Second Order Conic Optimization Formulations for Service System Design Problems with Congestion

Julio C. Góez[a], Miguel F. Anjos[b]

[a] *Department of Business and Management Science, NHH Norwegian School of Economics, Bergen, Norway*
[b] *GERAD & Polytechnique Montreal, Montreal, QC, Canada*

Abstract

We consider the service system design problem with congestion. This problem arises in a number of practical applications in the literature, and is concerned with determining the location of service facilities, the capacity level of each facility, and the assignment of customers to the facilities so that the customer demands are satisfied at total minimum cost. We present seven mixed integer second order cone optimization formulations for this problem, and compare their computational performances between them, and with the performance of other exact methods in the literature. Our results show that the conic formulations are competitive and may outperform the current leading exact methods. One advantage of the conic approach is the possibility of using off-the-shelf state-of-the-art solvers directly. More broadly, this study provides insights about different conic modeling approaches and the significant impact of the choice of approach on the efficiency of the resulting model.

Keywords: Service systems, Congestion, Second order cone optimization, Mixed integer optimization
2010 MSC: 90B18, 90C11

1. Introduction

We consider the service system design problem with congestion. The problem is to decide the location of service facilities, allocate each facility a capacity level, and assign customers to the facilities so that the demand of each customer is satisfied at minimum cost. This problem arises in a number of practical applications in the literature. For example, in the facility location literature it is known as the facility location problem with stochastic customer demand and immobile servers [8, 9, 30, 31]. Another example is the location of emergency service facilities such as medical clinics and preventive health care facilities [32]. It is also used to allocate optimally servers at different cloud locations to improve user request response times [21].

We index the set of customer locations by i, the possible locations for facilities by j, and the set of capacity levels for the facilities by k. We denote by $\lambda_i \geq 0$ the mean demand rate for consumer i, and by $\mu_{jk} \geq 0$ the mean service rate of server of type k at facility j. The goal is to minimize the sum of three costs:

Email addresses: Julio.Goez@nhh.no (Julio C. Góez), miguel-f.anjos@polymtl.ca (Miguel F. Anjos)

the cost p_{jk} of opening and allocating enough capacity to the facilities, the access cost c_{ij} of the customer i to the facility j, and the queuing delay cost per time unit d. For modeling the queuing delay we model each server facility as an independent $M/M/1$ queue. Under this assumption, we have the following mixed-integer nonlinear optimization model [2]:

$$\min \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m}\frac{\sum_{i=1}^{\ell}\lambda_i x_{ij}}{\sum_{k=1}^{n}\mu_{jk}y_{jk} - \sum_{i=1}^{\ell}\lambda_i x_{ij}} + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \tag{1}$$

$$\text{s.t.} \sum_{k=1}^{n}\mu_{jk}y_{jk} - \sum_{i=1}^{\ell}\lambda_i x_{ij} \geq 0 \quad j = 1,\ldots,m \tag{2}$$

$$\sum_{k=1}^{n} y_{jk} \leq 1 \quad j = 1,\ldots,m \tag{3}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad i = 1,\ldots,\ell \tag{4}$$

$$x_{ij}, y_{jk} \in \{0,1\} \quad i = 1,\ldots,\ell; \quad j = 1,\ldots,m; \quad k = 1,\ldots,n \tag{5}$$

Note that if $d < 0$, then the model will reward the queuing delay in the system. In general this goes against the purpose of service system design. Furthermore, if $d < 0$ then the objective function (1) will go to minus infinity, preferring unstable solutions. For these reasons, we assume throughout this paper that $d \geq 0$.

Different solution approaches have been proposed in the literature for this problem. Amiri [2] proposed a heuristic based in Lagrangian relaxation. Elhedhli proposed an equivalent mixed-integer linear optimization (MILO) formulation [12], and designed an exact algorithm based on outer linear approximations [13]. This problem has been also used as a testing problem for several techniques used to solve mixed-integer nonlinear problems, including disjunctive cuts [22], perspective reformulations [16], outer-inner approximation [18], and strong-branching inequalities [23].

In this paper we explore mixed-integer second order cone optimization (MISOCO) as a practical option to solve the service system design problem with congestion. Our contribution is a set of equivalent MISOCO formulations for the problem (1)–(5). This approach seeks to profit from the recent methodological developments, see e.g. [3, 4, 6, 5, 11, 15, 24, 25, 26], and from the power of off-the-shelf solvers capable of solving MISOCO problems [17, 20, 27]. Our computational results show that the MISOCO formulations are competitive and may outperform the leading exact methods in the literature. More broadly, our study also provides insights about different MISOCO modeling approaches and the significant impact of the choice of approach on the efficiency of the resulting model.

This paper is structured as follows. In section 2 we present the previously proposed MILO formulation in [13] and the outer linear approximation approach in [13]. The new MISOCO formulations are derived in Section 3, and Section 4 reports the results of our computational testing to determine the most efficient formulation. Finally, we provide some concluding remarks and thoughts on future research in Section 5.

2. Review of existing exact approaches

In this section we summarize two approaches from the literature that are specialized for problem (1)–(5). Section 2.1 presents the MILO reformulation derived in [12], and Section 2.2 discusses the outer linear approximation proposed in [13].

2.1. MILO Approach

One way to deal with the nonlinearity in (1) is to use the result in [14, 28] that allows to linearize the product of a binary variable and a continuous variable. This is the same approach that is suggested in [12].

We start by introducing new variables s_i for the facilities via the inequalities

$$\frac{\sum_{i=1}^{\ell} \lambda_i x_{ij}}{\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij}} \leq s_j, \quad s_j \geq 0, \quad j = 1, \ldots, m, \tag{6}$$

and observing that

$$\frac{\sum_{i=1}^{\ell} \lambda_i x_{ij}}{\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij}} \leq s_j \iff \sum_{i=1}^{\ell} \lambda_i x_{ij} \leq s_j \left(\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \right), \tag{7}$$

where the equivalence follows from (2).

Let $\bar{s}$ be an upper bound for the value of the s_j variables. Also let $w_{jk} = s_j y_{jk}$ and $z_{ij} = s_j x_{ij}$. Then, because x_{ij} and y_{jk} are binary, equation $w_{jk} = s_j y_{jk}$ can be modeled as follows [14, 28]:

$$w_{jk} \leq y_{jk} \bar{s}, \quad w_{jk} \leq s_j, \quad y_{jk} \bar{s} + s_j - w_{jk} \leq \bar{s}.$$

Similarly, equation $z_{ij} = s_j x_{ij}$ can be modeled as:

$$z_{ij} \leq x_{ij} \bar{s}, \quad z_{ij} \leq s_j, \quad x_{ij} \bar{s} + s_j - z_{ij} \leq \bar{s}.$$

This leads to the following MILO problem:

$$\min \sum_{i=1}^{\ell} \sum_{j=1}^{m} c_{ij} x_{ij} + d \sum_{j=1}^{m} s_j + \sum_{j=1}^{m} \sum_{k=1}^{n} p_{jk} y_{jk} \tag{8}$$

$$\text{s.t.} \sum_{k=1}^{n} \mu_{jk} w_{jk} - \sum_{i=1}^{\ell} \lambda_i z_{ij} \geq \sum_{i=1}^{\ell} \lambda_i x_{ij} \quad j = 1, \ldots, m \tag{9}$$

$$w_{jk} \leq y_{jk} \bar{s},\ w_{jk} \leq s_j,\ y_{jk} \bar{s} + s_j - w_{jk} \leq \bar{s},\ j = 1, \ldots, m,\ k = 1, \ldots, n \tag{10}$$

$$z_{ij} \leq x_{ij} \bar{s},\ z_{ij} \leq s_j,\ x_{ij} \bar{s} + s_j - z_{ij} \leq \bar{s},\ i = 1, \ldots, \ell,\ j = 1, \ldots, m \tag{11}$$

$$s_j, w_{jk}, z_{ij} \geq 0 \quad i = 1, \ldots, \ell,\ j = 1, \ldots, m,\ k = 1, \ldots, n \tag{12}$$

$$(3)\text{–}(5)$$

This reformulation has two main characteristics. First, the number of constraints increases significantly. Second, a good estimation of the upper bound $\bar{s}$ is needed. One option for this bound is that if the problem is feasible, an initial feasible solution can be obtained, and the value of the objective function associated with that solution can serve as $\bar{s}$.

2.2. Elhedhli's Exact Algorithm

The approach in [13] proposes an outer linear approximation to solve (1)–(5). Elhedhli's approach starts with a relaxation of the original nonlinear problem to a MILO problem. This relaxation is then iteratively refined using linear cuts to approximate the nonlinear constraints in the original problem. This refinement continues until the procedure reaches a difference between the upper and lower bounds below a given tolerance $\epsilon > 0$. The purpose of this section is to review the approach in [13].

The approach developed in [13] also introduces the variables s_i but defines them as follows:

$$s_j = \frac{\sum_{i=1}^{\ell} \lambda_i x_{ij}}{\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij}}. \tag{13}$$

This allows to define the following linear relation between s_j and the x_{ij} and y_{jk} variables :

$$\sum_{i=1}^{\ell} \lambda_i x_{ij} = \sum_{k=1}^{n} \mu_{jk} z_{jk}, \quad z_{jk} = \frac{s_j}{1+s_j} y_{jk}.$$

Thus, using (13) we obtain the following equivalent reformulation:

$$\begin{aligned}
\min \ & \sum_{i=1}^{\ell} \sum_{j=1}^{m} c_{ij} x_{ij} + d \sum_{i=j}^{m} s_j + \sum_{j=1}^{m} \sum_{k=1}^{n} p_{jk} y_{jk} \\
\text{s.t.} \ & \sum_{k=1}^{n} \mu_{jk} z_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} = 0 \quad j = 1, \ldots, m \\
& z_{jk} = \frac{s_j}{1+s_j} y_{jk} \quad j = 1, \ldots, m, k = 1, \ldots, n \\
& s_j \geq 0, j = 1, \ldots, m \\
& (3)\text{–}(5)
\end{aligned} \tag{14}$$

An important feature of this reformulation is that it guarantees the stability of the system since $z_{jk} < 1$ for $j = 1, \ldots, m$ and $k = 1, \ldots, n$. This captures the property in the original problem that an unstable system will drive the objective function to $+\infty$.

The nonlinearity of the problem is now found in the second constraint of (14). The basis for the approach in [13] is to exploit the fact that the function $f(s_j) = \frac{s_j}{1+s_j}$ is concave. Elhedhli's approach uses first-order approximations of the function $f(s_j)$ to reformulate the problem. In particular, using the set $\mathcal{H}$ of all $s_j^h \geq 0$ where $h \in \mathcal{H}$, then we have that:

$$\frac{s_j}{1+s_j} \leq \frac{1}{(1+s_j^h)^2} s_j + \left(\frac{s_j^h}{1+s_j^h} \right), \quad \forall h \in \mathcal{H}.$$

Using this result, we obtain the following equivalent reformulation of the problem [13]:

$$
\begin{aligned}
\min\ & \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m} s_j + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \\
\text{s.t.}\ & \sum_{k=1}^{n} \mu_{jk}z_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} = 0 \quad j = 1,\dots,m, \\
& z_{jk} \le y_{jk} \quad j = 1,\dots,m; k = 1,\dots,n, \\
& z_{jk} - \frac{1}{(1+s_j^h)^2} s_j \le \left(\frac{s_j^h}{1+s_j^h}\right)^2 \quad j = 1,\dots,m; k = 1,\dots,n,; h \in \mathcal{H}, \\
& s_j \ge 0 \quad j = 1,\dots,m, \\
& z_{jk} \ge 0, \quad j = 1,\dots,m; k = 1,\dots,n. \\
& (3)\text{–}(5).
\end{aligned}
\tag{15}
$$

Note that $|\mathcal{H}| = \infty$ in (15). For that reason, the approach in [13] proposes to start with a relaxation of (15) in which a finite subset $\hat{\mathcal{H}} \subseteq \mathcal{H}$ is used, and new s_j^h are added to $\hat{\mathcal{H}}$ iteratively such that the relaxation is refined at each iteration of the procedure. This procedure is summarized in Algorithm 1.

Algorithm 1 Elhedhli's approach

1: $UB \leftarrow +\infty$

2: $LB \leftarrow -\infty$

3: $\mathcal{H} \leftarrow$ well-placed initial set of cuts

4: **while do** $\frac{UB-LB}{UB} > \epsilon$

5: Solve problem (15) and let (x^*, y^*, z^*, s^*) be the optimal solution

6: $$LB \leftarrow \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij}^* + d\sum_{j=1}^{m} R_j^* + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk}^*$$

7: $$UB \leftarrow \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij}^* + d\sum_{j=1}^{m} \frac{\sum_{i=1}^{\ell}\lambda_i x_{ij}^*}{\sum_{k=1}^{n}\mu_{jk}y_{jk}^* - \sum_{i=1}^{\ell}\lambda_i x_{ij}^*} + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk}^*$$

8: **if** $\frac{UB-LB}{UB} > \epsilon$ **then**

9: **for** $j \leftarrow 1,\dots,m$ **do**

10: $\mathcal{H} \leftarrow \mathcal{H} \cup \left\{ \frac{\sum_{i=1}^{\ell}\lambda_i x_{ij}^*}{\sum_{k=1}^{n}\mu_{jk}y_{jk}^* - \sum_{i=1}^{\ell}\lambda_i x_{ij}^*} \right\}$

11: **end for**

12: **end if**

13: **end while**

An important remark about this procedure is necessary. Note that the solution of a relaxation of the

problem (15) can be such that

$$\sum_{k=1}^{n} \mu_{j^*k} y^*_{j^*k} - \sum_{i=1}^{\ell} \lambda_i x^*_{ij^*} = 0 \text{ for some } j^*.$$

This makes $s_{j^*} = +\infty$, which implies that the objective function of the original problem for the solution (x^*, y^*) is $+\infty$. In other words, the solution (x^*, y^*) is suboptimal for the original problem. This may happen because the reformulation (15) guarantees stability only when the whole set of s_j values is considered for $j = 1, \ldots, m$.

To illustrate this problem, let us consider a case with four customer locations points D_1, D_2, D_3, and D_5, and two possible server locations, each with the option of having a service rate of 10 or 20. This case is illustrated in Figure 1.

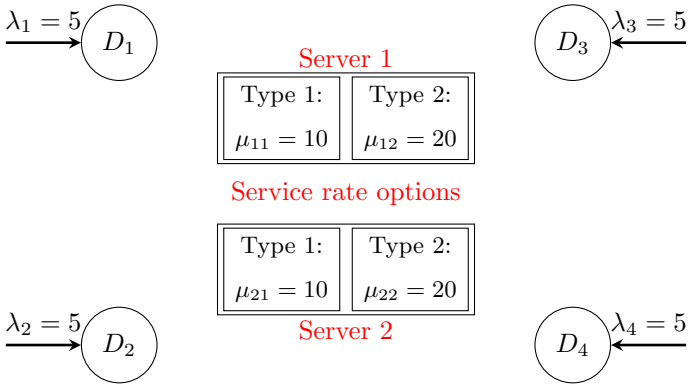

Figure 1: Example of a service system design problem

We further assume that the connection from any demand point to any server has a cost of \$15, the installation of a server with a service rate of 10 costs \$100, and the installation of a server with a service rate of 20 has a cost of \$500. Finally the cost associated with congestion is one dollar for each unit of time spent in the system. The optimization problem associated with this example is:

$$\min \; 15 \sum_{i=1}^{4} \sum_{j=1}^{2} x_{ij} + \sum_{j=1}^{2} \frac{5 \sum_{i=1}^{4} x_{ij}}{10y_{j1} + 20y_{j2} - 5 \sum_{i=1}^{4} x_{ij}} + 100 \sum_{j=1}^{2} y_{j1} + 500 \sum_{j=1}^{2} y_{j2} \tag{16}$$

$$\text{s.t. } 5 \sum_{i=1}^{4} x_{ij} - 10y_{j1} - 20y_{j2} \leq 0, j = 1, 2 \tag{17}$$

$$\sum_{j=1}^{2} x_{ij} = 1, i = 1, \ldots, 4 \tag{18}$$

$$\sum_{k=1}^{2} y_{jk} \leq 1, j = 1, 2 \tag{19}$$

$$x_{ij}, y_{jk} \in \{0, 1\}, i = 1, \ldots, 4, j = 1, 2, k = 1, 2. \tag{20}$$

Algorithm 1 is initialized using a "well-placed initial set of cuts" [13]. The criteria for a "well-placed cut" is crucial for this algorithm. In this case we can take advantage of the size of the problem to add all the

valid inequalities that algorithm 1 would generate. We define this as our criterion for well-placed cuts. In particular, given the size of this problem we can enumerate all the stable solutions for problem (16)–(20). This guarantees that those are all the possible cuts generated with Algorithm 1 because unstable solutions make step 10 fail.

We compute first all the possible values for the s_j that can be obtained running Algorithm 1. This enumeration is necessary because the valid inequalities are characterized by the values of the s_j. Note that the servers with service rate 10 can satisfy the demand of at most one of the demand sources. Also note that the servers with service rate 20 can satisfy the demand of at most three demand sources. Hence, the list of possible configurations for the servers in this problem for a solution to be stable are:

	Server 1	Server 2
Configuration 1	Type 1	Type 2
Configuration 2	Type 2	Type 1
Configuration 3	Type 2	Type 2.

For configurations 1 and 2 we can distribute the demand as follows: assign one demand source to the server with service rate 10, and assign three demand sources to the server with service rate 20. Given that all the demands are equal we can conclude the following:

- For configuration 1 we obtain for Server 1 $s_1^1 = \frac{5}{10-5} = 1$, and for Server 2 $s_2^1 = \frac{15}{20-15} = 3$.
- For configuration 2 we obtain that for Server 1 $s_1^2 = \frac{15}{20-15} = 3$, and for Server 2 $s_2^2 = \frac{5}{10-5} = 1$.

We note that for each of these configurations there are $\binom{4}{3} = 4$ ways of distributing the demand. However, for all the 4 cases in each configuration we will obtain the same values for s_j in the two servers. This means that any of those solutions will generate the same valid inequalities. For that reason, it suffices to consider each value of s_j once.

For configuration 3 we can distribute the demand in the following manner:

- Assign one demand source to Server 1 and three demand sources to Server 2. Using this configuration we obtain for Server 1 $s_1^3 = \frac{5}{20-5} = \frac{1}{3}$, and for Server 2 $s_2^3 = \frac{15}{20-15} = 3$.
- Assign two demand sources to each server. Using this configurations we obtain $s_1^4 = s_2^4 = \frac{10}{20-10} = 1$.
- Assign three demand sources to Server 1 and one demand source to Server 2. Using this configuration we obtain for Server 1 $s_1^5 = \frac{15}{20-15} = 3$, and for Server 2 $s_2^5 = \frac{5}{20-5} = \frac{1}{3}$.

Here the first and third distributions of demand also have 4 different ways to distribute the demand. However, it is again the case that those different ways will lead to the same values of s_j. The same is also true for the second demand distribution on the list. Hence, each of these values of s_j need to be considered only once.

Finally, note that $s_1^1 = s_1^4$, $s_2^2 = s_2^4$, $s_2^1 = s_2^3$, and $s_1^2 = s_1^5$. This leaves us with $s_1^1 = 1$, $s_2^1 = 3$, $s_1^2 = 3$, $s_2^2 = 1$, $s_1^3 = \frac{1}{3}$, and $s_2^5 = \frac{1}{3}$, as the possible values to generate the initial valid inequalities. Hence, the

algorithm can be initialized with the following MILO relaxation:

$$\min\ 15\sum_{i=1}^{4}\sum_{j=1}^{2} x_{ij} + \sum_{j=1}^{2} s_j + 100\sum_{j=1}^{2} y_{j1} + 500\sum_{j=1}^{2} y_{j2} \tag{21}$$

$$\text{s.t. } 5\sum_{i=1}^{4} x_{ij} - 10z_{j1} - 20z_{j2} = 0, j = 1,2 \tag{22}$$

$$\sum_{j=1}^{2} x_{ij} = 1, i = 1,\ldots,4 \tag{23}$$

$$\sum_{k=1}^{2} y_{jk} \leq 1, j = 1,2 \tag{24}$$

$$z_{jk} - y_{jk} \leq 0, j = 1,2, k = 1,2 \tag{25}$$

$$z_{jk} - \frac{1}{4}s_j \leq \frac{1}{4}, j = 1,2, k = 1,2 \tag{26}$$

$$z_{jk} - \frac{1}{16}s_j \leq \frac{9}{16}, j = 1,2, k = 1,2 \tag{27}$$

$$z_{jk} - \frac{9}{16}s_j \leq \frac{1}{16}, j = 1,2, k = 1,2 \tag{28}$$

$$x_{ij}, y_{jk} \in \{0,1\}, i = 1,\ldots,4, j = 1,2, k = 1,2 \tag{29}$$

$$s_j, z_{jk} \geq 0, j = 1,2, k = 1,2. \tag{30}$$

Let us analyze the structure of Problem (21) – (30). The first observation is that the first term in the objective function can be treated as a constant. The second observation is that the difference in cost between servers with capacity 10 and servers with capacity 20 is 400. This tells us that to use a server of capacity 20 one need savings of 400 or more. The third observation is that those savings must come from lowering the time the clients spend in the system, which is in this case given by the s_j. Given this structure it follows that there are 12 optimal solutions to problem (21) – (30) with optimal objective value equal to 290. The common structure for those solutions is that all of them will use the two servers of capacity 10. Example of optimal solutions are:

$$x_{11} = x_{12} = x_{23} = x_{24} = 1, y_{11} = z_{11} = y_{21} = z_{21} = 1, s_1 = s_2 = 7$$

$$x_{21} = x_{13} = x_{22} = x_{14} = 1, y_{11} = z_{11} = y_{21} = z_{21} = 1, s_1 = s_2 = 7$$

$$x_{21} = x_{22} = x_{13} = x_{14} = 1, y_{11} = z_{11} = y_{21} = z_{21} = 1, s_1 = s_2 = 7$$

Note that these solutions are suboptimal for the original problem. In this situation it is not possible to compute new values for R_j^h, which makes impossible to generate new valid inequalities. Hence, the algorithm will finish with a suboptimal solution.

In summary, Algorithm 1 ignores that it is possible that no new s_j^{new} may be computed. We note that in practice, with a careful approach for the definition of the initial cuts, this problem may be in general avoided. For example, in [12, 29] the initial set of cuts is chosen so that the approximation achieved with the tangents is arbitrarily precise. However, that still does not guarantee the elimination of unstable solutions.

3. New MISOCO Formulations

Before deriving our MISOCO formulations we need to reformulate the objective function of Problem (1)–(5), which can lead to the indeterminate form $\frac{0}{0}$ in the second term when a server is not deployed. To avoid this situation we linearize that objective function as follows. Consider the nonlinear term in the objective function (1)

$$d\sum_{j=1}^{m}\frac{\sum_{i=1}^{\ell}\lambda_i x_{ij}}{\sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}}.$$

Using again the inequalities (6), we obtain the following formulation:

$$\min \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m}\sum_{j=1}^{m} s_j + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \tag{31}$$

$$\text{s.t. } s_j \geq \begin{cases} \frac{\sum_{i=1}^{\ell}\lambda_i x_{ij}}{\sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}}, & \text{if } \sum_{i=1}^{\ell}\lambda_i x_{ij} > 0 \\ 0, & \text{otherwise.} \end{cases} \quad j = 1,\ldots,m \tag{32}$$

$$(2)\text{–}(5)$$

We now show that the reformulation (31)–(32) is equivalent to (1)–(5).

Lemma 1. *Formulation* (31)–(32) *is equivalent to* (1)–(5).

Proof. We need to show that (31)–(32) does not ignore the effect that the cost d has in the objective function (1) for unstable solutions. Note that an unstable solution in (31)–(32) requires that $\sum_{k=1}^{n}\mu_{jk}y_{jk} - \sum_{i=1}^{\ell}\lambda_i x_{ij} = 0$ for some server j. There are two different cases in which this may happen. First, consider the case when $\sum_{j=1}^{n}\lambda_i x_{ij} = 0$. Because (31) is minimizing we obtain that $s_j = 0$ and $\sum_{k=1}^{n}\mu_{jk}y_{jk} = 0$, which is a stable solution. Second, consider the case when $\sum_{j=1}^{n}\lambda_i x_{ij} = \sum_{k=1}^{n}\mu_{jk}y_{jk} > 0$. Then to satisfy (32) we need that $s_j \to +\infty$, which makes (31) go to $+\infty$. This shows that the effect of the cost d in the original formulation for unstable solutions is still accounted for in (31)–(32). □

We may now use (31)–(32) to derive several equivalent MISOCO formulations. We start in Section 3.1 with what we consider straightforward MISOCO formulations where we use the binary constraint on the x vector. Then, we show in Section 3.2 that it is also possible to use the y vector to obtain alternative MISOCO formulations. In this case we must use an additional algebraic step. Finally, in Section 3.3 we show how alternative formulations may be obtained by introducing the traffic intensity of an $M/M/1$ queuing system in the formulation.

3.1. MISOCO Models Based on the x Variables

Our first MISOCO formulations arise from using the binary nature of x to deal with the nonlinearity in constraint (32).

3.1.1. MISOCO Formulation 1

In this first approach we manage the nonlinearity in (32) as follows. First, from (7) and the binary constraint over the x vector, we have that

$$\frac{\sum_{i=1}^{\ell} \lambda_i x_{ij}}{\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij}} \leq s_j \qquad \Longleftrightarrow \qquad \sum_{i=1}^{\ell} \lambda_i x_{ij}^2 \leq s_j \left(\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \right). \tag{33}$$

Notice that thanks to the minimization direction in (31), the right-hand side inequality in (33) captures all the properties of (32).

Next we introduce a variable $t_j \geq 0$ to obtain the following reformulation of the inequality on the right-hand side of (33):

$$\sum_{i=1}^{\ell} \lambda_i x_{ij}^2 \leq s_j t_j \tag{34}$$

$$\left(\sum_{k=1}^{n} \mu_{ij} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \right) \geq t_j. \tag{35}$$

Notice that the inequality on the right of (33) and (34)–(35) are not equivalent in general. However, in this context their equivalence comes from the minimizing direction of the problem and the assumption that $d \geq 0$; this forces (35) to be always binding. Also notice that (35) dominates (2) and that (34) is a rotated second order cone (SOC). Hence, we obtain our first MISOCO reformulation for problem (31)–(32):

$$\begin{aligned}
\min \ & \sum_{i=1}^{\ell} \sum_{j=1}^{m} c_{ij} x_{ij} + d \sum_{j=1}^{m} s_j + \sum_{j=1}^{m} \sum_{k=1}^{n} p_{jk} y_{jk} \\
\text{s.t.} \ & \sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \geq t_j \quad j = 1, \ldots, m \\
& \sum_{i=1}^{\ell} \lambda_i x_{ij}^2 \leq s_j t_j \quad j = 1, \ldots, m \\
& t_j, s_j \geq 0 \quad j = 1, \ldots, m \\
& (3) - (5).
\end{aligned} \tag{MISOCO 1}$$

From this reformulation process we obtain the following result.

Lemma 2. *The formulation* (MISOCO 1) *is equivalent to the problem formulation* (31)–(32).

The dimensions of the problem affect formulation (MISOCO 1) as follows:

- The number of SOCs depends on the number of service locations m, which is usually significantly smaller than the number of demand locations.
- The dimension of the cones depends on the number of demand locations ℓ.

3.1.2. MISOCO Formulation 2

Here we manage the nonlinear constraint (32) in a slightly different way. Because

$$\frac{\sum_{i=1}^{\ell}\lambda_i x_{ij}}{\sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}}=\sum_{i=1}^{\ell}\frac{\lambda_i x_{ij}}{\sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}},$$

we can reformulate the left-hand side inequality of (33) as:

$$\sum_{i=1}^{\ell}u_{ij}\le s_j,\quad \frac{\lambda_i x_{ij}}{\sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}}\le u_{ij}. \tag{36}$$

Note that the minimization will force the second inequality in (36) to be binding. Furthermore, using the binary constraint over x and (2), we obtain

$$\frac{\lambda_i x_{ij}}{\sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}}\le u_{ij}\qquad\Longleftrightarrow\qquad \lambda_i x_{ij}^2\le u_{ij}\left(\sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}\right). \tag{37}$$

Again, the inequality on the right in (37) captures all the properties of (32). Here, we also introduce auxiliary variables t_j to reformulate the rotated SOC (37), and hence obtain our second MISOCO reformulation for problem (31)–(32):

$$\begin{aligned}
\min\ & \sum_{i=1}^{\ell}\sum_{j=1}^{m}c_{ij}x_{ij}+d\sum_{j=1}^{m}s_j+\sum_{j=1}^{m}\sum_{k=1}^{n}p_{jk}y_{jk}\\
\text{s.t.}\ & \sum_{k=1}^{n}\mu_{jk}y_{jk}-\sum_{i=1}^{\ell}\lambda_i x_{ij}\ge t_j\quad j=1,\dots,m\\
& \sum_{i=1}^{\ell}u_{ij}\le s_j\quad j=1,\dots,m\\
& \lambda_i x_{ij}^2\le t_j u_{ij}\quad i=1,\dots,\ell;\quad j=1,\dots,m\\
& t_j,u_{ij}\ge 0\quad i=1,\dots,\ell;\quad j=1,\dots,m\\
& (3)\ -\ (5).
\end{aligned}\tag{MISOCO 2}$$

From this reformulation process we obtain the following result.

Lemma 3. *The formulation* (MISOCO 2) *is equivalent to the problem formulation* (31)–(32).

The dimensions of the problem affect formulation (MISOCO 2) as follows:

- The number of SOCs depends on the number of demand sources ℓ and service locations m. Specifically, for each additional demand source we need to add m new rotated SOCs, for a total of ℓm additional cones.
- On the other hand, the dimension of the cones is fixed to 3.

3.2. MISOCO Models Based on the y Variables

Our next group of MISOCO models shows how we can use the binary constraint over y to deal with the nonlinearity in constraint (32).

3.2.1. MISOCO Formulation 3

For this model we start by adding the term $\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij}$ to both sides of the inequality on the right-hand side of (7) to obtain

$$\sum_{k=1}^{n} \mu_{jk} y_{jk} \leq (1 + s_j) \left(\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \right). \tag{38}$$

Using the binary constraint over y and (2), the inequality (38) can be reformulated as a rotated SOC as follows

$$\sum_{k=1}^{n} \mu_{jk} y_{jk}^2 \leq (1 + s_j) \left(\sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \right). \tag{39}$$

Similarly to what we did in Section 3.1.1, we may introduce an auxiliary variable t_j to reformulate the rotated SOC (39). Hence, we obtain our third MISOCO reformulation for problem (31)–(32):

$$\begin{aligned}
\min \ & \sum_{i=1}^{\ell} \sum_{j=1}^{m} c_{ij} x_{ij} + d \sum_{j=1}^{m} s_j + \sum_{j=1}^{m} \sum_{k=1}^{n} p_{jk} y_{jk} \\
\text{s.t.} \ & \sum_{k=1}^{n} \mu_{ik} y_{ik} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \geq t_j \quad j = 1, \ldots, m \\
& \sum_{k=1}^{n} \mu_{jk} y_{jk}^2 \leq (1 + s_j) t_j \quad j = 1, \ldots, m \\
& t_j \geq 0 \quad j = 1, \ldots, m \\
& (3) - (5).
\end{aligned} \tag{MISOCO 3}$$

From this reformulation process the following result follows.

Lemma 4. *The formulation* (MISOCO 3) *is equivalent to the problem formulation* (31)–(32).

The dimensions of the problem affect formulation (MISOCO 3) as follows:

- The number of SOCs depends on the number of service locations m.
- The dimensions of the cones depend on the number of service levels n.

Furthermore, we note that the rotated cones have smaller dimension than for (MISOCO 1) because this formulation works on the y variables instead of the x variables (in a way similar to what was done in Section 3.1.1).

3.2.2. MISOCO Formulation 4

For this model we use the constraint (3) and the rotated SOC constraint in (MISOCO 3) to derive a new MISOCO formulation. Recall the constraint

$$\sum_{k=1}^{n} y_{jk} \leq 1 \quad j = 1, \ldots, m, \tag{40}$$

which requires that for any feasible solution at most one server level is deployed. Hence, the rotated SOC constraint in (MISOCO 3) is equivalent to the following set of constraints

$$\mu_{jk} y_{jk}^2 \leq (1 + s_j) t_j \quad j = 1, \ldots, m; k = 1, \ldots, n. \tag{41}$$

Note that if $\mu_{jk} = 0$, then capacity k for server j is zero and that server level can simply be ignored in the formulation.

We thus obtain our fourth MISOCO reformulation for problem (31)–(32):

$$\begin{aligned}
\min \ & \sum_{i=1}^{\ell} \sum_{j=1}^{m} c_{ij} x_{ij} + d \sum_{j=1}^{m} s_j + \sum_{j=1}^{m} \sum_{k=1}^{n} p_{jk} y_{jk} \\
\text{s.t.} \ & \sum_{k=1}^{n} \mu_{jk} y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \geq t_j \quad j = 1, \ldots, m \\
& \mu_{jk} y_{jk}^2 \leq (1 + s_j) t_j \quad j = 1, \ldots, m; \quad k = 1, \ldots, n \\
& t_j \geq 0 \quad j = 1, \ldots, m \\
& (3) - (5).
\end{aligned} \tag{MISOCO 4}$$

From this reformulation process the following result follows.

Lemma 5. *The formulation* (MISOCO 4) *is equivalent to the problem formulation* (31)–(32).

The dimensions of the problem affect formulation (MISOCO 4) as follows:

- The number of SOCs depends on the number of service locations m and levels n. Specifically, with each additional level, we need to include m new rotated cones.
- The advantage that the dimension of these cones is fixed to 3.

3.2.3. MISOCO Formulation 5

For our fifth formulation we want to exploit further the binary constraint on the variables y and constraint (3). We first define the variables:

$$u_j = \sum_{k=1}^{n} \sqrt{\mu_{jk}} y_{jk}, \ j = 1, \ldots, m. \tag{42}$$

and use them to reformulate (38) for each $j = 1, \ldots, m$ as follows:

$$u_j^2 \leq (1 + s_j) t_j. \tag{43}$$

Hence, we obtain our fifth MISOCO reformulation for problem (31)–(32):

$$
\begin{aligned}
\min\ & \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m} s_j + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \\
\text{s.t.}\ & \sum_{k=1}^{n} \mu_{jk}y_{jk} - \sum_{i=1}^{\ell} \lambda_i x_{ij} \geq t_j \quad j = 1,\ldots,m \\
& u_j^2 \leq (1+s_j)t_j \quad j = 1,\ldots,m \\
& u_j = \sum_{k=1}^{n} \sqrt{\mu_{jk}}y_{jk},\ j = 1,\ldots,m \\
& t_j, u_j \geq 0 \quad j = 1,\ldots,m \\
& (3)\ -\ (5).
\end{aligned}
\tag{MISOCO 5}
$$

From this reformulation process the following result follows.

Lemma 6. *The formulation* (MISOCO 5) *is equivalent to the problem formulation* (31)–(32).

The dimensions of the problem affect formulation (MISOCO 5) as follows:

- The number of SOCs depends on the number of service locations m.
- The dimension of the cones is fixed to 3.

Notice that this formulation does not requires the direct use of the binary constraint over y to obtain the rotated SOC. For this reason, it avoids the slight relaxation that happens in the two previous relaxations when we squared the variables y.

3.3. MISOCO Models based on the Traffic Intensity

For our two final MISOCO formulations we use the traffic intensity of an $M/M/1$ queuing system defined (in our case) as:

$$\rho_j = \frac{\sum_{i=1}^{\ell} \lambda_i x_{ij}}{\sum_{k=1}^{n} \mu_{jk} y_{jk}}, \quad j = 1,\ldots,m. \tag{44}$$

Using (44) we can reformulate (31)–(32) as follows:

$$\min \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m}\sum_{j=1}^{m} s_j + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \tag{45}$$

$$\text{s.t.}\ \frac{\rho_j}{1-\rho_j} \leq s_j \quad j = 1,\ldots,m \tag{46}$$

$$\rho_j \geq \begin{cases} \frac{\sum_{i=1}^{\ell} \lambda_i x_{ij}}{\sum_{k=1}^{n} \mu_{jk} y_{jk}}, & \text{if } \sum_{i=1}^{\ell} \lambda_i x_{ij} > 0 \\ 0, & \text{otherwise.} \end{cases} \quad j = 1,\ldots,m \tag{47}$$

$$0 \leq \rho_j \leq 1, s_j \geq 0 \quad j = 1,\ldots,m \tag{48}$$

$$(2)\ -\ (5).$$

The inequality in (47) follows from the minimization direction. Specifically, note that for some $\hat{\rho} < \bar{\rho}$ one has that

$$\frac{\hat{\rho}}{1-\hat{\rho}} < \frac{\bar{\rho}}{1-\bar{\rho}}.$$

Note also that here the possibility of an indeterminate form is handled with (47).

Now, we need to manage the nonlinearities in (46) and (47). For (46) we have that:

$$\frac{\rho_j}{1-\rho_j} \leq s_j \Leftrightarrow \rho_j \leq s_j\,(1-\rho_j) \Leftrightarrow 1 \leq (1+s_j)\,(1-\rho_j)\,,$$

where the last inequality is SOC-representable [7, Chapter 3], since $1-\rho_j \geq 0$ and $1+s_j \geq 0$.

For (47) we have that

$$\frac{\sum_{i=1}^{\ell} \lambda_i x_{ij}}{\sum_{k=1}^{n} \mu_{jk} y_{jk}} \leq \rho_j \Leftrightarrow \sum_{i=1}^{\ell} \lambda_i x_{ij} \leq \rho_j \sum_{k=1}^{n} \mu_{jk} y_{jk}. \tag{49}$$

Notice that in this formulation (49) is equivalent to (47). Note also that (49) dominates (2), which allows us to greatly simplify the problem formulation as follows:

$$\min \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m} s_j + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \tag{50}$$

$$\text{s.t.} \sum_{i=1}^{\ell} \lambda_i x_{ij} \leq \rho_j \sum_{k=1}^{n} \mu_{jk} y_{jk} \quad j = 1,\ldots,m \tag{51}$$

$$1 \leq (1+s_j)\,(1-\rho_j) \quad j = 1,\ldots,m \tag{52}$$

$$0 \leq \rho_j \leq 1, s_j \geq 0 \quad j = 1,\ldots,m \tag{53}$$

$$(3) - (5).$$

From this reformulation process we obtain the following result.

Lemma 7. *The formulation* (50)–(53)*, is equivalent to the problem formulation* (31)–(32)*.*

To obtain MISOCO reformulations using this path it remains to deal with the nonlinearity in (51). We present two alternatives in the following sections.

3.3.1. MISOCO Formulation 6

The first alternative is to use the binary constraint on the x variables to obtain the following MISOCO reformulation:

$$\begin{aligned}
&\min \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m} s_j + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \\
&\text{s.t.} \sum_{i=1}^{\ell} \lambda_i x_{ij}^2 \leq \rho_j \sum_{k=1}^{n} \mu_{jk} y_{jk} \quad j = 1,\ldots,m \qquad \text{(MISOCO 6)} \\
&(3)\text{–}(5),\ (52)\text{–}(53).
\end{aligned}$$

The dimensions of the problem affect formulation (MISOCO 6) as follows:

- The number of SOCs depends on the number of service locations m. Specifically, this formulation requires $2m$ rotated SOCs.
- The m rotated SOCs in (52) have fixed dimension 3, while the dimension of the rotated SOCs in the first constraint of (MISOCO 6) depends on the number of demand locations ℓ.

3.3.2. MISOCO Formulation 7

The second alternative is to consider the constraints

$$\sum_{k=1}^{n} y_{jk} \leq 1, \quad 0 \leq \rho_j \leq 1,$$

and to introduce new variables $z_{jk} = \rho_j y_{jk}$. Using these for each j makes it possible to reformulate (51) as:

$$\begin{aligned} \min \ & \sum_{i=1}^{\ell}\sum_{j=1}^{m} c_{ij}x_{ij} + d\sum_{j=1}^{m} s_j + \sum_{j=1}^{m}\sum_{k=1}^{n} p_{jk}y_{jk} \\ \text{s.t.} \ & \sum_{i=1}^{\ell} \lambda_i x_{ij} \leq \sum_{k=1}^{n} \mu_{jk} z_{jk} \quad j = 1, \ldots, m \\ & \sum_{k=1}^{n} z_{jk} \leq \rho_j \quad j = 1, \ldots, m \\ & z_{jk} \leq y_{jk} \quad j = 1, \ldots, m; \quad k = 1, \ldots, n \\ & (3)\text{–}(5),\ (52)\text{–}(53). \end{aligned} \tag{MISOCO 7}$$

Note that the minimization will always make the second set of constraints in (MISOCO 7) bounding.

The dimensions of the problem affect formulation (MISOCO 7) as follows:

- The number of rotated SOCs in this formulation equals the number of service locations m.
- The m rotated SOCs in (52) have fixed dimension 3.

Note that (MISOCO 7) is closely related to formulation (15). What differentiates the two formulations is the way each handles the nonlinearity in (14). In (15) the nonlinearity is handled by outer linearization, whereas (MISOCO 7) uses the constraints (52), (53), and:

$$\sum_{k=1}^{n} z_{jk} \leq \rho_j, \quad j = 1, \ldots, m.$$

3.4. Summary of MISOCO formulations

We have shown how the service system design problem with congestion under the $M/M/1$ assumption can be cast as a MISOCO in seven different ways. The process we followed starts with a straightforward approach to find an initial MISOCO model. We then show how, using some extra simple algebra steps, we can obtain different MISOCO models that are affected in different ways by the dimensions of the original problem.

We can compare the seven models using three criteria: the number of additional (scalar) variables needed, the final number of SOCs, and the dimension of the cones. Table 1 shows the comparison among the models.

MISOCO model	1	2	3	4	5	6	7
# additional variables	$2m$	$(2+\ell)m$	$2m$	$2m$	$3m$	$2m$	$(2+\ell)m$
# SOCs	m	ℓm	m	mn	m	$2m$	m
Dimension of the SOCs	$2+\ell$	3	$2+n$	3	3	3 and $2+\ell$	3

Table 1: Comparison of the MISOCO models

The interest in this comparison comes from the possibility of using interior-point methods to solve the continuous relaxations of these problems within a branch-and-cut framework. It is well known that the iteration complexity of interior-point methods for SOC optimization depends directly on the number of SOCs in the problem [1]. For this reason, we performed some computational experiments to analyze the behaviour of these different models, and in particular to evaluate the impact of the number of SOCs. The results are reported in the next section.

4. Computational Testing

For our computational tests we used the test sets from Holmberg et al. [19]. This set contains 71 different instances, whit different numbers of demand and service locations. We modified these instances using the procedure described in [13]. In that procedure we increase the cost of a time unit in the system d each time by a factor of 10, starting with $d = 0.1$ until $d = 1000$, which resulted in a total of 350 instances. We use as a baseline our implementation of Elhedhli's method described in Section 2.2.

Because the formulations are not always ranked in the same order for different problems, we use a performance profile (PP) as proposed in [10]. The key for building a PP is $r_{p,s}$: the ratio between the time it took to solve problem p with formulation s and the best solution time (over all formulations). In our PP the vertical axis has the probability that $\log_2(r_{p,s}) \leq \tau$, where τ is the coordinate on the horizontal axis. For each level of τ, we consider a formulation successful if $\log_2(r_{p,s}) \leq \tau$. Hence, for each choice of τ this gives the proportion of problems for which each formulation was successful. Our results are summarized in the PP in Figure 2, which uses CPU time as the performance measure. We ran these experiments in a computer running linux centos with two Intel Xeon X5650 processors, each of which has 6 cores, and 36 GB or RAM. For these experiments we set a time limit of $10,800$ seconds and used CPLEX 12.6.3 to solve both the MISOCO problems and for our implementation of Elhedhli's method.

Our results show that the probability of formulation MISOCO 7 beign the fastest on a given problem is 0.41. The next best choice would be MISOCO 6 with a probability greater than 0.2 of being the fastest on a given problem, and third is Elhedhli's method with a probability of about 0.18 of being fastest on a given problem. For all the other formulations, the probability is below 0.05.

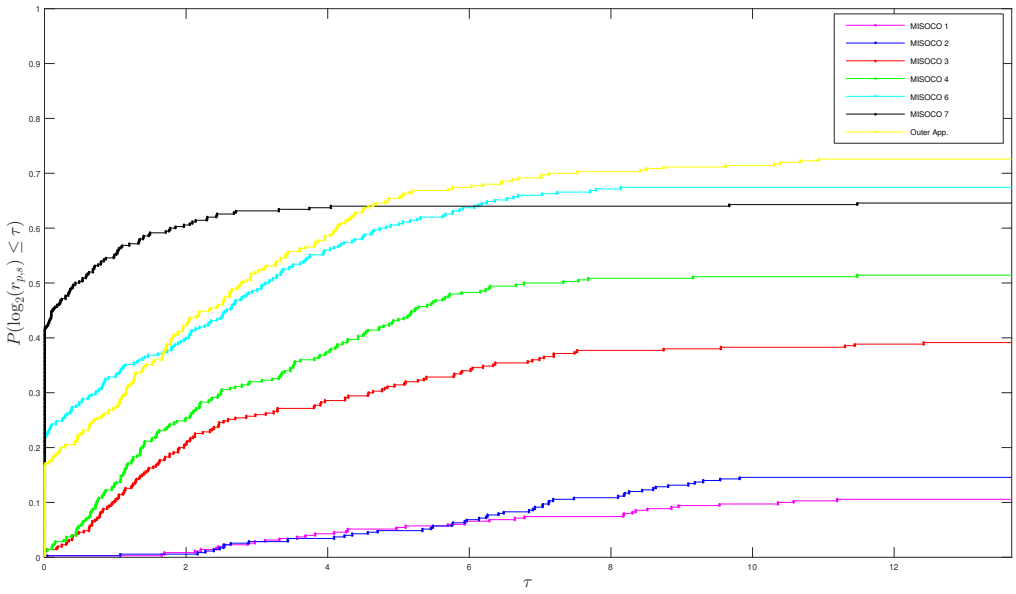

Figure 2: Performance profile based in cpu time in seconds

It turned out that the MILO and the MISOCO 5 reformulations were completely impractical because CPLEX was not successful in solving any of the instances with these models. For that reason they were omited in the PP. A remark about the MILO approach presented in Section 2.1 is in place here. The MILO formulation always ended with an integratility gap greater than 5% within our time limit. We noted that the integer solution available at the moment we stopped CPLEX was in several cases very close to the optimal solution of the problem. This is an indicator that the linear relaxation of MILO reformulation is not very tight, which makes difficult for the solver to improve the lower bound.

Notice that the choice between Elhedhli's method and MISOCO 6 becomes indifferent for $\tau \geq 2$, and also that these two options follow a similar trend. Recall that we are using the base 2 logarithm of the ratio, hence this result translates to accepting being withing a factor of 4 of the best solver to have Elhedhli's method match MISOCO 6 performance.

The gap between Elhedhli's method and MISOCO 7 is even more significant. Elhedhli's method catches up with the performance of MISOCO 7 only for $\tau > 4$. In other words, if the target is to be within a factor greater than 16 of the best approach, then any of MISOCO 7, MISOCO 6, or Elhedhli's method suffice. It is also clear from the PP that these three options dominate all the other formulations.

From our results it is difficult to identify precisely the reason for the dominance of MISOCO 7. One thing that is observable in Table 1 is that MISOCO 7 is among those with the lowest number of SOCs, and all its cones have dimension 3. This is not the whole story though; for instance MISOCO 6 has twice the

number of cones of MISOCO 1, and the same number of cones with dimension $2 + \ell$, but still MISOCO 6 dominates MISOCO 1 (and in fact it dominates all formulations but MISOCO 7).

Finally, notice that the PP at the far right gives the results of the testing while ignoring the ratio comparison. We see that Elhedhli's method was able to solve more than 73 % of the problems while MISOCO 6 and MISOCO 7 were able to solve 68 % and 65 % of the problems respectively. In summary, none of these three approaches was able to solve the full set of problems, and they showed a similar behaviour in terms of scalability.

5. Conclusions

In this paper we presented seven different MISOCO formulation for the service system design problem with congestion. Our computational results show that some of our conic formulations have a better performance when compared with existing exact approaches. This opens the possibility to use current off-the-shelf solvers like CPLEX, MOSEK, and GUROBI. Through the comparison of several conic formulations for the same problem, our experiments suggest that formulations with cones with small dimensions tend to dominate the formulations with cones of higher dimensions. This is observed when comparing MISOCO 1 with MISOCO 2, when comparing MISOCO 3 with MISOCO 4, and when comparing MISOCO 6 with MISOCO 7. Our results also show the importance of choosing the appropriate modelling technique. In particular our experiments show that for obtaining reasonable solution times, the modeler should use MISOCO models 6 and 7.

An important assumptions for all these formulations is that the servers behave as $M/M/1$ queues. This assumption could be relaxed by using an $M/G/1$ approach that allows to consider a general distribution for the service times. Given the encouraging results obtained with the MISOCO formulations presented in this work, our next step is to extend these formulations to consider the $M/G/1$ model instead of the $M/M/1$.

References

[1] Alizadeh, F., Goldfarb, D., 2003. Second-order cone programming. Mathematical Programming 95 (1), 3–51.

[2] Amiri, A., 1997. Solution procedures for the service system design problem. Computers & Operations Research 24 (1), 49 – 60.

[3] Andersen, K., Jensen, A., 2013. Intersection cuts for mixed integer conic quadratic sets. In: Goemans, M., Correa, J. (Eds.), Integer Programming and Combinatorial Optimization. Vol. 7801 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 37–48.

[4] Atamtürk, A., Narayanan, V., 2010. Conic mixed-integer rounding cuts. Mathematical Programming 122 (1), 1–20.

[5] Belotti, P., Góez, J., Pólik, I., Ralphs, T., Terlaky, T., September 2015. Disjunctive conic cuts for mixed integer second order cone optimization. Tech. Rep. G-2015-98, GERAD.

[6] Belotti, P., Góez, J. C., Pólik, I., Ralphs, T. K., Terlaky, T., 2015. A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization. In: Al-Baali, M., Grandinetti, L., Purnama, A. (Eds.), Numerical Analysis and Optimization: NAO-III, Muscat, Oman, January 2014. Springer International Publishing, Cham, pp. 1–35.

[7] Ben-Tal, A., Nemirovski, A., 2001. Lectures on Modern Convex Optimization. Society for Industrial and Applied Mathematics.

[8] Berman, O., Krass, D., 2002. Facility location problems with stochastic demands and congestion. In: Drezner, Z., Hamacher, H. (Eds.), Facility location: applications and theory, 1st Edition. Springer-Verlag Berlin Heidelberg, New York, pp. 329–371.

[9] Castillo, I., Ingolfsson, A., Sim, T., 2009. Social optimal location of facilities with fixed servers, stochastic demand, and congestion. Production and Operations Management 18 (6), 721–736.

[10] Dolan, D., Moré, J., 2002. Benchmarking optimization software with performance profiles. Mathematical Programming 91 (2), 201–213.

[11] Drewes, S., 2009. Mixed integer second order cone programming. Ph.D. thesis, Technische Universität Darmstadt, Germany.

[12] Elhedhli, S., 2005. Exact solution of a class of nonlinear knapsack problems. Operations Research Letters 33 (6), 615 – 624.

[13] Elhedhli, S., 2006. Service system design with immobile servers, stochastic demand, and congestion. Manufacturing & Service Operations Management 8 (1), 92–97.

[14] Glover, F., 1975. Improved linear integer programming formulations of nonlinear integer problems. Management Science 22 (4), 455–460.

[15] Góez, J., 2013. Mixed integer second order cone optimization disjunctive conic cuts: Theory and experiments. Ph.D. thesis, Lehigh University.

[16] Günlük, O., Linderoth, J., 2012. Perspective reformulation and applications. In: Lee, J., Leyffer, S. (Eds.), Mixed Integer Nonlinear Programming. Vol. 154 of The IMA Volumes in Mathematics and its Applications. Springer New York, pp. 61–89.

[17] GUROBI, 2014. gurobi.
URL `http://www.gurobi.com/resources/documentation`

[18] Hijazi, H., Bonami, P., Ouorou, A., 2014. An outer-inner approximation for separable mixed-integer nonlinear programs. INFORMS Journal on Computing 26 (1), 31–44.

[19] Holmberg, K., Ronnqvist, M., Yuan, D., March 1999. An exact algorithm for the capacitated facility location problem with single sourcing. European Journal of Operational Research 113 (3), 544–559.

[20] IBM, 2013. IBM ILOG CPLEX Optimization Studio V12.4.
URL `http://publib.boulder.ibm.com/infocenter/cosinfoc/v12r4/index.jsp`

[21] Keller, M., Karl, H., 2014. Response time-optimized distributed cloud resource allocation. In: Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing. DCC '14. ACM, New York, NY, USA, pp. 47–52.

[22] Kılınç, M., Linderoth, J., Luedtke, J., 2010. Effective separation of disjunctive cuts for convex mixed integer nonlinear programs. Optimization Online.

[23] Kılınç, M., Linderoth, J., Luedtke, J., Miller, A., 2014. Strong-branching inequalities for convex mixed integer nonlinear programs. Computational Optimization and Applications 59 (3), 639–665.

[24] Kılınç-Karzan, F., Yıldız, S., 2014. Two-term disjunctions on the second-order cone. In: Lee, J., Vygen, J. (Eds.), Integer Programming and Combinatorial Optimization. Vol. 8494 of Lecture Notes in Computer Science. Springer International Publishing, pp. 345–356.

[25] Modaresi, S., Kılınç, M. R., Vielma, J. P., 2016. Intersection cuts for nonlinear integer programming: convexification techniques for structured sets. Mathematical Programming 155 (1), 575–611.

[26] Modaresi, S., Kılınç, M. R., Vielma, J. P., 2015. Split cuts and extended formulations for mixed integer conic quadratic programming. Operations Research Letters 43 (1), 10 – 15.

[27] MOSEK, 2013. The MOSEK optimization tools manual, Version 7.0.
URL `http://mosek.com/resources/doc/`

[28] Torres, F., 1990. Linearization of mixed-integer products. Mathematical Programming 49 (1-3), 427–428.

[29] Vidyarthi, N., Jayaswal, S., 2014. Efficient solution of a class of location-allocation problems with stochastic demand and congestion. Computers & Operations Research 48, 20 – 30.

[30] Wang, Q., Batta, R., Rump, C. M., 2002. Algorithms for a facility location problem with stochastic customer demand and immobile servers. Annals of Operations Research 111 (1), 17–34.

[31] Wang, Y., 2015. Service system design with immobile servers, stochastic demand and economies of scale. Master's thesis, University of Waterloo.

[32] Zhang, Y., Berman, O., Verter, V., 2009. Incorporating congestion in preventive healthcare facility network design. European Journal of Operational Research 198 (3), 922 – 935.