A Game-theoretic Algorithm for Non-Linear Single-Path Routing Problems

Josselin Vallet, Olivier Brun, Balakrishna Prabhu

To cite this version:
Josselin Vallet, Olivier Brun, Balakrishna Prabhu. A Game-theoretic Algorithm for Non-Linear Single-Path Routing Problems. 7th International Network Optimisation Conference (INOC 2015), May 2015, Varsovie, Poland. hal-01245236v2

HAL Id: hal-01245236
https://hal.archives-ouvertes.fr/hal-01245236v2

Submitted on 14 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Game-theoretic Algorithm for Non-linear Single-Path Routing Problems

Josselin Vallet\(^1\), Olivier Brun\(^2\), Balakrishna Prabhu\(^3\)

CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
Univ. de Toulouse, LAAS, 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. We propose a Best Response algorithm, based on Game Theory, providing 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

1 Introduction

We consider single-path routing problems with an additive and non-linear objective function. These routing problems belong 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, both from a theoretical and from a practical point of view. Even in the simplest case with binary variables, quadratic function and equality constraints, they are known to be NP-hard ([3]).

Several approaches have been proposed to solve non-linear integer programming problems. Some transform the discrete problem into a continuous one (see for example [8]). 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 [7], which seems to scale better while providing fairly good approximations (see Section 3.1 for details).

Heuristics and meta-heuristics have also been used to solved non-linear integer programming problems. Among others, ant-inspired optimization techniques are known to be efficient for solving various routing problems ([10], [6]). In this paper, we use the heuristic method proposed in [6] for comparison purposes (see Section 3.2 for details).

We propose an approximation algorithm, inspired from Game Theory [5], for solving non-linear single-path routing problems. Indeed, we assume that individual flows are allowed to independently select their path to minimize their own cost function. We note that a similar algorithm was proposed in [1] for scheduling of strictly periodic tasks. 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.

1.1 Problem statement

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

We are given a set $\mathcal{K}$ of flows, each one associated with a traffic demand $\lambda_k$. We let $s_k$ and $t_k$ be the origin and destination of flow $k$. Each traffic flow has to be routed in the network over a single path. We let $\Pi_k$ be the set of all paths available for routing traffic between $s_k$ and $t_k$. A routing strategy for flow $k$ is defined as a vector $x_k \in S_k = \{x_k \in \{0, 1\}^{|\Pi_k|} : \sum_{\pi \in \Pi_k} x_{k,\pi} = 1\}$, where $S_k$ is the set of all feasible routing strategies for flow $k$, and the binary variable $x_{k,\pi}$ is 1 if flow $k$ is routed over path $\pi$, and 0 otherwise. The vector $x = (x_k)_{k \in \mathcal{K}}$ will be called a routing strategy. It belongs to the product strategy space $\mathcal{S} = \bigotimes_{k \in \mathcal{K}} S_k$. For each edge $e \in E$, we define the function $y_e : \mathcal{S} \rightarrow \mathbb{R}_+$ as the amount of traffic flowing through link $e$ under strategy $x$. We define the cost of a routing strategy $x \in \mathcal{S}$ as $\phi(x) = \sum_{e \in E} \ell_e(y_e(x))$. We
seek for a routing strategy with minimum cost, that is for the optimal solution of the following optimization problem:

\[
\begin{align*}
\text{minimize} \quad & \phi(x) = \sum_{e \in E} \ell_e(y_e) \\
\text{subject to} \quad & y_e = \sum_{k \in K} \sum_{\pi \in \Pi_k} \delta^e_{\pi} x_{k,\pi} \lambda_{k} \\
& \sum_{\pi \in \Pi_k} x_{k,\pi} = 1 \\
& y_e \leq c_e \\
& x_{k,\pi} \in \{0, 1\}
\end{align*}
\] (OPT-R)

\[1\]

\[2\]

\[3\]

\[4\]

2 Best-response algorithm

The best-response algorithm we propose takes its inspiration from an algorithm of the same name in Game theory [5]. 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 the best response that the player can give for a given strategy of the others. The algorithm consists of players taking turns to adapt their strategy based on the most recent known strategy of the others. This algorithm is similar to the Gauss-Seidel iterative scheme.

The proposed Best-response algorithm converts the optimization problem (OPT-R) into the following game. Let us think of individual flows as players. The number of players is therefore \(N = |K|\). In the following, we denote by \(P = \{1, 2, \ldots, N\}\) the set of players. The game is assumed to be played sequentially with players taking turns in some fixed order until the strategies of the players converge. Let \(\pi_j(n)\) denote the strategy (equivalently, the path) of player \(j\) at the beginning of the \(n\)th turn. If \(k(j)\) is the flow corresponding to player \(j\), we thus have \(\pi_j(n) \in \Pi_{k(j)}\). The vector \(\pi(n)\) describes the strategy of the \(N\) players at iteration \(n\). Using standard Game Theory notation, \(\pi_{-j}(n) = [\pi_1(n), \pi_2(n), \ldots, \pi_{j-1}(n), \pi_{j+1}(n), \ldots, \pi_N(n)]\) is the vector of paths of individual flows other than flow \(j\). We define \(y_e(\pi(n))\) as the traffic routed through link \(e\) at turn \(n\): \(y_e(\pi(n)) = \sum_{j \in P} \delta^e_{\pi_j(n)} \lambda_{k(j)}\)

In its turn, player \(j\) computes its path (using any shortest path algorithm) so as to minimize its own cost, that is, player \(j\) solves the following problem:

\[
\begin{align*}
\text{minimize}_{\pi \in \Pi_{k(j)}} u_j(\pi, \pi_{-j}(n)) = \sum_{e \in \pi} \ell_e(\lambda_{k(j)} + y_e(\pi_{-j}(n))) \\
\end{align*}
\] (OPT-\(j\)
and sets $\pi_j(n+1)$ to that value of $\pi$ that gives the solution. Note that player $j$ solves the same problem as (OPT-R) except that it takes into account only the terms that are affected by its own traffic. Note also that player $j$ deviates from strategy $\pi_j(n)$ to a new strategy $\pi'_j(n)$ if and only if $u_j(\pi'_j(n), \pi_{-j}(n)) < u_j(\pi(n))$. The pseudocode for the heuristic is given in Algorithm 1.

**Algorithm 1** Best-response

**Require:** $\pi(0)$

1: $n \leftarrow 0$
2: repeat
3: for $j = 1 \ldots N$ do
4: $\pi_j(n+1) \leftarrow \text{argmin } (\text{OPT}-j)$
5: end for
6: $n \leftarrow n + 1$
7: until $\pi(n) \neq \pi(n-1)$.
8: return $\pi(n)$

The general convergence of the algorithm is not addressed in this paper. However, when all flows have the same traffic demands ($\lambda_k = \lambda, \forall k$), the convergence of the algorithm directly follows from the fact that the game is a congestion game [9]. For heterogeneous traffic demands, see [4] for a proof of convergence in the case of linear latency functions. Also, in [2], the authors found that the price of anarchy is bounded for affine and polynomial cost functions with non negative coefficients.

3 Global Smoothing Algorithm and Ant Colony Optimization

In this section, we present two existing approximation techniques that can be used to solve (OPT-R). We will use those algorithms for comparison purposes.

3.1 Global Smoothing Algorithm (GSA)

This algorithm was introduced in [7]. Designed to handle non linear optimization with binary variables, it may provide good integer solutions. Its original approach is to use a sequence of non-linear optimization without integer constraints. It optimizes the cost with two penalty functions : 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 are weighted by one parameter,
which is updated at each iteration to gradually bring the current point on
the integer grid. The sequence of penalized problem should form a trajectory
leading to a good approximation of the solution. We refer to [7] for further
details.

3.2 Ant Colony Optimization (ACO)

This meta-heuristic algorithm belongs to the ant colony algorithms family,
which are efficient on problem that requires to find good paths in a graph.

The algorithm we consider was presented in [6]. It is designed to optimize
an MPLS network - a direct application of our problem - and provides good
approximated results within reasonable execution times. Basically, \( L \) ants are
defined, each one exploring the space of solution depending on probability
distributions. At each iteration, each ant construct a solution by routing each
flow, leaving a trace of pheromones on its way, influencing the next ants. The
algorithms stops after a fixed maximum number of iterations. This algorithm
gives a solution after the first iteration, and each subsequent iteration can
improve the solution.

4 Numerical Results

We analyse here the efficiency of the Best Response (BR) algorithm on two
non linear cost functions, compared to GSA and ACO. The optimal multipath
routing is considered as the lower bound to which we will compute a relative
“error”. Simulations are performed on 8 real network topologies (see Table 1),
found in IEEE literature and in [11]. We consider at most 2 shortest paths (in
terms of number of hops) for each source/destination pair. For each network,
we consider a set of 100 full random traffic matrices. For all matrices, there
exists a single-path routing solution such that no link capacity is excedeed.
The number of variables associated to each scenario is shown in Table 1.

For ACO, we consider at most 50 iterations and 5 ants : experiments
on our problems revealed that increasing those parameters did not improve
the results, while increasing execution times. For GSA, two different mul-
triouted starting points were tested : a random point, and a point close to
the monorouted solution obtained with the BR algorithm. Results presented
in the next sections are composed of the best solutions obtained from these
two starting points, and were obtained using Matlab 2013b, on a Intel Core
i5-2430M processor at 2.4 \( GHz \), running under Linux with 4 \( GB \) of memory.
4.1 Quadratic cost function

We consider here the following quadratic cost function: $\ell_e(y_e) = \left(\frac{y_e}{c_e}\right)^2$

The results are summarized in Table 2. First, we can observe that each algorithm provide good optimization results on this cost functions, as the average relative error remains below 4.37% for all configurations. Secondly, GSA seems to provides the best results, but when looking at the execution times shown on table 4, we observe that BR outperforms clearly GSA and ACO, as its worst execution time remains below 1.08 seconds, while ACO reaches 50 seconds and GSA 300 seconds.

<table>
<thead>
<tr>
<th>Topology</th>
<th>N</th>
<th>M</th>
<th>Binary variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOVENET</td>
<td>19</td>
<td>68</td>
<td>664</td>
</tr>
<tr>
<td>ARPANET</td>
<td>24</td>
<td>100</td>
<td>1104</td>
</tr>
<tr>
<td>BHVAC</td>
<td>19</td>
<td>46</td>
<td>684</td>
</tr>
<tr>
<td>EON</td>
<td>19</td>
<td>74</td>
<td>684</td>
</tr>
<tr>
<td>METRO</td>
<td>11</td>
<td>84</td>
<td>220</td>
</tr>
<tr>
<td>NSF</td>
<td>8</td>
<td>20</td>
<td>112</td>
</tr>
<tr>
<td>PACBELL</td>
<td>15</td>
<td>42</td>
<td>420</td>
</tr>
<tr>
<td>VNSL</td>
<td>9</td>
<td>22</td>
<td>140</td>
</tr>
</tbody>
</table>

Table 1
Topologies : number of nodes (N), links (M) and number of binary variables

<table>
<thead>
<tr>
<th>Topology</th>
<th>BR min</th>
<th>BR max</th>
<th>BR avg</th>
<th>GSA min</th>
<th>GSA max</th>
<th>GSA avg</th>
<th>ACO min</th>
<th>ACO max</th>
<th>ACO avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABOVENET</td>
<td>1,42</td>
<td>5,81</td>
<td>4,16</td>
<td></td>
<td></td>
<td></td>
<td>0,17</td>
<td>4,87</td>
<td>2,98</td>
</tr>
<tr>
<td>ARPANET</td>
<td>1,20</td>
<td>3,01</td>
<td>1,95</td>
<td>1,09</td>
<td>5,40</td>
<td>2,57</td>
<td>4,42</td>
<td>8,49</td>
<td>5,99</td>
</tr>
<tr>
<td>BHVAC</td>
<td>0,15</td>
<td>1,46</td>
<td>0,69</td>
<td>0,09</td>
<td>4,35</td>
<td>0,94</td>
<td>3,93</td>
<td>8,30</td>
<td>5,78</td>
</tr>
<tr>
<td>EON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0,09</td>
<td>4,92</td>
<td>1,34</td>
</tr>
<tr>
<td>METRO</td>
<td>2,23</td>
<td>22,67</td>
<td>9,43</td>
<td>1,64</td>
<td>8,90</td>
<td>4,47</td>
<td>2,63</td>
<td>13,32</td>
<td>7,41</td>
</tr>
<tr>
<td>NSF</td>
<td>0,10</td>
<td>16,92</td>
<td>4,07</td>
<td>0,04</td>
<td>4,43</td>
<td>1,76</td>
<td>0,15</td>
<td>7,46</td>
<td>2,21</td>
</tr>
<tr>
<td>PACBELL</td>
<td>2,04</td>
<td>5,46</td>
<td>3,72</td>
<td>0,32</td>
<td>3,00</td>
<td>1,44</td>
<td>2,30</td>
<td>8,00</td>
<td>4,76</td>
</tr>
<tr>
<td>VNSL</td>
<td>0,21</td>
<td>7,27</td>
<td>2,31</td>
<td></td>
<td></td>
<td></td>
<td>0,09</td>
<td>4,92</td>
<td>1,34</td>
</tr>
<tr>
<td>All</td>
<td>0,00</td>
<td>22,67</td>
<td>3,60</td>
<td></td>
<td></td>
<td></td>
<td>0,09</td>
<td>13,32</td>
<td>4,37</td>
</tr>
</tbody>
</table>

Table 2
Quadratic function relative error to OLS (%)
4.2 M/M/1 cost function

In this section, we study the results obtained with the cost function associated to an M/M/1 queueing model (Kleinrock’s delay function): \[ \ell_e(y_e) = \frac{y_e}{c_e - y_e} \]

Results are presented in Table 3: ACO and BR are clearly better than GSA in terms of obtained cost. GSA encounters difficulties on some instances, providing some bad maximum errors. BR provides the best average optimization results, while ACO is best for minimizing the maximum error. But as shown in Table 4, BR seems to offer the best trade-off between optimization efficiency and execution times, as the worst time remains below 0.6 seconds.

<table>
<thead>
<tr>
<th>Topology</th>
<th>BR</th>
<th>GSA</th>
<th>ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
<td>avg</td>
</tr>
<tr>
<td>ABOVENET</td>
<td>≃ 0</td>
<td>1.10</td>
<td>0.46</td>
</tr>
<tr>
<td>ARPANET</td>
<td>≃ 0</td>
<td>2.99</td>
<td>0.31</td>
</tr>
<tr>
<td>BHVAC</td>
<td>≃ 0</td>
<td>1.54</td>
<td>0.38</td>
</tr>
<tr>
<td>EON</td>
<td>≃ 0</td>
<td>4.03</td>
<td>0.52</td>
</tr>
<tr>
<td>METRO</td>
<td>≃ 0</td>
<td>12.90</td>
<td>1.46</td>
</tr>
<tr>
<td>NSF</td>
<td>≃ 0</td>
<td>20.85</td>
<td>1.03</td>
</tr>
<tr>
<td>PACBELL</td>
<td>≃ 0</td>
<td>2.07</td>
<td>0.45</td>
</tr>
<tr>
<td>VNSL</td>
<td>≃ 0</td>
<td>2.05</td>
<td>0.66</td>
</tr>
<tr>
<td>All</td>
<td>≃ 0</td>
<td>20.85</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Table 3
MM1 function relative error to OLS (%)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Quadratic function</th>
<th>MM1 function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BR</td>
<td>GSA</td>
</tr>
<tr>
<td>Min</td>
<td>0.02</td>
<td>1.02</td>
</tr>
<tr>
<td>Max</td>
<td>1.08</td>
<td>297.73</td>
</tr>
<tr>
<td>Average</td>
<td>0.24</td>
<td>30.85</td>
</tr>
</tbody>
</table>

Table 4
Execution time (s) for all tested scenarios

5 Conclusion

The numerical results obtained with the proposed Best Response algorithm are promising: the relative error to optimal solution remains very modest (equivalent to other techniques), while the execution times are substantially
lower than the others. Those results make us think that this algorithm should clearly be considered for large scale problems, where other techniques show their limits. Future work will try to consider the theoretical results associated with the algorithm.

References


