1. Introduction

In [?], we have proposed a separable augmented Lagrangian algorithm with multidimensional scaling (SALAMS) for nonlinear large-scale programs. This algorithm can be seen as an extension of the proximal decomposition method (see [?]) to non convex programs. It was designed for nonlinear separable problems with coupling constraints and is based on the separable augmented Lagrangian algorithm [?] and a rewriting of the feasible set where this latter is multiplied by a diagonal positive definite matrix. SALAMS was extensively studied in [?] and [?] where we showed that it possessed the global convergence properties, when the objective function is assumed convex and smooth and constraints are linear. An extension of this main convergence result is presented in [?] where near optimal solutions are considered for the subproblems generated by SALAMS.

In this paper, we propose a new version of SALAMS for routing problems. This version is based on the identification of the optimal paths combined with a descent method to meet the required traffic demand. Notice that the descent directions belong to the kernel of the constraints matrix and are obtained via a Newton method. This paper is presented as follows: in Section ??, we present an arc-path formulation of the routing problem which arises as a subproblem in telecommunication networks design. Section ?? presents the new and previous versions of SALAMS. Section ?? concerns the numerical experiments. In that section, we compare the two versions, the results show that the new one is more efficient than the previous version. Finally Section ?? will conclude this paper.

2. The Routing Problem: the mathematical model

Data routing arises in telecommunication networks design when packets have to be forwarded from a specific source to a specific destination. Here, we consider a datagram network where the packets forwarded from one router to another can take different paths simultaneously because of congestion. In others words, different routes are used simultaneously between the same origin node and same destination node. We also consider a static routing model where the objective function represents the average packet delay given by Kleinrock’s function [?].

We consider here an arc-path formulation. The network is represented by a graph \( G = (N, A) \) where \( N \) is the set of nodes (routers) and \( A \) the set of link between the nodes. For each link \( a \) between two nodes \( i \) and \( j \), we associate the link capacity \( C_a \) given in bits/sec. We denote by \( K \) the number of communications and for each communication \( k \), we denote by \( r_k \) the required flow between source node \( S_k \) and destination node \( D_k \). Let \( N_k \) be the number of path used by the flow \( k \). We also denote by \( x_{a} \) the aggregate flow on link \( a \) and \( x_{kp} \) the amount of flow \( k \) using the \( p^{th} \) path of \( k \). Finally we set that \( X = (x_1, \ldots, x_n, x_{11}, \ldots, x_{1N_1}, \ldots, x_{K1}, x_{K2}, \ldots, x_{KN_K}) \), \( x_{kp} = 1 \) if the \( p^{th} \) path of flow \( k \) uses link \( a \) and 0 otherwise.
We next present the routing problem modeled as a nonlinear multi-commodity flow problem.

\[
\min \Phi(X) = \sum_{a=1}^{n} \Phi_a(x_a) = \frac{x_a}{C_a - x_a} \\
\text{s.t.} \\
\sum_{k=1}^{K} \sum_{p=1}^{N_k} \pi_{kp}^a x_{kp} = x_a, \quad \forall \ a = 1, \cdots, n \\
\sum_{p=1}^{N_k} x_{kp} = r_k, \quad \forall \ k = 1, \cdots, K \\
0 \leq x_a \leq C_a, \quad \forall \ a = 1, \cdots, n \\
0 \leq x_{kp} \quad \forall \ k = 1, \cdots, K; \ p = 1, \cdots, N_k
\]

Remark that \(N_k\) is not known in advance. In fact, the paths will be generated at each iteration. Constraints (2) ensure that the collection of flow whose paths use link \(a\) is equal to the total amount of flow on this link. Constraints (3) ensure that the flow requirement is met for each commodity \(k\) and finally constraints (4) and (5) are respectively capacity constraints and positivity constraints.

3. SALAMS for Routing Problem

SALAMS was designed for the following block-structured convex program:

\[
\min \sum_{j=1}^{n} f_j(x_j) \\
\text{s.t.} \\
\sum_{j=1}^{n} (A_j x_j - a_j) = 0 \\
x_j \in S_j, \quad \forall \ j = 1, \cdots, n
\]

where the functions \(f_j : \mathbb{R}^{n_j} \rightarrow \mathbb{R}\) are smooth and convex; \(A_j\) are \(m \times n_j\) matrices; \(a_j\) are a given \(m\) vectors and the nonempty set \(S_j\) are supposed to be convex, closed and bounded subsets of \(\mathbb{R}^{n_j}\).

We skip the details of the successive transformations leading to SALAMS, for more details see [?] where we showed that SALAMS possessed the global convergence properties. Recall that to obtain SALAMS, the problem (6) - (7) was rewritten by multiplying the coupling constraints (2) with a diagonal positive matrix \(\Lambda\) representing the scaling of these constraints. In addition allocation vectors \(y_j \in \mathbb{R}^m\) such as \(\Lambda(A_j x_j - a_j) = y_j\) and \(\sum y_j = 0\) were considered to decouple the model. So we obtained an equivalent problem. When we apply the Lagrangian algorithm on this latter, we obtain the following algorithm so-called SALAMS.

SALAMS

1. Initialization: choose \(u^1, \epsilon > 0, \Lambda_1 > 0\) and \(y_j^0 \in \mathbb{R}^m\) such that \(\sum y_j^0 = 0\)
2. \( \forall j = 1, \ldots, n \) compute
\[
x_j^t = \arg \min_{x_j \in S_j} \left\{ f_j(x_j) + \langle u^t, A_j x_j - a_j \rangle + \frac{1}{2} \| A_i (A_j x_j - a_j) + y_j^{t-1} \|^2 \right\}
\]

3. Residual computation
\[
R^t = \sum_{j=1}^{n} (A_j x_j - a_j); \quad \text{If } \| R^t \| < \epsilon \text{ STOP. Otherwise, go to 4.}
\]

4. \( y_j^t = -A_i (A_j x_j - a_j) + \frac{1}{n} \Lambda_i R^t, \quad \forall j = 1, \ldots, n; \quad u^{t+1} = u^t + \frac{1}{n} \Lambda_i R^t \)
If necessary update \( \Lambda_t \) to get \( \Lambda_{t+1} \) and return to 2.

An empirical update formula for the matrix of weights will be presented in Section 7. Note that for the convergence results every matrix in the sequence \( \Lambda_t \) must be positive definite (see [7]).

We now present our previous version of SALAMS for routing problem. Note that when SALAMS is applied to routing problem (13) - (14), a scaling factor \( \lambda_a \) is associated to each link \( a \) and a scaling factor \( \lambda_k \) is associated to each commodity \( k \), where these factors are the entries of the diagonal matrix \( \Lambda \). The problem is decomposed into \( n \) unidimensional subproblems and at least \( K \) shortest paths problems. The unidimensional subproblems are solved by combining bisection and Newton methods. Since the cost on each arc given by the derivative function \( \Phi_a \) is positive, we use Dijkstra’s algorithm to solve the shortest path problems. We obtain the following algorithm,

**SALAMS for routing problem**

1. Choose \( \epsilon > 0, u^1, \lambda_a^0 > 0, \lambda_k^0 > 0, \lambda_k^0 > 0, x_a^0 = 0, \delta_a^0 = 0 \) and \( n_a = 1 \).

2. \( \forall a = 1, \ldots, n \) compute
\[
x_a^t = \arg \min_{0 \leq x_a \leq C_a} \left\{ \Phi_a(x_a) - \lambda_a \left( u_a^t - \lambda_a^{-1} (x_a^{t-1} + \frac{\delta_a^t}{n_a}) \right) x_a + \frac{(\lambda_a)^2}{2} x_a^2 \right\}
\]

3. \( \forall k = 1, \ldots, K \) compute
\[
\alpha = \min_{1 \leq p \leq N_k} \sum_{a=1}^{n} \pi_{kp} \Phi_a(x_a^{t+1})
\]
Compute \( \beta \) the shortest paths between \( S_k \) and \( D_k \), when \( \Phi_a(x_a^t) \) is the cost associated to link \( a \).
If \( \beta < \alpha \) then \( N_k := N_k + 1 \). For \( p = 1, \ldots, N_k \) compute
\[
x_{kp}^t = \max \left\{ 0, \frac{1}{(\lambda_k^t)^2 + \sum_{a=1}^{n} (\lambda_a^t)^2 \pi_{kp}} \left( \lambda_k^t \lambda_k^{t-1} - \sum_{a=1}^{n} \lambda_a^t \lambda_a^{t-1} \pi_{kp} x_{kp}^{t-1} + (\lambda_k^t U_k^t - \sum_{a=1}^{n} \pi_{kp} \lambda_a^t u_a^t) \right) \right\}
\]
\[
\lambda_k^t \left( \frac{\lambda_k^{t-1} \sum_{p=1}^{N_k} x_{kp}^{t-1}}{N_k} - \sum_{a=1}^{n} \lambda_a^t \lambda_a^{t-1} \pi_{kp} \delta_a^t \right)
\]
If \( x_{kp}^{t+1} = 0 \) then \( N_k := N_k - 1 \) (Cancellation of paths carrying no flow)

4. \( \forall a = 1, \ldots, n \) compute \( \delta_a^t \) and \( \sigma_a^t \cdot \forall k = 1 \ldots, K \), compute \( \delta_k^t \) and \( \sigma_{kp}^t \).

IF \( \max \left( \max_{a} |\sigma_{a,kp}^t|, \max_{a} |\sigma_{a,kp}^t|, \max_{a} |\delta_a^t|, \max_{a} |\delta_a^t| \right) \) < \( \epsilon \) STOP. Otherwise go to 5.
Descent method

We next present the procedure allowing to improve the current solution given by SALAMS.

by generator vectors of obtained is not optimal we return to step current matrix constraint method. In fact method takes the current solution of SALAMS as its initial solution and improves it. When the solution change after a fixed number of the active links and paths. In this new version, when the number of paths for each commodity does not launch a descent method.

The descent method is launched from step 2. Choose the largest value \( \gamma^t \) such that \( \phi(\theta^t + \gamma^t \delta_t) - \phi(\theta^t) \leq \gamma^t m \nabla \phi(\theta^t) d_t \), where \( m \in (0, \frac{1}{\| \delta_t \|}) \)

3. We set \( \theta^{t+1} = \theta^t + \gamma_t d_t \). If \( \| \nabla \phi(\theta^{t+1}) \| \leq \epsilon' \) go to step 4, otherwise return to step 2.

New Version of SALAMS

We have observed that with the version above, the number of paths for each commodity stays often unchanged for a large number of iterations. The amount of flow forwarded on each path vary slowly and then the requirement constraints (??) is not met. We now propose a new version of SALAMS based on the knowledge of the active links and paths. In this new version, when the number of paths for each commodity does not change after a fixed number \( N \) of successive iterations, we stop the computation of the shortest paths and we launch a descent method.

The descent method is launched from step 3 of SALAMS when we observe that the number of paths for each commodity and the number of active link are fixed after \( N \) iterations. The descent method is based on the current matrix constraint \( A \) obtained by SALAMS. We first determine a matrix \( Z \) whose column are given by generator vectors of \( \text{Ker} A \) and we obtain the feasible direction \( \theta Z^T \), where \( \theta \) is obtained by Newton’s method. In fact \( \theta \) can be seen as a Newton iteration and \( \theta Z^T \) is a feasible iteration. We update the aggregate flow \( x_a \) on each link \( a \) and the individual flow \( x_{kp} \) by using the direction \( \theta Z^T \). Notice that this descent method takes the current solution of SALAMS as its initial solution and improves it. When the solution obtained is not optimal we return to step 2 of SALAMS.

We next present the procedure allowing to improve the current solution given by SALAMS.

Descent method

1. We set \( X^0 \) the current solution, choose \( \epsilon' > 0 \), \( \mu = 1 \) and \( \theta^0 = 0 \) a \( \sum_{k=1}^K (N_k - 1) \)-vector.

2. We set \( \theta^{t+1} = \theta^t + \gamma_t d_t \). If \( \| \nabla \phi(\theta^{t+1}) \| \leq \epsilon' \) go to step 4, otherwise return to step 2.
4. Compute \( X^{t+1} = X^t + \mu (\theta^{t+1} Z^T) \).

5. If \( x^{t+1}_a < C_a \) for all links \( a \) carrying a flow, go to step 6. Otherwise set \( \mu := \frac{\mu}{2} \) and return to step 4.

6. If \( \max \delta_j^t < \epsilon \) and \( \max \delta_k^t < \epsilon \), STOP. Otherwise go to step 2 of SALAMS.

Notice that \( \phi(\theta) = \Phi (X + \mu \theta Z^T) \) (delay function), where \( X \) is the current solution given by SALAMS.

\[ d_t = -\nabla \phi(\theta^t) [\nabla^2 \phi(\theta^t)]^{-1} \] and the columns of the matrix \( Z \) are given by the generating vectors of \( \text{Ker} \mathcal{A} \).

4. Numerical Experiments

We present here the preliminary results performed on the medium-sized networks presented on Table ??.

<table>
<thead>
<tr>
<th>Networks</th>
<th># of nodes</th>
<th># of links</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net1</td>
<td>19</td>
<td>68</td>
</tr>
<tr>
<td>Net2</td>
<td>21</td>
<td>68</td>
</tr>
<tr>
<td>Net3</td>
<td>60</td>
<td>280</td>
</tr>
</tbody>
</table>

Table 1: Test Problems

Notice that the initial values of the scaling factors are given by the following empirical formula,

\[ \lambda_a^0 = 0.5 \frac{\min C_a}{R} \quad \text{and} \quad \lambda_k^0 = \lambda_a^0 \quad \text{where} \quad R = \sum_{k=1}^{K} r_k. \]

We also use the following formulas to update the scaling factor

\[ \lambda_a^{t+1} = \begin{cases} 
0.9 \lambda_a^t & \text{if} \quad 0 \leq t \leq 100 \quad \text{and} \quad t \equiv 0 \mod 10 \\
\lambda_a^t & \text{otherwise}
\end{cases} \]

We now compare both, the early and new versions of SALAMS where \( \epsilon = 10^{-3} \) in both version. After \( N = 10 \) successive iterations, when the number of paths generated does not change for each commodity, we launch the descent method presented above.

Table ?? illustrates the results obtained. We observe that the new version of SALAMS is more efficient than the first one. Indeed, we remark that the CPU time is decreased from three times to twelve times when the new version is used. Notice the result obtained by the previous version with Net3 when \( K = 37 \) is not presented because the solution was not obtained after 5000 iterations. The last column in Table ?? entitled "Paths reduction" shows that when the new version is applied, the total number of paths used by all communications is decreased significantly. This remark is important from both computational and telecommunication points of view. Indeed in computational point of view, we have less variables to handle and this decreases the number of calculations performed. On the other hand, it is really important in telecommunication networks to use as few paths as possible. This allows a best resource management and helps to avoid congestion.
<table>
<thead>
<tr>
<th>K</th>
<th>Previous version</th>
<th>New version</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$K$</td>
<td># of Iter.</td>
</tr>
<tr>
<td>Net1</td>
<td>7</td>
<td>308</td>
</tr>
<tr>
<td>Net1</td>
<td>11</td>
<td>261</td>
</tr>
<tr>
<td>Net1</td>
<td>17</td>
<td>243</td>
</tr>
<tr>
<td>Net2</td>
<td>10</td>
<td>368</td>
</tr>
<tr>
<td>Net2</td>
<td>15</td>
<td>384</td>
</tr>
<tr>
<td>Net2</td>
<td>21</td>
<td>439</td>
</tr>
<tr>
<td>Net2</td>
<td>25</td>
<td>437</td>
</tr>
<tr>
<td>Net2</td>
<td>37</td>
<td>—</td>
</tr>
<tr>
<td>Net3</td>
<td>10</td>
<td>451</td>
</tr>
<tr>
<td>Net3</td>
<td>15</td>
<td>430</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the two versions of SALAMS

5. Conclusion

In this paper, we have presented a new version of SALAMS for routing problem in telecommunication networks. This new one is based on the combination of the classical SALAMS (see [2]) and the Newton descent method. The preliminary results performed on medium-sized networks showed that our new version improved the convergence of this method in term of CPU time, number of iterations and total number of paths used.

In the further works, we would like to analyze the behavior of this new version when it’s applied on large-scale networks. We also would like to compare our new approach to others methods such as proximal decomposition and flow deviation methods.

References


