A Penalized Best-Response Algorithm for Non-Linear Single-Path Routing Problems

Olivier Brun, Balakrishna Prabhu, Josselin Vallet

A Penalized Best-Response Algorithm for Non-Linear Single-Path Routing Problems*

O. Brun[1,2], B.J. Prabhu[1,2], J. Vallet[3]

[1]CNRS, LAAS, 7 Av. du Colonel Roche, F-31400 Toulouse, France

[2]Univ. Toulouse, LAAS, F-31400 Toulouse, France

[3]Viveris Technologies, 1 Avenue de l'Europe, F-31400 Toulouse, France

Abstract

This paper is devoted to non-linear single-path routing problems, which are known to be NP-hard even in the simplest cases. For solving these problems, we propose an algorithm inspired from Game Theory in which individual flows are allowed to independently select their path to minimize their own cost function. We design the cost function of the flows so that the resulting Nash equilibrium of the game provides an efficient approximation of the optimal solution. We establish the convergence of the algorithm and show that every optimal solution is a Nash equilibrium of the game. We also prove that if the objective function is a polynomial of degree $d \geq 1$, then the approximation ratio of the algorithm is $\left(2^{1/d}-1\right)^{-d}$. Experimental results show that the algorithm provides single-path routings with modest relative errors with respect to optimal solutions, while being several orders of magnitude faster than existing techniques.

Keywords: best response; single-path routing; game theory; non-linear programming; approximation algorithm; Nash equilibrium

*This is an extended version of the paper "A game-theoretic algorithm for non-linear single-path routing problems" that appeared in the Proceedings of INOC 2015 [38]. The work of the third author was carried out while he was a doctoral student at LAAS.

1 Introduction

The single-path routing problem, also known as the unsplittable or non-bifurcated multicommodity flow problem, naturally arises in a variety of contexts and, in particular, plays a central role in the traffic engineering of communication networks. It amounts to routing a given set of fixed traffic demands in a network, allocating each demand to a single path to minimize the total network cost, which is usually expressed as the sum of link costs (see [25, 33, 39] and Chapter 8 of [34] for the related joint routing and congestion control problem). A common assumption is that the cost of a link is a linear function of the traffic flowing on that link. Under this assumption, the single-path routing problem [see p. 15] can be formulated as a Mixed-Integer Linear Programming (MILP) problem with binary variables [4, 6, 7, 8]. Although this assumption might be appropriate in some contexts, it is no longer viable when the cost of a link is interpreted as the delay incurred by a packet on that link since this delay depends non-linearly on the amount of traffic flowing over that link.

This paper is devoted to single-path routing problems with an additive and non-linear objective function. Our interest in these problems originates from traffic engineering in communication networks based on the Multi-Protocol Label Switching (MPLS) technology [16]. MPLS changes the usual hop-by-hop paradigm by enabling network flows to be routed along predetermined paths, which are called Label Switched Paths (LSPs). In an MPLS network, each flow is routed along a single path, but two different flows with the same source/destination pair can use two different paths. Traffic engineering in MPLS networks mainly amounts to optimizing the quality of service of network flows by tailoring the paths assigned to flows to the prevailing traffic conditions. As explained above, when the quality of service metric is the network delay, the objective function depends non-linearly on the traffic flowing on each link and the problem at hand becomes a non-linear single-path routing problem.

We aim to keep the discussion as general as possible by proposing a solution strategy that is also applicable in other application areas. We therefore focus on the essence of the non-linear single-path routing problem and omit on purpose many application-specific details of traffic engineering in MPLS networks. The problem we consider can be formulated as follows. Given some fixed traffic demands, the goal is to find a single-path routing strategy that minimizes the sum of link costs. The cost of a link e depends on the amount of traffic y_e flowing on that link (measured in packets/s)

and has the form $y_e\,\ell_e(y_e)$, where the function ℓ_e gives the delay per packet on link e as a function of y_e. We specifically assume that the link latency function ℓ_e is a non-linear function. A typical example is the M/M/1 delay function $\ell_e(x) = 1/(c_e - x)$, where c_e represents the capacity of link e. Note however that the results established in this paper hold for a broad range of cost functions.

The routing problem as defined above belongs to the class of non-linear mathematical programs involving integer variables (actually, binary variables). These mathematical programs are known to be extremely hard to solve, both from a theoretical and from a practical point of view (see Chapter 15 of [27] for a survey as well as [9] and references therein for a thorough literature review). Even in the simplest case with binary variables, quadratic function and equality constraints, they are known to be NP-hard [14]. Several general approaches have been proposed to solve non-linear integer programming problems. Some transform the discrete problem into a continuous one (see for example [32]). Despite their qualities, those techniques do not scale very well with the size of the problems. A recent alternative is the so-called *Global Smoothing Algorithm* [30], which seems to scale better while providing fairly good approximations (see Section 5.1 for details). Heuristics and meta-heuristics have also been used to find an approximate solution to non-linear integer programming problems. Among others, ant-inspired optimization techniques are known to be efficient for solving various routing problems [24, 36]. In this paper, we use the heuristic method proposed in [24] for comparison purposes, as well as exact NLP-based branch-and-bound algorithms [12, 26].

We propose an approximation algorithm, inspired from Game Theory [23], for solving non-linear single-path routing problems. We shall assume that individual demands are allowed to independently select their path to minimize their own cost function. We design the cost function of the demands so that the resulting Nash equilibrium of the game provides an efficient approximation of the optimal solution. We note that a similar algorithm was proposed in [1] for scheduling of strictly periodic tasks. We establish the convergence of the algorithm and show that every optimal solution is a Nash equilibrium of the game. We also prove that if the link latency functions ℓ_e are polynomial of degree $d \geq 0$, then the approximation ratio of the algorithm is $\left(2^{1/(d+1)} - 1\right)^{-(d+1)}$. As will be shown numerically, the main merit of this algorithm is that it is several orders of magnitude quicker than the method discussed above while providing good optimization results.

The rest of this paper is organized as follows. We introduce our notation and formally state the problem in Section 2. We then describe the proposed algorithm in Section 3. In Section 4, we show the convergence of the algorithm and obtain worst-case guarantees on the quality of the solutions

it produces. Section 5 is devoted to the empirical performance evaluation of the algorithm. Finally, some conclusions are drawn in Section 6.

2 Problem Statement

Consider a network represented by a directed graph $G = (V, E)$. To each edge $e \in E$ is associated a non-decreasing latency function $\ell_e : \mathbb{R}_+ \to \mathbb{R}_+$. For any set $\pi \subset E$, we define the constant δ_π^e as 1 if $e \in \pi$, and 0 otherwise.

We are given a set $\mathcal{K} = \{1, 2, \ldots, K\}$ of Origin-Destination (OD) pairs. Let s_k and t_k be the origin and destination of OD pair k, and let $\lambda_k \in \mathbb{N}$ be its traffic demand. Each traffic demand has to be routed in the network over a single path. We let Π_k be the set of all paths available for routing traffic between s_k and t_k. This set can contain all simple paths from s_k to t_k, or only a subset of those paths satisfying some constraints. We define a routing strategy as a vector $\boldsymbol{\pi} = (\pi_k)_{k\in\mathcal{K}} \in \Pi$, where π_k is the path assigned to traffic demand k and $\Pi = \Pi_1 \times \Pi_2 \times \cdots \times \Pi_K$. The goal is to find a routing strategy that minimizes the cost of the network $F(\boldsymbol{\pi}) = \sum_{e\in E} y_e(\boldsymbol{\pi})\ell_e(y_e(\boldsymbol{\pi}))$, where $y_e(\boldsymbol{\pi}) = \sum_{k\in\mathcal{K}} \delta_{\pi_k}^e \lambda_k$ is the total traffic flowing on link e in routing strategy $\boldsymbol{\pi}$. Formally, the problem is as follows:

$$\text{minimize } F(\boldsymbol{\pi}) = \sum_{e\in E} y_e(\boldsymbol{\pi})\, \ell_e(y_e(\boldsymbol{\pi})) \tag{OPT}$$

subject to

$$\pi_k \in \Pi_k \qquad\qquad k \in \mathcal{K}. \tag{1}$$

We note that the problem can be formulated as a 0-1 mathematical programming problem by introducing the binary variables

$$x_{k,\pi} = \begin{cases} 1 & \text{if demand } k \text{ is routed on path } \pi \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Indeed, we can rewrite problem (OPT) as follows:

$$\text{minimize} \sum_{e \in E} y_e \ell_e(y_e)$$

subject to

$$y_e = \sum_{k \in \mathcal{K}} \sum_{\pi \in \Pi_k} \lambda_k \delta_\pi^e x_{k,\pi} \qquad e \in E, \tag{3}$$

$$\sum_{\pi \in \Pi_k} x_{k,\pi} = 1 \qquad k \in \mathcal{K}, \tag{4}$$

$$x_{k,\pi} \in \{0,1\} \qquad \pi \in \Pi_k, k \in \mathcal{K}. \tag{5}$$

When the objective function F is non-linear function of the variables y_e, the above problem belongs to the class of non-linear mathematical programs involving integer variables (actually, binary variables in our case). These mathematical programs are known to be extremely hard to solve, thus motivating the development of efficient methods for finding approximate solutions.

3 Best-response algorithm

The best-response algorithm we propose takes its inspiration from an algorithm of the same name in Game Theory. In a game, the best-response of a player is defined as its optimal strategy conditioned on the strategies of the other players. It is, as the name suggests, the best response that the player can give for a given strategy of the others. The best-response algorithm then consists of players taking turns in some order to adapt their strategy based on the most recent known strategy of the others until an equilibrium point is reached.

In this section, we design the game so that the Nash equilibrium of the game provides an efficient approximation to problem (OPT). We follow the approach proposed in [15]. Let us think of the traffic demands as the players of the game. The strategy of player k is the path π_k it chooses in the set Π_k, and a strategy profile of the game is a vector $\boldsymbol{\pi} \in \Pi$. In other words, a strategy profile corresponds to a feasible solution of problem (OPT). Given the strategy of the other players $\boldsymbol{\pi}_{-i} = (\pi_1, \pi_2, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_K)$, the value $f_i(\pi, \boldsymbol{\pi}_{-i})$ associated to path $\pi \in \Pi_i$ by player i reflects the cost of this path, i.e.,

$$f_i(\pi, \boldsymbol{\pi}_{-i}) = \sum_{e \in \pi} \lambda_i \ell_e(y_e(\pi, \boldsymbol{\pi}_{-i})), \quad \forall \pi \in \Pi_i. \tag{6}$$

We note that

$$F(\boldsymbol{\pi}) = \sum_e \left(\sum_k \delta^e_{\pi_k} \lambda_k \right) \ell_e(y_e(\boldsymbol{\pi})) = \sum_k \lambda_k \sum_{e \in \pi_k} \ell_e(y_e(\boldsymbol{\pi})) = \sum_k f_k(\boldsymbol{\pi}). \tag{7}$$

When player i minimizes the cost given in (6), this game is called a weighted congestion game (see [10] and references therein). For these games, several authors, including [10] and [2], have investigated what is called the *Price of Anarchy* which is the ratio of the social cost (the function F in our case) at the worst-case Nash equilibrium and the social cost at the global optimum. For polynomial cost functions of degree d, it was shown in [2] that the Price of Anarchy for mixed equilibrium is Φ_d^{d+1}, where Φ_d is the solution of $(\Phi_d + 1)^d = \Phi_d^{d+1}$.

Although one could design a best-response algorithm based on (6), that algorithm will have two drawbacks: *(i)* a global optimum of F need not be a Nash equilibrium of the weighted congestion game as is shown in Example 1; and *(ii)* the convergence of the best-response algorithm for the weighted congestion game is known only in some special cases (linear latency functions or when the demands are the same).

Example 1. *Consider a network with two parallel links as shown in Figure 1. There are two flows, each of size* 1*, which wish to go from O to D. There are two possible routes: (i) the top-route with a latency function of* $l(x) = x$*; and (ii) the bottom-route with a latency function of* $l(x) = 0.4x$*.*

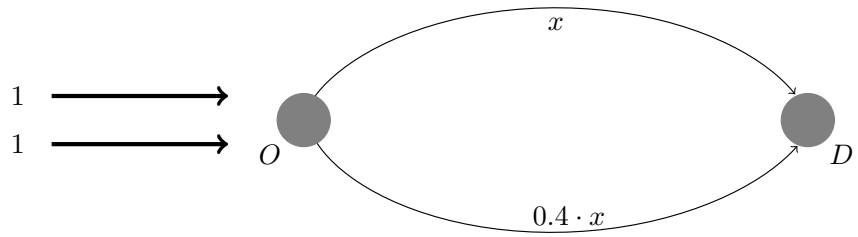


Figure 1: Example to show that the Nash equilibrium of a weighted congestion game need not be a global optimum.

It can be verified that at the global optimum one of the flows will be routed through the top-route and the other will be routed through the bottom-route. The cost for the flow on the top-route at

the optimal routing is 1 *while for the one on the bottom-route it is* 0.4. *In order to check that the optimum is not a Nash equilibrium for cost defined in* (6)*, we look at the best-response of the flow on the top-route. If this flow moves to the bottom-route its cost will be* $0.4(1+1) = 0.8$ *which is less than its current cost of* 1. *Thus, at the global optimum the flow on the top-route has an incentive to move to the bottom-route.*

In order to drive the players towards a local minimum of $F(\boldsymbol{\pi})$, we add a penalty term $p_i(\pi, \boldsymbol{\pi}_{-i})$ to the cost of player i. The role of the penalty term is that of an incentive for players to take into account the impact of their actions on other players. When player i routes its traffic on path π, the increase in the cost of each player $j \neq i$ using link $e \in \pi$ is

$$\lambda_j \left[\ell_e \left(y_e^{-i} + \lambda_i \right) - \ell_e \left(y_e^{-i} \right) \right], \tag{8}$$

where $y_e^{-i} = \sum_{k \neq i} \delta_{\pi_k}^e \lambda_k$ represents the total traffic flowing on link e due to all players other than i. As a consequence, we define the penalty term as follows:

$$\begin{aligned} p_i(\pi, \boldsymbol{\pi}_{-i}) &= \sum_{e \in \pi} \sum_{j \neq i} \lambda_j \delta_{\pi_j}^e \left[\ell_e \left(y_e^{-i} + \lambda_i \right) - \ell_e \left(y_e^{-i} \right) \right], \\ &= \sum_{e \in \pi} y_e^{-i} \left[\ell_e \left(y_e^{-i} + \lambda_i \right) - \ell_e \left(y_e^{-i} \right) \right]. \end{aligned} \tag{9}$$

In summary, player i computes its path to minimize its own cost, that is, player i solves the following problem:

$$\text{minimize}_{\pi \in \Pi_i} c_i(\pi, \boldsymbol{\pi}_{-i}) = f_i(\pi, \boldsymbol{\pi}_{-i}) + p_i(\pi, \boldsymbol{\pi}_{-i}). \tag{OPT-i}$$

In (OPT-i), the path π minimizing $c_i(\pi, \boldsymbol{\pi}_{-i})$ is known as the best response of player i to the strategy $\boldsymbol{\pi}_{-i}$ of the other players. The game is in a Nash equilibrium if and only if the current strategy of each player is its best response to the strategy of the others, implying that no player has an incentive to unilaterally deviate from its current strategy. Formally, $\boldsymbol{\pi}$ is a Nash equilibrium if and only if

$$c_i(\boldsymbol{\pi}) \leq c_i(\pi, \boldsymbol{\pi}_{-i}), \forall \pi \in \Pi_i, \forall i \in \mathcal{K}.$$

As we will show briefly, the best-response algorithm that we propose as an approximation to problem (OPT) computes a Nash equilibrium. The algorithm starts from an initial feasible solution $\boldsymbol{\pi}^{(0)}$. At each iteration, the players update their strategies in a given order by computing the optimal solution of problem (OPT-i) using any shortest path algorithm (e.g., Dijkstra's algorithm). Note that a player i deviates from its strategy $\pi_i^{(n)}$ at iteration n to a new strategy π' if and only if $c_i\left(\pi', \boldsymbol{\pi}_{-i}^{(n)}\right) < c_i\left(\boldsymbol{\pi}^{(n)}\right)$. The algorithm stops when no player can decrease its cost by unilaterally deviating from its strategy, that is, $\boldsymbol{\pi}^{(n+1)} = \boldsymbol{\pi}^{(n)}$. The pseudocode for the heuristic is given in Algorithm 1.

Algorithm 1 Penalized best-response

Require: $\boldsymbol{\pi}^{(0)}$
1: $n \leftarrow 0$
2: **repeat**
3: **for** $i = 1, \ldots, K$ **do**
4: $\pi_i^{(n+1)} \leftarrow \text{argmin}\,(\text{OPT-}i)$
5: **end for**
6: $n \leftarrow n + 1$
7: **until** $\boldsymbol{\pi}^{(n+1)} \neq \boldsymbol{\pi}^{(n)}$.
8: **return** $\vec{\pi}(n)$

In the following, we shall use the term *penalized best-response* to refer to Algorithm 1. The term *standard best-response algorithm* will refer to the case when each player i directly optimizes $f_i(\pi, \boldsymbol{\pi}_{-i})$ instead of the penalized cost function $c_i(\pi, \boldsymbol{\pi}_{-i}) = f_i(\pi, \boldsymbol{\pi}_{-i}) + p_i(\pi, \boldsymbol{\pi}_{-i})$.

4 Properties of the penalized best-response algorithm

In this section, we establish several properties of the penalized best-response algorithm. Notably, we will show that *(i)* it has guaranteed convergence; and *(ii)* the global optimum of F is an equilibrium of the algorithm. These two properities are not known to be true for the standard best-response algorithm.

We first show in Section 4.1 that the penalized best-response algorithm converges in a finite number of steps to a Nash equilibrium. In contrast, the convergence of the standard best-response algorithm in routing games is still largely an open issue. For the unsplittable routing games considered in the present paper, it has been proven only (a) for linear latency functions [22], and (b) when

all flows have the same traffic demands (that is, when $\lambda_k = \lambda$, $\forall k$), in which case the game is a congestion game [35]. It is also worth mentioning that for splittable routing games, the convergence of the standard best-response algorithm has been proven only in some special cases for networks of parallel links [3, 13, 29, 31].

In Section 4.2, we obtain performance guarantees for the solutions provided by the penalized best-response algorithm in the special case of polynomial link latency functions. In this special case, the algorithm is therefore an approximation algorithm since it returns a solution that is provably close to optimal (in contrast to other cases where it is only a heuristic that may or may not find a good solution). Our performance guarantees take the form of worst-case bounds on the approximation ratio of our algorithm, that is, bounds on the ratio between the result obtained by the algorithm and the optimal cost. We use techniques similar to those used in [5], where the authors obtain bounds on the approximation ratio of the standard best-response algorithm for affine and polynomial cost functions with non-negative coefficients. The idea of the proof is similar in that it uses the Hölder inequality to bound the difference between the Nash equilibrium and deviations from it. Since our problem is different from the one in [5], we get a tight upper bound.

4.1 Convergence of the penalized best-response algorithm

As proven in [15], the convergence of the penalized best-response algorithm directly follows from the fact that F is a potential function of the game. We briefly describe the proof below. The crucial argument of this proof is stated in Lemma 1 below.

Lemma 1. *The objective function of player i can be written as*

$$c_i(\boldsymbol{\pi}) = F(\boldsymbol{\pi}) - h(\boldsymbol{\pi}_{-i}), \tag{10}$$

where

$$h(\boldsymbol{\pi}_{-i}) = \sum_{e \in E} y_e^{-i}\, \ell_e(y_e^{-i}), \tag{11}$$

does not depend on the path π_i chosen by player i.

Proof. Note that $y_e(\boldsymbol{\pi}) = y_e^{-i}$ for $e \notin \pi_i$, while $y_e(\boldsymbol{\pi}) = y_e^{-i} + \lambda_i$ for $e \in \pi_i$. Hence

$$
\begin{aligned}
c_i(\boldsymbol{\pi}) &= f_i(\pi_i, \boldsymbol{\pi}_{-i}) + p_i(\pi_i, \boldsymbol{\pi}_{-i}), \\
&= \sum_{e \in \pi_i} \lambda_i \ell_e(y_e^{-i} + \lambda_i) + \sum_{e \in \pi_i} y_e^{-i} \left[\ell_e\left(y_e^{-i} + \lambda_i\right) - \ell_e\left(y_e^{-i}\right) \right], \\
&= \sum_{e \in \pi_i} \left(y_e^{-i} + \lambda_i\right) \ell_e(y_e^{-i} + \lambda_i) - \sum_{e \in \pi_i} y_e^{-i} \ell_e\left(y_e^{-i}\right), \\
&= \left(\sum_{e \in \pi_i} y_e(\boldsymbol{\pi})\, \ell_e(y_e(\boldsymbol{\pi})) + \sum_{e \notin \pi_i} y_e(\boldsymbol{\pi})\, \ell_e(y_e(\boldsymbol{\pi})) \right) - \left(\sum_{e \in \pi_i} y_e^{-i} \ell_e\left(y_e^{-i}\right) + \sum_{e \notin \pi_i} y_e^{-i} \ell_e\left(y_e^{-i}\right) \right), \\
&= F(\boldsymbol{\pi}) - h(\boldsymbol{\pi}_{-i}).
\end{aligned}
$$

□

Proposition 1. *The penalized best-response algorithm converges in a finite number of steps to a Nash equilibrium.*

Proof. A direct consequence of Lemma 1 is that the function F is a potential function of the game, i.e.,

$$
c_i(\boldsymbol{\pi}) - c_i(\pi, \boldsymbol{\pi}_{-i}) = F(\boldsymbol{\pi}) - F(\pi, \boldsymbol{\pi}_{-i}), \quad \forall \pi \in \Pi_i, \forall i \in \mathcal{K},
$$

implying that any best-response move results in a decrease of the global cost $F(\boldsymbol{\pi})$. Since $F(\boldsymbol{\pi})$ can accept a finite number of values, the sequence will reach a local minimum in a finite number of steps. □

Another consequence of Lemma 1 is that any global optimum is a Nash equilibrium, so that the penalized best-response algorithm stops in an optimal point if it ever reaches it.

Proposition 2. *Any global optimum is a Nash equilibrium of the routing game.*

Proof. Consider a global optimum $\boldsymbol{\pi}^*$, i.e., a point such that $F(\boldsymbol{\pi}^*) \leq F(\boldsymbol{\pi})$ for all $\boldsymbol{\pi} \in \times_i \Pi_i$ Assume to the contrary that $\boldsymbol{\pi}^*$ is not a Nash equilibrium. It follows that there exists a player i and a strategy $\pi_i \in \Pi_i$ such that $c_i(\pi_i, \boldsymbol{\pi}^*_{-i}) < c_i(\boldsymbol{\pi}^*)$. With Lemma 1, it yields $F(\pi_i, \boldsymbol{\pi}^*_{-i}) - h(\boldsymbol{\pi}^*_{-i}) < F(\boldsymbol{\pi}^*) - h(\boldsymbol{\pi}^*_{-i})$, from which we conclude that $F(\pi_i, \boldsymbol{\pi}^*_{-i}) < F(\boldsymbol{\pi}^*)$. This is clearly a contradiction to the global optimality of $\boldsymbol{\pi}^*$. Hence $\boldsymbol{\pi}^*$ is a Nash equilibrium. □

Remark 1. *The worst-case computational complexity of finding a pure Nash equilibrium in a potenial game can be exponential in the number of players (the number of flows in our problem)[20]. However, in [19] it is shown that on average the complexity is linear in the number of players. Thus, on random instances, the best-response algorithm can be expected to converge much faster than exact algorithms.*

4.2 Performance guarantees for polynomial link latency functions

An instance I of our problem is defined by the graph $G = (V, E)$, by the set of traffic demands $\mathcal{K}$ (including the demand value λ_k and the set of candidate paths Π_k for routing each demand k) and by the link latency functions $\ell_e : \mathbb{R}_+ \to \mathbb{R}_+$. Our goal in this section is to establish bounds on the approximation ratio of the penalized best-response algorithm that hold uniformly over all instances of the problem. More precisely, given an instance I of the problem, let $\boldsymbol{\pi}$ be any Nash equilibrium and let $\boldsymbol{\pi}^*$ be an optimal routing strategy for that instance. We look for an upper bound on the ratio $F(\boldsymbol{\pi})/F(\boldsymbol{\pi}^*)$ that holds for all instances I of the problem. In the following, we establish such a bound when the link latency functions are polynomial functions of the form $\ell_e(x) = \sum_{j=0}^{d} a_{e,j} x^j$, where $d \geq 0$ and the a_e are non-negative coefficients. Our main result is stated in Theorem 3.

Theorem 3. *If $\ell_e(x) = \sum_{j=0}^{d} a_{e,j} x^j$, then the approximation ratio of the penalized best-response algorithm is bounded above by $\left(2^{1/(d+1)} - 1\right)^{-(d+1)}$.*

Proof. See Appendix A. □

We can furthermore prove that the upper bound of Theorem 3 is tight, in the sense that there exists an instance of the single-path routing problem for which it is reached. The lower bound on the approximation ratio is stated in Proposition 4, whose proof is based on an adaptation of the instance given in [10] (see Lower Bound 2 and Lemma 4.5 in that paper).

Proposition 4. *If $\ell_e(x) = \sum_{j=0}^{d} a_{e,j} x^j$, then the approximation ratio of the penalized best-response algorithm is bounded below by $\left(2^{1/(d+1)} - 1\right)^{-(d+1)}$.*

Proof. We consider a problem instance with a monomial cost function, that is, $\ell_e(x) = a_e x^d$, and N origin-destination pairs. In addition to these nodes, there are two other nodes R_0 and R_1 which act as routers. Origin O_i is connected to nodes O_{i-1} and O_{i+1}, except O_1 which is connected to

R_0 and O_2, and O_N which is connected to O_{N-1} and R_1. Destination D_i is connected to O_{i-1} and O_{i+1}, except D_1 which is connected to R_0 and O_2, and D_N which is connected to O_{N-1} and R_1. In the following, we shall assume that links are numbered in such a way that link 0 is link R_0O_1, link i is link O_iO_{i+1} for $i = 1, \ldots, N-1$ and link N is link O_NR_1. For $i = 0, \ldots, N$, link i has a cost of $a_i y^{d+1}$ when it carries a traffic of y, whereas for all other links we have $a_e = 0$. The instance has a total of $3N + 1$ links of which $2N$ links have a cost of 0 and the $N + 1$ other links have a non-zero cost. Figure 2 shows an instance with $N = 4$ pairs.

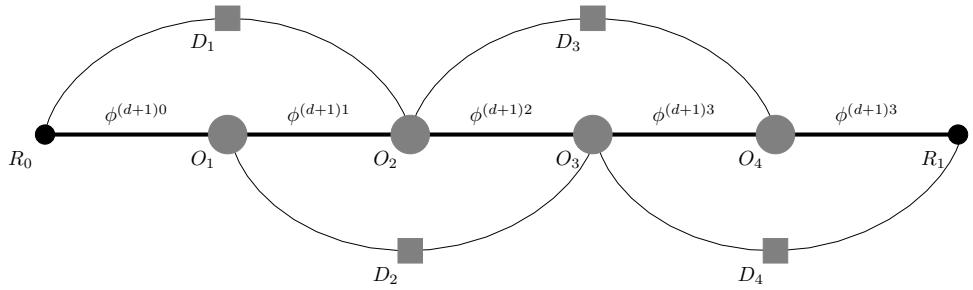


Figure 2: Problem instance with $N = 4$ origin-destination pairs for the lower bound on the approximation ratio.

Let λ_i be the amount of traffic sent from origin O_i to destination D_i. Each origin has two choices: either send the traffic to the left or to the right. In the global optimum (OPT) each origin will send its traffic to the right provided that

$$a_{i+1} \leq a_i \quad \text{and} \quad a_N = a_{N_1}. \tag{12}$$

Indeed, in that case, each flow is routed over the link with the smaller a_e, and over another link of cost $\ell_e(x) = 0$. Similarly, we shall establish the conditions under which routing the traffic to the left corresponds to a Nash equilibrium (NE). It is enough, according to Lemma 1, to show that we cannot decrease the network cost by deviating a single flow from its equilibrium route onto another route. Provided that $a_N = a_{N-1}$, the cost does not change when flow N is routed to the right on link O_NR_1 instead of being routed to the left on link $O_{N-1}O_N$. If flow $i = 1, \ldots, N_1$ is routed to the right on link i instead of being routed to the left on link $i - 1$, the traffic on link i increases from λ_{i+1} to $\lambda_i + \lambda_{i+1}$, whereas the traffic on link i becomes 0. We deduce that the cost variation is

$$a_i(\lambda_i + \lambda_{i+1})^{d+1} + 0 - \left[a_{i-1}\lambda_i^{d+1} + a_i\lambda_{i+1}^{d+1}\right],$$

so that routing to the left corresponds to a Nash equilibrium provided that

$$a_{i-1}\lambda_i^{d+1} + a_i\lambda_{i+1}^{d+1} \leq a_i(\lambda_i + \lambda_{i+1})^{d+1}, \quad i = 1, \ldots, N-1, \tag{13}$$

and $a_N = a_{N-1}$. Let us now assume that $a_i = \phi^{(d+1)i}$ for $0 \leq i < N$ and that $a_N = \phi^{(d+1)(N-1)}y^{d+1}$, where $\phi = 2^{1/(d+1)} - 1$. Note that the links $O_{N-1}O_N$ and O_NR_1 have the same cost. In Figure 2, for the links with a non-zero cost, the corresponding a_e is shown close to the mid-point of the link. We further assume that $\lambda_i = \phi^{N-i}$. Since $\phi < 1$, $a_N = a_{N-1}$ and $a_{i+1} = \phi^{(d+1)}a_i$ for $i = 1, \ldots, N-1$ implies that the conditions (12) are satisfied, so that routing to the right is indeed an optimal strategy. To see that routing to the left corresponds to a Nash equilibrium, observe that for $i < N$,

$$\begin{aligned} a_{i-1}\lambda_i^{d+1} + a_i\lambda_{i+1}^{d+1} &= 2\,\phi^{(d+1)(N-1)} \\ &= (1+\phi)^{d+1}\,\phi^{(d+1)(N-1)} \\ &= a_i(\lambda_i + \lambda_{i+1})^{d+1}, \end{aligned}$$

which proves that the network cost cannot be decreased by deviating unilaterally any flow. It is interesting to note that this instance has been designed in such a way that the corresponding Nash equilibrium is a weak equilibrium, that is, for each flow i, there exists a path π different from its equilibrium path π_i such that $c_i(\pi, \boldsymbol{\pi}_{-i}) = c_i(\boldsymbol{\pi})$.

For the instance with $N = 4$, in Table 1, we give the traffic on each link and the associated cost for each of the two solutions (for the clarity of the presentation, only links e such that $\ell_e(x) \neq 0$ are presented). Summing the costs of the links in Table 1, we observe that the optimal cost is $F_4(\boldsymbol{\pi}^*) = 3\phi^{4(d+1)} + \phi^{3(d+1)}$, whereas the cost at the Nash equilibrium is $F_4(\boldsymbol{\pi}) = 4\phi^{3(d+1)}$.

In general, it can be seen that $F_N(\boldsymbol{\pi}^*) = (N-1)\phi^{(d+1)N} + \phi^{(d+1)(N-1)}$ and that $F_N(\boldsymbol{\pi}) =$

Table 1: Traffic on the links and associated cost in the optimal solution and at the Nash equilibrium for the instance shown in Figure 2.

	NE			OPT		
Links	Flow	Traffic	Cost	Flow	Traffic	Cost
R_0O_1	1	ϕ^3	$\phi^{3(d+1)}$	–	0	0
O_1O_2	2	ϕ^2	$\phi^{3(d+1)}$	1	ϕ^3	$\phi^{4(d+1)}$
O_2O_3	3	ϕ^1	$\phi^{3(d+1)}$	2	ϕ^2	$\phi^{4(d+1)}$
O_3O_4	4	ϕ^0	$\phi^{3(d+1)}$	3	ϕ^1	$\phi^{4(d+1)}$
O_4R_1	–	0	0	4	ϕ^0	$\phi^{3(d+1)}$

$N\phi^{(d+1)(N-1)}$. As a consequence, the approximation ratio is

$$g_N(\phi) = \frac{N}{(N-1)\phi^{d+1}+1}. \tag{14}$$

By taking $N \to \infty$, we obtain an approximation ratio of $\phi^{-(d+1)} = (2^{1/(d+1)}-1)^{-(d+1)}$ as claimed. □

Remark 2. *For the sake of clarity, we have shown the lower bound on an undirected graph. By adding links and intermediate nodes, this instance can be modified to make the graph directed as well. However, this will make the figure and the notation cumbersome.*

As an immediate consequence of Theorem 3 and Proposition 4, we obtain the following corollary.

Corollary 5. *For large d, the approximation ratio is roughly $\left(\frac{d+1}{\log(2)}\right)^{d+1}$.*

In Table 2, we give the approximation ratios for different values of d for the penalized best-response algorithm and compare it with that of the non-penalized version, as established in [5] and [10]. We emphasize that it is not proven that the non-penalized version converges for polynomial latency functions, so that its approximation ratio provides only a guarantee on the quality of the solution when it converges.

Interestingly, in the case $d = 0$, that is, when the latency function $\ell_e(x)$ of each link e is a constant $a_e \geq 0$, our algorithm is guaranteed to provide an optimal solution. This is a direct consequence of Theorem 3, but can also be easily understood by noting that in this case

Table 2: Approximation ratio for the penalized best-response (BR) as well as for the standard best-response as a function of the degree of the polynomial. The values have been rounded-off at two decimal places.

degree (d)	Penalized BR	Standard BR [2, 10]
0	1	1
1	5.83	2.62
2	56.95	9.91
3	780.28	47.82
asymptotic	$\Theta\left(\left(\frac{d+1}{\log(2)}\right)^{d+1}\right)$	$\Theta\left(\left(\frac{d}{\log(d)}\right)^{d+1}\right)$

$$F(\boldsymbol{\pi}) = \sum_{e \in E} a_e y_e(\boldsymbol{\pi}) = \sum_{e \in E} a_e \sum_{k \in \mathcal{K}} \delta^e_{\pi_k} \lambda_k = \sum_{k \in \mathcal{K}} \lambda_k \left(\sum_{e \in \pi_k} a_e \right),$$

so that it is optimal to route each flow k on the path $\pi_k \in \Pi_k$ minimizing $\sum_{e \in \pi_k} a_e$, which can be seen as the length of the path. However, in that case (9) yields $p_i(\boldsymbol{\pi}) = 0$, from which it follows that $c_i(\pi, \boldsymbol{\pi}_{-i}) = f_i(\pi, \boldsymbol{\pi}_{-i}) = \lambda_i \left(\sum_{e \in \pi_i} a_e\right)$, so that our algorithm routes each flow on a shortest path, exactly as does the optimal solution.

When $d \geq 1$, the approximation ratio of the penalized best-response algorithm is significantly worse than that of the non-penalized best-response algorithm. Asymptotically, the non-penalized version is a factor $(\log(d))^{d+1}$ better than the penalized version. On the downside, the non-penalized version does not have a guaranteed convergence whereas the penalized one is guaranteed to converge.

It is however important to keep in mind that the above performance guarantees are obtained for worst-case scenarios that are not necessarily representative of instances met in practice. We conjecture that the worst-case performance of the algorithm occurs in very asymmetric scenarios in terms of link latency functions and in terms of traffic demands. Consider for example the instance used in Proposition 4 to obtain a lower bound on the approximation ratio. Assume that the link latency functions are monomials of degree d such that $a_N = a_{N-1}$ and $1 \leq \frac{a_i}{a_{i+1}} \leq x$ for some fixed x such that

$$1 \leq x \leq \min_i \frac{(\lambda_i + \lambda_{i+1})^{(d+1)} - \lambda_{i+1}^{(d+1)}}{\lambda_i^{d+1}}.$$

The above conditions impose a certain homogeneity between the coefficients a_i. It is readily verified that equations (12) and (13) are satisfied under these conditions, so that routing to the right is an optimal strategy, and routing to the left is a Nash equilibrium. Observe now that

$$\frac{F(\boldsymbol{\pi})}{F(\boldsymbol{\pi}^*)} = \frac{\sum_{i=0}^{N-1} a_i \lambda_{i+1}^{(d+1)}}{\sum_{i=1}^{N} a_i \lambda_i^{(d+1)}} \leq x \, \frac{\sum_{i=0}^{N-1} a_{i+1} \lambda_{i+1}^{(d+1)}}{\sum_{i=1}^{N} a_i \lambda_i^{(d+1)}} = x,$$

so that in this example we can obtain a solution as close as we want to an optimal solution by imposing a condition on the ratio $\frac{a_i}{a_{i+1}}$. Unfortunately, we were not able to explicitly characterize the conditions under which the approximation ratio will be lower than a given x in the general case. Nevertheless, the numerical experiments presented in Section 5 reveal that the penalized best-response algorithm often provides good quality solutions, contrary to what is suggested by the approximation ratio $\left(2^{1/(d+1)} - 1\right)^{-(d+1)}$. These experiments also show that the results of the penalized best-response algorithm are quite similar to those of its non-penalized counterpart.

5 Empirical Performance Evaluation

This section is devoted to the empirical performance evaluation of the best-response algorithm. We first describe the benchmark experiments that were performed, as well as our empirical algorithm comparison methodology in Section 5.1. Section 5.2 reports the numerical results obtained.

5.1 Benchmark Experiments

In our benchmark experiments, we have used 800 randomly generated instances obtained using 8 standard network topologies (see Table 3) collected from the IEEE literature and from the Rocketfuel project [37]. For each network topology, we consider 100 random traffic matrices generated with uniform distributions in such a way that there is positive traffic demand associated to each origin/destination pair. The traffic matrices are computed so that there exists a minimum-hop routing strategy in which the network congestion rate (that is, the maximum utilization rate of the network links) is equal to γ, where $\gamma > 0$ is a given parameter.

The empirical evaluation of the best-response algorithm is done for two different types of objective functions, as described below.

Table 3: Topologies : number of nodes and links.

Topology	# nodes	# links
ABOVENET	19	68
ARPANET	24	100
BHVAC	19	46
EON	19	74
METRO	11	84
NSF	8	20
PACBELL	15	42
VNSL	9	22

5.1.1 Piece-wise Linear Objective Function

The main motivation for considering a piece-wise linear objective function is that in that case we can compare the algorithm with the optimal solutions obtained from a MILP solver. Specifically, we consider the following increasing latency function proposed in [21]:

$$\ell_e(y) = \begin{cases} 1 & \text{if } 0 \leq \frac{y}{c_e} < \frac{1}{3}, \\ 3 - \frac{2}{3}\,\frac{c_e}{y} & \text{if } \frac{1}{3} \leq \frac{y}{c_e} < \frac{2}{3}, \\ 10 - \frac{16}{3}\,\frac{c_e}{y} & \text{if } \frac{2}{3} \leq \frac{y}{c_e} < \frac{9}{10}, \\ 70 - \frac{178}{3}\,\frac{c_e}{y} & \text{if } \frac{9}{10} \leq \frac{y}{c_e} < 1, \\ 500 - \frac{1468}{3}\,\frac{c_e}{y} & \text{if } 1 \leq \frac{y}{c_e} < \frac{11}{10}, \\ 5000 - \frac{16318}{3}\,\frac{c_e}{y} & \text{if } \frac{11}{10} \leq \frac{y}{c_e}, \end{cases}$$

where c_e represents the capacity of link e. This function expresses that it is cheap to send flow over a link with a small utilization rate, whereas, as the load approaches 100%, it becomes more expensive. Note that the utilization rate can be greater than 1, in which case we get heavily penalized. For the above latency function, we assume that the set Π_k contains all the possible path between s_k and t_k for all flows k. Problem (OPT) can then be formulated as a Mixed-Integer Linear Problem:

$$\text{minimize} \sum_{e \in E} \Phi_e \tag{OPT-PWL}$$

subject to

$$\sum_{e \in In(n)} x_{k,e} - \sum_{e \in Out(n)} x_{k,e} = \begin{cases} -1 & \text{if } n = s, \\ 1 & \text{if } n = t, \\ 0 & \text{otherwise.} \end{cases} \qquad n \in V, k \in \mathcal{K}, \tag{15}$$

$$y_e = \sum_{k \in \mathcal{K}} \lambda_k x_{k,e} \qquad e \in E, \tag{16}$$

$$\Phi_e \geq a y_e - b c_e \qquad (a,b) \in \mathcal{C}, e \in E, \tag{17}$$

$$x_{k,e} \in \{0,1\}, y_e \geq 0, \Phi_e \geq 0 \qquad e \in E, k \in \mathcal{K}, \tag{18}$$

where $\mathcal{C} = \left\{(1,0), (3, \frac{2}{3}), (10, \frac{16}{3}), (70, \frac{178}{3}), (500, \frac{1468}{3}), (5000, \frac{16318}{3})\right\}$ and $In(n)$ (resp. $Out(n)$) represents the set of all ingoing (resp. outgoing) links at node n. In the above node/link formulation, the decision variable $x_{k,e}$ is 1 if flow k is routed over link e, and 0 otherwise.

For each of the 800 instances, we have computed the relative error of the best-response algorithm with respect to the optimal solution of (OPT-PWL), which was computed using Gurobi 6.0 as the MILP solver [26]. We have limited the total time expended by the solver to 15 minutes per problem instance. The traffic matrices were computed with $\gamma = 1.2$ as threshold parameter, so that we can guarantee that there exists a feasible solution in which the utilization rate of the links is at most 120%.

5.1.2 Non-linear Objective Function

We consider two different link latency functions, each one yielding a non-linear network cost function. The first one is a linear latency function, $\ell_e(y) = \frac{y}{(c_e)^2}$, where c_e represents the capacity of link e, yielding the following quadratic objective function

$$F(\boldsymbol{\pi}) = \sum_{e \in E} \left(\frac{y_e(\boldsymbol{\pi})}{c_e} \right)^2. \tag{19}$$

The second link latency function corresponds to the well-known M/M/1 delay function (a.k.a.

Kleinrock function) [28], that is $\ell_e(y) = \frac{1}{c_e - y}$, so that

$$F(\boldsymbol{\pi}) = \sum_{e \in E} \frac{y_e(\boldsymbol{\pi})}{c_e - y_e(\boldsymbol{\pi})}. \tag{20}$$

This cost function is often used when one wants to minimize the mean packet delay in the network. For the above two objective functions, it becomes extremely hard to find an optimal solution to problem (OPT). We therefore compare our penalized best-response algorithm with a lower bound on the optimal cost obtained by relaxing the 0-1 constraints in problem (OPT), thereby solving the following multi-path routing problem

$$\text{minimize} \sum_{e \in E} y_e \ell_e(y_e) \tag{OPT-NL-MP}$$

subject to

$$y_e = \sum_{k \in \mathcal{K}} \sum_{\pi \in \Pi_k} \lambda_k \delta_\pi^e x_{k,\pi} \qquad e \in E, \tag{21}$$

$$\sum_{\pi \in \Pi_k} x_{k,\pi} = 1 \qquad k \in \mathcal{K}, \tag{22}$$

$$0 \leq x_{k,\pi} \leq 1 \qquad \pi \in \Pi_k, k \in \mathcal{K}, \tag{23}$$

which can easily be done with a projected gradient algorithm.

In addition to the comparison with the optimal multi-path solution, we compare our algorithm with four other algorithms:

- **Global Smoothing Algorithm (GSA)**: This algorithm was introduced in [30] for solving non-linear optimization problems with binary variables. Its original approach is to solve a sequence of non-linear optimization problems without integer variables. In each problem of the sequence, the goal is to minimize the sum of the original objective function plus two penalty terms: a logarithmic barrier penalty (designed to smooth the function and keep the current point away from the integer grid) and an "exact" penalty (describing how far the current point is from the integer grid). Both terms are weighted by one parameter, which is updated at each iteration in order to gradually bring the current point onto the integer grid. At each step, a continuous optimization problem is solved using a gradient algorithm

(we have used the modified conjugate gradient algorithm presented in [11]). The sequence of penalized problems should form a trajectory leading to a good approximation of the solution. We refer to [30] for further details.

- **Ant Colony Optimization (ACO)**: This heuristic algorithm was proposed in [24]. It belongs to the family of ant colony algorithms , which are known to be efficient for searching an optimal path in a graph [17, 18, 36]. The algorithm iteratively improves an initial solution to the problem through a random exploration of the solution space by agents called (artificial) ants. During one iteration of the algorithm, each ant builds a solution to the problem by randomly taking decisions, that is by randomly assigning paths to flows. A weight, called pheromone trail, is assigned to each decision taken by ants and is updated based on the experience acquired by ants during problem solving. These weights, which represent the learned desirability of the decisions, are used to guide the random exploration of the solution space.

- **Gurobi quadratic-programming based branch-and-bound algorithm**: The Gurobi MIP solver can solve models with a quadratic objective and/or quadratic constraints using a branch-and-bound algorithm [26]. This exact algorithm was used to compute the optimal solution of the single-path routing problem when the objective function is given by (19). As for the piece-wise linear objective function, we have limited the total time expended by the solver to 15 minutes per problem instance.

- **Bonmin NLP-based branch-and-bound algorithm**: Bonmin is an open-source software for solving general MINLP (Mixed-Integer Non-Linear Programming) problems [12]. Bonmin features several exact algorithms, including a NLP-based branch-and-bound algorithm and an outer-approximation decomposition algorithm. The former was used to compute the optimal solution of the single-path routing problem when the objective function is given by (20). As with Gurobi, we have limited the total time expended by Bonmin to 15 minutes per problem instance.

Given the size and complexity of the problem instances, and in order to keep the running times of these algorithms below a reasonable amount, we restrict ourselves to two possible paths for each OD flow (that is, $|\Pi_k| = 2$ for all $k \in \mathcal{K}$). These paths are obtained by solving a 2-shortest-path problem in which the link weights are equal to 1 [40]. The traffic matrices were computed with

$\gamma = 1.0$ as threshold parameter, so that we can guarantee that there exists a feasible solution in which the capacity of no link is exceeded, which is particularly important for the M/M/1 latency function.

The GSA and ACO algorithms have been implemented in Matlab 2013b. For GSA, two different initial multi-path solutions are considered: the first one is randomly generated, whereas the second one is obtained by adding a small random perturbation to the single-path solution obtained with the best-response algorithm (so as to obtain a multi-path initial solution). The results presented in the following for the GSA algorithms correspond to the best solution obtained from these two initial points. Regarding ACO, we consider at most 50 iterations and 5 ants since our experiments revealed that increasing these values does not improve significantly the obtained results while negatively impacting computing times.

5.2 Numerical Results

5.2.1 Piece-wise Linear network cost function

The numerical results obtained for this cost function are reported in Table 4. Interestingly, the worst results are obtained for the smallest topologies, NSF and METRO. Except for these topologies, the penalized BR algorithm performs quite well. Its average error over the 800 instances is only 3.31%, and it would be as low as 1.77% without the instances generated with the NSF topology. The average execution time of the algorithm is 0.51 seconds, and it never exceeds 3.2 seconds. This is to be compared to the average execution time of 82.6 seconds of the MILP solver (the time limit of 15 minutes was reached for 46 out of the 100 instances generated with the EON topology).

5.2.2 Quadratic network cost function

We first start by comparing the penalized BR algorithm with GSA and ACO for this cost function. Our numerical results are reported in Table 5. First of all, note that all three algorithms perform quite well since the average relative error is below 4.37% for each one. Over the 800 instances, the maximum relative error of the best-response algorithm is 20.16% (to be compared to the 13.32% of the other algorithms). We also note that on average GSA performs slightly better than ACO and the best-response algorithm. Remember however that GSA is initialized with the solution obtained with the best-response algorithm. If we now look at the computing times presented in Table 9, we

Table 4: Relative gap (%) to the optimal solution for the piece-wise linear objective function.

Topology	**min**	**max**	**avg**
ABOVENET	0.04	1.912	0.725
ARPANET	0.08	3.488	1.103
BHVAC	0.347	4.225	1.838
EON	0.938	6.390	3.022
METRO	0.177	10.297	3.969
NSF	0.081	63.021	14.128
PACKBEL	0.313	3.294	1.520
VNSL	0.010	2.556	0.240
Global	0.010	63.021	3.31

observe that the best-response algorithm is significantly faster than all other algorithms, since its worst computing time is 6.8 seconds, whereas the running times of ACO and GSA can be as high as 50 and 300 seconds, respectively.

It is also interesting to note that the Gurobi solver usually computes the optimal solution quickly (although always significantly slower that that of the penalized BR algorithm). Given these fast computing times of the Gurobi solver, and in order to better assess the performance of the penalized BR algorithm, we ran once again both of them, but this time with $k = 6$ candidate paths per OD flow. Gurobi found the optimal solution for all instances. Results are reported in Table 6. Except for the METRO topology, the relative gaps to the optimal solution are very modest and the average error over the 800 instances is only 1.07% (it would be as low as 0.2% without the instances generated with the METRO topology).

5.2.3 M/M/1 network cost function

The results obtained with this cost function are presented in Table 7. In this case, the penalized best-response and the ACO algorithms perform better than GSA, for which relative errors up to 87.54% are obtained on some instances. The best-response algorithm achieves the lower average relative error with only 1%, whereas ACO has a lower maximum relative error with 14.99%. However, as can be noted from Table 9, the best-response algorithm provides the best tradeoff between optimization quality and computing times. Over the 800 instances, its worst execution time is only 4.7 seconds, whereas the computing times of GSA and ACO can be as high as 400 and 50 seconds, respectively.

Table 5: Relative gap (%) to the optimal multi-path solution for the quadratic cost function.

Topology	**Penalized BR**			**GSA**			**ACO**		
	min	max	avg	min	max	avg	min	max	avg
ABOVENET	1.28	5.18	3.86	$\simeq 0$	13.32	7.19	0.17	4.87	2.98
ARPANET	1.05	2.83	1.84	1.09	5.40	2.57	4.42	8.49	5.99
BHVAC	0.24	1.16	0.65	0.09	4.35	0.94	3.93	8.30	5.78
EON	$\simeq 0$	3.37	2.31	$\simeq 0$	3.63	0.89	2.24	6.63	4.53
METRO	2.23	20.16	8.59	1.64	8.90	4.47	2.63	13.32	7.41
NSF	$\simeq 0$	12.97	3.29	0.04	4.43	1.76	0.15	7.46	2.21
PACBELL	1.89	5.27	3.61	0.32	3.00	1.44	2.30	8.00	4.76
VNSL	0.03	7.27	1.91	$\simeq 0$	8.57	3.07	0.09	4.92	1.34
Global	$\simeq 0$	20.16	3.26	$\simeq 0$	13.32	2.79	0.09	13.32	4.37

Table 6: Relative gap (%) to the Gurobi solution for the quadratic cost function.

Topology	**Penalized BR**		
	min	max	avg
ABOVENET	0.067	0.58	0.28
ARPANET	0.02	0.29	0.13
BHVAC	0	0.16	0.03
EON	0.05	0.97	0.35
METRO	0	18.3	7.17
NSF	0	3.8	0.39
PACBELL	0	0.81	0.16
VNSL	0	0.25	0.05
Global	0	18.3	1.07

We observe that the computing times of Bonmin are two orders of magnitude larger than that of the penalized BR algorithm. In practice, they often reach the time limit of 15 minutes for the largest instances, even though there are only $k = 2$ candidate paths per OD flow (we had to increase the time limit for the ARPANET network in order to always obtain a feasible solution). Table 8 shows the minimum, maximum and average relative gaps between the solution of the BR algorithm and the one computed by Bonmin. Here again, despite the fact that significant errors are obtained for the smallest topologies (NSF and METRO), the average error over the 800 instances is only

Table 7: Relative gap (%) to the optimal multi-path solution for the M/M/1 cost function.

Topology	**Penalized BR**			**GSA**			**ACO**		
	min	max	avg	min	max	avg	min	max	avg
ABOVENET	$\simeq 0$	1.52	0.72	$\simeq 0$	87.54	5.95	1.94	6.23	3.59
ARPANET	$\simeq 0$	0.50	0.13	$\simeq 0$	77.01	3.44	2.58	7.64	4.82
BHVAC	$\simeq 0$	1.45	0.35	$\simeq 0$	64.58	8.57	3.30	8.54	5.17
EON	$\simeq 0$	1.87	0.82	$\simeq 0$	58.09	3.15	1.71	8.92	3.63
METRO	$\simeq 0$	24.45	2.38	$\simeq 0$	30.89	3.62	0.20	14.99	2.77
NSF	$\simeq 0$	21.06	2.77	0.01	60.14	3.75	0.01	8.77	1.47
PACBELL	$\simeq 0$	1.19	0.20	$\simeq 0$	45.13	5.30	1.44	7.14	4.00
VNSL	$\simeq 0$	4.40	0.64	$\simeq 0$	24.07	3.51	0.34	6.82	2.46
Global	$\simeq 0$	24.45	1.00	$\simeq 0$	87.54	4.53	0.01	14.99	3.26

1.04%.

Table 8: Relative gap (%) to the Bonmin solution for the M/M/1 cost function.

Topology	**Penalized BR**		
	min	max	avg
ABOVENET	0	0.23	0.06
ARPANET	0	0.19	0.04
BHVAC	0	0.29	0.04
EON	0	0.72	0.21
METRO	0	6.18	0.76
NSF	0	26.6	5.95
PACBELL	0	0.87	0.18
VNSL	0	0.52	0.12
Global	0	26.6	1.04

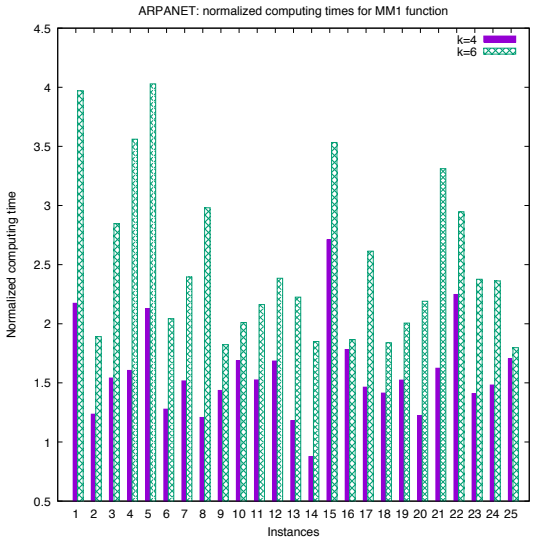
We remind the reader that all algorithms where run with only 2 candidate paths for each OD pair. In order to evaluate how increasing the number of paths affects the quality of the solution and the computing times, we also have run the penalized BR algorithm with 4 and 6 paths for the ARPANET topology (the largest one) and the M/M/1 cost function. These paths were chosen using Yen's algorithm for computing the k-shortest loopless paths in a graph [40]. Figure 3(a) presents the computing times of the penalized BR algorithm with 4 and 6 paths (normalized by

Table 9: Computing times (seconds) for the 800 problem instances.

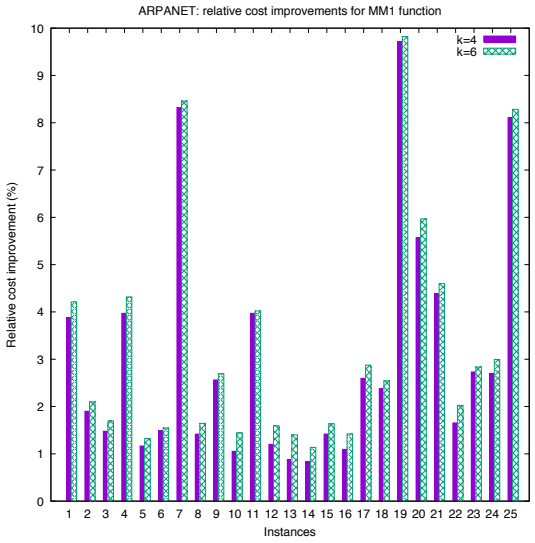
Scenario	**Quadratic Cost**				**M/M/1 Cost**			
	BR	GSA	ACO	Gurobi	BR	GSA	ACO	Bonmin
Min	0.04	1.02	4.73	0.11	0.03	0.75	4.98	0.6
Max	6.75	297.73	48.83	302.3	4.68	400.82	50.95	2638
Average	0.92	30.85	22.16	5.01	0.72	27.45	22.40	725

the computing times with 2 paths) for the first 25 instances. Over the whole set of 100 instances, the computing times increase only by a factor 1.69 (resp. 2.49) on average when passing from 2 to 4 paths (resp. 6). Similar results were obtained with the other topologies and other cost functions, which leads to the conclusion that running the penalized BR algorithm with $2n$ candidate paths yields roughly an increase of computing times by a factor n. This has usually a positive impact on the quality of the solution, and Figure 3(b) shows the relative cost improvements when using 4 and 6 candidate paths instead of 2 for the first 25 instances. Although significant improvements can be observed in some cases, the average relative improvement over the 100 instances is only 2.1 % with 4 paths, and 2.34 % with 6 paths. We however observed that, in contrast to the case of computing times, the benefits of using more candidate paths heavily depend on the particular instance considered.

Remark 3. *The results obtained with the standard best-response algorithm are comparable to those presented above for the penalized best-response algorithm (see Tables 10 and 11). In practice, the results of the standard best-response algorithm are sometimes better, and sometimes worse, but the difference is always in the order of a few percents. On the other hand, the computing times of the standard best-response algorithm are clearly better since its worst execution time over the* 800 *instances is 1.1 seconds for the M/M/1 cost. This is something expected since the penalty term implies an extra computation with respect to the standard best-response algorithm. It is also interesting to note that both versions of the BR algorithm require roughly the same number of iterations. For instance, for the quadratic objective function, the average number of iterations of the penalized BR algorithm over the 800 instances was* 4.63*, whereas it was* 4.53 *for the non-penalized BR algorithm.*



(a) Computing times with 4 and 6 paths (normalized by the computing times with 2 paths).



(b) relative cost improvements (%) with 4 and 6 paths instead of 2 paths.

Figure 3: Impact of passing from 2 candidate paths to 4 or 6 candidate paths on the quality of the solution and on computing times for the first 25 instances of the ARPANET topology.

Table 10: Relative gap (%) to the optimal multi-path solution for the quadratic cost function.

Topology	**penalized BR**			**standard BR**		
	min	max	avg	min	max	avg
ABOVENET	1.28	5.18	3.86	1.42	5.81	4.16
ARPANET	1.05	2.83	1.84	1.20	3.01	1.91
BHVAC	0.24	1.16	0.65	0.15	1.46	0.69
EON	$\simeq 0$	3.37	2.31	$\simeq 0$	3.85	2.47
METRO	2.23	20.16	8.59	2.23	22.67	9.43
NSF	$\simeq 0$	12.97	3.29	0.10	16.92	4.07
PACBELL	1.89	5.27	3.61	2.04	5.46	3.72
VNSL	0.03	7.27	1.91	0.21	7.27	2.31
Global	$\simeq 0$	20.16	3.26	$\simeq 0$	22.67	3.60

6 Conclusion

In this paper, we studied non-linear single-path routing problems, which are known to be extremely hard to solve. Yet, these problems naturally arise when one seeks to optimize a quality of service

Table 11: Relative gap (%) to the optimal multi-path solution for the M/M/1 cost function.

Topology	**penalized BR**			**standard BR**		
	min	max	avg	min	max	avg
ABOVENET	$\simeq 0$	1.52	0.72	$\simeq 0$	1.10	0.46
ARPANET	$\simeq 0$	0.50	0.13	$\simeq 0$	2.99	0.31
BHVAC	$\simeq 0$	1.45	0.35	$\simeq 0$	1.54	0.38
EON	$\simeq 0$	1.87	0.82	$\simeq 0$	4.03	0.52
METRO	$\simeq 0$	24.45	2.38	$\simeq 0$	12.90	1.46
NSF	$\simeq 0$	21.06	2.77	$\simeq 0$	20.85	1.03
PACBELL	$\simeq 0$	1.19	0.20	$\simeq 0$	2.07	0.45
VNSL	$\simeq 0$	4.40	0.64	$\simeq 0$	2.05	0.66
Global	$\simeq 0$	24.45	1.00	$\simeq 0$	20.85	0.67

metric such as the delay. For solving these problems, we have proposed an algorithm inspired from Game Theory in which individual flows are allowed to independently select their path to minimize their own cost function. Given the routing of the other flows, each flow seeks to minimize the sum of its own end-to-end delay, plus a penalty term which plays the role of an incentive for players to take into account the impact of their actions on the others. We have proven the convergence of this algorithm to a Nash equilibrium of the game. In the case of polynomial link latency functions, we have also established a tight upper bound on the approximation ratio of the algorithm. The numerical results obtained with this algorithm are quite interesting: the relative gap to the optimal multi-path solution remains modest (equivalent to that of GSA and ACO), while the execution times are substantially lower than that of the other algorithms. These results suggest that the best-response algorithm is an appropriate solution for large-scale single-path routing problems with non-linear cost functions, where other techniques show their limits.

Acknowledgement

This work was supported by French FUI project NEC (AAP FUI 12).

References

[1] A. Al Sheikh, O. Brun, P. Hladik, and B. Prabhu, A best-response algorithm for multiprocessor periodic scheduling, 23rd Euromicro Conference Real-Time Syst (ECRTS), July 2011, pp. 228–237.

[2] S. Aland, D. Dumrauf, M. Gairing, B. Monien, and F. Schoppmann, Exact price of anarchy for polynomial congestion games, SIAM J Comput 40 (2011), 1211–1233.

[3] E. Altman, T. Basar, T. Jiménez, and N. Shimkin, Routing into two parallel links: Game-theoretic distributed algorithms, J Parallel Distrib Comput 61 (September 2001), 1367–1381.

[4] F. Alvelos and J.V. de Carvalho, Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem, Proc INOC'2003, Evry-Paris, 2003, pp. 7–12.

[5] B. Awerbuch, Y. Azar, and A. Epstein, The price of routing unsplittable flow, Proc Thirty-seventh Ann ACM Symp Theory Comput (STOC '05), Baltimore, MD, ACM, 2005, pp. 57–66.

[6] C. Barnhart, C. Hane, and P. Vance, Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems, Oper Res 48 (2000), 318–326.

[7] M. Belaidouni and W. Ben-Ameur, A superadditive approach to solve the minimum cost single path routing problem: Preliminary results, Proc INOC'2003, Evry-Paris, 2003, pp. 67–71.

[8] M. Belaidouni and W. Ben-Ameur, On the minimum cost multiple-source unsplittable flow problem, RAIRO-Operations Res 41 (2007), 253–273.

[9] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, Mixed-integer nonlinear optimization, Acta Numerica 22 (2013), 1–131.

[10] K. Bhawalkar, M. Gairing, and T. Roughgarden, Weighted congestion games: The price of anarchy, universal worst-case examples, and tightness, ACM Trans. Economics Compu 2 (2014), 14:1–14:23.

[11] E.G. Boman, Infeasibility and negative curvature in optimization, Ph.D. Thesis, Stanford University, Stanford, CA, 1999.

[12] P. Bonami, L. Biegler, A. Conn, G. Cornuejols, I. Grossmann, C. Laird, J. Lee, A. Lodi, F. Margot, N.Sawaya, and A. Waechter, An algorithmic framework for convex mixed integer nonlinear programs, Discr Optim 5 (2008), 186–204.

[13] O. Brun, B.J. Prabhu, and T. Seregina, On the convergence of the best-response algorithm in routing games, Proc ValueTools 2013, Turin, Italy, ICST, 2013, pp. 136–144.

[14] E. Cela, The quadratic assignment problem: Theory and algorithms., Kluwer Academic Publishers, 1998.

[15] P. Coucheney, Auto-optimisation des réseaux sans fil : Une approche par la théorie des jeux, Ph.D. Thesis, Université de Grenoble, 2006.

[16] B. Davie and Y. Rekhter, MPLS technology and applications, Morgan Kaufmann, 2000.

[17] M. Dorigo, Optimization, learning and natural algorithms, Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[18] M. Dorigo, V. Maniezzo, and A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Trans Syst, Man, Cybernetics 26 (1996), 29–41.

[19] S. Durand and B. Gaujal, Complexity and optimality of the best-response algorithm in random potential games, Research report RR-8925, Inria - Research Centre Grenoble – Rhône-Alpes ; Grenoble 1 UGA - Université Grenoble Alpe, June 2016.

[20] A. Fabrikant, C. Papadimitriou, and K. Talwar, The complexity of pure Nash equilibria, Proc Thirty-sixth Ann ACM Symp Theory Comput, New York, NY, ACM, 2004, STOC '04 pp. 604–612.

[21] B. Fortz and M. Thorup, Internet traffic engineering by optimizing OSPF weights, Proc IEEE InfoCom 2000, 19th Ann Joint Conference IEEE Comput Commun Societies, Tel-Aviv, Israel, Vol. 2, IEEE, March 2000, pp. 519–528.

[22] D. Fotakis, S. Kontogiannis, and P. Spirakis, Selfish unsplittable flows, Theoret Comput Sci 348 (2005), 226–239.

[23] D. Fudenberg and J. Tirole, Game Theory, MIT Press, Cambridge, MA, 1991.

[24] J.M. Garcia, A. Rachdi, and O. Brun, "Optimal LSP placement with QoS constraints in Diffserv/MPLS networks," Proc 18th international teletraffic congress (ITC-18), J. Charzinski, R. Lehnert, and P. Tran-Gia (Editors), Elsevier, 2003, Vol. 5 of Teletraffic Science and Engineering, pp. 11 – 20.

[25] J. Geffard, A solving method for singly routed traffic demands in telecommunication networks, Annales des Télécommun 56 (2001), 140–149.

[26] Gurobi Optimization Inc., *Gurobi Optimizer Reference Manual* 2015.

[27] M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L.E. Wolsey (Editors), 50 years of Integer Programming 1958–2008, Springer Berlin, 2010.

[28] L. Kleinrock, Queueing Systems Vol. I: Theory, Wiley Interscience, 1975.

[29] G. Mertzios, Fast convergence of routing games with splittable flows, Proc 2nd Int Conference Theoret Math Foundations Comput Sci (TMFCS), Orlando, FL, July 2009, pp. 28–33.

[30] W. Murray and K.M. Ng, An algorithm for nonlinear optimization problems with binary variables, Comput Optim Appl 47 (2010), 257–288.

[31] A. Orda, R. Rom, and N. Shimkin, Competitive routing in multi-user communication networks, IEEE/ACM Trans Networking 1 (October 1993), 510–521.

[32] P.M. Pardalos, "Continuous approaches to discrete optimization problems," Nonlinear optimization and applications, G. Di Pillo and F. Giannessi (Editors), Springer US, 1996, pp. 313–325.

[33] S. Park, D. Kim, and K. Lee, An integer programming approach to the path selection problems, Proc INOC'2003, Evry-Paris, 2003, pp. 448–453.

[34] M. Pióro and D. Medhi, Routing, flow, and capacity design in communication and computer networks, Elsevier, 2004.

[35] R.W. Rosenthal, A class of games possessing pure-strategy Nash equilibria, Int J Game Theory 2 (1973), 65–67.

[36] R. Schoonderwoerd, O. Holl, J. Bruten, and L. Rothkrantz, Ant-based load balancing in telecommunications networks, Adaptive Behavior 5 (1996), 169–207.

[37] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, Measuring ISP topologies with Rocketfuel, IEEE/ACM Trans Networking 12 (Feb. 2004), 2 – 16.

[38] J. Vallet, O. Brun, and B. Prabhu, A game-theoretic algorithm for non-linear single-path routing problems, Electronic Notes in Discr Math 52 (2016), 77 – 84, INOC 2015 - 7th International Network Optimization Conference.

[39] M. Wang, C.W. Tan, A. Tang, and S.H. Low, How bad is single-path routing?, GLOBECOM 2009. IEEE, 2009, pp. 1–6.

[40] J.Y. Yen, Finding the k shortest loopless paths in a network, Manage Sci 17 (July 1971), 712–716.

A Proof of Theorem 3

Before proving Theorem 3, let us first introduce some notation. Let $\boldsymbol{\pi}$ be a solution computed with the best-response algorithm and let $\boldsymbol{\pi}^*$ be an optimal solution. In the following, to simplify notation, we denote by y_e and y_e^* the traffic volumes $y_e(\boldsymbol{\pi})$ and $y_e(\boldsymbol{\pi}^*)$, respectively. Since $\boldsymbol{\pi}$ is a Nash equilibrium, we have $c_i(\boldsymbol{\pi}) \leq c_i(\pi_i^*, \boldsymbol{\pi}_{-i})$, for all $i \in \mathcal{K}$, which according to Lemma 1 translates into $F(\boldsymbol{\pi}) \leq F(\pi_i^*, \boldsymbol{\pi}_{-i})$ for all i. The deviation of player i from path π_i to path π_i^* partitions the set of links in three separate subsets: those for which the load increases ($e \in \pi_i^* \setminus \pi_i$), those for which it decreases ($e \in \pi_i \setminus \pi_i^*$), and those for which it remains constant. Let us therefore define

$$\alpha_e^i = \begin{cases} 1 & \text{if } e \in \pi_i^* \setminus \pi_i, \\ 0 & \text{otherwise.} \end{cases} \quad \text{and} \quad \beta_e^i = \begin{cases} 1 & \text{if } e \in \pi_i \setminus \pi_i^*, \\ 0 & \text{otherwise.} \end{cases}$$

In order to simplify the presentation of the proof, we shall first prove the result for monomials, and then generalize to arbitrary polynomials. The monomial case contains most of the essential ingredients of the proof and has the added advantage of being notationally lighter. Note that in the course of the proof, we shall use two technical results which are proved at the end of this appendix.

Proposition 6. *If $\ell_e(x) = a_e x^d$, then the approximation ratio of the penalized best-response algorithm is bounded above by $\left(2^{1/(d+1)} - 1\right)^{-(d+1)}$.*

Proof. For all $i \in \mathcal{K}$, the condition $F(\boldsymbol{\pi}) \leq F(\pi_i^*, \boldsymbol{\pi}_{-i})$ can be written as follows

$$0 \le \sum_e a_e \left(\left(y_e + (\alpha_e^i - \beta_e^i)\lambda_i\right)^{d+1} - y_e^{d+1} \right) \tag{24}$$

$$= \sum_e a_e \left(\alpha_e^i \lambda_i \sum_{k=0}^{d} (y_e + \alpha_e^i \lambda_i)^k y_e^{d-k} - \beta_e^i \lambda_i \sum_{k=0}^{d} (y_e - \beta_e^i \lambda_i)^k y_e^{d-k} \right) \tag{25}$$

$$= \sum_e a_e \left(\alpha_e^i \lambda_i \sum_{k=1}^{d} (y_e + \alpha_e^i \lambda_i)^k y_e^{d-k} - \beta_e^i \lambda_i \sum_{k=1}^{d} (y_e - \beta_e^i \lambda_i)^k y_e^{d-k} \right) \tag{26}$$

$$+ \sum_e a_e (\alpha_e^i - \beta_e^i) \lambda_i y_e^d$$

$$\le \sum_e a_e \left(\alpha_e^i \lambda_i \sum_{k=1}^{d} (y_e + \alpha_e^i \lambda_i)^k y_e^{d-k} \right) + \sum_e a_e (\alpha_e^i - \beta_e^i) \lambda_i y_e^d \tag{27}$$

$$\le \sum_e a_e \left(\alpha_e^i \lambda_i \sum_{k=1}^{d} (y_e + y_e^*)^k y_e^{d-k} \right) + \sum_e a_e (\alpha_e^i - \beta_e^i) \lambda_i y_e^d, \tag{28}$$

where equality (25) follows from Lemma 2 with $\gamma = \alpha_e^i \lambda_i$ and $\delta = \beta_e^i \lambda_i$. To get to (26), we have separated the $k = 0$ term in each of the two sums and combined them together.

Summing the inequalities (28) over i and using the fact that $\sum_i (\alpha_e^i - \beta_e^i)\lambda_i = y_e^* - y_e$ yields

$$0 \le \sum_e a_e y_e^* \sum_{k=1}^{d} (y_e + y_e^*)^k y_e^{d-k} + \sum_e a_e y_e^* y_e^d - \sum_e a_e y_e^{d+1} \tag{29}$$

$$= \sum_e a_e y_e^* \sum_{k=0}^{d} (y_e + y_e^*)^k y_e^{d-k} - \sum_e a_e y_e^{d+1} \tag{30}$$

$$= \sum_e a_e \left(y_e + y_e^*\right)^{d+1} - \sum_e a_e y_e^{d+1} - \sum_e a_e y_e^{d+1} \tag{31}$$

$$= \sum_e a_e \left(y_e + y_e^*\right)^{d+1} - 2 \sum_e a_e y_e^{d+1} \tag{32}$$

$$\le (s^{1/(d+1)} + (s^*)^{1/(d+1)})^{d+1} - 2 \sum_e a_e y_e^{d+1}, \tag{33}$$

where the last inequality follows from Lemma 3 with $s = \sum_e a_e y_e^{d+1}$ and $s^* = \sum_e a_e (y_e^*)^{d+1}$. Rearranging the above inequality, we get

$$2F(\boldsymbol{\pi}) \le \left(F(\boldsymbol{\pi})^{1/(d+1)} + F(\boldsymbol{\pi}^*)^{1/(d+1)} \right)^{d+1}, \tag{34}$$

from which we deduce that

$$\frac{F(\boldsymbol{\pi})}{F(\boldsymbol{\pi}^*)} \leq \left(\frac{1}{2^{1/(d+1)}-1}\right)^{d+1}, \tag{35}$$

as was stated in the proposition. □

Proof of Theorem 3. For each monomial of degree j, we shall follow the same steps as for the monomial case until inequality (33) to arrive at

$$0 \leq \sum_j \left(s_j^{1/(j+1)} + (s_j^*)^{1/(j+1)}\right)^{j+1} - 2\sum_j \sum_e a_{e,j} y_e^{j+1}, \tag{36}$$

where $s_j = \sum_e a_{e,j} y_e^{j+1}$ and $s_j^* = \sum_e a_{e,j} (y_e^*)^{j+1}$. Since $(x_1^p + x_2^p)^{1/p} \leq (x_1^q + x_2^q)^{1/q}$ for $p \geq q > 0$, the above inequality becomes

$$2F(\pi) \leq \sum_j \left(s_j^{1/(d+1)} + (s_j^*)^{1/(d+1)}\right)^{d+1}. \tag{37}$$

Applying Lemma 3 to the sum on the right-hand side, we get

$$2F(\boldsymbol{\pi}) \leq \left(F(\boldsymbol{\pi})^{1/(d+1)} + F(\boldsymbol{\pi}^*)^{1/(d+1)}\right)^{d+1}, \tag{38}$$

from which the stated result follows. □

Lemma 2 (Decomposition)**.** *Let γ and δ be two non-negative numbers such that $\gamma\delta = 0$. Let m be a non-negative integer and $x \geq 0$. Then,*

$$(x + (\gamma - \delta))^{m+1} - x^{m+1} = \gamma \sum_{k=0}^{m} (x+\gamma)^k x^{m-k} - \delta \sum_{k=0}^{m} (x-\delta)^k x^{m-k}. \tag{39}$$

Proof. The proof is based on simple algebraic manipulations but we give it for the sake of com-

pleteness. Starting from the left-hand side, we have

$$(x+(\gamma-\delta))^{m+1}-x^{m+1} = (\gamma-\delta)\sum_{k=0}^{m}(x+(\gamma-\delta))^m x^{m-k} \tag{40}$$

$$= \gamma\sum_{k=0}^{m}\left(\sum_j \binom{k}{j}(x+\gamma)^j(-\delta)^{k-j}\right)x^{m-k} \tag{41}$$

$$-\delta\sum_{k=0}^{m}\left(\sum_j \binom{k}{j}(x-\delta)^j\gamma^{k-j}\right)x^{m-k} \tag{42}$$

$$= \gamma\sum_{k=0}^{m}(x+\gamma)^k x^{m-k} - \delta\sum_{k=0}^{m}(x-\delta)^k x^{m-k}, \tag{43}$$

where the last equality follows from the assumption that $\gamma\delta = 0$. □

Lemma 3. *For non-negative numbers a_e, x_e z_e, and non-negative integer m,*

$$\sum_e a_e(x_e+z_e)^{m+1} \le \left(s_x^{1/(m+1)} + s_z^{1/(m+1)}\right)^{m+1} \tag{44}$$

where $s_x = \sum_e a_e x_e^{m+1}$ and $s_z = \sum_e a_e z_e^{m+1}$.

Proof. The proof is an application of Hölder's inequality. Starting from the left-hand side,

$$\sum_e a_e(x_e+z_e)^{m+1} = \sum_e a_e \sum_{k=0}^{m+1}\binom{m+1}{k}x_e^k z_e^{m+1-k} \tag{45}$$

$$= \sum_{k=0}^{m+1}\binom{m+1}{k}\left(\sum_e a_e x_e^k z_e^{m+1-k}\right) \tag{46}$$

$$= \sum_{k=0}^{m+1}\binom{m+1}{k}\left(\sum_e (a_e x_e^{m+1})^{k/(m+1)}(a_e z_e^{m+1})^{(m+1-k)/(m+1)}\right) \tag{47}$$

$$\le \sum_{k=0}^{m+1}\binom{m+1}{k}\left(\sum_e a_e x_e^{m+1}\right)^{k/(m+1)}\left(\sum_e a_e z_e^{m+1}\right)^{(m+1-k)/(m+1)} \tag{48}$$

$$= \sum_{k=0}^{m+1}\binom{m+1}{k}s_x^{k/(m+1)}s_z^{(m+1-k)/(m+1)} \tag{49}$$

$$= \left(s_x^{1/(m+1)} + s_z^{1/(m+1)}\right)^{m+1}, \tag{50}$$

where we have used the Hölder's inequality with $p = (m+1)/k$ and $q = (m+1)/(m+1-k)$ to

get to (48) from (47). □