



HAL
open science

An Augmented Lagrangean Approach for the QoS Constrained Routing Problem

Christophe Duhamel, Antoine Mahul

► **To cite this version:**

Christophe Duhamel, Antoine Mahul. An Augmented Lagrangean Approach for the QoS Constrained Routing Problem. 2007. hal-00678369

HAL Id: hal-00678369

<https://hal.science/hal-00678369>

Submitted on 12 Mar 2012

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.

**An Augmented Lagrangean Approach for the QoS
Constrained Routing Problem**

Christophe Duhamel¹, Antoine Mahul²
LIMOS, UMR 6158-CNRS
Université Blaise-Pascal, BP 10125, 63173 Aubière, France.

Research Report LIMOS/RR07-15

¹christophe.duhamel@isima.fr

²antoine.mahul@isima.fr

Abstract

Given a directed graph with capacities on each arc and given a set of commodities, the QoS constrained Routing Problem (QCRP) consists in routing each commodity while respecting a end-to-end upper bound on the delay for each packet. This problem is NP-hard and we propose an augmented Lagrangean approach to solve it. A flow deviation algorithm as well as a projected gradient method are proposed to solve the inner problem in the augmented Lagrangean approach. The delay and the load functions are approximated either by an analytical function or by a neural network. Computational results are reported to show the effectiveness of the projected gradient method and the neural network approximation.

Keywords: routing, quality of service, nonlinear programming, augmented Lagrangean, flow deviation, projected gradient, neural network

Résumé

Étant donné un graphe muni de capacités sur les arcs ainsi qu'un ensemble de demandes, le problème de Routage avec Contraintes de Qualité de Service (QRCP) consiste à router chaque demande tout en respectant une borne supérieure sur le délai de bout en bout pour chaque paquet. Ce problème est NP-difficile et nous proposons une approche de type Lagrangien Augmenté pour le traiter. Nous proposons un algorithme de déviation de flot ainsi qu'une méthode de gradient projeté pour résoudre le sous-problème à chaque itération du Lagrangien Augmenté. Nous utilisons deux types d'approximation pour les fonctions de délai et de charge sur les arcs : une fonction analytique ou un réseau de neurones. Cette approche est testée sur des instances de taille réaliste. Les expérimentations montrent que l'utilisation de réseaux de neurones ainsi que du gradient projeté permettent d'obtenir de bons résultats dans des temps acceptables.

Mots clés : routage, qualité de service, programmation non linéaire, lagrangien augmenté, déviation de flot, gradient projeté, réseau de neurones

Introduction

The demand for a dynamic and efficient usage of network resources is one of the key points in the current evolution of the Internet. However, to support both existing and emerging (VoIP, peer-to-peer, ad-hoc, etc.) multimedia applications with specific Quality of Service (QoS) requirements, additional services and protocols are needed. IETF has proposed several approaches to address QoS requirements in the Internet. DiffServ-based architectures have received the greatest favors of operators and in the industry but the deployment of end-to-end QoS is still limited [11]. Although the basic DiffServ concept (differentiation of packets per traffic classes) is being currently exploited to support voice traffic over IP network (VoIP), handling the traffic in term of performance (delay, loss rate, jitter) is still an open issue [13].

One of the problems is to design effective routing policies that allow QoS constraints to be respected and resource occupation to be optimized. MultiProtocol Label Switching (MPLS) is being considered as one (standardized) approach for scalable QoS provisioning in the Internet and may be seen as a possible answer to Traffic Engineering (TE) needs. We address here the problem of routing a set of demands in a MPLS backbone with QoS requirements expressed in term of end-to-end delays. The classical optimization algorithms that are typically used in TE task to maximize network performance and to balance traffic load suffer from a major issue: they still rely on the assumption that the QoS in routers obeys the standard M/M/1 formula. In this work, we define a multicommodity flow optimization procedure in which the M/M/1 approximation can be substituted by a more realistic evaluation of the delay using a neural network trained on simulated examples. We restrict ourselves to the QoS constraint defined as the end-to-end delay: for each commodity, it corresponds to the delay incurred by a packet between its arrival time at the destination router and its departure time from the origin router. The QoS Constrained Routing Problem (QCRP) is then to route each commodity on the network in order to minimize the overall load on the network – defined as the overall sum of the arc loads – while respecting all QoS requirements. Note that a somewhat different cost function could have been used, namely the maximal congestion on the network (i.e. the utilization factor on the most saturated arc) as discussed in [7].

A mathematical formulation for (QCRP) is presented in Section 1. Then we describe in Section 2 an augmented Lagrangean approach to solve the problem. The subproblem reduces to a nonlinear minimum cost flow problem. It will be solved two different ways: first, by using the flow deviation method, second, by using the projected gradient method. Due to the nature of the objective function, the generation of the improving direction for both methods requires a specific algorithm. In Section 3, two approximations (the classical Kleinrock's delay function and a neural network-based function) are presented for the delay and the load functions. Numerical results will be presented in Section 4 before concluding remarks.

1 Mathematical formulations

Let $G = (N, A, c)$ be a directed network, where N is the set of nodes (routers) and A is the set of arcs (connections). A capacity (bandwidth) $c_a > 0$ is associated with each arc $a \in A$. The set of commodities (services) to be routed on G is defined as $\mathcal{K} = \{(o_i, d_i, q_i, s_i), i = 1 \dots K\}$. For each commodity $k \in \mathcal{K}$, o_k is the origin node, d_k is the destination node, q_k and s_k are respectively the requested bandwidth and the class of service. The bandwidth requirement q_k is considered to be fixed, i.e. the traffic is not subject to variations over the time. The set $\mathcal{S} = \{s_i, i = 1 \dots S\}$ contains the classes of service. To each class of service $s \in \mathcal{S}$, D_s is the upper bound on the end-to-end delay. Thus, for every commodity $k \in \mathcal{K}$, D_{s_k} corresponds to the maximal end-to-end delay for each of its packets.

Let $\mathcal{P}_k = \{p_i^k, i = 1 \dots P_k\}$ be the set of all paths for commodity $k \in \mathcal{K}$. Thus, $\mathcal{P} = \cup_{k \in \mathcal{K}} \mathcal{P}_k$ denotes the set of all the paths. Each path $p \in \mathcal{P}$ can be expressed as a set of arcs. For each commodity k , let $x_p^k \geq 0$ be the percent of bandwidth assigned to the path $p \in \mathcal{P}_k$. Throughout this work, the delay will be considered as an additive function over the arcs. Therefore, the end-to-end delay for a path $p \in \mathcal{P}$ is given by the sum of the delays on each arc a belonging to p . Let $\phi_s(x_a, c_a)$ and $\psi_s(x_a, c_a)$ be respectively the load function and the delay function for the class of service $s \in \mathcal{S}$ on the arc $a \in A$, where x_a is the aggregated bandwidth vector for each class of service. The (QCRP) can be formulated as follows:

$$\begin{array}{ll}
\text{Min} & \Phi(x) = \sum_{a \in A} \sum_{s \in \mathcal{S}} \phi_s(x_a, c_a) \\
\text{s.t.} & \\
& \sum_{p \in \mathcal{P}_k} x_p^k = 1 \quad \forall k \in \mathcal{K} \quad (1) \\
\text{(QCRP)} & x_a^s = \sum_{k \in \mathcal{K} | s_k = s} q_k \sum_{p \in \mathcal{P}_k | a \in p} x_p^k \quad \forall a \in A, \forall s \in \mathcal{S} \quad (2) \\
& d_p^k = \sum_{a \in A | a \in p} \psi_{s_k}(x_a, c_a) \quad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}_k \quad (3) \\
& x_p^k (D_{s_k} - d_p^k) \geq 0 \quad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}_k \quad (4) \\
& x_p^k \geq 0 \quad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}_k
\end{array}$$

Constraints (1) require all the bandwidth to be assigned to the paths of commodity k . Equations (2) compute the bandwidth used on each arc according to the bandwidth that is assigned on each path. Equations (3) compute the end-to-end delay of each path. Finally, constraints (4) are the QoS constraint for each active path, i.e. for each path on which a positive amount of bandwidth is routed. These constraints are nonlinear and nonconvex. It is worth noting that only (1) and (4) define constraints, since x_a^s and d_p^k are auxiliary variables. Moreover, the capacity constraints are not used in (QCRP) since the delay function ψ_s acts as a barrier on the arcs capacity. Thus, the capacity constraints are redundant in our context. (QCRP) is NP-hard as shown by Ben-Ameur and Ouorou [1]. In fact, even the computation of a shortest path given several QoS metrics is NP-hard, as shown by [15]. To our knowledge, [1] is the only work trying to take the end-to-end delay constraint into account. They consider a minimum cost flow problem subject to QoS constraints. They first give a mathematical model before proposing several lower and upper bounds by performing convexifications. No computational results are given, though.

2 Augmented Lagrangean method for (QCRP)

A wide range of efficient methods are well-suited to solve convex multicommodity minimum cost flow problems (see [14] for a comprehensive survey and comparison). Yet, most of them cannot be applied to this problem since the load and the delay functions, $\phi_s(x_a, c_a)$ and $\psi_s(x_a, c_a)$, are nonlinear. Moreover, there is an exponential number of nonlinear, nonconvex QoS constraints (4). Thus, they will be first dualized by performing a Lagrangean relaxation. Let $g_p^k(x) = x_p^k (d_p^k - D_{s_k})$. Then the objective function becomes

$$L(x, \lambda) = \sum_{a \in A} \sum_{s \in \mathcal{S}} \phi_s(x_a, c_a) + \sum_{k \in \mathcal{K}, p \in \mathcal{P}_k} \lambda_p^k g_p^k(x) \quad (1)$$

where $\lambda_p^k \geq 0$ are the Lagrangean coefficients associated with each relaxed QoS constraint. The resulting problem consists in minimizing the nonlinear objective function (1) with linear constraints.

The Lagrangean relaxation cannot be solved directly as the objective function $L(x, \lambda)$ is non-convex (due to $g_p^k(x)$). An augmented Lagrangean relaxation scheme can be built by adding a quadratic penalty term to the objective function (see [6]). Let $h_p^k(x, \lambda, c) = \max\{0, \lambda_p^k + c g_p^k(x)\}$. Then the augmented Lagrangean function is

$$L_c(x, \lambda) = \sum_{a \in A} \sum_{s \in \mathcal{S}} \phi_s(x_a, c_a) + \frac{1}{2c} \sum_{k \in \mathcal{K}, p \in \mathcal{P}_k} \left(\left(h_p^k(x, \lambda, c) \right)^2 - \left(\lambda_p^k \right)^2 \right) \quad (2)$$

$L_c(x, \lambda)$ is locally convex at the neighborhood of x and the resulting problem can be solved by any classical optimization method (i.e. subgradient method, bundle method). At each iteration of the augmented Lagrangean method, the Lagrangean multipliers λ are fixed (λ_t) and the subproblem ($L_{c,t}^+$) to be solved is as follows:

$$(L_{c,t}^+) \quad \begin{array}{l} \text{Min} \quad L_c(x, \lambda_t) \\ \text{s.t.} \\ \sum_{p \in \mathcal{P}_k} x_p^k = 1 \quad \forall k \in \mathcal{K} \\ x_p^k \geq 0 \quad \forall k \in \mathcal{K}, \forall p \in \mathcal{P}_k \end{array} \quad (1)$$

At iteration t , let x_t^+ be the solution of ($L_{c,t}^+$). The multipliers are updated using the formula

$$\lambda_{t+1} = \max\{0, \lambda_t + c_t g(x_t^+)\} \quad (3)$$

The multipliers method is applied on the QoS constraints removal (partial constraint removal). The remaining constraints are linear. This ensures the convergence of the method ([3, 8]). However, as $L(x, \lambda)$ is nonconvex, the augmented Lagrangean method converges towards a local optimum. Thus, the computed solution x^* is only a local optimum. However, this solution is still interesting for two reasons: first, it is feasible, especially with respect to the QoS constraints. Moreover, a lower bound can be computed within the method. This will give us an estimation of the quality of our primal solution.

The choice for the serie $\{c_t\}$ is derived from [6]. It consists in multiplying c_t by a constant factor β (here $\beta = 10$) each time the overall violation of the QoS constraints has not been decreased by a factor γ (here $\gamma = 1.0005$). Thus, the main steps of the algorithm are summarized in Algorithm 1.

2.1 Computation of the gradient

The multipliers method is used in the augmented Lagrangean scheme. At each step, a ($L_{c,t}^+$) subproblem has to be solved. Two approaches will be considered: a flow deviation method and a pro-

Algorithm 1 Multipliers method for (QRCP)

set $\lambda_0 = 0, c_0 = 0, t = 0$

repeat

 solve $(L_{c,t}^+)$ to get x_t

 // multipliers update

$\lambda_{t+1} = \max\{0, \lambda_t + c_t g(x_t)\}$

 // penalty update

if $\|\max\{0, g(x_{t+1})\}\| < \gamma \|\max\{0, g(x_t)\}\|$ **then**

$c_{t+1} = c_t$

else

$c_{t+1} = \beta c_t$

end if

$t \leftarrow t + 1$

until $\|\max\{0, g(x_t)\}\| > 0$

return x_t

jected gradient method. Both methods require the computation of the gradient of the augmented Lagrangean function, $\nabla_x L_c(x, \lambda)$.

Proposition 1. Let λ be a multiplier vector and let c be a strictly positive penalty coefficient. The component of the gradient $\nabla_x L_c(x, \lambda)$ for the path $p \in \mathcal{P}_k$ of the commodity $k \in \mathcal{K}$ is :

$$\frac{\partial L_c(x, \lambda)}{\partial x_p^k} = \sum_{a \in p} \ell_a^{s_k} + h_p^k(x, \lambda, c) \left(d_p^k - D_{s_k} \right) \quad (4)$$

where, for a given arc $a \in A$ and for a given CoS $s \in \mathcal{S}$, the quantity ℓ_a^s is defined as:

$$\ell_a^s = \sum_{s' \in \mathcal{S}} \frac{\partial \phi_{s'}}{\partial x_a^s}(x_a, c_a) + \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k | a \in p} x_p^k h_p^k(x, \lambda, c) \frac{\partial \psi_s}{\partial x_a^s}(x_a, c_a) \quad (5)$$

Proof.

$$\begin{aligned} \frac{\partial L_c(x, \lambda)}{\partial x_p^k} &= \sum_{a \in A} \sum_{s \in \mathcal{S}} \frac{\partial \phi_s}{\partial x_p^k}(x_a, c_a) + \sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} h_{p'}^{k'}(x) \frac{\partial g_{p'}^{k'}(x)}{\partial x_p^k} \\ &= \sum_{a \in A} \sum_{s \in \mathcal{S}} \frac{\partial \phi_s}{\partial x_p^k}(x_a, c_a) + \sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} h_{p'}^{k'}(x) \frac{\partial x_{p'}^{k'}}{\partial x_p^k} \left(\sum_{a \in p'} \psi_{s_{k'}}(x_a, c_a) - D_{s_{k'}} \right) \\ &\quad + \sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} h_{p'}^{k'}(x) x_{p'}^{k'} \frac{\partial}{\partial x_p^k} \left[\sum_{a \in p'} \psi_{s_{k'}}(x_a, c_a) - D_{s_{k'}} \right] \end{aligned} \quad (6)$$

The first term of (6) can be written as:

$$\begin{aligned} \sum_{a \in A} \sum_{s \in \mathcal{S}} \frac{\partial \phi_s}{\partial x_p^k}(x_a) &= \sum_{a \in A} \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{S}} \frac{\partial \phi_s}{\partial x_a^r}(x_a) \frac{\partial x_a^r}{\partial x_p^k} \\ &= \sum_{a \in p} \sum_{s \in \mathcal{S}} \frac{\partial \phi_s}{\partial x_a^{s_k}}(x_a, c_a) \quad \text{since } \frac{\partial x_a^r}{\partial x_p^k} = \begin{cases} 1 & \text{if } a \in p \text{ and } r = s_k \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \quad (7)$$

The second term of equation (6) can also be simplified:

$$\sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} h_{p'}^{k'}(x) \frac{\partial x_{p'}^{k'}}{\partial x_p^k} \left(\sum_{a \in p'} \psi_{s_{k'}}(x_a, c_a) - D_{s_{k'}} \right) = h_p^k(x) \left(\sum_{a \in p} \psi_{s_k}(x_a, c_a) - D_{s_k} \right) \quad (8)$$

Considering the third term of equation (6):

$$\sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} x_{p'}^{k'} h_{p'}^{k'}(x) \frac{\partial}{\partial x_p^k} \left[\sum_{a \in p'} \psi_{s_{k'}}(x_a, c_a) - D_{s_{k'}} \right] \quad (9)$$

$$= \sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} x_{p'}^{k'} h_{p'}^{k'}(x) \left(\sum_{a \in p'} \sum_{s \in \mathcal{S}} \frac{\partial \psi_{s_{k'}}}{\partial x_a^s}(x_a, c_a) \frac{\partial x_a^s}{\partial x_p^k} \right) \quad (10)$$

$$= \sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} x_{p'}^{k'} h_{p'}^{k'}(x) \sum_{a \in p' \cap p} \frac{\partial \psi_{s_{k'}}}{\partial x_a^{s_k}}(x_a, c_a) \quad (11)$$

The term (11) can now be defined on the arcs:

$$\sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'}} x_{p'}^{k'} h_{p'}^{k'}(x) \sum_{a \in p' \cap p} \frac{\partial \psi_{s_{k'}}}{\partial x_a^{s_k}}(x_a, c_a) = \sum_{a \in p} \sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'} | a \in p'} x_{p'}^{k'} h_{p'}^{k'}(x) \frac{\partial \psi_{s_{k'}}}{\partial x_a^{s_k}}(x_a, c_a) \quad (12)$$

Thus, by considering (7), (8) and (12), the equation (6) can be reformulated:

$$\begin{aligned} \frac{\partial L_c(x, \lambda)}{\partial x_p^k} &= \sum_{a \in p} \left[\sum_{s \in \mathcal{S}} \frac{\partial \phi_s}{\partial x_a^s}(x_a, c_a) + \sum_{k' \in \mathcal{K}} \sum_{p' \in \mathcal{P}_{k'} | a \in p'} x_{p'}^{k'} h_{p'}^{k'}(x) \frac{\partial \psi_{s_{k'}}}{\partial x_a^{s_k}}(x_a, c_a) \right] \\ &\quad + h_p^k(x) \left(\sum_{a \in p} \psi_{s_k}(x_a, c_a) - D_{s_k} \right) \quad \square \end{aligned}$$

This result is important as it shows that the gradient is made of two components: a pseudo cost ℓ_a^s on the arcs and a path-based cost. Both the following methods will use this result to compute an improving direction.

2.2 Solving (L_t^+) using the flow deviation method

The inner problem $(L_{c,t}^+)$ in the multipliers method consists in minimizing a convex objective function subject to linear constraints. Thus it can be solved by an adaptation of the Frank-Wolfe

method to network routing problems: the flow deviation method [10, 5]. At each iteration, a linearization of the objective function is performed at the current point x_t to get an extremal point \bar{x}_t . Then, the linear combination $x_{t+1} = x_t + \theta(\bar{x}_t - x_t)$ minimizing the objective function $L_c(x, \lambda_t)$ is computed. The method stops as soon as no sufficient improvement can be obtained. In the context of classical routing problems, the problem can be separated over the commodities. Computing the extremal point reduces to the computation of a shortest path for each commodity and computing the optimal linear combination can be seen as a fair re-routing of some amount of flow from the old flow x_t to the new flow \bar{x}_t . In the context of (QCRP), the computation is a bit more complex due to the structure of the gradient, as previously seen in Section (2.1).

The computation of the solution minimizing the gradient (4) cannot be reduced to the computation of a shortest path anymore as the last term of this expression depends on the path. However, it can still be separated over the commodities. Thus, one has to compute an extreme point (that is, a path) for each commodity. Let ℓ_a^s be the arc-based cost and let c_p^k be the path-based cost in the gradient.

$$c_p^k = h_p^k(x) \left(\sum_{a \in p} \psi_{s_k}(x_a, c_a) - D_{s_k} \right) \quad (13)$$

One can note c_p^k can only be positive for paths that have already been used in the previous solutions. Let $\mathcal{H}^k(x) = \{p \in \mathcal{P}^k \mid h_p^k \neq 0\}$ be the set containing the active paths violating the QoS constraints as well as some paths that once were active. Then, for each commodity, the algorithm works in two steps (see Algorithm 2): first a shortest path \bar{p} is computed using the first term ℓ_a^s as the arc cost. Then the path $p^* \in \mathcal{H}^k(x) \cup \{\bar{p}\}$ minimizing the whole expression (4) is computed.

Algorithm 2 Computation of the improving path in flow deviation

```

set  $\delta^* = \infty, p^* = \emptyset$ 
compute the shortest path using  $\ell_a^s$  to get  $\bar{p}$ 

for all path  $p \in \mathcal{H}^k(x) \cup \{\bar{p}\}$  do
  compute  $\delta_p^k = \sum_{a \in A} \ell_a^s + c_p^k$ 
  if  $\delta_p^k < \delta^*$  then
     $\delta^* \leftarrow \delta_p^k$ 
     $p^* \leftarrow p$ 
  end if
end for

return  $p^*$ 

```

In the classical version of the flow deviation method, the first order Taylor approximation is a lower bound as the problem is convex. Thus, it can be used to define a stopping criterion as the gap between the lower bound and the current solution (upper bound) is known. This cannot be used in our context as $L(x, \lambda)$ is not convex. Another stopping criterion have to be used. Some consider the norm of the gradient, which is hard to compute here since only the components related to active paths are known. Moreover L_c is minimized on the set of compatible multiflows, thus the first order optimality conditions do not reduce to a null gradient. A solution consists in stopping the algorithm when the improving direction does not improve the current point enough.

2.3 Solving (L_t^+) using the projected gradient method

Another approach consists in solving the routing problem over a fixed set of paths before adding new paths in a way similar to the flow deviation iteration. A full optimization of the flows over the set of active paths is performed before looking for new paths. This approach is similar to the column generation method in mixed integer linear programming. The new paths to be added are the shortest paths, with respect to the first derivatives of the objective function. This technique has been cited by Bertsekas [4] within a projected Newton method and the algorithm is summarized in Algorithm 3.

Algorithm 3 Column generation algorithm for (QCRP)

```

// initialization
let  $x_0$  be an initial feasible solution
let  $\overline{\mathcal{P}}_k$  be the set of active paths for commodity  $k$ 
set stop = false

repeat
  // fixed-dimension optimization
  compute the optimal flows over the set  $\cup_{k \in \mathcal{K}} \overline{\mathcal{P}}_k$ 

  // column generation
  compute the shortest paths  $\overline{p}_k, \forall k \in \mathcal{K}$  with respect to the gradient

  // stopping criterion
  if  $\overline{p}_k \in \overline{\mathcal{P}}_k, \forall k \in \mathcal{K}$  then
    stop  $\leftarrow$  true
  end if

  // update the pool
   $\overline{\mathcal{P}}_k \leftarrow \overline{\mathcal{P}}_k \cup \{\overline{p}_k\}, \forall k \in \mathcal{K}$ 
until stop = true

return  $p^*$ 

```

The projected gradient will be used in the first step of the loop to compute the optimal flows over the set of active paths. The method has been proposed by Goldstein in 1964 and relies on a projection operator P to compute the projection $P_{\mathcal{X}}(x)$ of x on a convex set \mathcal{X} . At each iteration, the improving direction at the current point is computed. Then the optimal step on this direction is computed and the resulting point is projected on \mathcal{X} to give the new point. The algorithm stops when the new point is close enough to the old point.

In the context of (DCRP), \mathcal{X} is the convex set of compatible multiflows and the optimal step is computed by using the generalized Armijo rule proposed by Bertsekas [2].

The column generation step consists in generating improving paths once the optimal flows have been computed. The procedure is simpler than the computation of the improving path in the flow deviation method as $c_p^k \neq 0$ only if $p \in \mathcal{H}^k(x)$, due to the nature of the algorithm. Thus we are looking for new paths such that $c_p^k = 0$. It is sufficient to compute a shortest path with respect to ℓ_a^s and check if this is a new and improving path. This simplified version is summarized in Algorithm 4.

Algorithm 4 Computation of the improving path in column generation

$p^* = \emptyset$
compute the shortest path using ℓ_a^s to get \bar{p}

if $\sum_{a \in \bar{p}} \ell_a^s \leq \sum_{a \in p} \ell_a^{s_k} + c_p^k, \forall p \in \mathcal{H}^k(x)$ **then**
 $p^* \leftarrow \bar{p}$
end if

return p^*

3 Approximating the functions ϕ_s and ψ_s

We propose two ways to compute the load and the delay functions, $\phi_s(x_a, c_a)$ and $\psi_s(x_a, c_a)$. The first approach (the classical one) is based on an analytical formulation. It assumes strong hypotheses on the traffic (Poisson traffic, independent flows). Let $f_a = \sum_{s \in \mathcal{S}} x_a^s$ be the aggregated flow on the arc a . Under such hypotheses, the average delay is then $\widetilde{\phi}_s(x_a, c_a) = \frac{f_a}{c_a - f_a}$. This approximation is mostly referred as the Kleinrock's delay function. By using Little's theorem, the load can be obtained from the delay: $\widetilde{\phi}_s(x_a, c_a) = x_a^s \widetilde{\psi}_s(x_a, c_a)$. Those functions have interesting properties: they are \mathcal{C}^∞ , convex and they define a barrier on the arc capacity. As both $\widetilde{\phi}_s(x_a, c_a)$ and $\widetilde{\psi}_s(x_a, c_a)$ are computed on the aggregated flow, they are identical for each class of service which means that no differentiation of service can be done.

The second approach tries to compute a more realistic approximation with respect to the differentiation of service (DiffServ). The QoS is usually expressed in terms of average response time and / or loss rate. It is difficult to express it analytically in terms of the incoming traffic characteristics. Thus, it may be particularly interesting to train a feed-forward neural network as a "black-box" on simulation data for the evaluation of the QoS values. As the delay is monotonically increasing with respect to the incoming traffic rates, inclusion of this prior knowledge into the training procedure can lead to better generalization. Here, neural networks (multi-level perceptron) are trained to approximate $\phi_s(x_a, c_a)$ and $\psi_s(x_a, c_a)$. By construction, these two functions are derivable. The monotonicity is imposed by a penalty algorithm during the learning procedure (see [12]). However, convexity can only be assumed.

In the augmented Lagrangean method, traffic flows are only characterized by their bandwidth. After aggregating these traffic rates per class of service, the neural network will be able to estimate the mean delay for each class since it was trained to approximate the function $\{x_a^s\}_{s \in \mathcal{S}} \rightarrow \{d_a^s\}_{s \in \mathcal{S}}$, where d_a^s is the average delay for the class s on the arc a .

4 Numerical results

We consider several randomly generated planar network instances, whose size ranges from 10 nodes, 20 edges and 40 commodities to 30 nodes, 78 edges and 180 commodities. The program has been coded in C and compiled with gcc version 3.3.2 on a Pentium 4 computer with 4Gb memory running linux 2.6.6.

Table 1 displays the results for all the instances and for the analytical delay and load approximations (by using Kleinrock's function). For each instance, five strategies are compared. The first two strategies, "FD" and "FDh", correspond to the flow deviation algorithm running on (QCRP) without the QoS constraints. "FDh" uses some path removal procedures as explained in [9]. The

last three strategies, “AL+FD”, “AL+FDh”, and “AL+PG”, consists in solving (QRCP) using the augmented Lagrangean method described in Section 2. The subproblem is solved by respectively the classical flow deviation, the flow deviation with path removal and the projected gradient method. For each strategy, the value of the optimal solution $\Phi^* = \Phi(x^*)$, the maximal violation $\|g\|_\infty$ of the solution over all the paths, the number n_{vio} of active paths violating QoS constraints over the active paths n_{tot} , and the CPU time in seconds are reported. The maximal violation is computed as follows: $\|g\|_\infty = \max_{p \in \mathcal{P}_k, k \in \mathcal{K}} \max \{0, g_p^k(x^*)\}$, where $g_p^k(x^*)$ denotes the QoS constraint on the path p for the demand k (cf. the constraints (4) in (QCRP)). In the objective function, the analytical approximation function ϕ_s is kept in all cases for two reasons: first, this evaluation function is not critical because it only gives a direction for the optimization, and also because it is a barrier function for the capacity on the arcs, which simplifies sub-problems.

instance	strategy	Φ^*	$\ g\ _\infty$	n_{vio}/n_{tot}	cpu (s)
T_10_20	FD	27.780763	1.266620×10^{-1}	13 / 48	44
	FDh	27.780651	1.260117×10^{-1}	12 / 46	12
	AL+FD	30.307937	4.733780×10^{-5}	9 / 48	20228
	AL+FDh	28.215005	9.886708×10^{-7}	4 / 48	477
	AL+PG	29.214939	0.	0 / 37	131
T_10_60	FD	60.211874	3.386479×10^{-1}	4 / 105	23
	FDh	60.211124	3.406511×10^{-1}	2 / 101	13
	AL+FD	61.975698	1.664052×10^{-5}	3 / 165	2841
	AL+FDh	61.733936	1.488083×10^{-8}	1 / 240	50851
	AL+PG	60.460260	0.	0 / 99	1
T_20_40	FD	46.993231	2.243466×10^{-1}	25 / 152	72
	FDh	46.993028	2.237437×10^{-1}	25 / 150	45
	AL+FD	48.402162	1.055846×10^{-6}	3 / 305	10796
	AL+FDh	48.299828	8.354941×10^{-7}	4 / 273	2069
	AL+PG	47.494344	0.	0 / 107	81
T_20_120	FD	92.888822	1.461434×10^{-1}	14 / 349	198
	FDh	92.888238	1.454175×10^{-1}	15 / 358	159
	AL+FD	95.899077	1.599609×10^{-6}	9 / 638	8184
	AL+FDh	93.326225	3.660244×10^{-7}	1 / 409	7692
	AL+PG	92.929370	0.	0 / 303	24
T_30_180	FD	164.440426	5.961343×10^{-3}	28 / 1022	718
	FDh	164.437816	4.764601×10^{-3}	29 / 1034	562
	AL+FD	164.437284	0.	0 / 973	31270
	AL+FDh	164.437470	9.992447×10^{-7}	3 / 982	2539
	AL+PG	164.437230	0.	0 / 939	109

Table 1: results for the analytical approximation.

As can be seen, taking the QoS into account (strategy AL) consumes a lot more time as it calls iteratively the FD algorithm. The difference can be really high, as for the instance T_10_60. However, the computed solutions using the AL strategy produce a very limited violation on the QoS, and for only few paths. In general, the number of paths carrying the traffic is higher since AL tries to use more paths to reduce the delay on each path. This has an impact on the value of the objective function. Values computed by FD and FDh are lower because QoS constraints are not taken into account. For the instance T_30_180, the FD strategy behaves rather poorly and stops too early. Thus, in that case, the AL strategy produces a better solution.

Table 2 displays a comparison between the analytical (KL) approximation and the neural network (NN) approximation (column “approx.”). For the neural network, two traffic scenarios (Poissonian sources and OnOff sources, which better approximate the nature of the traffic) are

considered. As the neural network consumes a lot of time, the experiments have only been done on a small instance (10 nodes, 22 edges and 40 commodities). In order to measure the impact of the global congestion, several throughputs ($\tau = 0.9, 1.0$ and 1.1) have been tested. Each throughput τ is a common multiplying factor for all the demands. Thus, the bandwidth demands are both equally and globally increased. For each throughput and each approximation, three strategies have been considered: the flow deviation with path reduction and without the QoS constraints (“FDh”), the augmented Lagrangean with either the FDh approach (“AL+FDh”) or the projected gradient (“AL+PG”).

τ	approx.	strategy	Φ^*	$\ g\ _\infty$	n_{vio}/n_{tot}	cpu (s)
0.9	KL	FDh	18.190555	3.76×10^{-3}	2 / 42	12
		AL+FDh	18.190524	0.	0 / 39	640
		AL+PG	18.190524	0.	0 / 38	0.02
	Poisson	FDh	18.190555	1.67×10^{-3}	1 / 42	13
		AL+FDh	18.190524	0.	0 / 39	5358
		AL+PG	18.190524	0.	0 / 38	0.2
	OnOff	FDh	18.190555	1.90×10^{-3}	2 / 42	13
		AL+FDh	18.190524	0.	0 / 39	6461
		AL+PG	18.190524	0.	0 / 38	0.2
1.0	KL	FDh	22.455972	2.02×10^{-2}	7 / 45	16
		AL+FDh	22.483721	0.	0 / 43	3320
		AL+PG	22.483877	1.48×10^{-7}	1 / 36	0.7
	Poisson	FDh	22.455972	1.04×10^{-2}	4 / 45	17
		AL+FDh	22.456047	0.	0 / 40	13312
		AL+PG	22.456090	0.	0 / 38	4
	OnOff	FDh	22.455972	2.25×10^{-1}	5 / 45	17
		AL+FDh	22.730098	8.08×10^{-8}	1 / 61	6575
		AL+PG	22.475819	0.	0 / 40	41
1.1	KL	FDh	27.780724	8.28×10^{-2}	8 / 46	14
		AL+FDh	28.384249	1.71×10^{-5}	3 / 69	15051
		AL+PG	28.029748	0.	0 / 37	84
	Poisson	FDh	27.780724	9.22×10^{-2}	5 / 46	16
		AL+FDh	28.384249	9.00×10^{-9}	2 / 69	10526
		AL+PG	27.803175	2.62×10^{-6}	2 / 45	41
	OnOff	FDh	27.780724	1.38	8 / 46	16
		AL+FDh		Failed		
		AL+PG	28.218878	4.67×10^{-7}	1 / 40	288

Table 2: results for both Kleinrock and NN approximations.

First, we can observe that the AL+FDh approach converges towards a feasible solution or a *quasi*-feasible solution. As the value $\|g\|_\infty$ remains small, output from the algorithm can be used after a cleaning step. This holds for all values of τ . The FDh approach does not take the QoS constraints into account. Therefore, the QoS model cannot have any impact on the behavior of this method. This explains why, for any given value of τ , the optimal value of FDh remains the same whatever the approximation. For the same reason, the number of violated QoS constraints n_{vio} is higher than for AL. This number increases with higher congestion (higher throughput), along with the number of active paths (which is a classical result). About the AL approach, it can be noted that the impact of the delay model on the solution increases as the overall traffic load (i.e. the throughput τ) increases. For moderate values of τ , AL gives solutions with fewer active paths. However this situation becomes different when the network comes close to saturation (to the QoS constraint point of view). At this point, it is difficult to keep a small number of active paths and the solution must use a lot more paths (i.e. $\tau = 1.0$ with OnOff neural approximation of delay).

The last line of the table should indicate that the problem has become infeasible with respect to the OnOff QoS constraint. It still remains feasible with less accurate QoS approximations, namely Kleinrock and neural approximation with Poissonian traffic.

Overall, these experiments are useful in proving that the way QoS is approximated has a strong impact on the routing solution computed. According to the QoS approximation used, the routes that are obtained can differ significantly. It is worth mentioning that the higher the overall demand, the better balanced the load throughout the network. Moreover, the numerical results show that the neural network approach provides reliable estimates of the delay and it can be successfully applied on a simple context. It shows promises for QoS-based traffic control, routing and congestion avoidance in multiservice telecommunication networks.

Conclusion

The (QRCP) corresponds to the integration of QoS constraints, namely the end-to-end delay constraints, into routing problems. Such constraints are hard to take into account as they rely on path rather than on arcs. We have proposed an augmented Lagrangean method to solve the nonlinear problem. This approach involves the computation of an optimal solution to a routing problem with convex costs. Two algorithms have been proposed: a flow deviation approach and a projected gradient approach. The numerical results show that the latter approach is a lot more efficient in many aspects (CPU time, quality of the solution, number of active paths in the solution). It allows telecom instances of realistic size to be solved in a reasonable amount of time. Two functions have been tested to model the delay: the analytical M/M/1 approximation and a neural network. The neural network provides a better approximation as it is able to take the differentiation of service (DiffServ) into account. It has been trained on two traffic scenarios (Poisson and OnOff). The numerical results show that the overhead when using a neural network instead of an analytical function can be significant. The solutions produced can also differ significantly since the neural network provides a more realistic approximation.

As the augmented Lagrangean method cannot guaranty the optimality of the solution, an exact method still remain to be developed, even if the gaps produced here are small. Another approach could be to embed the augmented Lagrangean into a multistart procedure to try to reach the optimal solution. Other lines of work would be to deal with the minimization of the maximal congestion and consider end-to-end delay and loss rate constraints.

References

- [1] Ben-Ameur, W., and Ouorou, A., “Mathematical models of the delay constrained routing problem”, *Algorithmic Operations Research*, 1(2), 2006.
- [2] Bertsekas, D., “On the Goldstein-Levitin-Polyak gradient projection method”, *IEEE Transactions on Automatic Control*, AC-21(2), pp. 164–174, 1976.
- [3] Bertsekas, D., “Constrained Optimization and Lagrange Multiplier Methods”, Academic Press, 1982.
- [4] Bertsekas, D., and Gafni, E., “Projected Newton methods and optimization of multicommodity flows”, *IEEE Transactions on Automatic Control*, AC-28(12), pp. 1090–1096, 1983.
- [5] Bertsekas, D., and Gallager, R., *Data Networks, second edition*, Prentice-Hall, 1992.

- [6] Bertsekas, D., *Nonlinear Programming, second edition*, Athena Scientific, 1999.
- [7] Bienstock, D., and Raskina, O., “Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem”, *Mathematical Programming*, 91(3), pp. 479–492, 2002.
- [8] Conn, A.R., and Gould, N., and Sartenaer, A., and Toint, P.L., “Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints”, *SIAM J. on Optimization*, 6(3), pp. 674–703, 1996.
- [9] Duhamel, C., and Mahey, P., “Multicommodity flow problems with a bounded number of paths: A flow deviation approach”, *Networks*, 49(1), pp. 80–89, 2007.
- [10] Fratta, L., Gerla, M., and Kleinrock, L., “The flow deviation method: An approach to store-and-forward communication network design”, *Networks*, 3, pp. 97–133, 1973.
- [11] Giordano, S., Potts, M., Smirnov, M., and Ventre, G., “Advances in QoS”, *IEEE Communications Magazine*, 41(1), pp. 28–29, 2003.
- [12] Mahul, A., and Aussem, A., “Learning with monotonicity requirements for optimal routing with end-to-end quality of service constraints”, in *Proceedings of the 14th European Symposium on Artificial Neural Networks (ESANN’2006)*, pp. 455–460, Bruges, Belgium, 2006.
- [13] Mykoniati, E., and al., “Admission control for providing QoS in DiffServ IP networks: the TEQUILA approach”, *IEEE Communications Magazine*, 41(1), pp. 38–44, 2003.
- [14] Ouorou, A., Mahey, P., and Vial, J.-P., “A survey of algorithms for convex multicommodity flow problems”, *Management Science*, 47(1), pp. 126–147, 2000.
- [15] Wang, Z., “On the complexity of quality of service routing”, *Information Processing Letters*, 69, pp. 111–114, 1999.