MASTER'S THESIS

# Interuniversity Master in Statistics and Operations Research UPC-UB

**Title: Video-On-Demand Optimization Using an Interior-point Algorithm**

**Author:** Luis Felipe Urquiza Aguiar

**Advisor:** Jordi Castro

**Department:** Statistics and Operations Research.

**University:** Universitat Politècnica de Catalunya (UPC)

**Academic year**: 2017/2018



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Master Thesis

# Video-On-Demand Optimization Using an Interior-point Algorithm

Luis Felipe Urquiza Aguiar

Advisor: Jordi Castro

Department of Statistics and Operations Research, Research Group
GNOM (Group on Numerical Optimization and Modeling)

To my daughter Nuria and my mom Yolanda.
They are two angels in my life equally beautiful.

# Abstract

**Keywords:** CDN, VoD, optimal placement, interior point methods, BlockIP

**MSC2000:** 90-02, 90B15, 90B18

A Content Delivery Network (CDN) aims to provide efficient movement of massive digital content (multimedia and files) across the Internet. This is achieved by putting the content in servers closer to the costumer. Video-On-Demand service is an application of CDN where videos have to be located strategically to avoid network congestion and servers' saturation. Therefore, the problem of optimal placement of videos arises. This problem has a block diagonal structure with linking constraints on links' and servers' capacities. In this project, we solve huge instances of a video placement problem over three real network topologies with a specialized interior point solver named BlockIP. The evaluated instances range from 7 to 300 millions of variables and the difficulty of the instances depends on the size of servers, links' bandwidth and network topology. Our results: 1) verified characteristics of BlockIP like regularization and the intensive computation in the last iterations and 2) showed that BlockIP found optimal solution in all the evaluated instances with a good optimality gap. On the contrary, state-of-art CPLEX cannot reach an optimal, feasible solution in some difficult instances and needs almost twice the memory of BlockIP. However, CPLEX solved most of feasible instances at least twice faster than BlockIP

# Contents


Chapter 1. Introduction 1
1. Interior Point Methods 2
1.1. Primal-dual path-following methods 2
2. BlockIP 5
2.1. Solving the normal equations by PCG 7
2.2. Improving the spectral radius 8
2.3. Estimating the spectral radius 9
2.4. Implementation details 9
3. Optimization for Content Delivery Networks 10
3.1. Content Delivery Networks Overview 10
3.2. Optimization models for Content Delivery Networks 12

Chapter 2. Optimal Placement for Video On Demand Systems 13
1. The problem formulation 13
1.1. Parameters and variables 13
1.2. The model 14
2. Input Instance 18

Chapter 3. Implementation and Results 27
1. Implementation Details 27
2. Results for small instances 28
2.1. Ebone 28
2.2. Sprintlink 31
2.3. Tiscali 34
3. Results for big instances 38
3.1. Ebone 38
3.2. Sprintlink 40
3.3. Tiscali 42

Chapter 4. Conclusions 45

References 47

# Chapter 1
# Introduction

Linear programing [**21**] is a well-established field of operational research where optimization problems are modeled by a linear cost function and linear constraints. Linear programing allows researchers and practitioners of many different areas of sciences and engineering to get an optimal solution for a variety of problems. In particular, Telematics engineering is a increasing consuming discipline of linear programing tools because many design issues can be expressed as linear optimization problems.

In this master project, we deal with the optimal placement of videos in a content distribution network. The problem at hand is a current research topic in Telematics engineering because videos has to be stored in different locations and satisfy every client request. A bad videos' distribution could saturate the network or make impossible to serve clients' demands. Traditionally, linear optimization problems are solved by using the Simplex method. This method iterates on the vertices (extreme points) of the feasible region until reach the one that minimizes the objective function. Although Simplex Method is non-polynomial time, in practice it is efficient, visiting only a small fraction of the total number of vertices [**36**]. Despite the good performance of simplex method, other methods called Interior Point Methods (IPM) seems to be more suitable for large-scale problems. These methods reach the optimal solution (i.e., an extreme point) starting from a point inside the feasible region. Moreover, IPMs take advantage of any block matrix structure in the linear algebra operations [**28**]. The video placement problem to be solved in this thesis is precisely of this kind, i.e., large-scale, block matrix structure,because variables and constraints are associate with each video that has to be stored.

In this chapter, we summarize the most widely used IPM called primal-dual, path-following (PD-PF). After that, we concentrate on describing an specialized IPM solver for large-scale problems with a block matrix structure named `BlockIP`. This chapter finalizes with an overview of content delivery networks, for which video distribution network is only an special case.

# 1. Interior Point Methods

Interior point methods are a family of non-simplex method for Linear Programing, which appeared in 1984 with Karmarkar algorithm. These IPM algorithms iterate through the interior of the feasible region.

Broadly, they can be classified in four categories [**17**]:

**Affine-scaling methods:** It is the simplest IPM. It computes a movement direction in a scaled feasible region. The idea is to center the search of a improvement direction to avoid borders. In these methods each scaled point $\hat{x_i}$ is computed as:

$$\hat{x_i} = \beta_i x_i.$$

**Methods based on projective transformations:** A representative method of this kind is the Karmarkar algorithm. In general, scaled point in these methods are obtained as:

$$\hat{x_i} = \frac{\beta_i x_i}{\sum_{j=1}^{n} \beta_j x_j}.$$

**Path-following methods:** currently the most used methods. They do no scale the variables. Instead, they follow the central path to reach the optimum. The specialized solver employed in this project employs this kind of method. The algorithm used by stat-of-art CPLEX solver also belongs to this family

**Potential-reduction methods:** The computed directions are decided by measuring their quality through the reduction of a potential function.

**1.1. Primal-dual path-following methods.** The primal and dual formulations for a linear problem (LP) are:

$$(1)\qquad \begin{array}{lll} \min & c^T x & \\ \text{s.t.} & Ax = b & \\ & u \geq x \geq 0, & \end{array} \qquad\qquad \begin{array}{lll} \max & b^T\lambda - u^T w & \\ \text{s.t.} & A^T\lambda - w + z = c & \\ & z, w \geq 0. & \end{array}$$

The Karush-Kuhn-Tucker (KKT) optimality conditions are the following:

$$(2)\qquad \begin{array}{lcccl} r_c & \equiv & A^\top\lambda + z - w + c & = & 0, \quad \text{[dual feasibility]} \\ r_b & \equiv & Ax - b & = & 0, \quad \text{[primal feasibility]} \\ r_{xz} & \equiv & XZe & = & 0, \quad \text{[complementarity]} \\ r_{sw} & \equiv & SWe & = & 0, \quad \text{[complementarity]} \\ & & (x, s, z, w) & \geq & 0, \end{array}$$

where $x, u \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $e \in \mathbb{R}^n$ is a vector of 1's; $\lambda \in \mathbb{R}^m$, $z \in \mathbb{R}^n$ and $w \in \mathbb{R}^n$ are, respectively, the vectors of Lagrange multipliers of the equality constraints, lower and upper bounds; $s = u - x$; and matrices $X, Z, S, W \in \mathbb{R}^{n\times n}$ are diagonal matrices made up of vectors $x, z, s, w$.

We define function $F(x, \lambda, z, w) : \mathbb{R}^{3n+m} \longrightarrow \mathbb{R}^{3n+m}$ as:

$$F(x^k, \lambda^k, s^k) = \begin{bmatrix} r_c \\ r_b \\ r_{xz} \\ r_{sw} \end{bmatrix}. \tag{3}$$

By using the Newton's method a feasible direction $\Delta = (\Delta_x, \Delta_\lambda, \Delta_z, \Delta_w)$ can be obtained as:

$$\nabla_{(x,\lambda,z,w)} F(x^k, \lambda^k, z^k, w^k)\Delta = -F(x^k, \lambda^k, z^k, w^k), \tag{4}$$

where $(x^k, \lambda^k, z^k, w^k)$ is the current point and if it is feasible then the two first vector of the right hand side (i.e., primal and dual feasibility) are zero. The next point $(x^{k+1}, \lambda^{k+1}, z^{k+1}, w^{k+1}) = (x^k, \lambda^k, z^k, w^k) + \alpha(\Delta_x, \Delta_\lambda, \Delta_z, \Delta_w)$, where $\alpha$ is the step length and it is used to guarantee $(x, z, w, s) > 0$. In practice, $\alpha \ll 1$ because points are very close to the boundary $(x, z, w) = 0$ and the progress to reach the optimal solution is slow [**18**]. To get a more "interior or central" direction, a perturbed $F_\tau(x, \lambda, z, w)$ is used instead of $F(x, \lambda, z, w)$ in (4), which leads to:

$$\nabla_{(x,\lambda,z,w)} F_\tau(x^k, \lambda^k, z^k, w^k)\Delta = -F_\tau(x^k, \lambda^k, z^k, w^k), \tag{5}$$

where:

$$F_\tau(x, \lambda, z, w) = \begin{bmatrix} r_c \\ r_b \\ r_{xz} - \tau e \\ r_{sw} - \tau e \end{bmatrix}, \qquad \tau \in \mathbb{R}^+. \tag{6}$$

In addition, $(x, z, w) \geq 0$ changes to $(x, z, w) > 0$. The set of solutions for (5) with different values of $\tau$ is known as the primal-dual central path. Notice that, when a point is feasible the first two vectors in 6 (i.e., $r_c$ and $r_b$) are zero. In addition, Fiacco et. al. [**23**] show that when $\tau \to 0$ the solution of (5) tends to a primal-dual solution to the LP KKT optimality conditions.

Primal-dual path-following (PD-PF) IPMs approximately follow the central path to the optimum. This is done by computing only one damped Newton direction for a value of $\tau$, then $\tau$ is reduced and a new damped Newton direction for the new $\tau$ is computed, and so on. In this way, the computed points are not in the central path, but they follow it. $\tau$ is reduced until 0, as follows:

$$\tau = \sigma\mu, \qquad \sigma \in (0, 1),\ \mu = \frac{x^T z + s^T w}{2n}. \tag{7}$$

Notice that the perturbed complementary $\mu$ is obtained as the average of $x_i z_i$ and $s_i w_i$ because in general $x_i z_i,\ s_i w_i \neq \mu, \forall i$. Moreover, if $(x, \lambda, z, w)$ are primal and dual feasible then $2n\mu$ is the duality gap.

$$\begin{aligned} 2n\mu &= x^T(c - A^T\lambda + w) + s^T w u &= c^T x - (Ax)^T\lambda + (u - s)^T w + s^T w \\ 2n\mu &= c^T x - (b^T\lambda - u^T w). \end{aligned}$$

Regarding the reduction factor $\sigma$, on one hand if it reduces quickly then it forces an aggressive direction, as in Newton's method, and the algorithm will progress slowly. On the other hand, if $\sigma \approx 1$ then it forces a direction that leads to the central path allowing significant progress in next iteration, however, the algorithm

does not move to optimality. Thus, $\sigma$ is updated according to the particular IPM used.

So, at each iteration the PD-PF the next system of equations has to be solved:

$$\begin{bmatrix} 0 & A^T & I & -I \\ A & 0 & 0 & 0 \\ Z^k & 0 & X^k & 0 \\ -W^k & 0 & 0 & S^k \end{bmatrix} \begin{bmatrix} \Delta_x^k \\ \Delta_\lambda^k \\ \Delta_z^k \\ \Delta_w^k \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ r_{\hat{xz}} \\ r_{\hat{sw}} \end{bmatrix}, \tag{8}$$

where $r_{\hat{xz}} = -X^k Z^k e + \sigma_k \mu_k e$ and $r_{\hat{sw}} = -S^k W^k e + \sigma_k \mu_k e$. Depending on the movement length from $(x^k, \lambda^k, z^k, w^k)$ to $(x^{k+1}, \lambda^{k+1}, z^{k+1}, w^{k+1})$ PD-PF algorithm has two variants: 1) Short-step method which has the best theoretical complexity, but the worse in practice. 2) It is the most efficient and used in practice. In addition, both variants are polynomial time algorithm for LP [**18**].

We concentrate on PD-PF long step method, which iterates in a wide neighborhood of the central path. Some features of a PD-PF long step method are [**18**]:

- In practice an infeasible (i.e., $r_c \neq 0, r_b \neq 0$), interior (i.e., $(x^0, z^0, w^0) > 0$) is considered to initialize the algorithm.
- $\sigma_k \in [\sigma_{min} \sigma_{max}]$ to avoid large or small reduction of $\mu$.
- $\alpha_k$ is in practice the maximum step length such that $(x, z, w) \geq 0$, reduced by parameter $\rho \geq 0.95$.
- It has a polynomial complexity of $O(n^2 log\frac{1}{\epsilon})$, where $n$ is the number of variables and $\epsilon$ is and optimality precision.
- It is the most used approach by most solvers: CPLEX, XPRESS, LIPSOL, etc.

An excellent discussion about the theoretical properties of this and other interior-point algorithms can be found in [**41, 46, 50**]. As we said, at each iteration the system of equation (8) has to be solved. For this purpose, two techniques are commonly used [**18**]: 1) augmented system, which is preferred for quadratic programming and general nonlinear programing; and 2) Normal equations that is more effective for most LP problems. Regarding the latter method, after eliminating $\Delta w$ and $\Delta z$, as follows:

$$\Delta z = X^{-1} r_{\hat{xz}} - X^{-1} Z \Delta x \tag{9}$$

$$\Delta w = S^{-1} r_{\hat{sw}} + S^{-1} W \Delta x, \tag{10}$$

we obtain the augmented system form

$$\begin{bmatrix} -\Theta^{-1} & A^\top \\ A & \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} r \\ -r_b \end{bmatrix}, \tag{11}$$

where $\Theta$ and $r$ are defined as

$$\Theta = (ZX^{-1} + WS^{-1} + \nabla^2 f(x))^{-1} \qquad r = -r_c + S^{-1} r_{\hat{sw}} - X^{-1} r_{\hat{xz}}. \tag{12}$$

Note that, if the objective function is separable, $\Theta$ is an easily computable diagonal matrix. Additionally, eliminating $\Delta x$ from the first group of equations of (11), the normal equations are obtained:

$$(A\Theta A^\top)\Delta\lambda = -r_b + A\Theta r \tag{13}$$

$$\Delta x = \Theta(A^\top \Delta\lambda - r). \tag{14}$$

In methods based on normal equations, the Newton direction is obtained by solving (13), (14), (9) and (10).

The most expensive step of the normal equations is $A\Theta A^T$. Nonetheless, if $A$ is full row rank, it can be solved very efficiently with sparse Cholesky factorization. The Cholesky factorization has to deal with three well-known difficulties:

- **fill-in.** Factorization can degrade sparsity by increasing the number of non-zeros positions considerably. The solution is to find a permutation matrix $P$ that minimize the fill-in.
- **0 pivots.** They can appear because $A$ is not full-row rank or the solution is degenerate. Some Cholesky factorization packages like cholmod[1] tackle this issue by replacing the zero pivot with a large value.
- **Dense columns** If A has at least one dense column then $A\Theta A^T$ is a dense matrix. This makes it computationally very expensive. One way to deal with dense columns is to factorize $A\Theta A^T$ as the sum of sparse matrix plus low rank additional matrix. Other strategy for this issue is to use augmented system, which is insensible to dense column. A third option, that will be employed in this project, is to reformulate the problem at hand in such way that avoid the presence of dense columns. This last approach might come at the price of a higher number of variables than the initial problem.

# 2. BlockIP

Block-angular structures are used as a modeling tool in many situations, such as multiperiod or multicommodity problems, two-stage stochastic problems, and, in general, models involving linking variables or linking constraints. The resulting optimization problems have in common a huge number of variables, and a large number of linear constraints. Interior-point methods (IPMs) are in general competitive against other techniques in those cases.

This section summarizes an efficient interior-point solver for block-angular convex optimization problems named `BlockIP` proposed in [**19**], which relies on a combination of Cholesky factorizations and preconditioned conjugate gradient (PCG) for the solution of the normal equations at each interior-point iterations. `BlockIP` includes three most relevant additional features from its Matlab predecessor in [**14**]: (i) it may solve convex nonlinear separable optimization problems; (ii) the use of quadratic regularization, which may significantly improve the quality of the preconditioner as shown in [**16**]; (iii) an estimation of the spectral radius of a certain matrix which intervenes in the preconditioner, following [**11**], which can be used as a measure of the quality of preconditioner.

The standard form of the linearly constrained convex block-angular problems solved by `BlockIP` is

[1]Created by Tim Davis and used in Matlab

$$
\begin{aligned}
&\min \quad \sum_{i=0}^{k} f_i(x^i) \\
&\text{s. to} \quad \begin{bmatrix} A_1 & & & \\ & \ddots & & \\ & & A_k & \\ L_1 & \dots & L_k & I \end{bmatrix} \begin{bmatrix} x^1 \\ \vdots \\ x^k \\ x^0 \end{bmatrix} = \begin{bmatrix} b^1 \\ \vdots \\ b^k \\ b^0 \end{bmatrix} \\
&\qquad\qquad 0 \le x^i \le u^i \quad i = 0, \dots, k.
\end{aligned} \tag{15}
$$

Matrices $A_i \in \mathbb{R}^{m_i \times n_i}$ and $L_i \in \mathbb{R}^{l \times n_i}$, $i = 1, \dots, k$ define the block and linking constraints, respectively, $k$ being the number of blocks. Vectors $x^i \in \mathbb{R}^{n_i}, i = 1, \dots, k$, are the variables for each block. $x^0 \in \mathbb{R}^l$ are the slacks of the linking constraints. $b^i \in \mathbb{R}^{m_i}, i = 1, \dots, k$ is the right-hand-side vector for each block of constraints, whereas $b^0 \in \mathbb{R}^l$ is for the linking constraints. The upper bounds for each group of variables are defined by $u^i, i = 0, \dots, k$. Note that with this standard formulation linking constraints are of the form $b^0 - u^0 \le \sum_{i=1}^{k} L_i x^i \le b^0$. Slacks of linking constraints play a significant role in the quality of the preconditioner, and they should not be removed. Equality linking constraints can be formulated by setting $u^0 \approx 0$. Functions $f_i : \mathbb{R}^{n_i} \to \mathbb{R}$, $i = 0, \dots, k$, are assumed to be convex. Although the specialized IPM to be described is valid for any $f_i$, for the sake of efficiency we will restrict to separable functions, i.e., $\nabla^2 f_i(x^i)$ are (positive semidefinite) diagonal matrices. Note that any convex quadratic problem can be transformed into a separable equivalent one by the addition of extra variables and constraints

Exploiting the structure of $A$, and appropriately partitioning $\Theta$ of (12), as follows

$$
A = \begin{bmatrix} A_1 & & & \\ & \ddots & & \\ & & A_k & \\ L_1 & \dots & L_k & I \end{bmatrix} \qquad \Theta = \begin{bmatrix} \Theta_1 & & & \\ & \ddots & & \\ & & \Theta_k & \\ & & & \Theta_0 \end{bmatrix},
$$

the matrix of system (13) can be recast as

$$
A\Theta A^\top = \left[\begin{array}{ccc|c} A_1\Theta_1 A_1^\top & & & A_1\Theta_1 L_1^\top \\ & \ddots & & \vdots \\ & & A_k\Theta_k A_k^\top & A_k\Theta_k L_k^\top \\ \hline L_1\Theta_1 A_1^\top & \dots & L_k\Theta_k A_k^\top & \Theta_0 + \sum_{i=1}^{k} L_i\Theta_i L_i^\top \end{array}\right] = \begin{bmatrix} B & C \\ C^\top & D \end{bmatrix}, \tag{16}
$$

$B \in \mathbb{R}^{\tilde{n} \times \tilde{n}}$ ($\tilde{n} = \sum_{i=1}^{k} n_i$), $C \in \mathbb{R}^{\tilde{n} \times l}$ and $D \in \mathbb{R}^{l \times l}$ being the blocks of $A\Theta A^\top$, and $\Theta_i$, $i = 0, \dots, k$, the submatrices of $\Theta$ associated with the $k+1$ groups of variables in (15), i.e., $\Theta_i = (Z_i X_i^{-1} + W_i S_i^{-1} + \nabla^2 f_i(x^i))^{-1}$. Denoting by $g$ the right-hand-side of (13), and appropriately partitioning $g$ and $\Delta\lambda$, the normal equations can be

1. **Algorithm** *PCG(*$S, M, \bar{g}, \Delta\lambda_{2_0}, \epsilon, i_{\max}$*)*
2. // Solve $S\Delta\lambda_2 = \bar{g}$ by PCG with preconditioner $M$
3. **Initializations:** $i := 0$**;** $r_0 := \bar{g} - S\Delta\lambda_{2_0}$**;**
4. Solve $Mz_0 = r_0$; $p_0 := z_0$;
5. **while** $||r_k|| > \epsilon$ and $i < i_{\max}$ **do**
6. $q_i := Sp_i$;
7. $\alpha_i := (z_i^\top r_i)/(p_i^\top q_i)$;
8. $\Delta\lambda_{2_{i+1}} := \Delta\lambda_{2_i} + \alpha_i p_i$;
9. $r_{i+1} := r_i - \alpha_i q_i$;
10. Solve $Mz_{i+1} = r_{i+1}$;
11. $\beta_i := (z_{i+1}^\top r_{i+1})/(z_i^\top r_i)$;
12. $p_{i+1} := z_{i+1} + \beta_i p_i$;
13. $i := i + 1$;
14. **end while**
15. Return $\Delta\lambda_2 := \Delta\lambda_{2_i}$;
16. **End_algorithm**

FIG. 1. The PCG algorithm for the solution of $S\Delta\lambda_2 = \bar{g} \equiv g_2 - C^\top B^{-1} g_1$ with preconditioner $M$

written as

$$\begin{bmatrix} B & C \\ C^T & D \end{bmatrix} \begin{bmatrix} \Delta\lambda_1 \\ \Delta\lambda_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}. \tag{17}$$

**2.1. Solving the normal equations by PCG.** Eliminating $\Delta\lambda_1$ from the first group of equations of (17), we obtain

$$(D - C^\top B^{-1} C)\Delta\lambda_2 = (g_2 - C^\top B^{-1} g_1) \tag{18}$$

$$B\Delta\lambda_1 = (g_1 - C\Delta y_2). \tag{19}$$

System (19) is solved by performing $k$ Cholesky factorizations, one for each diagonal block $A_i \Theta_i A_i^\top, i = 1 \ldots k$, of $B$. System (18) with the Schur complement

$$S = D - C^\top B^{-1} C, \tag{20}$$

of dimension $l$—the number of linking constraints—, may exhibit a large fill-in, and it is prohibitive if computed by Cholesky factorization. It will be solved by a PCG, which is outlined in Figure 1. The preconditioner used in this work is from [**12**],[**12**]. It relies on the fact that (20) is a *P-regular splitting*, i.e., $S$ is symmetric and positive definite, $D$ is nonsingular and $D + C^\top B^{-1} C$ is positive definite. Therefore, the spectral radios $\rho(D^{-1}(C^\top B^{-1} C)) < 1$ is guaranteed by [**6**],[**44**, pp. 254–255] This allows us to compute the inverse of $S$ [**12**, Prop. 4], as

$$(D - C^\top B^{-1} C)^{-1} = \left( \sum_{i=0}^{\infty} (D^{-1}(C^\top B^{-1} C))^i \right) D^{-1}. \tag{21}$$

The preconditioner $M^{-1}$ is an approximation of $S^{-1}$ obtained by truncating the infinite power series (21) at some term $h$. For instance, for $h = 0$ and $h = 1$ we have

$$\begin{aligned} M^{-1} &= D^{-1} && \text{if } h = 0, \\ M^{-1} &= (I + D^{-1}(C^\top B^{-1} C))D^{-1} && \text{if } h = 1. \end{aligned}$$

1. **Algorithm** $Mz = r(D, C, B, r, h)$
2. $v := D^{-1}r$;
3. $z_0 := v$;
4. **for** $j := 1$ to $h$ **do**
5. $\quad z_j := D^{-1}(C^\top(B^{-1}(Cz_{j-1}))) + v$;
6. **end for**
7. Return $z := z_h$

FIG. 2. Algorithm for computing $z = M^{-1}r$

The larger $h$, the better the approximation of the inverse. On the other hand, systems $Mz = r$ (for some vectors $z$ and $r$) have to be solved at each PCG iteration (step 10 of PCG algorithm of Figure 1), and any extra term in the series means an additional linear system solution with matrix $B$. This is clearly seen in the algorithm of Figure 2, which shows how $Mz = r$ is iteratively computed in the specialized interior-point solver. Note that matrix $C^\top B^{-1}C$ does not need to be built, and aside from the solution of systems with $B$ and $D$, only matrix-vector products with $C$ and $C^\top$ (i.e., with $L_i$, $L_i^\top$, $A_i$ and $A_i^\top$, $i = 1, \ldots, k$) are required. This also applies to the PCG algorithm of Figure 1. It is thus possible to partially apply the matrix-free paradigm [**29**]. Efficient implementations of this matrix-vector products for particular $A_i$ and $L_i$, $i = 1, \ldots, k$, matrices can significantly speed the computational efficiency. It is worth noting that, although (21) is valid for any primal block-angular problem, the preconditioner is only useful in practice for separable problems; otherwise, nondiagonal $\Theta_i$ matrices make systems with $B$ prohibitive.

**2.2. Improving the spectral radius.** The quality of the preconditioner depends on $\rho$, which is always in $[0, 1)$: the farther from 1, the closer $M^{-1}$ is to $S^{-1}$. In practice it has been observed that $\rho$ comes closer to 1 as we approach the optimal solution for most instances. However, as shown in [**16**], non-zero Hessians reduce $\rho$, and this opens the possibility for improving the preconditioner by the addition of a quadratic regularization term

In practice, if the problem is linear or the Hessian is close to 0, a non-zero Hessian can be added by a quadratic regularization. The interior-point solver `BlockIP` implements two types of regularization: a proximal point and a quadratic regularization. The later is based on the addition of a quadratic term to the standard logarithmic barrier function $B(x, \mu)$ of (1)

$$B(x, \mu) \triangleq f(x) + \mu \left( -\sum_{i=1}^{n} \ln x_i - \sum_{i=1}^{n} \ln(u_i - x_i) \right), \tag{22}$$

$\mu \in \mathbb{R}^+$ being the barrier parameter. In the quadratic regularization introduced in [**16**] $B(x, \mu)$ is replaced by

$$B_Q(x, \mu) \triangleq f(x) + \mu \left( \frac{1}{2} x^\top Q_R x - \sum_{i=1}^{n} \ln x_i - \sum_{i=1}^{n} \ln(u_i - x_i) \right), \tag{23}$$

$Q_R$ being a diagonal positive semidefinite matrix. The reduction of $B_Q$ to 0 is controlled by $\mu$, the standard barrier parameter.

Using either $B$ or $B_Q$ only changes the dual feasibility of KKT conditions and matrix $\Theta$, defined in (2) and (12), respectively. Dual feasibility becomes

$$A^\top \lambda + z - w = \nabla f(x) \qquad \text{for } B, \text{and} \tag{24}$$

$$A^\top \lambda + z - w = \nabla f(x) + \mu Q_R x \qquad \text{for } B_Q. \tag{25}$$

(25) is equivalent to (24) when $\mu$ tends to zero (i.e., when we approach the optimal solution).The $\Theta$ matrices are

$$\Theta = (ZX^{-1} + WS^{-1} + \nabla^2 f(x))^{-1} \qquad \text{for } B, \text{ and} \tag{26}$$

$$\Theta = (\mu Q_R + ZX^{-1} + WS^{-1} + \nabla^2 f(x))^{-1} \qquad \text{for } B_Q. \tag{27}$$

$\mu Q_R$ tends to zero with $\mu$ and therefore (27) approximates (26) when they are close to the optimal solution.

**2.3. Estimating the spectral radius.** Although knowing the spectral radius $\rho$ of $D^{-1}(C^\top B^{-1}C)$ would be instrumental to forecast the efficiency of the preconditioner, its computation is impractical. However, a procedure to estimate $\rho$ was recently introduced in [**11**] for $h = 0$, i.e., $M^{-1} = D^{-1}$. By linear algebra, if $\sigma_{\min}$ is the minimum eigenvalue of $I - D^{-1}(C^\top B^{-1}C)$ then $1 - \sigma_{\min}$ is the spectral radius of $D^{-1}(C^\top B^{-1}C)$.

The minimum eigenvalue $\sigma_{\min}$ of $I - D^{-1}(C^\top B^{-1}C)$ can be estimated from the solution of (13) by PCG, using the Ritz values from the relation between PCG and Lanczos method (see [**35**], [**33**, Chapter 9],[**38**] for details). In general, the extreme eigenvalues of the preconditioned matrix (the ones we are interested in, for the estimation of the spectral radius) are well approximated already during early PCG iterations [**38**]. The estimation of $\sigma_{\min}$ have been extended in [**19**] to consider any number $h \geq 0$ of terms in the power series preconditioner. For these cases $\rho$ can be easily estimated as $\sqrt[h+1]{1 - \tilde{\sigma}_{\min}}$, where $\tilde{\sigma}_{\min}$ is the smallest Ritz value.

**2.4. Implementation details.** `BlockIP` is written in C++, using the object oriented paradigm. It is roughly about 14000 lines of source code, aside from the external package for Cholesky factorization. Actually, `BlockIP` is only linked to the Ng-Peyton block sparse Cholesky package [**42**], which only implements an approximate minimum degree ordering; other more recent Cholesky packages can be added in the future, which may likely improve the performance of the solver. The package can be obtained for research purposes from `http://www-eio.upc.edu/~jcastro/BlockIP.html`. a reference manual and an example illustrating the use of the package.

The main features that `BlockIP` includes are:

(1) `BlockIP` may handle linear, quadratic and convex linearly constrained block-angular optimization problems.
(2) `BlockIP` stops at the feasible point of iteration $j$ when the relative gap between the primal and dual objectives, denoted by $p^j$ and $d^j$, is below some optimality gap. The relative gap is computed as $(p^j - d^j)/(1 + p^j)$.
(3) `BlockIP` implements two types of directions: the standard Newton direction and the second order heuristic direction of [**37**], which requires the solution of two systems (13) with different right-hand-sides. Both directions can be

computed by solving the normal equations by either a Cholesky factorization or by the PCG-based approach of Subsection 2.1. In general, the reduction of iterations caused by the second order direction is not worthwhile when using PCG, since, as far as we know, the solution of the first system can not be efficiently used as a warm-start for the second one. Indeed, from the results of [**12**] this strategy was not successful for multicommodity flows problems.

(4) The solver implements quadratic regularizations presented in Subsection 2.2.Following [**16**], the quadratic regularization matrix $Q_R$ of (23) is heuristically updated at each interior-point iteration $i$ as

$$Q_R := \delta \cdot i \cdot \mu_i / \mu_0 I,$$

where $\mu_0$ and $\mu_i$ are the barrier parameter at the starting and current point, and $\delta$ is a (usually small) initial regularization value. The product by $i$, the iteration counter, is an attempt to compensate for the quick reduction of $\mu_i$ when the optimal solution is approached.

(5) `BlockIP` may compute the Ritz values, and thus the spectral radius $\rho$, for any number $h$ of terms in the preconditioner.

(6) Problems can be provided in four different formats.
   (a) The most efficient way is using the `BlockIP` callable library, which provides routines to create problems from matrices and vectors. Particular matrix formats (network—oriented and nonoriented—, general rowwise or columnwise, diagonal, identity, etc.) can be exploited through the callable library. Nonlinear objective functions—including gradient and Hessian evaluations—have to be provided as C++ routines.
   (b) Problems can also be efficiently provided by an input file using a specific format for `BlockIP`. This format consists on a set of scalars, vectors and sparse matrices defining the problem parameters.
   (c) Input files can also be in *structured MPS format*, an extension of the well-known MPS format created for `BlockIP`. Standard packages can read structured MPS files without modification.
   (d) The last format is based on SML [**22**], a structure-conveying modeling language based on the popular AMPL [**26**] modeling language. In addition to hooking `BlockIP` to it, SML was extended to deal with nonlinear separable problems (see [**34**] for details).

# 3. Optimization for Content Delivery Networks

This section introduces the main ideas about Content Delivery Networks (CDN) and how operational research has been used in this context.

**3.1. Content Delivery Networks Overview.** A CDN is a collection of network elements spanning the Internet, where content is replicated over mirrored servers (i.e., point of presence (PoP), edge or replica servers), located at the border of the Internet service providers' (ISP') networks to which end-users are connected [**40**]. A high level architecture of a CDN is depicted in Fig. 3. The idea is to put the most likely requested content from users in the PoPs closer to customers. In short,

a CDN mitigates that every request from users has to be served by the (usually distant) origin server.

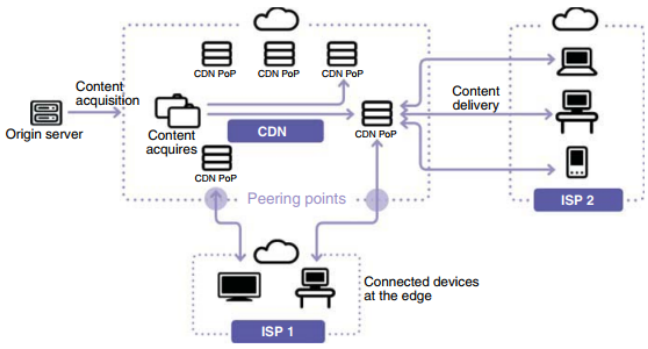



FIG. 3. High-level architecture of a CDN. [**40**]

A CDN can be very useful to provide high quality, live streaming coverage of major events, distributing user generated content (e.g., YouTube), providing fast and efficient software downloads to customers, or enhancing the performance of an e-commerce website. A classical application, dealt in this thesis, is the Video on Demand distribution (VoD). VoD service can be seen as a virtual video rental store in which a user can choose and watch any program on request, at the convenience of their time. Video is a very sensitive application from packet losses, end-to-end delay or delay variation (jitter). Without a CDN, bandwidth is overused as each request for the same content is retransmitted over and over from origin server and video performance degrades (i.e., higher packet losses, delay and jitter), especially when the number of users increases. On the other hand, With a CDN, bandwidth usage of the content provider is optimized, as if a single end-user has requested unique content only once. No additional investment is required by the content provider to increase bandwidth capacity, as media content is packaged for delivery by the CDN infrastructure, once it has been placed in the PoPs.

Two types of CDNs can be distinguished according the level of awareness of the underlying network [**40**]

**Pure-play CDN:** CDNs that provide over-the-top (OTT) delivery of video and audio content without the ISPs being involved in the control and distribution of the content itself.

**Carrier/Telco CDN:** Broadband providers and Telcos, that provide content delivery as a means to reduce the demands on the network backbone and reduce infrastructure investment

A CDN can be useful for large-scale video streaming. Good Quality of Experience (QoE) in user for video services requires high network bandwidth and low network loading to avoid contention. Thus, centralized video server may obtain satisfactory performance with low video demands because of big Internet infrastructure between them. As demand grows, QoE gets increasingly worse. A CDN can improve content delivery by ensuring that network resources are utilized efficiently. On one hand,

Without a CDN, bandwidth is overused as each request for the same content is retransmitted over and over. On the other hand, with a CDN, bandwidth usage of the content provider is optimized as media content is packaged for delivery by the CDN infrastructure

**3.2. Optimization models for Content Delivery Networks.** Currently, stringent requirements on quality-of-service (QoS) mechanisms, ever-increasing amount of content to be distributed and the wild CDN market competition makes efficient use of available resources highly relevant. Therefore, in the deployment of a CDN arises optimization problems for resources management, allocation, and pricing. We will concentrate on resource management and allocations in CDNs. VoD applications primarily differ from other CDN services by their requirement of significant amounts of bandwidth.

There are three main problems about managing and allocating resources in a CDN [**48**]:

**Proxy Server Location Problem:** Consists of finding the number and location of a given number of proxy (also called cache or mirror) servers to be deployed, such that a predefined measure (e.g., traffic flow, total cost) is minimized. The mathematical models for this problem are based on or variations of, the uncapacitated p-median or facility location problems.

**Request Routing:** Routing in a computer network refers to sending data from one or more sources to one or more destinations so as to minimize the total traffic flowing in the network. Request routing in a CDN is the process of guiding a client's request to a suitable proxy server that is able to serve the corresponding request. The problem is formally defined as selecting a proxy server to address a request for an object such that a cost function is minimized.

**Object Placement:** Most of the formulations for CDNs assume that the whole content is entirely replicated onto the caching servers. Unfortunately, this may not always be possible in situations where the objects are significantly large in size (i.e., multimedia files) or the number of objects is huge. In those cases only a partial replication can be performed due to the limited storage capacity of the caching servers (i.e., PoPs). Therefore, any caching server can only hold a subset of the content. Determining which objects should be placed at each caching server under storage capacity restrictions is known as the object placement problem.

A good review of the optimization problems associated with CDNs can be found in [**48**] and the references therein. Mathematical models for resource allocation in CDNs has substantially grown and become more complex problems by including simultaneous consideration of one or more of the three main problems above.

In this project, we deal with a combined problem of video placement and request routing. In addition, the problem to be solved considers a Carrier CDN because we use network topologies from three major Telecommunication companies. Therefore, the objective is to reduce the network load while guarantee that all videos are stored at some server and user demands are satisfied without exceeding carrier network capacities.

# Chapter 2
# Optimal Placement for Video On Demand Systems

This chapter presents the optimization model employed in this project to assign the videos to storage locations (i.e., PoP) across a CDN. The model was originally proposed in [**4**] and [**3**]. The objective of the model is to minimize the total data transfer between nodes to satisfy the video demand which are not available in the local repository. In addition, the chapter describes the topology networks and data used to test the model.

## 1. The problem formulation

As we anticipated, the optimization problem aims to find the assignment strategy for videos to minimize the network load. To provide realism to the solution, the model considers a credible request pattern of videos for the peak hours of the weekend. Moreover, the model considers links and disk capacity constraints.

In general, videos on CDN must be completely stored in a location. This makes the optimization model a Mixed Integer Problem (MIP) and therefore challenging because the solutions should not be fractional. Nonetheless, even with the relaxation of the integer variables, the problem remains difficult because of its huge number of variables and constraints.

**1.1. Parameters and variables.** Table 1 summarizes the input data and decisions variables used in the optimization model for location of videos in a CDN. Let $V$ denote the set of CDN's PoPs. To agree with the terminology of Applegate et al. [**3**], we refer to PoPs as Video Hub Offices (VHOs). The VHOs store videos to distribute them to the local clients. $L$ represent the set of links that connect the nodes with an associated capacity $B_l$ Mbps. The library of videos to be optimally located among the VHO $i \in V$ is denoted by $M$. Every VHO $i$ has an storage capacity, also known as disk capacity, $D_i$. Alike, each video $m \in M$ has a size $S^m$ GB and needs a bitrate of $r^m$ Mbps to be transmitted along the network. To communicate any two nodes $i, j \in V$, they use the shortest directed path $P_{i,j} \in L$, computed in advanced (i.e., the path is fixed in the model and does not change

TABLE 1. Parameters and Decision variables used in model for VoD placement.

| **Parameter** | **Value** |
|---|---|
| $V$ | set of VHO (vertices) |
| $L$ | set of directed links |
| $M$ | set of videos |
| $T$ | set of times slices |
| $D_i$ | disk capacity at $i \in V$ (in GB) |
| $S^m$ | size of video $m \in M$ (in GB) |
| $P_{ij}$ | set of links on path from $i \in V$ to serve request at $j \in V$ |
| $B_l$ | capacity of link $l \in L$ (in Mbps) |
| $r^m$ | bitrate of video $m \in M$ (in Mbps) |
| $a_j^m$ | aggregate number of request for video $m$ at VHO $j$ |
| $f_j^m(t)$ | number of request for video $m$ at VHO $j$ active at time $t$ |
| $c_{ij}$ | cost of transferring one GB from VHO $i$ to VHO $j$ |
| **Decision Variable** | **Meaning** |
| $y_i^m$ | binary variable indicating if video $m$ is stored at VHO $i$ |
| $x_{ij}^m$ | fraction of request for video $m$ at VHO $j$ served by VHO $i$ |

regardless the traffic in the network). The path $P_{i,j}$ is used to serve a request of a video $m$ in node $j$ from a distant VHO $i$. Thus, if a video is served locally, the path is empty $P_{i,i} = \emptyset$.

The total demand of a video $m \in M$ from each VHO $j \in V$ for the week is denoted by $a_j^m$. Moreover, the simultaneous demand for videos $f_j^m(t)$ from a node $j$ at a peak hour $t \in T$ is used to ensure that the links $l \in L$ are not overloaded. Finally, $c_{ij}$ refers the cost to transfer 1GB from node $i$ to $j$. Recalling that to reach $j$, the data has to travel along the path $P_{ij}$, authors that proposed the VHO location model suggest to use:

$$c_{ij} = \alpha |P_{ij}| + \beta, \tag{28}$$

where $|P_{ij}|$ refers to the number of nodes that traverses the path between $i$ and $j$, $\alpha$ is the cost of sending 1GB over any link in $L$, and $\beta$ is a fixed cost for satisfying a request such as lookup operations or local delivery.

Regarding the decision variables, $x_{ij}^m$ is the fraction of requests of video $m$ at VHO $j$ that are fulfilled from VHO $i$. Thus, $x_{ii}$ is the portion of videos served locally.The other set variables $y_i^m$ indicates if node $i$ stores the video $m$. $y_i^m$ is a binary variable because a video cannot be stored partially.

**1.2. The model.** The objective of the model is to minimize the total cost of retrieving videos from remote locations during a period, subject to constraints of disk space and link bandwidth. The formulation of the linear relaxation of this MIP is as follows:

$$\min_{x_{ij}^m,\ y_i^m} \quad \sum_{m\in M}\sum_{i,j\in V} S^m a_j^m c_{ij} x_{ij}^m \tag{29}$$

$$\text{s.t.} \quad \sum_{i\in V} x_{ij}^m = 1, \qquad \forall m\in M,\ j\in V \tag{30}$$

$$x_{ij}^m \le y_i^m, \qquad \forall m\in M,\ i,j\in V \tag{31}$$

$$\sum_{m\in M} S^m y_i^m \le D_i, \qquad \forall i\in V \tag{32}$$

$$\sum_{m\in M}\sum_{\substack{i,j\in V:\\ l\in P_{ij}}} r^m f_j^m(t) x_{ij}^m \le B_l, \qquad \forall l\in L,\ \forall t\in T \tag{33}$$

$$x_{ij}^m \ge 0, \qquad \forall m\in M,\ i,j\in V \tag{34}$$

$$y_i^m \ge 0, \qquad \forall m\in M,\ i,j\in V. \tag{35}$$

The objective function expressed by (29) is the total cost of transferring videos among VHOs to serve the total requests of videos $a_j^m$ from all $j \in V$. Constraint (30) guarantees that the total demand of video $m$ in any VHO $j$ is satisfied by the sum of fractions $x_{ij}$ served from the set of nodes $i \in V$. Therefore, $x_{ij}$ have to be non-negative as constraint (34) states. Constraint (31) satisfies the fact that only a node that stores a video $m$ can deliver requests of it. Constraint (32) refers to the limited storage capacity of each VHO. Constraint (33) assures that the bandwidth of each link $B_l$ is not overflow during the rush hours of the weekend. Originally, since a whole video has to be located in a location, this $y_i^m$ must be 0 or 1, however, in this project we focus on solving the linear relaxation expressed by constraint (35). Indeed the (optimal) solution of the binary formulation is beyond the limits of current technology. Solving the linear relaxation is already a hard task.

After an inspection of the problem, reader can realize that variables $x_{ij}^m$ and $y_i^m$ from different values $m$ only appears together in constraints (33) and (32), respectively. Hence, constraints (30) and (31) are arranged in $|M|$ independent, separable, diagonally located blocks of equations. This is precisely, the structure of problems for which the specialized solver `BlockIP` was designed.

Now, we carefully study the structure of one of these identical $|M|$ blocks. To do this, we define $x_j^m$, $y^m$, $D(x_j^m)$ and $D(y^m)$ as:

$$x_j^m = \begin{pmatrix} x_{1j}^m \\ x_{2j}^m \\ \vdots \\ x_{nj}^m \end{pmatrix}, y^m = \begin{pmatrix} y_1^m \\ y_2^m \\ \vdots \\ y_n^m \end{pmatrix} D(x_j^m) = \begin{pmatrix} x_{1j}^m & 0 & \dots & 0 \\ 0 & x_{2j}^m & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & x_{nj}^m \end{pmatrix}, D(y^m) = \begin{pmatrix} y_1^m & 0 & \dots & 0 \\ 0 & y_2^m & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & y^m \end{pmatrix}.$$

Then the block of constraints corresponding to video $m$ is given by (36).

$$N^m = \left(\begin{array}{cccc|cccc|c} {x_1^m}^T & 0 & \cdots & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & {x_2^m}^T & \cdots & 0 & 0 & \vdots & \ddots & & 0 \\ \vdots & \cdots & \ddots & \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & {x_n^m}^T & 0 & 0 & \cdots & \cdots & 0 \\ \hline D(x_1^m) & 0 & \cdots & 0 & -D(y^m) & D(s_1^m) & 0 & \cdots & 0 \\ 0 & D(x_2^m) & \cdots & 0 & -D(y^m) & 0 & D(s_2^m) & \cdots & 0 \\ \vdots & \cdots & \ddots & \vdots & -D(y^m) & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & \cdots & D(x_n^m) & -D(y^m) & 0 & 0 & \cdots & D(s_n^m) \end{array}\right), \tag{36}$$

where the rows above the horizontal line are the expansion of constraint (30). On the other hand, rows below the horizontal line match constraint (31). In order to agree with `BlockIP` format these rows include slack variables $s_{ij}$ to write the equations as equalities, where $s_j^m$ and $D(s_j^m)$ are:

$$s_j^m = \begin{pmatrix} s_{1j}^m \\ s_{2j}^m \\ \vdots \\ s_{nj}^m \end{pmatrix}, \qquad D(s_j^m) = \begin{pmatrix} s_{1j}^m & 0 & \dots & 0 \\ 0 & s_{2j}^m & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & s_{nj}^m \end{pmatrix}.$$

The dimensions of $N^m$ (36) are $(n + n^2) \times (2n^2 + n)$, where $n = |V|$ and has $4n^2$ non-zero positions. It is worth noting that the matrix $N^m$ is sparse, the columns of $x_{ij}^m \forall i, j \in V$ only have two non-zero positions and slack variables $s_{ij}^m \forall i, j \in V$ have a single non-zero position. That is not the case of $y^m$ columns where all of them have $n$ non zeros. The dense $y^m$ columns makes that the percentage of non-zeros positions in the computation of $N^m {N^m}^T$ is considerably higher than the $N^m$ ones. $N^m {N^m}^T$ is used by `BlockIP` in its operation, so the a denser matrix leads to a slower operation of the solver and more memory consumption because of the higher number of operations.

To overcome the issue of dense columns $D(y^m)$ in (36), we will use $y_{ij}^m$ instead of $y_i^m \in y^m$ , so variables $y_{ij}^m$ match one to one to $x_{ij}^m$. In order to preserve the constraints of the model we replace constraint (31) by (37) and (38). The later equation guarantees that $y_{ij}^m$ have the same value for all $j \in V$. Therefore, the results obtained with this constraint replacement leads to the same results of the original model.

$$x_{ij}^m \leq y_{ij}^m, \qquad \forall m \in M,\ i,j \in V \tag{37}$$

$$y_{ij}^m - y_{ij+1}^m = 0, \qquad \forall m \in M,\ i \in V, j \in V \setminus \{n\} \tag{38}$$

Eq.(39) shows the structure of a modified block of constraint $\hat{N}^m \ \ \forall m \in M$.

(39)

$$\hat{N}^m = \left(\begin{array}{cccc|cccccccc}
{x_1^m}^T & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & 0 & \cdots & \cdots & 0 \\
0 & {x_2^m}^T & \cdots & 0 & 0 & \ddots & & 0 & 0 & \ddots & & 0 \\
\vdots & \cdots & \ddots & \vdots & \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\
0 & 0 & \cdots & {x_n^m}^T & 0 & \cdots & \cdots & 0_{n,2n^2} & 0 & \cdots & \cdots & 0 \\
\hline
D(x_1^m) & 0 & \cdots & 0 & -D(y_1^m) & & & & D(s_1^m) & 0 & \cdots & 0 \\
0 & D(x_2^m) & \cdots & 0 & & -D(y_2^m) & & & 0 & D(s_2^m) & \cdots & 0 \\
\vdots & \cdots & \ddots & \vdots & & & \ddots & & \vdots & \cdots & \ddots & \vdots \\
0 & 0 & \cdots & D(x_n^m) & & & & -D(y_n^m) & 0 & 0 & \cdots & D(s_n^m) \\
\hline
0 & 0 & \cdots & 0 & D(y_1^m) & -D(y_2^m) & 0 \cdots & \cdots 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & D(y_2^m) & -D(y_3^m) & \vdots & 0 & 0 & \cdots & 0 \\
\vdots & \cdots & \ddots & \vdots & & & \ddots & & \vdots & \cdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & & & & D(y_{n-1}^m) - D(y_n^m) & 0 & 0 & \cdots & 0
\end{array}\right),$$

where $y_i^m$ and $D(y_i^m)$ are:

$$y_j^m = \begin{pmatrix} y_{1j}^m \\ y_{2j}^m \\ \vdots \\ y_{nj}^m \end{pmatrix}, \qquad D(y_j^m) = \begin{pmatrix} y_{1j}^m & 0 & \ldots & 0 \\ 0 & y_{2j}^m & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & y_{nj}^m \end{pmatrix}.$$

As can be seen the total number of variables are $3n^2$ and the number of rows increases in $n^2 - n$ to reach $2n^2$ constraints. Despite the increment in the dimensions of the matrix $\hat{N}^m$ compared to $N^m$, its structure is more convenient for computations of `BlockIP` since none column has more than 3 non-zero positions.

Regarding the linking constraints of the model expressed in equations (32) and (33), the structure of each $m$ block of constraint $L^m$ is shown in (40).

(40)

$$\hat{L}^m = \left(\begin{array}{cccc|c|c} 0 & 0 & \cdots & 0 & S^m D(y_1^m)\ 0\ \cdots 0 & 0\cdots\cdots 0 \\ \hline r^m f_1(1) P_1 D(x_1^m)\, r^m f_2(1) P_2 D(x_2^m) \cdots r^m f_n(1) P_n D(x_n^m) & & & & 0 \quad \cdots\cdots 0 & 0\cdots\cdots 0 \\ r^m f_1(2) P_1 D(x_1^m)\, r^m f_2(2) P_2 D(x_2^m) \cdots r^m f_n(2) P_n D(x_n^m) & & & & 0 \quad \cdots\cdots 0 & 0\cdots\cdots 0 \end{array}\right),$$

where $P_j$ is a matrix of shortest path from every node $i \in V$ to destination $j$. More precisely, $P_j$ has the following structure:

$$p_{ij} = \begin{pmatrix} l_1^{ij} \\ l_2^{ij} \\ \vdots \\ l_{|L|}^{ij} \end{pmatrix}, \qquad P_j = \begin{pmatrix} p_{1j}\ p_{2j}\ \ldots\ p_{nj} \end{pmatrix}.$$

Notice that each column $p_{ij}$ of matrix $P_j$ represents the set of links that form the shortest path between nodes $i$ and $j$, where $l_k^{ij} = 1$ means that link $k$ is in the shortest path $p_{ij}$, otherwise (i.e., $l_k^{ij} = 0$) link $k$ is not in the path.

The matrix $\hat{L}^m$ is written for the optimization model with constraints (37) and (38) instead of (31), so $\hat{L}^m$ has $3n^2$ columns. The first $n$ rows (above the horizontal line) correspond to constraint in (32) in which if a portion $y_{i1}$ of video $m$ is stored in node $i$ then the disk space of node $i$ $D_i$ must be reduced in $S^m y_{i1}$. Notice that there is not need of the aforementioned constraint for other $y_{ij}, i \neq 1$ because (38) guarantees that $y_{ik} = y_{ij}, \forall i, j, k \in V$. The rows below line in (40) are from constraint (33). For the representation we use two time slices, since we will use only two snapshots to test the model. For each time slice $t$ there are $|L|$ rows affected by the request patterns of videos $f_j(t)$ and the data rate needed to send the video $m$. The last $n^2$ columns (after the last vertical line) are for slack variables of 37, which do not play any role in the linking constraints. Despite that the structure of all $\hat{L}^m$ is identical among them, the actual values of the different position depend on the size of videos $S^m$, request pattern of the video at different locations $f_j(t)$ and the rate $r^m$. In practice, to resolve a instance of the problem, we will introduce $|M|$ different $\hat{L}^m$ matrices. Note that the structure of $\hat{L}^m$ is far from ideal for the PGC computation since $D$, defined in (16), is not diagonal. Moreover, $\hat{L}^m$ have many null columns such as those that contains $p_{ii}$ paths (i.e., null paths).

## 2. Input Instance

We will use the same test suite of input instances of [**4**]. This test suite consists of three topologies taken from Rocketfuel [**47**] and six library sizes of videos (5k, 10k, 20k, 50k, 100k, 200k and 1000K). Authors that originally formulated the optimization problem choose two different trade-offs between disk size and link

bandwidth, yielding to 42 instances (42 = 3x7x2). These synthetic instances are designed to mimic salient features of a proprietary data set of the instances' authors.

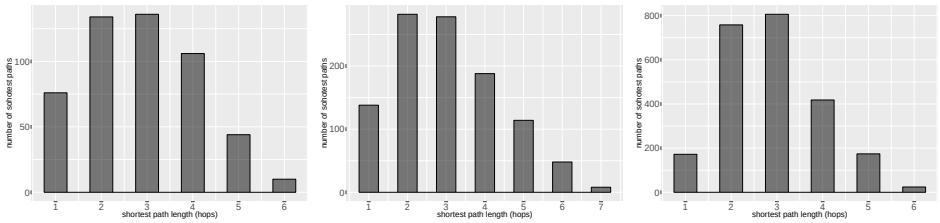
An instance is specified by the data parameters listed in Table 1. The set $T$ contains two time slices, representing the peak viewing hour on Friday and Saturday, respectively. The reason is that the demand during non-peak periods does not overload any links. In addition, the time slices of peak hours in weekend capture the maximum link utilization over an entire week [**4**]. On the other hand, a greater number of time slices (e.g., each second during a 7-day interval) might guarantee that we never exceed the link constraint, however it makes the problem computationally hard because the size of constraints (33) is proportional to the number of time slices in $|T|$.

The three topologies are taken from the following three undirected networks published as part of the Rocketfuel [**47**] data sets:

| **Network Name** | **Rocketfuel Code** | **number of nodes** | **number of undirected edges** | **number of links** | **links/nodes ratio** |
|---|---|---|---|---|---|
| Tiscali | AS3257 | 49 | 86 | 172 | 3.51 |
| Sprintlink | AS1239 | 33 | 69 | 138 | 4.18 |
| Ebone | AS1755 | 23 | 38 | 76 | 3.30 |

TABLE 2. Simulation Settings.

To get a network with full-duplex between each pair of nodes, the original undirected edges were bidirected. In these networks, each node resides in a different city, primarily major cities. Ebone spans Western Europe, plus New York City. Tiscali spans Western Europe, plus New York, Chicago, and Washington D.C. Sprintlink spans the United States and Europe, plus Sydney, Tokyo, Singapore and Hong Kong. The directed path between each pair of nodes is a shortest path with respect to hop count. Fig. 1 shows the distributions of shortest paths' length for the three network topologies.




(A) Ebone Network. (B) Sprintlink Network. (C) Tiscali Network.

FIG. 1. Histogram of shortest paths.

As can be seen, most of the shortest paths range from two to four hops. However, the links' utilization in terms of the number of shortest paths that use each link, depicted in Fig. 2 for the three network topologies, is not homogeneous. On one hand, there are a few links whose utilization is dis-proportionally high compared

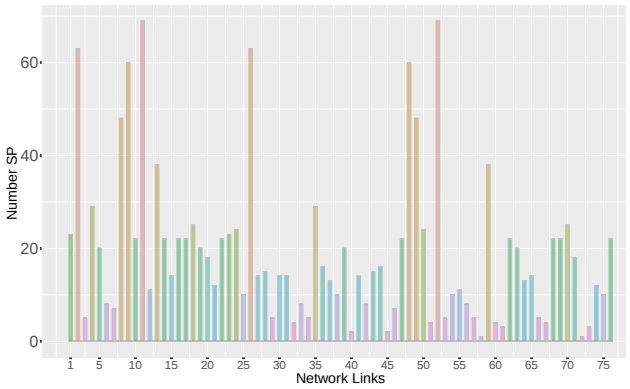
to the average value. On the other, Fig. 2 shows that roughly one third of the links has a low utilization, therefore that all links have the same capacity seems not to be a realistic assumption. It is more likely that links with high utilization have a bandwidth capacity larger than the other links. Nevertheless, the authors of instances use the the same bandwidth for all links, which depends of the video library size and disk size.

The disk size and request volume in each node (VHO) was based on the population size of the surrounding metropolitan area that serves each node. Within each VHO, 2% of the requests were assigned to each peak period. Thus, for each VHO and data instance there are three distributions for the video requests: one main distribution, used for the 96% of the requests that are off-peak, and one distribution for each of the two peak periods (2% for each one). The videos were identified by an index between 1 and the library size, with lower indices indicating higher aggregate popularity.

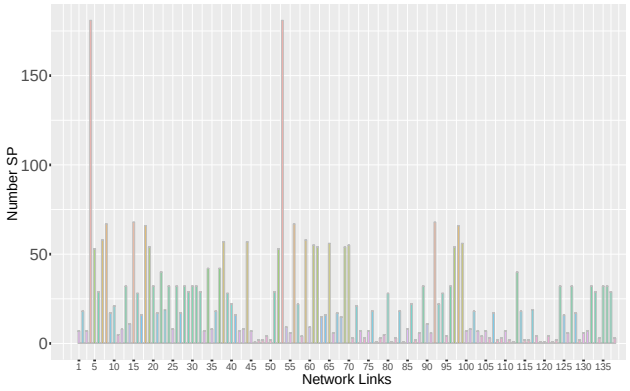
These request distributions do not consider that popularity of videos is affected by the geographic diversity. Thus, each VHO draws from the same three distributions. The main video popularity distribution was derived from the YouTube distribution published by Cha et al. [**20**]. Compared to the proprietary data set of [**4**], the top videos in the YouTube data were disproportionately popular. Therefore, the top 5000 videos were dropped from the YouTube distribution before rescaling it to the library size for the instance at hand. The two peak distributions differ both from each other and the main distribution.To measure how different are these distributions among them, each one is represented as a vector of probabilities, indexed by video, and quantify the difference using cosine similarity. The two peak distributions are perturbations of the main distribution so that the cosine similarity between the two peak distributions is about 0.98, and the cosine similarities between the main distribution and each peak distribution is about 0.74. For each library size, all of generated instances draw their requests from the same distributions.

Regarding the video properties, all videos in the instances have bitrates of 2 Mbps. Each video is assigned one of four sizes 100MB, 500MB, 1GB, or 2GB, which are meant to correspond roughly to 5-minute clips, 30-minute and 1-hour TV shows, and 2-hour movies. Each video index was assigned one of these four sizes independently, with probabilities of approximately 0.29, 0.28, 0.19, and 0.24, respectively. For each combinations of topology and library size, there are two instances, one with a small total amount of disk space, and another with a large amount of it. There are three disk sizes. The disk of a VHO can be large, medium or small, according to the population it serves. Each large VHO has twice the disk space of each medium VHO, which has twice the disk space of each small VHO. For the small-disk instances, the disk sizes are chosen so that the total disk space over all VHOs is twice the size of the entire library. For the large-disk instances, the disk sizes are chosen so that the disk space at a medium VHO can hold 20% of the entire library.

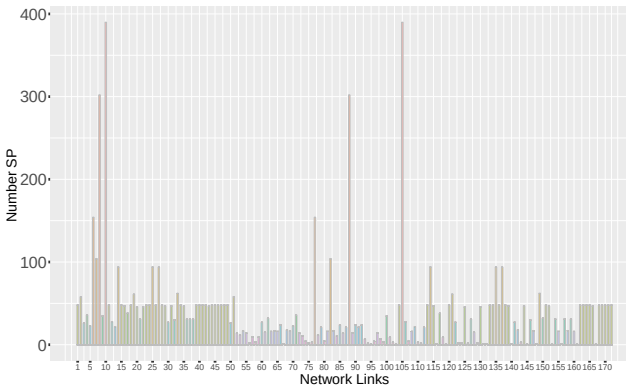
As we anticipate, all links have the same bandwidth within each instance . Authors of instances chose the actual value in such a way that both the disk capacity and link bandwidth constraints are challenging to the optimization. To this end, the link bandwidth for each instance is the smallest round number for which the instance




(A) Ebone Network.




(B) Sprintlink Network.




(C) Tiscali Network.

FIG. 2. Links utilization in terms of number of shortest paths that go through the links.

is feasible. Since the instances generated have very tight values of disk and link capacities, in this project we create a third combination with the largest values of capacity for disks and links. We will see that even this more feasible combination becomes hard to solve for optimization tools.

Table 3 shows the values of disk size and bandwidth used to build the problem instance for five video library sizes in the three network topologies. There are two additional sizes of libraries (i.e., 200k and 1000k videos). However, we do not test them because they are too large to be executed in our server.

TABLE 3. Disk sizes and link capacities for instance generation of VHO problem.

| | | | **Sprintlink** | | **Tiscali** | | **Ebone** | |
|---|---|---|---|---|---|---|---|---|
| Number of videos | Disk type | Link type | Disk size (GB) | Link BW (Mbps) | Disk size (GB) | Link BW (Mbps) | Disk size (GB) | Link BW (Mbps) |
| 5k | small | large | 260 | 2000 | 180 | 3750 | 440 | 2000 |
| | large | small | 860 | 425 | 870 | 150 | 1000 | 900 |
| | large | large | 860 | 2000 | 870 | 3750 | 1000 | 2000 |
| 10k | small | large | 540 | 3750 | 360 | 8000 | 880 | 4500 |
| | large | small | 1780 | 700 | 1760 | 300 | 2020 | 1700 |
| | large | large | 1780 | 3750 | 1760 | 8000 | 2020 | 4500 |
| 20k | small | large | 1070 | 7500 | 720 | 16000 | 1740 | 8500 |
| | large | small | 3530 | 1500 | 3530 | 650 | 4000 | 3250 |
| | large | large | 3530 | 7500 | 3530 | 16000 | 4000 | 8500 |
| 50k | small | large | 2700 | 19000 | 1810 | 37500 | 4380 | 21000 |
| | large | small | 8910 | 3750 | 8880 | 1500 | 10070 | 8500 |
| | large | large | 8910 | 19000 | 8880 | 37500 | 10070 | 21000 |
| 100k | small | large | 5400 | 38000 | 3600 | 77500 | 8760 | 45000 |
| | large | small | 17800 | 7200 | 17600 | 3250 | 20140 | 17000 |
| | large | large | 17800 | 38000 | 17600 | 77500 | 20140 | 45000 |

To conclude this section we show the topologies of the three networks, i.e., Ebone, Sprintlink and Tiscali in Figs. 3, 4 and 5, respectively. These plots show the connectivity among VHOs, their comparative disk size and the link utilization. It is clear that they are difficult instance for the video placement problem because Links with number of shortest path through them are not necessarily connected to VHOs with big disk size.

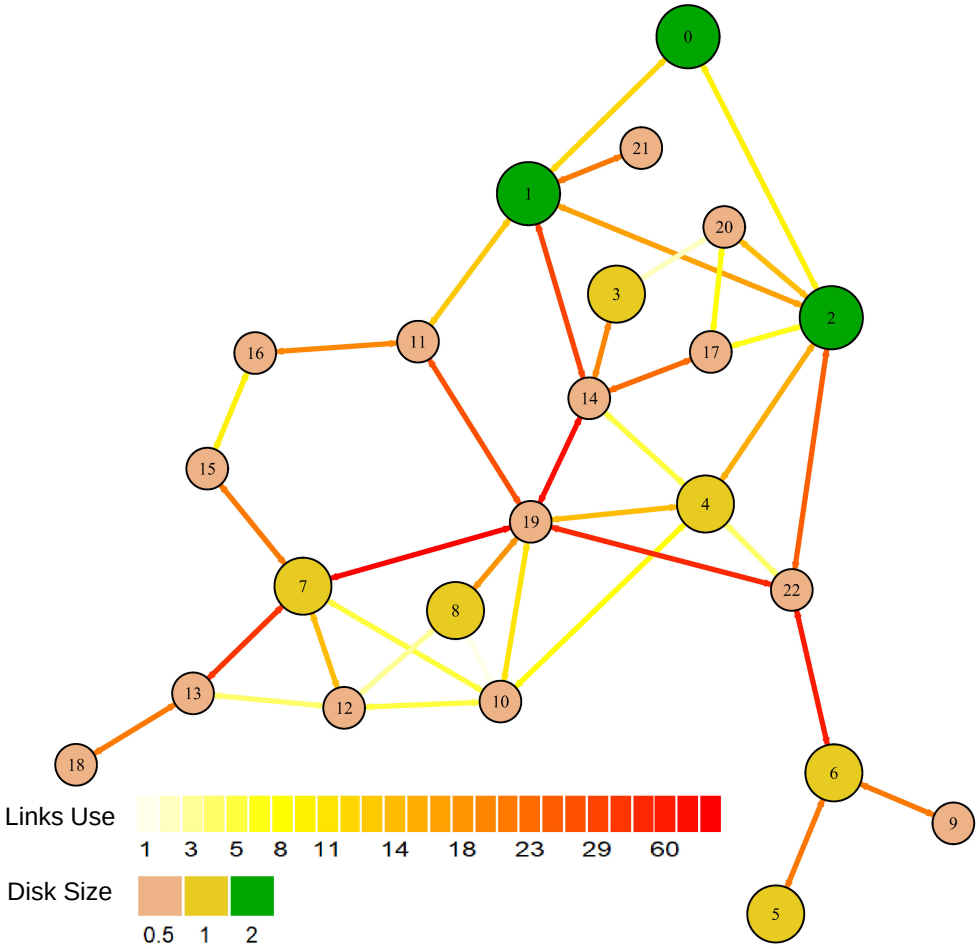



FIG. 3. Ebone Network.

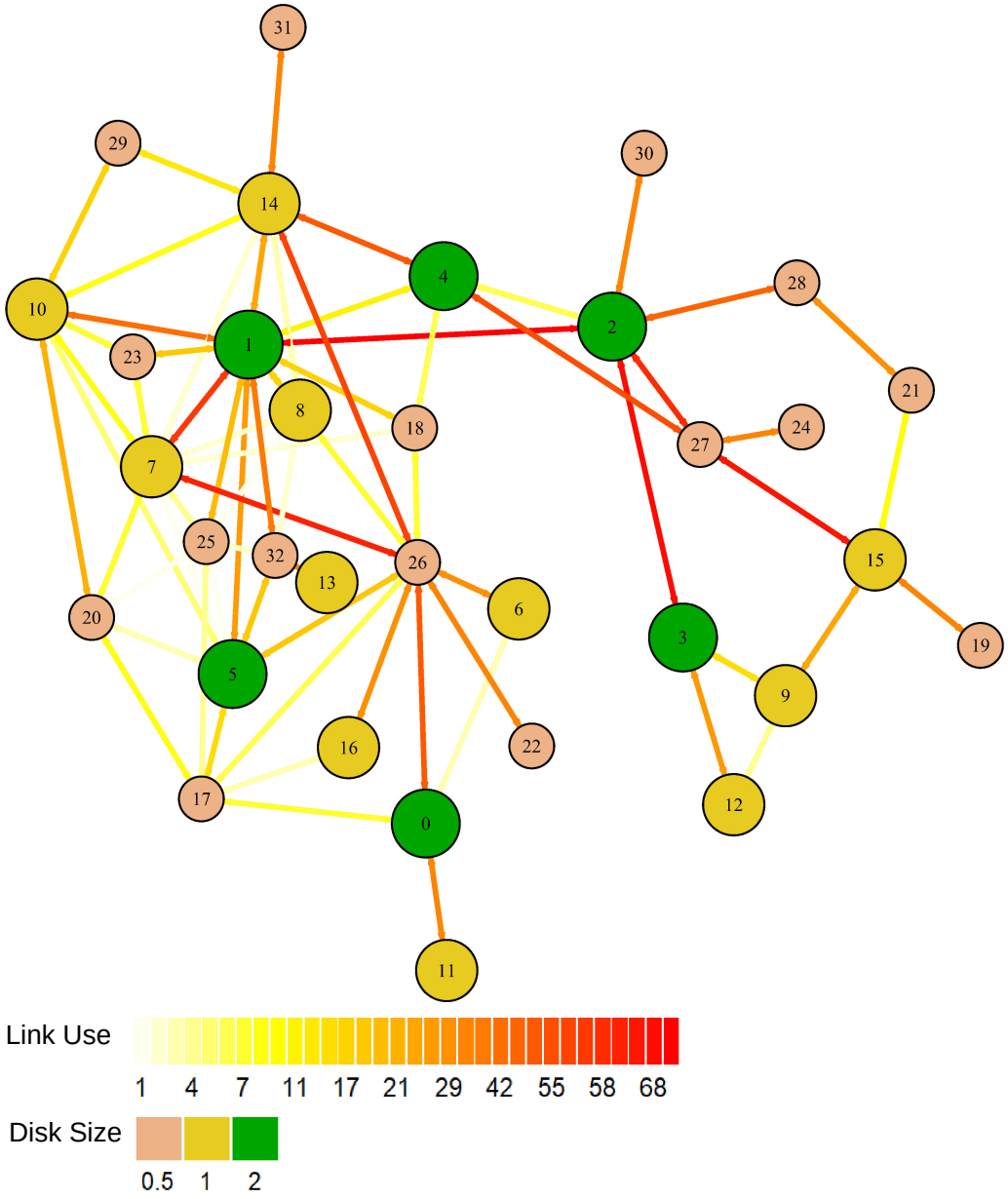



FIG. 4. Sprintlink Network.

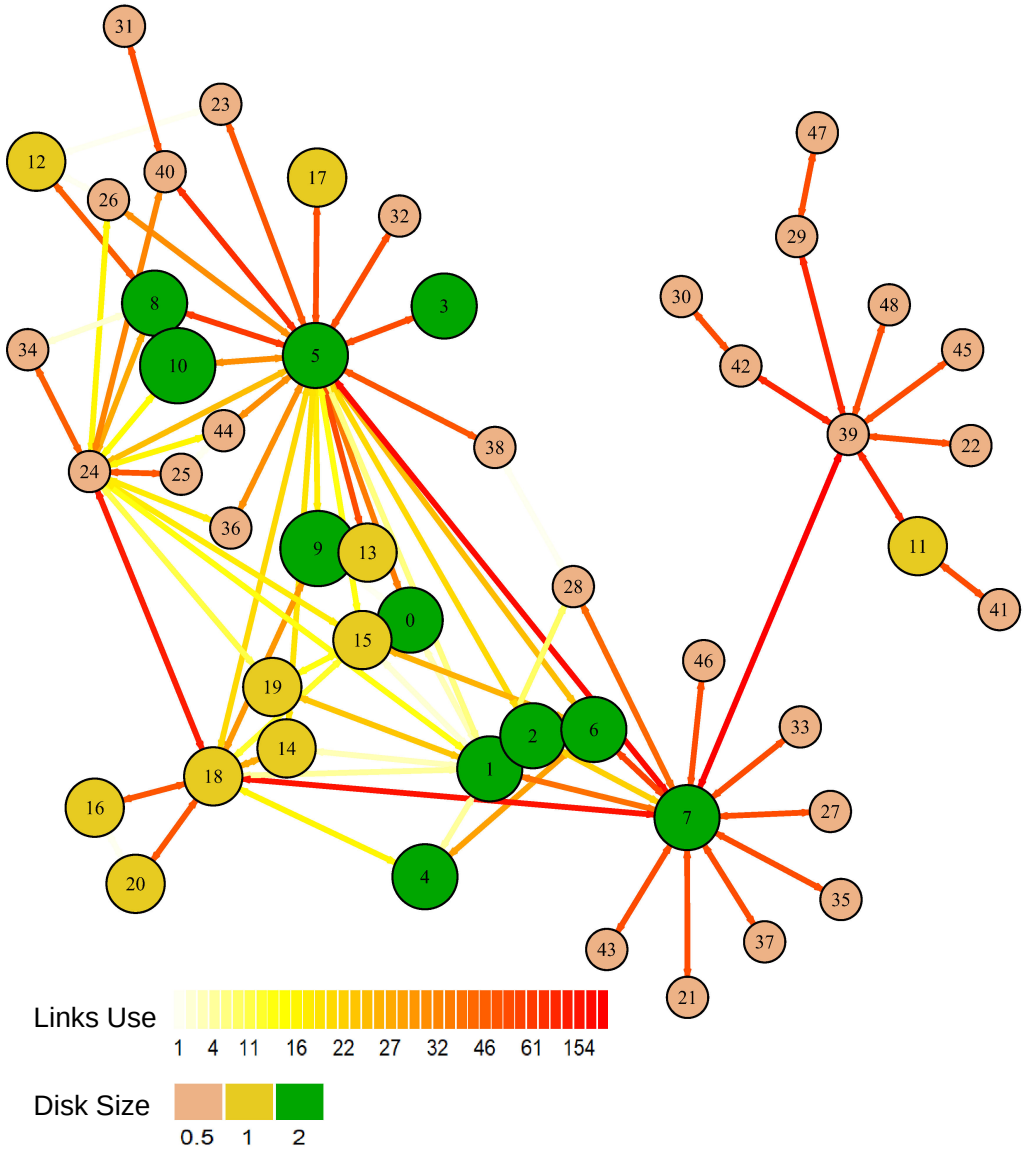



FIG. 5. Tiscali Network.

# Chapter 3
# Implementation and Results

This chapter presents the implementation details and results from testing the VHO instances in Ebone, Sprintlink and Tiscali networks with `BlockIP` solver. More precisely, we test the `BlockIP`'s solver options with the smallest instances (i.e., 5000 videos in the library) for these three networks. Afterwards, we only evaluate bigger instances with `BlockIP` set up that provided best results. We decided to follow this path because the time to solve an instance increases considerably with the number of variables. Afterwards, in order to compare the performance of `BlockIP`, we solve the same instances with CPLEX, properly configured to get a fair analysis.

## 1. Implementation Details

To solve the optimal video location problem in network of Video Hub Offices, we used the `BlockIP` callable library. The first step was to read the problem information, which includes: transfer cost for videos, aggregate an peak hours demand, disk sizes, and paths among nodes. Then, with these data, we build block matrix 39 and linking matrices 40, After that, we create the problem with the routines provided by `BlockIP` library. The resulting program, written in C++ as `BlockIP`, as well as the input data can be download from `http://www.lfurquiza.com/research/codes`. The zip file includes an awl script to run a batch of problems and does not include `BlockIP`. `BlockIP` can be downloaded from `http://www-eio.upc.edu/~jcastro/BlockIP.html`.

The options of `BlockIP` that we will test are:

**Type of direction:** `BlockIP` has two ways to compute the direction of movement: automatic and second order (predictor-corrector). The first one use the Newton direction. The latter can be more precise than the automatic option, nevertheless it comes at the cost of a greater number of conjugate gradient iterations because it solves two systems of equations. A value $h = 0$ was considered for the number of terms of the power series preconditioner.

**Infeasibility tolerance:** `BlockIP` allow us to set the infeasibility tolerance of the solution provided by the solver. A high tolerance could lead to a shorter time for getting a solution.

**Quadratic Regularization factor:** The quadratic regularization helps linear optimization problems to find an optimal solution faster than without regularization because quadratic regularization decreases the spectral radius and makes PCG more efficient. As the regularization factor increases the importance of quadratic terms in the objective function increases as well and this could modify the value of the objective function and the dual feasibility condition.

We compare `BlockIP` results with state-of-the-art CPLEX 12.5 package. Regarding CPLEX, we set it to use only one thread because `BlockIP` do not exploit parallelism. In addition, we set the barrier algorithm 3 and the optimality gap equal to the obtained by `BlockIP`. The CPU time provided for CPLEX is only for the barrier iterations, without crossover. All runs were carried out on a server at 2.60 GHz Intel Xeon E5-2690 CPUs with 192 gigabytes of memory, under a GNU/Linux operating system (OpenSuse 13.2)

## 2. Results for small instances

The smallest instance for the three different network topology are obtained by using the library of 5000 videos. So, the resulting problem has five thousand diagonal blocks. The actual number of variables into a block depends on the numbers of VHO in the network. Among the three networks, Tiscali will produce the largest problem and Ebone the smallest one.

**2.1. Ebone.** As we said, the smallest instances produce 5000 blocks. In the case of Ebone Network this number of blocks leads to a total number of variables of 7935175. The detailed information of variables and constraints are depicted in Table 1.

| **Parameter** | **Value** |
|---|---|
| Block Variables | 1587 |
| Block Constraints | 1058 |
| Linking constraints | 175 |
| Total Variables | 7935175 |
| Total constraints | 5290175 |

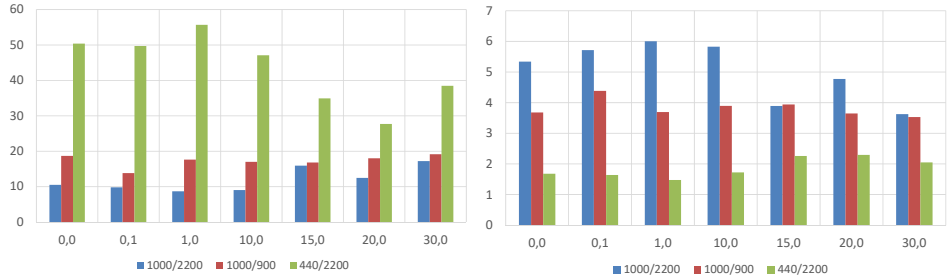
TABLE 1. Descriptive information of smallest instance in Ebone network with a library of 5000 videos

Table 2 shows the results of testing the aforementioned `BlockIP` options in the 5k instance of Ebone network. The fist two rows of each section of the Table 2 show the results of using automatic direction, none regularization factor (i.e., Q=0) and the infeasibility gaps of 1e-7 and 1e-2. These results indicates that a tight infeasibility gap do not affect the solution time. Hence, we test the other options combinations of `BlockIP` that uses automatic direction with an infeasibility of 1e-7.

Regarding the option second order direction, we test this option with a infeasibility gap of 1e-2. Results show that the intensive computation operations that this

options demands is not worthing for this particular problem because the optimality gap is not better than the obtained with automatic direction. Moreover, a more strict infeasibility gap of 1e-7 leads to the same results and increases even more the solution time as it can be seen for the instance with disks of 440GB and links of 2200 Mbps.

The best results for the Ebone network for a library size of 5000 videos were obtained with automatic direction and a quadratic regularization factor. The best regularization factor depends on the disk size and link capacity of the instance at hand. For the small disks and large links (i.e., 440GB and 2200 Mbps) a regularization factor of 20 gets and optimality gap in the order of 1e-5 with smallest number of PCG iterations per `BlockIP` major iteration (See Fig. 1a) and therefore the greatest average number of BlockIP iterations per minute (Fig. 1b) and in the shortest time.




(A) PCG iterations per `BlockIP` iteration vs Quadratic regularization factor

(B) `BlockIP` iterations per minute vs Quadratic regularization factor

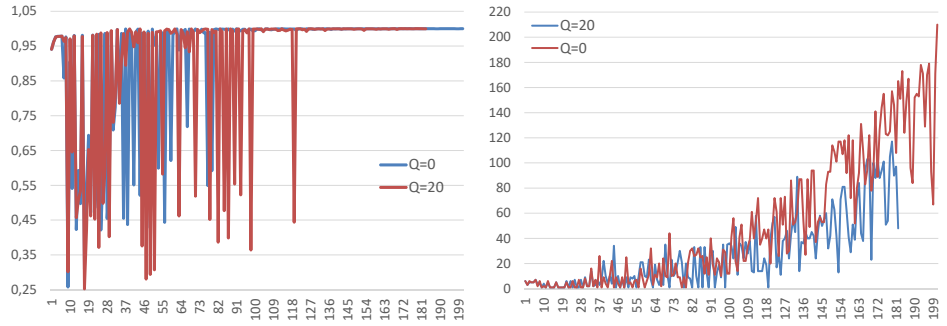
FIG. 1. `BlockIP` performance for different values of quadratic regularization in Ebone network with a library of 5000 videos

On the other hand, in the scenario with "big" disks and tight links capacities (1000 GB and 900 Mbps), a factor of 1.0 or 100 lead to 2% optimal solution. The more feasible instance (1000 GB and 2200 Mbps) needs a more conservative regularization factor of 0.1 or 1.0 to reach a 0.01% optimal solution. Notice that, the optimality gap improvement got in the latter instance due to the use of quadratic regularization is not as important as in the two previous, more challenging instances. Additionally, the use of a regularization factor comes at the cost of a slightly increment in the infeasibility of dual solutions, however it is not significant.

We also include Fig. 2 to show the performance of `BlockIP` decreases while it approaches to the optimum solution, as it was pointed out in [**19**]. Results showed in Fig. 2 are from the most time demanding instance of "small" disks - "large" links. As in the reader can notice in the last 25% of iterations the estimated spectral radius of the inverse of (20) is 1 (See Fig. 2a) and therefore the number of PCG iterations needed in those cases increases considerable (Fig. 2b). From these results, it is reasonable to conclude that most of the solution time is due to the last iterations. However, for VHO problem is not possible to switch to full Cholesky factorization to overcome this issue, as [**19**] suggested, because of the size of the problem.

| Type | | Option | | BlockIP | Primal Objective | | Dual Objective | | | Opt. | C. Grad. | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Dir. | Feas. | Reg. | Iterat. | Value | Feas. | Gap | Value | Feas. | Gap | Gap | iterat. | (min) |
| 440/2200 | auto | 1e-7 | 0.0 | 201 | 3.053808e+07 | 3.44e-02 | 2.845779e+07 | 4.20e-16 | 6.81e-02 | 10128 | 119.69 | | |
| 440/2200 | auto | 1e-2 | 0.0 | 201 | 3.053808e+07 | 3.44e-02 | 2.845779e+07 | 4.20e-16 | 6.81e-02 | 10128 | 120.07 | | |
| 440/2200 | S.O. | 1e-7 | 0.0 | 86 | 3.756273e+07 | 9.05e-03 | -1.733239e+07 | 3.30e-15 | 1.46e+00 | 16760 | 208.64 | | |
| 440/2200 | S.O. | 1e-2 | 0.0 | 86 | 3.756273e+07 | 9.05e-03 | -1.733239e+07 | 3.30e-15 | 1.46e+00 | 16760 | 181.27 | | |
| 440/2200 | auto | 1e-7 | 0.1 | 199 | 3.054501e+07 | 3.43e-02 | 2.845959e+07 | 1.10e-06 | 6.83e-02 | 9896 | 121.49 | | |
| 440/2200 | auto | 1e-7 | 1.0 | 218 | 3.025824e+07 | 2.69e-02 | 2.875234e+07 | 1.19e-05 | 4.98e-02 | 12141 | 147.51 | | |
| 440/2200 | auto | 1e-7 | 10.0 | 225 | 2.991432e+07 | 1.76e-02 | 2.966640e+07 | 7.08e-05 | 8.29e-03 | 10589 | 130.41 | | |
| 440/2200 | auto | 1e-7 | 15.0 | 204 | 3.014656e+07 | 2.34e-02 | 3.004562e+07 | 1.05e-04 | 3.35e-03 | 7121 | 90.34 | | |
| 440/2200 | auto | 1e-7 | 20.0 | 182 | 3.045123e+07 | 3.11e-02 | 3.044938e+07 | 1.45e-04 | 6.09e-05 | 5038 | 79.30 | | |
| 440/2200 | auto | 1e-7 | 30.0 | 216 | 2.988471e+07 | 1.57e-02 | 3.071943e+07 | 1.56e-04 | 2.79e-02 | 8312 | 105.25 | | |
| 1000/900 | auto | 1e-7 | 0.0 | 123 | 1.008823e+07 | 2.90e-03 | 9.761668e+06 | 9.19e-06 | 3.24e-02 | 2294 | 33.42 | | |
| 1000/900 | auto | 1e-2 | 0.0 | 123 | 1.008823e+07 | 2.90e-03 | 9.761668e+06 | 9.19e-06 | 3.24e-02 | 2294 | 33.45 | | |
| 1000/900 | S.O. | 1e-2 | 0.0 | 95 | 1.082892e+07 | 5.86e-05 | 7.942960e+06 | 7.11e-06 | 2.67e-01 | 15691 | 172.30 | | |
| 1000/900 | auto | 1e-7 | 0.1 | 104 | 1.041669e+07 | 1.21e-02 | 9.433713e+06 | 6.26e-06 | 9.44e-02 | 1437 | 23.73 | | |
| 1000/900 | auto | 1e-7 | 1.0 | 121 | 1.010332e+07 | 3.43e-03 | 9.807502e+06 | 9.76e-06 | 2.93e-02 | 2131 | 32.76 | | |
| 1000/900 | auto | 1e-7 | 10.0 | 122 | 1.006919e+07 | 2.62e-03 | 1.027935e+07 | 4.73e-05 | 2.09e-02 | 2073 | 31.32 | | |
| 1000/900 | auto | 1e-7 | 15.0 | 128 | 1.007763e+07 | 2.83e-03 | 1.054288e+07 | 7.17e-05 | 4.62e-02 | 2151 | 32.49 | | |
| 1000/900 | auto | 1e-7 | 20.0 | 141 | 1.007381e+07 | 2.76e-03 | 1.074588e+07 | 8.64e-05 | 6.67e-02 | 2539 | 38.66 | | |
| 1000/900 | auto | 1e-7 | 30.0 | 156 | 1.006752e+07 | 2.17e-03 | 1.113910e+07 | 1.22e-04 | 1.06e-01 | 2982 | 44.22 | | |
| 1000/2200 | auto | 1e-7 | 0.0 | 154 | 9.689621e+06 | 6.77e-05 | 9.687891e+06 | 3.95e-08 | 1.79e-04 | 1613 | 28.84 | | |
| 1000/2200 | auto | 1e-2 | 0.0 | 154 | 9.689621e+06 | 6.77e-05 | 9.687891e+06 | 3.95e-08 | 1.79e-04 | 1613 | 28.53 | | |
| 1000/2200 | S.O. | 1e-2 | 0.0 | 71 | 9.824971e+06 | 2.26e-05 | 9.610221e+06 | 6.96e-07 | 2.19e-02 | 2846 | 37.07 | | |
| 1000/2200 | auto | 1e-7 | 0.1 | 150 | 9.688294e+06 | 7.73e-05 | 9.687697e+06 | 1.45e-07 | 6.17e-05 | 1469 | 26.26 | | |
| 1000/2200 | auto | 1e-7 | 1.0 | 145 | 9.688574e+06 | 5.49e-05 | 9.687784e+06 | 6.25e-07 | 8.16e-05 | 1257 | 24.16 | | |
| 1000/2200 | auto | 1e-7 | 10.0 | 154 | 9.689392e+06 | 4.93e-05 | 9.697077e+06 | 1.44e-06 | 7.93e-04 | 1386 | 26.43 | | |
| 1000/2200 | auto | 1e-7 | 15.0 | 154 | 9.687499e+06 | 6.41e-05 | 9.688836e+06 | 8.59e-08 | 1.38e-04 | 2451 | 39.55 | | |
| 1000/2200 | auto | 1e-7 | 20.0 | 193 | 9.688062e+06 | 8.01e-06 | 9.697415e+06 | 1.01e-06 | 9.65e-04 | 2406 | 40.43 | | |
| 1000/2200 | auto | 1e-7 | 30.0 | 239 | 9.688306e+06 | 4.69e-06 | 9.712916e+06 | 2.68e-06 | 2.54e-03 | 4108 | 65.95 | | |

TABLE 2. Results with `BlockIP` for Ebone network with a library of 5000 videos (diagonal blocks). This problem has 7935175 variables




(A) Spectrum ratio vs BlockIP iterations (B) PCG iterations vs BlockIP iterations

FIG. 2. BlockIP performance along iterations in Ebone network with a library of 5000 videos. The results are from an instance with average disk size of 440GB and link's bandwidth of 2200 Mbps

The results obtained with CPLEX are shown in Table 3. CPLEX is faster than BlockIP for the VHO problem. Nevertheless, CPLEX is only able to find an optimal solution for the most feasible problem among the three evaluated instances. CPLEX was unable to find feasible primal solutions for the challenging combinations of small disks/large link capacities and big disks / small link capacities.

| Type | Iter | Primal Objective | | | Dual Objective | | | Opt. | Time |
|---|---|---|---|---|---|---|---|---|---|
| | | Value | Feas. | Gap | Value | Feas. | Gap | Gap | (min) |
| 440/2200* | 241 | 3.0255e+07 | 5.14e-01 | | 2.8462e+07 | 1.57e-6 | | 5.92e-02 | 18.64 |
| 1000/900* | 186 | 1.0163e+07 | 1.35e-01 | | 9.8543e+06 | 4.31e-07 | | 3.05e-02 | 14.88 |
| 1000/2200 | 157 | 9.6879e+06 | 2.98e-05 | | 9.6879e+06 | 3.28e-07 | | 1.55e-07 | 12.71 |

TABLE 3. Results with CPLEX for Ebone network with a library of 5000 videos. Non-optimal solutions are marked with *

For the small instance in the Ebone network we can conclude that BlockIP finds optimal and feasible solutions in very tight instances (i.e., problems with a small feasible region). However, when VHO instances are more feasible, CPLEX clearly outperforms BlockIP in terms of processing time.

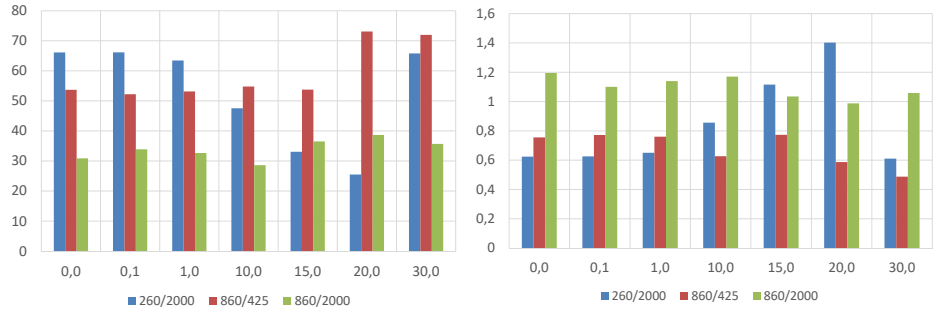
**2.2. Sprintlink.** For the Sprintlink network the 5000 blocks reach a total number of 16335309 variables. The details are in Table 4. Table 5 shows the results of testing options of BlockIP options in the 5k instance of Sprintlink network. As for Ebone network, the fist two rows of each section of the table show the results of using automatic direction, none regularization factor (i.e., Q=0) and the infeasibility gaps of 1e-7 and 1e-2. These results agree with Ebone ones regarding that a relaxed infeasibility gap (i.e., 1e-2) does not degrade the solution but in these instances it reduces the solution time. Hence, we test the other options combinations of BlockIP that uses automatic direction with an infeasibility of 1e-2. Regarding the option second order direction, we test this option with a infeasibility gap of 1e-2. Results show that the intensive computation operations that this option demands

| Parameter | Value |
|---|---|
| Block Variables | 3267 |
| Block Constraints | 2178 |
| Linking constraints | 309 |
| Total Variables | 16335309 |
| Total constraints | 10890309 |

TABLE 4. Descriptive information of smallest instance in Sprintlink network with a library of 5000 videos

is not worthing for this particular problem, except for the more feasible instance. Contrary to what is expected, the optimality gap is worse than the obtained with automatic direction for the two complicate instances (i.e., small disk- large links and vice versa). Again here, as happened in the Ebone instance, a more strict infeasibility gap of 1e-7 leads to the same results and increases even more the solution time as it can be seen for the instance with average disks of 260GB and links of 2000 Mbps.

The results for the Sprintlink network clearly improves when automatic direction and a quadratic regularization factor are applied together. As in the Ebone case, the best regularization factor depends on the disk size and link capacity of the instances. For the small disks and large links (i.e., 260GB and 2000 Mbps) a regularization factor of 20 (the same factor obtained for Ebone) gets an optimality gap in the order of 1e-5 with smallest number of PCG iterations per `BlockIP` major iteration (See Fig. 3a), the highest average number of BlockIP iterations per minute (Fig. 3b) and in the shortest time. On the other hand, in the scenario with "big" disks and tight links capacities (860 GB and 425 Mbps), factors of 0.1 or 1.0 lead to a 6% optimal solution. To solve the more feasible instance (1000 GB and 2200 Mbps) the same conservative regularization factors of 0.1 or 1.0 can be used to reach a 0.01% optimal solution. It is important to note that in the latter instance, `BlockIP`, using second-order direction without regularization factor, finds an optimal solution faster than using automatic direction and quadratic regularization.




(A) PCG iterations per `BlockIP` iteration vs Quadratic regularization factor

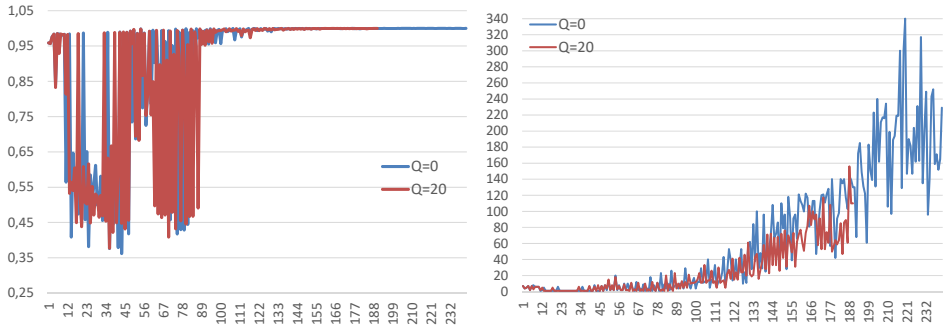
(B) `BlockIP` iterations per minute vs Quadratic regularization factor

FIG. 3. `BlockIP` performance for different values of quadratic regularization in Sprintlink network with a library of 5000 videos

| Type | Option | | BlockIP | Primal Objective | | Dual Objective | | Opt. | C. Grad. | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| | Dir. | Feas. | Reg. | Iterat. | Value | Feas Gap | Value | Feas Gap | Gap | iterat. | (min) |
| 260/2000 | auto | 1e-7 | 0.0 | 241 | 3.630455e+07 | 1.42e-03 | 3.592152e+07 | 1.29e-05 | 1.06e-02 | 15935 | 459.30 |
| 260/2000 | auto | 1e-2 | 0.0 | 241 | 3.630455e+07 | 1.42e-03 | 3.592152e+07 | 1.29e-05 | 1.06e-02 | 15935 | 386.47 |
| 260/2000 | S.O. | 1e-7 | 0.0 | 149 | 4.359295e+07 | 1.33e-03 | 1.826213e+06 | 3.23e-06 | 9.58e-01 | 41273 | 918.63 |
| 260/2000 | S.O. | 1e-2 | 0.0 | 149 | 4.359295e+07 | 1.33e-03 | 1.826213e+06 | 3.23e-06 | 9.58e-01 | 41273 | 815.95 |
| 260/2000 | auto | 1e-2 | 0.1 | 240 | 3.630495e+07 | 1.42e-03 | 3.592576e+07 | 1.29e-05 | 1.04e-02 | 15876 | 383.49 |
| 260/2000 | auto | 1e-2 | 1.0 | 238 | 3.628998e+07 | 1.25e-03 | 3.596898e+07 | 1.33e-05 | 8.85e-03 | 15106 | 365.50 |
| 260/2000 | auto | 1e-2 | 10.0 | 228 | 3.631611e+07 | 1.36e-03 | 3.631861e+07 | 4.76e-05 | 6.90e-03 | 10837 | 266.48 |
| 260/2000 | auto | 1e-2 | 15.0 | 205 | 3.665937e+07 | 4.01e-03 | 3.666259e+07 | 1.24e-04 | 8.78e-05 | 6785 | 183.69 |
| 260/2000 | auto | 1e-2 | 20.0 | 189 | 3.711548e+07 | 8.48e-03 | 3.711836e+07 | 2.27e-04 | 7.75e-05 | 4820 | 134.83 |
| 260/2000 | auto | 1e-2 | 30.0 | 282 | 3.619438e+07 | 3.22e-04 | 3.668512e+07 | 7.56e-05 | 1.36e-02 | 18557 | 462.15 |
| 860/425 | auto | 1e-7 | 0.0 | 162 | 1.040228e+07 | 3.30e-02 | 9.395164e+06 | 5.65e-06 | 9.68e-02 | 8701 | 216.79 |
| 860/425 | auto | 1e-2 | 0.0 | 162 | 1.040228e+07 | 3.30e-02 | 9.395164e+06 | 5.65e-06 | 9.68e-02 | 8701 | 214.47 |
| 860/425 | S.O. | 1e-2 | 0.0 | 112 | 1.035840e+07 | 2.50e-05 | 8.538779e+06 | 4.68e-06 | 1.76e-01 | 40854 | 904.98 |
| 860/425 | auto | 1e-2 | 0.1 | 160 | 1.040095e+07 | 3.29e-02 | 9.406151e+06 | 5.62e-06 | 9.56e-02 | 8354 | 207.15 |
| 860/425 | auto | 1e-2 | 1.0 | 163 | 1.032483e+07 | 2.86e-02 | 9.603340e+06 | 1.00e-05 | 6.99e-02 | 8665 | 214.41 |
| 860/425 | auto | 1e-2 | 10.0 | 206 | 9.849424e+06 | 4.79e-03 | 1.026505e+07 | 3.44e-05 | 4.22e-02 | 11289 | 328.63 |
| 860/425 | auto | 1e-2 | 15.0 | 218 | 9.852558e+06 | 4.84e-03 | 1.055625e+07 | 4.97e-05 | 7.14e-02 | 11722 | 281.97 |
| 860/425 | auto | 1e-2 | 20.0 | 349 | 9.054713e+06 | 1.32e-04 | 9.246526e+06 | 1.17e-05 | 2.12e-02 | 25503 | 593.97 |
| 860/425 | auto | 1e-2 | 30.0 | 400 | 9.332795e+06 | 5.62e-04 | 1.000494e+07 | 3.96e-05 | 7.20e-02 | 28795 | 819.89 |
| 860/2000 | auto | 1e-7 | 0.0 | 238 | 9.465295e+06 | 3.86e-06 | 9.460459e+06 | 6.91e-07 | 5.11e-04 | 7355 | 240.93 |
| 860/2000 | auto | 1e-2 | 0.0 | 238 | 9.465295e+06 | 3.86e-06 | 9.460459e+06 | 6.91e-07 | 5.11e-04 | 7355 | 199.19 |
| 860/2000 | S.O. | 1e-2 | 0.0 | 117 | 9.466474e+06 | 5.79e-07 | 9.459769e+06 | 1.55e-13 | 7.08e-04 | 6925 | 177.27 |
| 860/2000 | auto | 1e-2 | 0.1 | 243 | 9.464631e+06 | 3.71e-06 | 9.461497e+06 | 4.04e-07 | 3.31e-04 | 8243 | 220.89 |
| 860/2000 | auto | 1e-2 | 1.0 | 239 | 9.464049e+06 | 3.03e-06 | 9.462984e+06 | 2.71e-07 | 1.13e-04 | 7804 | 209.73 |
| 860/2000 | auto | 1e-2 | 10.0 | 264 | 9.462953e+06 | 1.84e-06 | 9.469305e+06 | 5.08e-07 | 6.71e-04 | 7557 | 225.66 |
| 860/2000 | auto | 1e-2 | 15.0 | 309 | 9.462722e+06 | 2.91e-06 | 9.465142e+06 | 1.60e-07 | 2.56e-04 | 11284 | 298.78 |
| 860/2000 | auto | 1e-2 | 20.0 | 347 | 9.462714e+06 | 1.13e-06 | 9.464845e+06 | 1.44e-07 | 2.25e-04 | 13423 | 351.28 |
| 860/2000 | auto | 1e-2 | 30.0 | 400 | 9.462765e+06 | 8.63e-07 | 9.474848e+06 | 7.96e-07 | 1.28e-03 | 14290 | 377.80 |

TABLE 5. Results with `BlockIP` for Sprintlink network with a library of 5000 videos. This problem has 16335309 variables

We also include Fig. 4 to show how the performance of BlockIP decreases while it approaches to the optimum solution because the spectral radius of the inverse of S (20) is 1 in those cases (See Fig. 4a)).




(A) Spectrum ratio vs BlockIP iterations (B) PCG iterations vs BlockIP iterations

FIG. 4. BlockIP performance along iterations in Sprintlink network with a library of 5000 videos. The results are from an instance with average disk size of 260GB and link's bandwidth of 2000 Mbps

The results obtained with CPLEX are shown in Table 6. CPLEX is faster than BlockIP for the VHO problem. Nevertheless, CPLEX was unable to find feasible primal solutions for the challenging combination of big disks/small link capacities.

| Type | Iter | Primal Objective | | Dual Objective | | Opt. | Time |
|---|---|---|---|---|---|---|---|
| | | Value | Feas Gap | Value | Feas Gap | Gap | (min) |
| 260/2000 | 346 | 3.61158e+07 | 3.2e-04 | 3.61133e+07 | 3.2e-04 | 6.8e-05 | 71.03 |
| 860/425* | 264 | 9.85287e+06 | 5.8e-01 | 9.65577e+06 | 1.9e-06 | 2.0e-02 | 65.72 |
| 860/2200 | 200 | 9.46269e+06 | 3.8e-06 | 9.46269e+06 | 4.4e-07 | 1.1e-08 | 41.73 |

TABLE 6. Results with CPLEX for Sprintlink network with a library of 5000 videos. Non-optimal solutions are marked with *

For the small instance in the Sprintlink network we can conclude that BlockIP finds optimal and feasible solutions in very tight instances (i.e., problems with a small feasible region) while CPLEX can not perform well in one of them. When VHO instances become more feasible, second-order direction should be considered an strong option to solve the problem. However, in the same more feasible instances, CPLEX clearly outperforms BlockIP in terms of processing time and optimality gap.

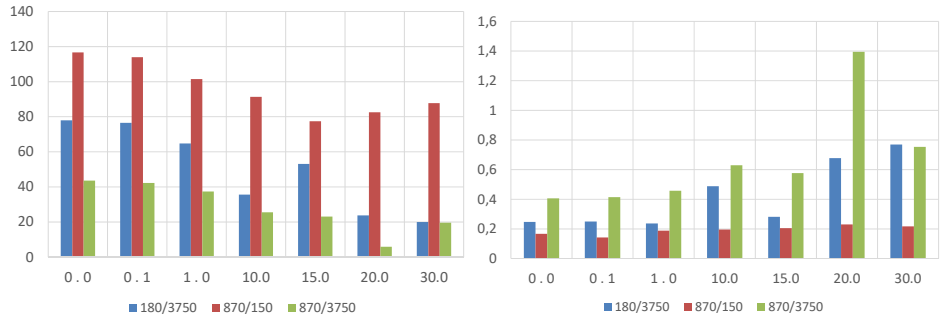
**2.3. Tiscali.** The results shows that Tiscali network is the most difficult and large topology among the three tested in this project. The smallest problem for Tiscali, with 5000 videos reach to a total number of 36015393 variables. The details are in Table 7.

| Parameter | Value |
|---|---|
| Block Variables | 7203 |
| Block Constraints | 4802 |
| Linking constraints | 393 |
| Total Variables | 36015393 |
| Total constraints | 24010393 |

TABLE 7. Descriptive information of smallest instance in Tiscali network with a library of 5000 videos

Table 8 shows the results of testing options of `BlockIP` options in the 5k instance of Tiscali network. In Tiscali network, unexpectedly a relaxed infeasibility gap (i.e., 1e-2) increases the solution time but it does not degrade the optimality gap. Hence, we test the other options combinations of `BlockIP` that uses automatic direction with an infeasibility of 1e-7. Regarding the option second order direction, we test it with a infeasibility gap of 1e-2. Results show that the intensive computation of this option is worthing only for the more feasible instance of this problem. This behavior was also observed in Sprintlink network. For the instance with large disk and small link bandwidth, the optimality gap is worse than the obtained with automatic direction. Also here, as happened in the two previous networks, the second-order direction with a more strict infeasibility gap of 1e-7 only increases the solution time as it can be seen for the instance with average disks of 180GB and links of 3750 Mbps.

A quadratic regularization factor with automatic direction improves considerably the optimality gap and/or solution time. For the small disks and large links (i.e., 180GB and 3750 Mbps) Q=30, the largest one the tested values, gets an optimality gap in the order of 1e-5 with smallest number of PCG iterations per `BlockIP` major iteration (See Fig. 5a), the highest average number of `BlockIP` iterations per minute (Fig. 5b) and in the shortest time. Regarding the scenario with "big" disks and small links capacities (870 GB and 150 Mbps), only factor Q=1.0 leads to a 6% optimal solution.




(A) PCG iterations per `BlockIP` iteration vs Quadratic regularization factor

(B) `BlockIP` iterations per minute vs Quadratic regularization factor

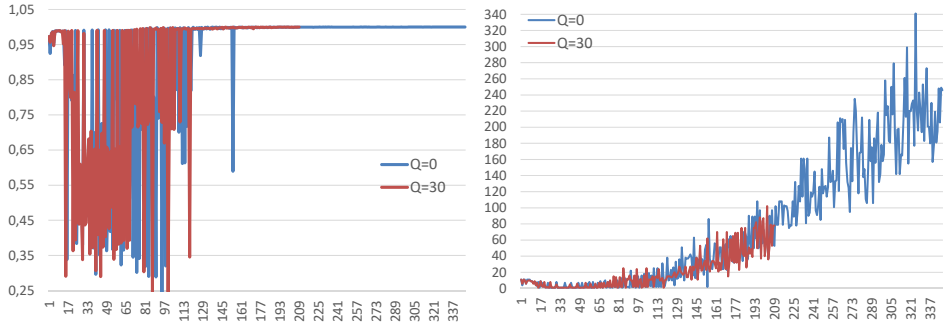
FIG. 5. `BlockIP` performance for different values of quadratic regularization in Tiscali network with a library of 5000 videos

| Type | Option | | BlockIP | Primal Objective | | Dual Objective | | Opt. | C. Grad. | Time |
|---|---|---|---|---|---|---|---|---|---|---|
| | Dir. | Feas. | Reg. | Iterat. | Value | Feas Gap | Value | Feas Gap | Gap | iterat. | (min) |
| 180/3750 | auto | 1e-7 | 0.0 | 347 | 3.286729e+07 | 8.00e-05 | 3.264088e+07 | 2.96e-05 | 6.89e-03 | 27063 | 1404.20 |
| 180/3750 | auto | 1e-2 | 0.0 | 347 | 3.286729e+07 | 8.00e-05 | 3.264088e+07 | 2.96e-05 | 6.89e-03 | 27063 | 1602.98 |
| 180/3750 | S.O. | 1e-7 | 0.0 | 257 | 3.290480e+07 | 1.26e-07 | 3.264874e+07 | 8.73e-10 | 7.78e-03 | 46361 | 2383.04 |
| 180/3750 | S.O. | 1e-2 | 0.0 | 257 | 3.290480e+07 | 1.26e-07 | 3.264874e+07 | 8.73e-10 | 7.78e-03 | 46361 | 2280.45 |
| 180/3750 | auto | 1e-7 | 0.1 | 344 | 3.287040e+07 | 8.37e-05 | 3.264313e+07 | 2.98e-05 | 6.91e-03 | 26341 | 1379.39 |
| 180/3750 | auto | 1e-7 | 1.0 | 329 | 3.286754e+07 | 7.98e-05 | 3.266426e+07 | 2.77e-05 | 6.19e-03 | 21303 | 1392.46 |
| 180/3750 | auto | 1e-07 | 10.0 | 276 | 3.288487e+07 | 6.48e-05 | 3.288208e+07 | 4.86e-05 | 8.50e-05 | 9825 | 565.65 |
| 180/3750 | auto | 1e-7 | 15.0 | 361 | 3.276618e+07 | 2.15e-06 | 3.277439e+07 | 9.82e-07 | 2.50e-04 | 19153 | 1283.98 |
| 180/3750 | auto | 1e-7 | 20.0 | 227 | 3.330268e+07 | 6.17e-04 | 3.330358e+07 | 1.64e-04 | 2.70e-05 | 5394 | 334.86 |
| 180/3750 | auto | 1e-7 | 30.0 | 208 | 3.395353e+07 | 2.11e-03 | 3.395216e+07 | 3.40e-04 | 4.03e-05 | 4154 | 270.60 |
| 870/150 | auto | 1e-7 | 0.0 | 289 | 6.884542e+06 | 3.23e-02 | 6.029003e+06 | 7.15e-06 | 1.24e-01 | 33702 | 1738.94 |
| 870/150 | auto | 1e-2 | 0.0 | 289 | 6.884542e+06 | 3.23e-02 | 6.029003e+06 | 7.15e-06 | 1.24e-01 | 33702 | 1945.87 |
| 870/150 | S.O. | 1e-2 | 0.0 | 165 | 6.787085e+06 | 3.41e-06 | 5.279135e+06 | 6.21e-06 | 2.22e-01 | 88150 | 4675.00 |
| 870/150 | auto | 1e-7 | 0.1 | 291 | 6.892304e+06 | 3.30e-02 | 6.072066e+06 | 7.11e-06 | 1.19e-01 | 33151 | 2053.68 |
| 870/150 | auto | 1e-7 | 1.0 | 284 | 6.862573e+06 | 3.12e-02 | 6.426435e+06 | 1.40e-05 | 6.36e-02 | 28830 | 1511.57 |
| 870/150 | auto | 1e-7 | 10.0 | 400 | 6.368229e+06 | 3.34e-03 | 7.169245e+06 | 3.51e-05 | 1.26e-01 | 36554 | 2050.80 |
| 870/150 | auto | 1e-7 | 15.0 | 400 | 6.079956e+06 | 6.74e-03 | 7.947198e+06 | 7.45e-05 | 3.07e-01 | 30978 | 1950.62 |
| 870/150 | auto | 1e-7 | 20.0 | 400 | 6.526174e+06 | 7.77e-03 | 9.065555e+06 | 1.05e-04 | 3.89e-01 | 33001 | 1733.30 |
| 870/150 | auto | 1e-7 | 30.0 | 400 | 6.744848e+06 | 1.16e-02 | 1.223570e+07 | 2.32e-04 | 8.14e-01 | 35107 | 1846.42 |
| 870/3750 | auto | 1e-7 | 0.0 | 358 | 5.859181e+06 | 2.45e-06 | 5.858959e+06 | 7.47e-09 | 3.79e-05 | 15602 | 883.21 |
| 870/3750 | auto | 1e-2 | 0.0 | 358 | 5.859181e+06 | 2.45e-06 | 5.858959e+06 | 7.47e-09 | 3.79e-05 | 15602 | 890.64 |
| 870/3750 | S.O. | 1e-2 | 0.0 | 145 | 5.859295e+06 | 2.74e-06 | 5.858931e+06 | 7.38e-17 | 6.22e-05 | 6614 | 377.79 |
| 870/3750 | auto | 1e-7 | 0.1 | 357 | 5.859300e+06 | 2.80e-06 | 5.858873e+06 | 2.87e-08 | 7.29e-05 | 15093 | 862.63 |
| 870/3750 | auto | 1e-7 | 1.0 | 350 | 5.859179e+06 | 2.2e-06 | 5.859497e+06 | 2.05e-08 | 5.42e-05 | 13090 | 766.18 |
| 870/3750 | auto | 1e-7 | 10.0 | 400 | 5.890367e+06 | 2.07e-05 | 6.436744e+06 | 2.65e-05 | 9.28e-02 | 10231 | 635.19 |
| 870/3750 | auto | 1e-7 | 15.0 | 400 | 5.894077e+06 | 9.12e-05 | 7.273558e+06 | 6.40e-05 | 2.34e-01 | 9248 | 694.94 |
| 870/3750 | auto | 1e-7 | 20.0 | 142 | 1.729022e+07 | 4.83e-02 | 1.729068e+07 | 1.21e-03 | 2.62e-05 | 840 | 101.84 |
| 870/3750 | auto | 1e-7 | 30.0 | 400 | 5.980893e+06 | 1.07e-04 | 9.494350e+06 | 1.54e-04 | 5.87e-01 | 7842 | 530.64 |

TABLE 8. Results with `BlockIP` for Tiscali network with a library of 5000 videos. This problem has 36015393 variables

The most feasible instance tested (i.e., big disk and high bandwidth capacities) is the fastest to be solved. In this case Q=1.0 provides the best results in terms of optimality gap (5e-5) and in a competitive time. A solution with a shorter gap was found with Q=20, however, we do not consider it because the value of the objective function is much higher than the obtained with other regularization factors. This atypical result require further analysis. As with the Sprintlink network, `BlockIP` using second-order direction without regularization factor finds an optimal solution much faster than using automatic direction and quadratic regularization in this instances.

We also include Fig. 4 to show the performance of `BlockIP` decreases in terms of PCG iterations while it approaches to the optimum solution because the spectral radius of the inverse of S (20) is 1 in those cases (See Fig. 4a)). Hence, if the optimality gap is reduced then the solution time could improve significantly.




(A) Spectrum ratio vs `BlockIP` iterations (B) PCG iterations vs `BlockIP` iterations

FIG. 6. `BlockIP` performance along iterations in Tiscali network with a library of 5000 videos. The results are from an instance with average disk size of 260GB and link's bandwidth of 2000 Mbps

The results obtained with CPLEX are shown in Table 9. CPLEX cannot reach optimal, feasible primal solutions for both challenge combinations (i.e., small disk /large link capacities and viceversa). However, CPLEX is faster than `BlockIP` for the VHO problem in the third instance (the most feasible).

| Type | Iter | Primal Objective | | Dual Objective | | Opt. | Time |
|---|---|---|---|---|---|---|---|
| | | Value | Feas Gap | Value | Feas Gap | Gap | (min) |
| 180/3750* | 363 | 3.29959e+07 | 2.24e-1 | 3.27562e+7 | 1.95e-6 | 7.2e-3 | 177.1 |
| 870/150* | 260 | 6.75114e+06 | 2.23 | 6.04147e+06 | 1.49e-6 | 1.05e-1 | 126.53 |
| 870/3750 | 161 | 5.85910e+06 | 1.70e-3 | 5.85910e+06 | 2.13e-6 | 9.55e-7 | 83.2 |

TABLE 9. Results with CPLEX for Tiscali network with a library of 5000 videos. * means not optimal/feasible solutions

Our results show that `BlockIP` finds optimal and feasible solutions in both tight instances, even when its performance degrade quickly in the last iterations. On the

other hand, CPLEX outperforms BlockIP in both processing time and optimality gap, only in the instance with big disks and large links' bandwidth. It is important that, as happened in Sprintlink network, BlockIP with second-order direction it is a strong option to solve the problem with the third instance because it reach the optimum faster than with automatic direction.

# 3. Results for big instances

In this section, we test the best BlockIP settings in the most feasible instances of the three networks with video libraries from 10000 to 200000 videos (blocks in the problem). Table 10 summarizes the options that we will try in the big disk, large link capacities instances.

| **Network name** | **BlockIP options** | | |
|---|---|---|---|
| | **Type of Direction** | **Feasibility Gap** | **Quadratic Regularization factor** |
| Ebone | auto | 1e-7 | 1.0 |
| Sprintlink | auto | 1e-2 | 1.0 |
| | second-order | 1e-2 | 0.0 |
| Tiscali | auto | 1e-7 | 1.0 |
| | second-order | 1e-2 | 0.0 |

TABLE 10. Best BlockIP settings for most feasible instances of VHO problem with a library of 5000 videos.

**3.1. Ebone.** For this network we present results considering the best setting (quadratic factor Q=1.0, feasibility gap of FG=1e-7) and with default setting (Q=0, FG=1e-7). We aim to get a better idea on how quadratic regularization help in these big instances.

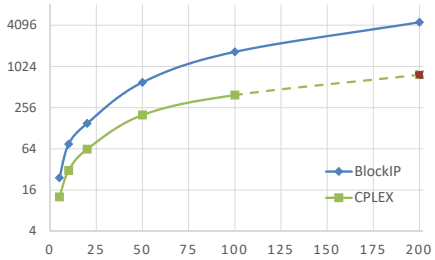
Table 11 reports the size of the video library, the total number of variables and constraints, the options used in BlockIP, the number of iterations, the value of the objective function, the total number PCG iterations and the solution time. The same table, also includes the corresponding values, if applicable, for the solutions obtained by CPLEX solver.

Results show that as the library size doubles its size, BlockIP , in average, triples its solution time. On the contrary, CPLEX increases its solution time in the same proportion that the video library does. This fact can be seen in Fig. 7a.

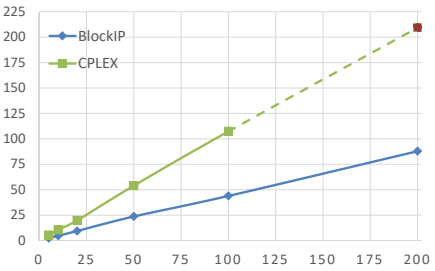
Fig. 7b shows the low consumption of RAM memory of BlockIP compared to CPLEX. In fact, BlockIP needed almost the half of memory than CPLEX for the VHO problems that have solved. This careful use of memory of BlockIP allowed to solve a very huge instance in Ebone network (200k videos, last result in Table 11) while CPLEX ran out of memory in the process.

| Problem Info | | | Option | | | BlockIP | Primal Objective | | Opt. | C. Grad. | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N° blocks | Variables | Constraints | Dir. | Feas. | Reg. | Iterat. | Value | Feas Gap | Gap | iterat. | (min) |
| 5k | 7935175 | 5290175 | auto | 1e-7 | 0.0 | 154 | 9.689621e+06 | 6.77e-05 | 1.79e-04 | 1613 | 28.84 |
| | | | auto | 1e-7 | 1.0 | 145 | 9.688574e+06 | 5.49e-05 | 8.16e-05 | 1257 | 24.16 |
| | | | | CPLEX | | 157 | 9.6879e+06 | 2.98e-05 | 1.55e-07 | N/A | 12.71 |
| 10k | 15870175 | 10580175 | auto | 1e-7 | 0.0 | 182 | 1.919569e+07 | 1.85e-05 | 1.82e-04 | 2351 | 78.51 |
| | | | auto | 1e-7 | 1.0 | 168 | 1.919669e+07 | 1.98e-05 | 2.21e-04 | 1717 | 75.17 |
| | | | | CPLEX | | 178 | 1.919362e+07 | 4.47e-03 | 3.6e-06 | N/A | 30.96 |
| 20k | 31740175 | 21160175 | auto | 1e-7 | 0.0 | 200 | 3.815495e+07 | 1.98e-05 | 1.08e-03 | 2179 | 151.24 |
| | | | auto | 1e-7 | 1.0 | 200 | 3.814375e+07 | 1.68e-05 | 1.33e-04 | 2151 | 150.35 |
| | | | | CPLEX | | 197 | 3.8140117e+07 | 1.66e-02 | 1.2e-05 | N/A | 63.2 |
| 50k | 79350175 | 52900175 | auto | 1e-7 | 0.0 | 256 | 9.352068e+07 | 4.14e-05 | 3.30e-03 | 3985 | 600.55 |
| | | | auto | 1e-7 | 1.0 | 248 | 9.351105e+07 | 3.60e-05 | 1.47e-03 | 3110 | 599.30 |
| | | | | CPLEX **non-optimal** | | 242 | 9.3508341e+07 | 1.57e+00 | 1.2e-03 | N/A | 199.43 |
| 100k | 158700175 | 105800175 | auto | 1e-7 | 0.0 | 339 | 1.914746e+08 | 2.44e-06 | 4.23e-04 | 6100 | 1811.05 |
| | | | auto | 1e-7 | 1.0 | 323 | 1.915064e+08 | 3.99e-06 | 9.51e-05 | 5590 | 1669.76 |
| | | | | CPLEX | | 218 | 1.9146171e+08 | 4.77e-02 | 2.0e-05 | N/A | 389.9 |
| 200k | 317400175 | 21160175 | auto | 1e-7 | 0.0 | 399 | 3.804385e+08 | 3.24e-05 | 2.97e-03 | 7407 | 4521.5 |

TABLE 11. Results with BlockIP for Ebone network with video libraries from 5000 to 200000 videos (diagonal blocks).




(A) Solution Time (log scale) vs Library size (thousands of videos)




(B) RAM Memory (GB) vs Library size (thousands of videos)

FIG. 7. Resource use comparison between `BlockIP` and CPLEX as function of the number of videos for Ebone network. Results are from the most feasible instance (big disks and high link capacity). Dashed lines and red points were drawn from estimations

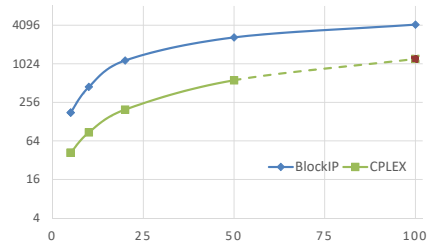
It is important to notice that, as happen in the small instances, sometimes CPLEX cannot find a feasible, optimal solution. This is the case of Ebone instance with 50000 videos. Although `BlockIP` is slower than CPLEX, `BlockIP` solves this instance satisfactorily.

**3.2. Sprintlink.** For this network we present results considering the best setting (quadratic factor Q=1.0, feasibility gap of FG=1e-2) and with default setting (Q=0, FG=1e-2). In addition, we have tested `BlockIP` with the second-order direction for the instance with 10000 videos.

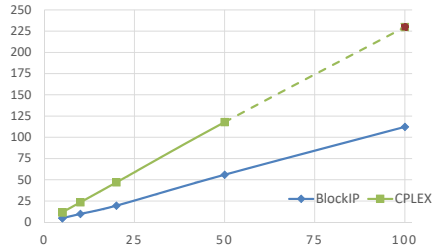
Table 12 reports the results for Sprintlink with the same format used for Ebone network. Notice that for a video library of 10000 videos, second-order direction does not improves solution time of automatic direction. Fig. 8a and Fig. 8b shows the same behavior observed in Ebone. As the video library doubles its size, the solution time of `BlockIP` , increases at 2.5 times. On the contrary, CPLEX increases its solution time at the same rate that the video library does. Regarding memory use, `BlockIP` clearly outperforms CPLEX. CPLEX needs almost twice the memory space than `BlockIP`. Due to the fact that `BlockIP` needs less memory than CPLEX, it solved an instance with 100000 videos satisfactorily –this instance has more 326 millions of variables– while CPLEX got out of memory. The last instance that CPLEX tried to solved was the one with a video library of 50000 videos. In this case, CPLEX did not reach an feasible primal solution and it ended with a feasibility gap of 8.54. In the same instance `BlockIP` reached a remarkable feasibility gap of 1e-4.

| Problem Info | | | Option | | | BlockIP | Primal Objective | | Opt. | C. Grad. | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N° blocks | Variables | Constraints | Dir. | Feas. | Reg. | Iterat. | Value | Feas Gap | Gap | iterat. | (min) |
| 5k | 16335309 | 10890309 | auto | 1e-2 | 0.0 | 238 | 9.465295e+06 | 3.86e-06 | 5.11e-04 | 7355 | 199.19 |
| | | | S.O | 1e-2 | 0.0 | 117 | 9.466474e+06 | 5.79e-07 | 7.08e-04 | 6925 | 177.27 |
| | | | auto | 1e-2 | 1.0 | 239 | 9.464049e+06 | 3.03e-06 | 1.13e-04 | 7804 | 209.73 |
| | | | | CPLEX | | 200 | 9.46269e+06 | 3.8e-06 | 1.1e-08 | N/A | 41.73 |
| 10k | 32670309 | 21780309 | auto | 1e-2 | 0.0 | 289 | 1.876374e+07 | 8.48e-06 | 1.53e-03 | 8854 | 529.49 |
| | | | S.O | 1e-2 | 0.0 | 141 | 1.875593e+07 | 7.80e-07 | 4.47e-04 | 9533 | 484.68 |
| | | | auto | 1e-2 | 1.0 | 279 | 1.875968e+07 | 5.32e-06 | 2.83e-04 | 7634 | 445.61 |
| | | | | CPLEX | | 196 | 1.8750325e+07 | 3.00e-03 | 7.4e-07 | N/A | 86.91 |
| 20k | 65340309 | 43560309 | auto | 1e-2 | 0.0 | 339 | 3.755364e+07 | 9.30e-06 | 1.68e-03 | 10847 | 1176.16 |
| | | | auto | 1e-2 | 1.0 | 330 | 3.752920e+07 | 2.14e-06 | 9.66e-05 | 10981 | 1153.84 |
| | | | | CPLEX | | 211 | 3.7519346e+07 | 1.36e-02 | 1.2e-07 | N/A | 197.43 |
| 50k | 163350309 | 108900309 | auto | 1e-2 | 0.0 | 400 | 9.480438e+07 | 2.98e-04 | 2.71e-02 | 8269 | 2434.41 |
| | | | auto | 1e-2 | 1.0 | 400 | 9.419190e+07 | 4.70e-05 | 1.46e-03 | 9126 | 2642.27 |
| | | | | CPLEX | **non-optimal** | 272 | 9.3956454e+07 | 8.54e+00 | 1.3e-03 | N/A | 566.65 |
| 100k | 326700309 | 217800309 | auto | 1e-2 | 1.0 | 400 | 1.884211e+08 | 2.44e-06 | 7.92e-03 | 6279 | 4191.52 |

TABLE 12. Results with BlockIP for Sprintlink network with video libraries from 5000 to 100000 videos (diagonal blocks).




(A) Solution Time (log scale) vs Library size (thousands of videos)




(B) RAM Memory (GB) vs Library size (thousands of videos)

FIG. 8. Resource use comparison between `BlockIP` and CPLEX as function of the number of videos for Sprintlink network. Results are from the most feasible instance (big disks and high link capacity). Dashed lines and red points were drawn from estimations

**3.3. Tiscali.** Tiscali is the most complex and large network that we used in this project. Solve large instances of this network takes too much time. Therefore, we only analyzed the instance of 10000 videos and compared its results against the smallest instance of 5000 videos. For the instance with 10000 videos, we configure a optimality gap of 0.01% in an effort to reduce solving time. Even so, `BlockIP` lasted around 18 hours to solve this instance. The solution time ratio between `BlockIP` and CPLEX is around 5, which is similar to the obtained in the two previous network topologies.

Finally, it is worth noting that the solution time difference between `BlockIP` with second-order direction and automatic direction is not as big in the 10000 videos instance as in the smallest one. We recall that the same behavior was also present in Sprintlink network. This empirical result could indicate that in this particular problem, second-order direction is more valuable in small-size instances.

| Problem Info | | | Option | | | BlockIP | Primal Objective | | Opt. | C. Grad. | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N° blocks | Variables | Constraints | Dir. | Feas. | Reg. | Iterat. | Value | Feas Gap | Gap | iterat. | (min) |
| | | | auto | 1e-7 | 1.0 | 350 | 5.859179e+06 | 2.2e-06 | 5.42e-05 | 13090 | 766.18 |
| 5k | 36015393 | 24010393 | S.O. | 1e-2 | 0.0 | 145 | 5.859295e+06 | 2.74e-06 | 6.22e-05 | 6614 | 377.79 |
| | | | | CPLEX | | 161 | 5.85910e+06 | 1.70e-3 | 9.55e-7 | N/A | 83.2 |
| | | | auto | 1e-7 | 1.0 | 373 | 1.221095e+07 | 1.18e-04 | 9.54e-03 | 7951 | 1093.68 |
| 10k | 72030393 | 48020393 | S.O | 1e-2 | 1.0 | 153 | 1.202256e+07 | 2.09e-07 | 9.20e-03 | 7512 | 1053.17 |
| | | | | CPLEX | | 182 | 1.1953577e+07 | 4.65e-03 | 3.17e-06 | N/A | 188.31 |

TABLE 13. Results with BlockIP for Ebone network with video libraries from 5000 and 10000 videos (diagonal blocks).

# Chapter 4
# Conclusions

In this thesis, we have solved large instances of a problem of optimal placement of videos for the VoD service by using the specialized `BlockIP`, which was designed for problems with block diagonal structure. Results show that, given a fixed amount of memory, `BlockIP` can deal with problems that contain twice the number of variables compared to the problems that state-of-art CPLEX can solve.

`BlockIP` was able to solve instances with more than 300 millions of variables with optimality and feasibility gaps around $10^{-3}$. However, this memory efficiency has a downside in the time needed by `BlockIP` to find an optimal solutions, which is two or three times slower than CPLEX. In addition, we corroborate the propositions in [**19**] about the degradation of `BlockIP` performance when it approaches to the optimum and how a quadratic regularization factor can speed up the solution time. Other interesting result is that `BlockIP` found feasible, optimal solutions in some complicate instances when CPLEX cannot reach feasibility.

PCG performance is completely related with structure of linking constraints. In the video location problem of this thesis, the linking constraints depends on: size of disks, links capacities and network topology. The latter is particular important because solution time varies considerably from one topology to another although the instances have similar number of variables.

The VoD service considered in this work, has huge video libraries, however, videos could be grouped if they have similar request pattern and/or size. This grouping strategy could make the problem easier to solve and should be tested.

In this thesis, we used a Carrier CDN with very challenging topologies. On the other hand, overlay CDNs usually have much simpler topologies by connecting sites through paths from one to three hops. These straightforward topologies could potentially be favorable for the PCG computations. Finally, modifications in the problem model as the use of additional costs to extend links or disk capacities are relevant from a CDN's management point of view and constitute future work.

# References


[1] A. Altman and J. Gondzio, *Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization*, Optimization Methods & Software 11 (1999), pp. 275–302.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney and D. Sorensen, *LAPACK Users' Guide, Third edition*, SIAM, Philadelphia, PA, 1999.

[3] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, *Optimal Content Placement for a Large-Scale VoD System*, IEEE/ACM Transactions on Networking 24 (2016), no. 4, pp. 2114–2127.

[4] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. K. Ramakrishnan, *Optimal Content Placement for a Large-scale VoD System*, Proceedings of the 6th International Conference (New York, NY, USA), Co-NEXT '10, ACM, 2010, pp. 4:1–4:12.

[5] S. Bellavia, J. Gondzio and B. Morini, *A matrix-free preconditioner for sparse symmetric positive definite systems and least-squares problems*, SIAM Journal on Scientific Computing 35 (2013), pp. A192-A211.

[6] M. Benzi, *Splittings of symmetric matrices and a question of Ortega*, Linear Algebra and its Applications 429 (2008), pp. 2340-2343.

[7] L. Bergamaschi, J. Gondzio and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Computational Optimization and Applications 28 (2004), pp. 149–171.

[8] D. Bienstock, *Potential Function Methods for Approximately Solving Linear Programming Problems. Theory and Practice*, Kluwer: Boston, 2002.

[9] D. Bienstock and O. Raskina, Asymptotic analysis of the flow deviation method for the maximum concurrent flow problem, *Mathematical Programming* 91 (2002), pp. 479–492.

[10] R. E. Bixby, *Solving real-world linear programs: a decade and more of progress*, Operations Research, 50 (2002), pp. 3–15.

[11] S. Bocanegra, J. Castro and A.R.L. Oliveira, *Improving an interior-point approach for large block-angular problems by hybrid preconditioners*, European Journal of Operational Research 231 (2013), pp. 263–273.

[12] J. Castro, *A specialized interior-point algorithm for multicommodity network flows*, SIAM Journal on Optimization 10 (2000), pp. 852–877.

[13] J. Castro, *Solving difficult multicommodity problems through a specialized interior-point algorithm*, Annals of Operations Research, 124 (2003), pp. 35–48.

[14] J. Castro, *An interior-point approach for primal block-angular problems*, Computational Optimization and Applications 36 (2007), pp. 195–219.

[15] J. Castro, *Recent advances in optimization techniques for statistical tabular data protection*, European Journal of Operational Research 216 (2012), pp. 257-269.

[16] J. Castro and J. Cuesta, *Quadratic regularizations in an interior-point method for primal block-angular problems*, Mathematical Programming 130 (2011), pp. 415–445.

[17] J. Castro, *Interior Point Methods [Lecture Notes]*, (2014), pp. 1–36.

[18] J. Castro, *Primal-dual path-following methods [Lecture Notes]*, (2014), pp. 1–90.

[19] J. Castro, *Interior-point solver for convex separable block-angular problems*, Optimization Methods and Software 31 (2016), no. 1, pp. 88–109.

[20] M. Cha, H. Kwak, P. Rodriguez, Y-Y. Ahn, and S. Moon, *I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system*, Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement (New York, NY, USA), IMC '07, ACM, 2007, pp. 1–14.

[21] J. J. Cochran, *An Introduction to Linear Programming*, Wiley Encyclopedia of Operations Research and Management Science, Major Reference Works, 2010.

[22] M. Colombo, A. Grothey, J. Hogg, K. Woodsend and J. Gondzio, *A structure-conveying modelling language for mathematical and stochastic programming*, Mathematical Programming Computation, 1 (2009), pp. 223–247.

[23] A. V. Fiacco and G. P. McCormick, *Nonlinear programming: Sequential unconstrained minimization techniques*, 2nd Ed. SIAM, 1990.

[24] K. Fountoulakis and J. Gondzio, A second-order method for strongly convex l1-regularization problems, Technical Report ERGO-14-005, School of Mathematics, The University of Edinburgh, 2014.

[25] K. Fountoulakis, J. Gondzio and P. Zhlobich, *Matrix-free interior point method for compressed sensing problems*, Mathematical Programming Computation 6 (2014), pp. 1–31.

[26] R. Fourer, D.M. Gay and D.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming, Second edition*, Thomson Brooks/Cole, Toronto, Canada, 2003.

[27] A. Frangioni and C. Gentile, *New preconditioners for KKT systems of network flow problems*, SIAM Journal on Optimization 14 (2004), pp. 894–913.

[28] J. Gondzio and A. Grothey, *Direct Solution of Linear Systems of Size* $10^9$ *Arising in Optimization with Interior Point Methods.* In R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Waśniewski (Eds.), Parallel Processing and Applied Mathematics Springer Berlin Heidelberg, 2006, pp. 513–525.

[29] J. Gondzio, *Matrix-free interior point method*, Computational Optimization and Applications 51 (2012), pp. 457–480.

[30] J. Gondzio, *Convergence analysis of an inexact feasible interior point method for convex quadratic programming*, SIAM Journal on Optimization 23 (2013), pp. 1510–1527.

[31] J. Gondzio and R. Sarkissian, *Parallel Interior Point Solver for Structured Linear Programs*, Mathematical Programming 96 (2003) pp. 561–584.

[32] C. Keller, N.I.M. Gould and A.J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM Journal on Matrix Analysis and Applications 21 (2000), pp. 1300–1317.

[33] G.H. Golub and C.F. Van Loan, *Matrix Computations, Third edition*, The Johns Hopkins University Press, Baltimore, MA, 1996.

[34] X. Jiménez, *A modelling and optimization environment for large-scale block-angular problems* (in Catalan), MSc thesis, Barcelona School of Informatics, Universitat Politècnica de Catalunya, 2012.

[35] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research of the National Bureau of Standards 45 (1950), pp. 225–280.

[36] David G Luenberger and Yinyu Ye, *The Simplex Method BT - Linear and Nonlinear Programming*, Springer International Publishing, Cham, 2016, pp. 33–82.

[37] S. Mehrotra, *On the implementation of a primal–dual interior point method*, SIAM Journal on Optimization 2 (1992), pp. 575–601.

[38] G. Meurant, *The Lanczos and Conjugate Gradient Algorithms: from Theory to Finite Precision Computations*, SIAM, Philadelphia, PA, 1006.

[39] P. Munari and J. Gondzio, *Using the primal-dual interior point algorithm within the branch-price-and-cut method*, Computers & Operations Research 40 (2013), pp. 2026–2036.

[40] P. Mukaddim, *Cloud-Based Content Delivery and Streaming*, ch. 1, pp. 1–31, Wiley-Blackwell, 2014.

[41] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer, Boston, MA, 2004.

[42] E. Ng and B.W. Peyton, *Block sparse Cholesky algorithms on advanced uniprocessor computers*, SIAM Journal on Scientific Computing 14 (1993), pp. 1034–1056.

[43] A.R.L. Oliveira and D.C. Sorensen, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, Linear Algebra and its Applications 394 (2005), pp. 1–24.

[44] J.M. Ortega, *Introduction to Parallel and Vector Solutions of Linear Systems*, Plenum Press, New York, NY, 1988.

[45] M.G.C. Resende and G. Veiga, *An implementation of the dual affine scaling algorithm for minimum-cost flow on bipartite uncapacitated networks* SIAM Journal on Optimization, 3 (1993), pp. 516–537.

[46] C. Roos, T. Terláky and J.-P. Vial, *Interior Point methods for linear optimization, 2nd Ed.*, Springer, Boston, MA, 2006.

[47] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, *Measuring ISP topologies with Rocketfuel*, IEEE/ACM Transactions on Networking 12 (2004), no. 1, pp. 2–16.

[48] B. Tolga and E. Ozgur, *CDN Modeling*, Wiley-Blackwell, 2014, , ch. 9, pp. 179–202.

[49] R.J. Vanderbei, *Linear Programming: Foundations and Extensions*, Kluwer, Boston, MA, 1996.

[50] S.J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, PA, 1996.