.66	23.38	15.10
		800,000	23.5	25.1	19.9	45.14	45.88	41.90
		1,200,000	35.9	30.1	20.4	60.73	63.05	57.30
		1,600,000	31.8	36.7	27.5	84.54	81.46	82.87
		2,000,000	30.9	30.1	28.7	99.68	109.98	105.14
	60,000	600,000	21.8	16.1	15.8	39.21	31.20	22.96
		1,200,000	29.9	26.8	21.7	65.34	72.31	59.27
		1,800,000	28.7	32.4	26.2	92.61	87.70	78.63
		2,400,000	31.8	35.3	23.0	124.92	141.53	104.93
		3,000,000	36.0	32.2	34.1	170.26	161.07	135.22
	80,000	800,000	21.8	21.4	16.5	57.94	51.97	37.74
		1,600,000	28.4	27.0	28.7	89.25	90.94	81.56
		2,400,000	31.6	31.8	26.5	130.03	143.72	108.45
		3200,000	35.0	35.9	33.9	148.90	168.41	164.46
		4,000,000	37.0	38.3	31.4	223.11	214.16	207.19
	100,000	1,000,000	22.3	22.5	17.7	66.68	58.03	43.85
		2,000,000	28.7	27.3	17.5	134.51	115.37	105.83
		3,000,000	30.3	32.3	25.4	194.02	212.43	174.07
		4,000,000	35.7	35.4	30.0	229.23	216.92	199.08
		5,000,000	36.0	34.7	30.5	320.52	308.50	265.57

Mean of the number of candidate $s-t$ shortest paths P_j and mean of the computing time (CPU time in seconds)


(a) Networks of Set 1

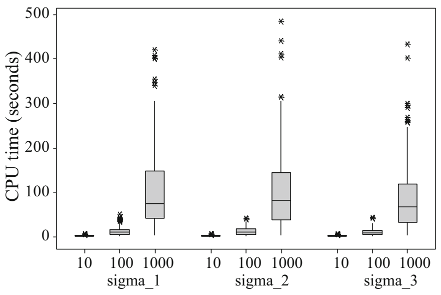

(b) Networks of Set 2

Fig. 2 Boxplots of CPU time depending on the number of capacities and the value of σ

Table 4 Dimension and destination nodes of the network in Set 3

Road network	n	m	Dest. 1	Dest. 2	Dest. 3	Dest. 4
NY	264,346	733,846	264,346	132,173	857	20
BAY	321,270	800,172	321,270	160,635	567	18
COL	435,666	1,057,066	435,666	217,833	660	19
FLA	1,070,376	2,712,798	1,070,376	535,188	1035	21
NE	1,524,453	3,897,636	1,542,453	762,227	1235	21
CAL	1,890,815	4,657,742	1,890,815	945,408	1375	21
LKS	2,758,119	6,885,658	2,758,119	1,379,060	1661	23

Table 5 Arcs delay empirical distribution

$l(u, v)$	11	16	25	42	73	128	227	410	744	1365	2520	4681	8700
%	2.3	5.4	8.5	10.0	12.0	11.0	10.0	11.0	8.5	7.0	5.0	5.0	4.3

the road network [20]. The capacity is computed from the parameter time of the road network. The range of the arc times is partitioned in 100 intervals of equal length. In order to have integer capacities, the intervals are rounded off by applying the ceiling function to the upper endpoint and properly adjusting the intervals. For instance, if $(a_1, a_2]$, $(a_2, a_3]$ are the first two intervals of the partition, the resulting intervals would be $(a_1, \lceil a_2 \rceil]$, $(\lceil a_2 \rceil, \lceil a_3 \rceil]$. Then, if an arc time is in the interval $(a, b]$, the arc capacity is b. Therefore, problems with 100 distinct capacities are obtained.

The second group of instances built with the USA road networks takes the arc delay and capacity from the empirical distributions proposed in [6], which are displayed in Tables 5 and 6. For this group, 10 instances have been generated for each problem.

Table 7 provides the results. Now the first column displays the name of the network and the second column shows the destination node. The other columns display, for the first group, the number of candidate shortest paths and the CPU time depending on the size σ. For the second group, the columns which contain the number of candidate shortest paths and CPU time provide the average of the 10 instances. Note that the

Table 6 Arcs capacity empirical distribution

$c(u,v)$	1360	64	128	256	800	1680	2640	4000	8000
%	51.30	7.15	5.30	0.88	4.40	19.47	4.40	2.70	4.40

Table 7 EQPA test results: Set 3

	Dest.	First group						Second group					
		# Shortest paths			CPU time			# Shortest paths			CPU time		
		σ_1	σ_2	σ_3	σ_1	σ_2	σ_3	σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
NY	4	2	2	0	1.36	1.36	1.36	2.2	2.2	0.0	0.44	0.40	0.31
	3	2	2	1	1.36	1.36	1.36	3.2	3.1	2.3	0.29	0.29	0.31
	2	1	1	0	1.45	1.45	1.36	3.2	3.7	0.1	0.49	0.50	0.31
	1	1	1	0	1.39	1.40	1.36	2.2	2.7	0.0	0.43	0.44	0.31
BAY	4	1	1	0	1.46	1.45	1.46	1.6	1.6	0.0	0.38	0.37	0.35
	3	1	1	0	1.49	1.48	1.46	1.9	1.6	0.0	0.41	0.41	0.35
	2	1	1	0	1.51	1.51	1.46	1.8	2.2	0.0	0.45	0.47	0.35
	1	1	1	0	1.54	1.55	1.45	1.7	1.6	0.0	0.50	0.50	0.35
COL	4	1	1	0	2.02	2.02	2.03	1.4	1.9	0.3	0.48	0.47	0.47
	3	1	1	0	2.02	2.02	2.03	3.0	3.2	0.0	0.48	0.46	0.47
	2	1	0	0	2.04	2.02	2.03	2.7	3.0	0.0	0.50	0.51	0.47
	1	1	0	0	2.14	2.03	2.02	1.0	0.9	0.0	0.70	0.71	0.47
FLA	4	1	1	0	3.89	3.88	3.88	1.2	1.1	0.0	1.24	1.17	1.18
	3	1	1	0	3.88	3.87	3.89	1.0	1.2	0.0	1.24	1.19	1.18
	2	1	1	0	3.97	3.98	3.89	1.2	1.0	0.0	1.27	1.28	1.18
	1	1	1	0	3.90	3.93	3.88	1.3	1.1	0.0	1.20	1.21	1.18
NE	4	1	1	0	9.57	9.57	9.60	3.5	4.0	0.3	1.98	1.89	1.92
	3	1	1	0	9.56	9.57	9.59	2.7	3.2	0.2	2.18	2.09	1.94
	2	1	1	0	9.89	9.97	9.60	1.6	1.4	0.0	3.49	3.47	1.93
	1	1	1	0	9.79	9.92	9.60	1.1	0.9	0.0	3.27	3.36	1.93
CAL	4	1	1	0	9.93	9.91	9.92	2.1	2.1	0.0	2.20	2.21	2.32
	3	1	0	0	9.94	9.90	9.94	1.0	1.0	0.0	2.30	2.32	2.32
	2	1	0	0	10.34	9.91	9.93	1.0	1.0	0.0	2.80	2.81	2.32
	1	1	0	0	10.12	9.93	9.93	1.0	1.0	0.0	2.50	2.48	2.32
LKS	4	1	1	0	14.78	14.76	14.83	2.0	1.9	0.9	3.43	3.40	3.56
	3	1	1	0	14.87	14.85	14.82	3.5	2.9	0.0	3.60	3.49	3.58
	2	1	1	0	15.48	15.37	14.83	2.7	2.6	0.0	5.26	5.20	3.56
	1	1	1	0	14.84	14.87	14.83	2.6	2.4	0.0	4.10	4.22	3.55

First group: Number of candidate $s-t$ shortest paths P_j and computing time. Second group: Mean of the number of candidate $s-t$ shortest paths P_j and mean of the computing time. (CPU time in seconds)

Table 8 BEQPA test results

r	n	m	#Efficient paths P_j			# Efficient paths			CPU time		
			σ_1	σ_2	σ_3	σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
2	1000	10,000	6.0	4.8	3.9	5.7	4.8	3.9	0.01	0.01	0.00
		20,000	5.3	4.4	3.5	4.6	4.1	3.2	0.03	0.02	0.01
		50,000	5.3	4.9	4.4	4.4	4.4	4.3	0.06	0.04	0.03
	5000	50,000	6.2	5.7	4.7	5.5	5.2	4.4	0.09	0.07	0.04
		100,000	6.6	6.5	6.1	6.4	6.3	5.9	0.19	0.15	0.10
		250,000	5.4	5.4	5.4	5.0	5.1	5.2	0.47	0.45	0.26
	10,000	100,000	5.9	5.9	4.2	5.0	5.3	4.0	0.27	0.19	0.11
		200,000	6.3	6.3	6.0	5.7	6.1	5.9	0.48	0.46	0.25
		500,000	6.9	6.9	6.9	6.1	6.6	6.6	1.43	1.11	0.71
5	1000	10,000	9.7	8.5	5.7	6.0	5.7	4.0	0.02	0.01	0.01
		20,000	10.9	9.9	8.2	5.7	6.1	5.3	0.04	0.03	0.02
		50,000	13.2	12.5	10.5	7.2	7.8	7.5	0.11	0.08	0.05
	5000	50,000	11.6	10.1	7.7	5.9	6.5	5.6	0.17	0.11	0.06
		100,000	13.2	13.1	12.0	7.7	8.9	8.7	0.38	0.29	0.17
		250,000	15.3	15.3	14.9	10.5	11.3	11.3	1.10	0.92	0.63
	10,000	100,000	12.7	10.9	7.9	7.5	7.7	6.8	0.50	0.31	0.11
		200,000	13.7	13.4	12.3	10.2	10.8	10.1	1.07	0.80	0.58
		500,000	16.3	16.2	15.8	8.8	10.5	11.0	2.86	2.37	1.51
10	1000	10,000	20.9	15.8	9.0	6.1	5.8	4.6	0.04	0.02	0.01
		20,000	22.8	19.1	12.4	10.8	10.3	8.1	0.08	0.05	0.03
		50,000	23.5	22.6	18.3	10.2	10.9	10.1	0.20	0.15	0.09
	5000	50,000	21.5	18.9	12.3	8.2	9.0	6.8	0.29	0.20	0.09
		100,000	27.7	26.3	19.8	10.6	12.7	11.4	0.68	0.51	0.29
		250,000	29.8	29.2	27.6	12.6	13.8	13.4	1.98	1.58	1.02
	10,000	100,000	20.2	17.0	10.5	7.4	8.1	6.6	0.91	0.58	0.20
		200,000	30.3	29.0	26.0	10.0	11.8	11.7	1.83	1.45	0.84
		500,000	31.4	31.2	30.1	12.8	14.7	16.3	5.64	4.23	2.65
20	1000	10,000	38.1	29.8	15.0	5.4	6.3	5.0	0.07	0.04	0.01
		20,000	40.1	32.6	26.5	10.3	11.0	9.7	0.14	0.09	0.05
		50,000	44.3	41.4	35.7	11.5	13.4	12.8	0.34	0.24	0.14
	5000	50,000	47.3	42.5	23.0	9.4	10.4	8.0	0.56	0.37	0.15
		100,000	46.6	42.8	36.5	7.9	8.6	8.9	1.21	0.87	0.51
		250,000	56.7	55.9	52.8	15.5	18.6	20.3	3.92	2.95	1.96
	10,000	100,000	43.5	35.6	24.4	8.3	8.3	7.8	1.60	0.95	0.42
		200,000	53.7	52.1	45.6	7.0	9.4	10.6	3.57	2.74	1.64
		500,000	66.0	65.6	62.0	16.0	19.0	22.0	9.30	7.11	4.96

Table 8 continued

r	n	m	#Efficient paths P_j			# Efficient paths			CPU time		
			σ_1	σ_2	σ_3	σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
30	1000	10,000	57.8	51.1	26.8	5.2	6.6	4.9	0.09	0.06	0.02
		20,000	67.5	52.4	37.1	8.9	9.7	9.2	0.21	0.13	0.06
		50,000	68.1	65.7	54.9	13.4	14.5	15.2	0.49	0.36	0.19
	5000	50,000	64.1	58.4	37.1	8.6	9.7	8.9	0.63	0.40	0.17
		100,000	65.6	63.2	56.4	10.6	11.9	12.7	1.44	1.03	0.59
		250,000	80.5	79.4	76	14.9	17.4	20	4.61	3.45	2.31
	10,000	100,000	63.7	51.7	30.9	9.8	10.9	9.1	1.67	1.05	0.40
		200,000	79.2	76.1	67.7	11.8	14.3	15.4	4.08	2.99	1.81
		500,000	100.7	99.7	96.7	16.6	20.3	23.8	11.85	9.06	6.30

Mean of the number of candidate $s - t$ efficient paths P_j, mean of the cardinality of the complete set of efficient paths of the BEQPP and mean of the computing time (CPU time in seconds)

number of candidates as well as the CPU time are small. It is worth pointing out that the USA road networks considered are not very dense. In fact, the average degree of the nodes is 2.6.

6.2 The BEQPA performance evaluation

As mentioned above, it is harder to solve these problems. We present the results of a set of smaller networks which have been generated using NETGEN. The parameters have been assigned as follows. Delay time and capacity coefficients are generated from uniform distributions in the range [1, 50] and [10, 50], respectively. The energy rate of the arc (u, v) is computed as $\omega(u, v) = 0.01c(u, v)l^2(u, v)$. The power at the nodes has been fixed at 3×10^5. There are 45 problem groups defined by the number of nodes $n = 1000$, 5000 and 10,000, the number of arcs $m = 10n$, $20n$ and $50n$ and the number of distinct capacities $r = 2, 5, 10, 20$ and 30. For each problem group, we have generated 10 instances. The size of the message has been taken to be $\sigma_1 = 10{,}000$, $\sigma_2 = 20{,}000$ and $\sigma_3 = 50{,}000$.

Table 8 presents the results provided by the BEQPA. The first to third columns show the value of the parameters r, n and m. The fourth to sixth columns display the mean of the candidate $s - t$ efficient paths computed by the algorithm in the 10 runs by solving problem (8), depending on the size of σ. The seventh to ninth columns show the mean of the cardinality of the complete set of efficient paths of the BEQPP computed by the algorithm in the 10 runs. Finally, the tenth to twelfth columns display the mean CPU time in seconds of the 10 runs for the different values of σ. We can see that the number of efficient solutions is reasonably small, as suggested in practical applications for the BSPP [17]. Moreover, computing times are also short, less than twelve seconds on average for all the problems.

7 Conclusions

In this paper we have introduced the energy-constrained quickest path problem, a variant of the QPP with a side constraint on the consumption of energy at the nodes. Taking into account its properties, this problem can be reformulated as the problem of finding shortest paths with respect to the delay time on a sequence of as many subnetworks of $\mathcal{G}$ as different capacities. These subnetworks satisfy, by construction, that there is energy available at the nodes for transmitting the data units. A polynomial algorithm has been developed with the same time complexity as the algorithms developed to solve the QPP. The bi-objective variant of the energy-constrained quickest path problem is also considered which aims to minimize transmission time and consumed energy. The problem is transformed into finding a complete set of efficient shortest paths in the same networks. The results of the computational study show the good performance of the algorithms.

Acknowledgements This research work has been funded by the Gobierno de Aragón under Grant E58 (FSE) and by UZ-Santander under Grant UZ2012-CIE-07. The authors gratefully acknowledge the anonymous referee for his/her valuable suggestions to improve the presentation of the paper.

References

1. Calvete, H., del-Pozo, L.: The quickest path problem with batch constraints. Oper. Res. Lett. **31**(4), 277–284 (2003)
2. Calvete, H., del-Pozo, L., Iranzo, J.: Algorithms for the quickest path problem and the reliable quickest path problem. Comput. Manag. Sci. **9**(2), 255–272 (2012)
3. Chen, G., Hung, Y.: Algorithms for the constrained quickest path problem and the enumeration of quickest paths. Comput. Oper. Res. **21**, 113–118 (1994)
4. Chen, Y.: Finding the *k* quickest simple paths in a network. Inf. Process. Lett. **50**, 89–92 (1994)
5. Chen, Y., Chin, Y.: The quickest path problem. Comput. Oper. Res. **17**(2), 153–161 (1990)
6. Clímaco, J., Pascoal, M., Craveirinha, J., Captivo, M.: Internet packet routing: application of a k-quickest path algorithm. Eur. J. Oper. Res. **181**, 1045–1054 (2007)
7. Fredman, M., Tarjan, R.: Fibonacci heaps and their uses in improved network optimization algorithms. J. Assoc. Comput. Mach. **34**(3), 596–615 (1987)
8. Hamacher, H., Tjandra, S.: Mathematical modelling of evacuation problems: a state of the art. In: Schreckenberg, M., Sharma, S. (eds.) Pedestrian and Evacuation Dynamics, pp. 227–266. Springer, Berlin (2002)
9. Klingman, D., Napier, A., Stutz, J.: Netgen: a program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. Manag. Sci. **20**(5), 814–821 (1974)
10. Martins, E., Santos, J.: An algorithm for the quickest path problem. Oper. Res. Lett. **20**(4), 195–198 (1997)
11. Moore, M.: On the fastest route for convoy-type traffic in flowrate-constrained networks. Transp. Sci. **10**(2), 113–124 (1976)
12. Park, C.K., Lee, S., Park, S.: A label-setting algorithm for finding a quickest path. Comput. Oper. Res. **31**(14), 2405–2418 (2004)
13. Pascoal, M., Captivo, M., Clímaco, J.: An algorithm for ranking quickest simple paths. Comput. Oper. Res. **32**(3), 509–521 (2005)
14. Pascoal, M., Captivo, M., Clímaco, J.: A comprehensive survey on the quickest path problem. Ann. Oper. Res. **147**(1), 5–21 (2006)
15. Pascoal, M., Captivo, M., Clímaco, J.: Computational experiments with a lazy version of a *k* quickest simple path ranking algorithm. TOP **15**(2), 372–382 (2007)
16. Pelegrín, B., Fernández, P.: On the sum-max bicriterion path problem. Comput. Oper. Res. **25**(12), 1043–1054 (1998)

17. Raith, A., Ehrgott, M.: A comparison of solution strategies for biobjective shortest path problems. Comput. Oper. Res. **36**(4), 1299–1331 (2009)
18. Rao, N., Grimmell, W., Radhakrishan, S., Bang, Y., Manickam, N.: Quickest paths for different network router mechanisms. In: Proceedings of ninth International Conference on Advanced Computing and Communications (2001)
19. Rosen, J., Sun, S., Xue, G.: Algorithms for the quickest path problem and the enumeration of quickest paths. Comput. Oper. Res. **18**(6), 579–584 (1991)
20. Sedeño-Noda, A., González-Barrera, J.: Fast and fine quickest path algorithm. Eur. J. Oper. Res. **238**(2), 596–606 (2014)