Demand allocation with latency cost functions

Alessandro Agnetis · Enrico Grande · Andrea Pacifici

Received: 28 July 2009 / Accepted: 2 July 2010 / Published online: 23 July 2010
© Springer and Mathematical Optimization Society 2010

Abstract We address the exact resolution of a Mixed Integer Non Linear Programming model where resources can be activated in order to satisfy a demand (a covering constraint) while minimizing total cost. For each resource, there is a fixed activation cost and a variable cost, expressed by means of latency functions. We prove that this problem is \( \mathcal{NP} \)-hard even for linear latency functions. A branch and bound algorithm is devised, having two important features. First, a dual bound (equal to that obtained by continuous relaxation) can be computed very efficiently at each node of the enumeration tree. Second, to break symmetries resulting in improved efficiency, the branching scheme is \( n \)-ary (instead of binary). These features lead to a successful comparison against two popular commercial and open-source solvers, CPLEX and Bonmin.

Keywords M.I.N.L.P · Latency functions · Convex functions · Branch and bound · Resource allocation

Mathematics Subject Classification (2000) 90C11 · 90C25 · 90C27 · 90C30 · 90C57 · 49M37 · 65K05 · 65K10

1 Introduction

Congestion phenomena arise in many real world problems. It is therefore a key issue in modeling techniques to address those features. Latency functions (see [15]) are a...
useful tool to model a nonlinear increase of the cost of a resource with respect to its usage (e.g., traffic jams). Studies about the behavior of (possibly) congested systems where users act autonomously, i.e. selfishly, have addressed the problem of quantifying the loss of efficiency compared to a centrally optimized situation (see [4,16–18]). There have also been efforts in the direction of pricing edges of networks in order to influence users’ behavior (see [5,7,12,14]).

We study a Mixed Integer Non Linear Program (MINLP) which models demand allocation problems where there is (i) a single commodity demand that has to be satisfied, (ii) a set of potential resources, (iii) a fixed activation cost for each resource and (iv) a congestion effect, that heavily affects the cost of a resource. We aim to a socially desirable solution, that is, a global optimum. In [9] a class of related MINLP problems are addressed and a solution approach is devised, based on solving a semi-infinite MILP obtained using a new class of cuts, the so called perspective cuts. Other approaches to related MINLP problems are devised in [2,6,10], where reformulations of the original MINLPs as Mixed Integer Second-Order Cone programs are proposed.

We denote by $Q$ the set of the $q$ available resources. The cost $\gamma_i(x_i)$ of using resource $i \in Q$ at level $x_i \in [0, 1]$ is given by:

$$\gamma_i(x_i) = \begin{cases} 0 & \text{if } x_i = 0 \\ c_i + x_i f_i(x_i) & \text{if } 0 < x_i \leq 1 \end{cases} \quad (1)$$

where $f_i(\cdot)$ is the latency function. Examples of such situations include the following:

- **Traffic flow on disjoint routes.** A unit flow has to be routed on $q$ disjoint routes. For each route $i$ there is an activation cost $c_i$ and the delay (cost) encountered by a fraction $x$ of flow on this route is $xf_i(x)$.

- **Call center design.** Incoming calls must be dispatched to $q$ possible workstations, each having an installation cost $c_i$. Operational costs are related to the congestion level for each workstation, which can be measured again by means of suitable latency functions.

- **Unidimensional radar sensor location.** A linear portion of space (e.g. river, road) of unit length has to be covered by a set of radar sensors, each having an installation cost $c_i$ and an operational cost which is nonlinearly related to its covering radius [1].

In certain applications, not all resources are different from each other. Two resources $u$ and $v$ are called identical copies if $\gamma_u(\cdot) = \gamma_v(\cdot)$. We will exploit the presence of identical copies to speed up the solution algorithm presented in Sect. 5.

Throughout the paper we assume the following:

**Assumption 1** (i) $xf(x)$ is strictly convex, (ii) $f_i(0) = 0$, and (iii) the derivative $df_i(x) \over dx > 0$ for $x > 0$.

Note that this assumption does not include the case in which $f_i$ is constant. This case is however trivial (see Sect. 3.1.3).
The problem we address is the following:

\[
    z^* = \min \sum_{i \in Q} (c_i y_i + x_i f_i(x_i)) \tag{2}
\]

s.t. \( x_i \leq y_i \), for all \( i \in Q \) \tag{3}

\[
    \sum_{i \in Q} x_i \geq 1 \tag{4}
\]

\[
    x \in \mathbb{R}_+^q \tag{5}
\]

\[
    y \in \{0, 1\}^q \tag{6}
\]

where \( x_i \) indicates the fraction of demand allocated to resource \( i \) (level of usage of resource \( i \)) and \( y_i \) is a binary variable indicating selection of resource \( i \). Note that, because of the fixed costs \( c_i \), the objective function is not convex in the domain \( x \geq 0 \), although it is for \( x > 0 \).

In the above MINLP formulation all the constraints are linear with the \( y \) variables restricted to be integer. Note that the covering constraint (4) will be always active at optimality. We will often refer to (2)–(6) as \( P \).

The plan of the paper is as follows. In Sect. 2 we introduce a related subproblem which is very important for subsequent developments. Section 3 is devoted to the problem complexity. Section 4 describes how to efficiently derive lower bounds. These bounds are used in the branch-and-bound algorithm, outlined in Sect. 5. The results of a thorough experimental campaign are reported in Sect. 6. Finally, some conclusions are drawn.

2 A related subproblem

In this section, we address a related subproblem \( P_S \), which is problem \( P \) when the set \( S \subseteq Q \) of selected resources (i.e., active facilities) is fixed. In this case, the fixed costs are known and variables \( y_i \) vanish from Program (2–6). The resulting problem is:

\[
    z(S) = \sum_{i \in S} c_i + \min \left\{ \sum_{i \in S} x_i f_i(x_i) : \sum_{i \in S} x_i \geq 1; \ x_i \in \mathbb{R}_+, \ i \in S \right\} \tag{7}
\]

Since the objective function is strictly convex, \( P_S \) may be solved by applying Karush-Kuhn-Tucker (KKT) optimality conditions. In what follows, we omit the constant term \( \sum_{i \in S} c_i \). The Lagrangian function \( L(x, \mu, \lambda) \) is defined as:

\[
    L(x, \mu, \lambda) = \sum_{i \in S} x_i f_i(x_i) + \lambda \left( 1 - \sum_{i \in S} x_i \right) - \sum_{i \in S} \mu_i x_i
\]

where \( x \in \mathbb{R}_+^{|S|}, \lambda \in \mathbb{R}_+, \mu \in \mathbb{R}_+^{|S|} \). The KKT conditions for the triple \((x^*, \mu^*, \lambda^*)\), are

\[
    \n\]
\[
\frac{\partial L(x^*, \lambda^*, \mu^*)}{\partial x_i} = f_i(x_i^*) + x_i^* \dot{f}_i(x_i^*) - \lambda^* - \mu_i^* = 0 \quad \text{for all } i \in S \quad (8)
\]

\[
\sum_{i \in S} x_i^* = 1 \quad (9)
\]

\[
\mu_i^* x_i^* = 0 \quad \text{for all } i \in S \quad (10)
\]

\[
x_i^* \geq 0 \quad \text{for all } i \in S \quad (11)
\]

\[
\mu_i^* \geq 0 \quad \text{for all } i \in S \quad (12)
\]

\[
\lambda^* \geq 0 \quad (13)
\]

**Lemma 1** In an optimal solution \(x^*\) of \(P_S\), \(x_i^* > 0\) for all \(i \in S\).

**Proof** Consider any \(i \in S\) such that \(x_i^* > 0\) (there must be at least one such \(i\)). From Assumption 1, \(f_i(x_i^*)\) and \(\dot{f}_i(x_i^*)\) are both positive, so \(\mu_i^* = 0\) and \(\lambda^* = f_i(x_i^*) + x_i^* \dot{f}_i(x_i^*) > 0\). Suppose now that there exists \(x_j^* = 0\), then by Assumption 1, \(f_j(x_j^*) = 0\) and therefore from (8), \(\lambda^* + \mu_j^* = 0\), which implies, by (12) and (13), \(\lambda^* = 0\), a contradiction. \(\square\)

Lemma 1 implies that \(\mu_i^* = 0\) for all \(i \in S\). Then, letting \(g_i(x_i) = \frac{d}{dx_i}(x_i f_i(x_i)) / \frac{d}{dx_i} = f_i(x_i) + x_i \dot{f}_i(x_i)\), from (8) one has

\[
x_i^* = g_i^{-1}(\lambda^*) \quad (14)
\]

which is well defined because of Assumption 1. Hence, \(\lambda^*\) can be computed through

\[
\sum_{i \in S} g_i^{-1}(\lambda^*) = 1 \quad (15)
\]

Depending on the specific functions \(f_i(\cdot)\), the value of \(\lambda^*\) and \(x_i^*\) can be computed either in closed form or via numerical methods. For instance, if \(f_i(x_i) = b_i x_i^p\) (for any \(p \in \mathbb{R}\)), one obtains

\[
\lambda^* = \left( \frac{1}{\sum_{i \in S} (b_i (1 + p))^{-\frac{1}{p}}} \right)^p \quad \text{and} \quad x_i^* = \frac{1}{\sum_{j \in S} \left( \frac{b_j}{b_i} \right)^{\frac{1}{p}}} \quad (16)
\]

If \(f_i(x_i) = b_i \ln(1 + x_i)\), then the \(x_i^*\) and \(\lambda^*\) must satisfy Eq. (8), i.e., \(b_i \ln(1 + x_i^*) + \frac{x_i^*}{1 + x_i^*} = \lambda^*\), for all \(i \in S\), and Eq. (9). In this case a closed form expression for such values cannot be obtained and one should therefore employ numerical methods.

### 3 Complexity issues

In this section we prove that \(P\) is \(\mathcal{NP}\)-hard, even if \(f(\cdot)\) is a linear function (see Sects. 3.2). Nevertheless, as discussed respectively in Sect. 3.1.1, 3.1.2 and 3.1.3, the
problem is easy when the cost functions are all identical, or there are no fixed costs (i.e., $c_i = 0$ for all $i \in Q$), or $f(\cdot)$ is constant.

3.1 Easy cases

3.1.1 Identical cost functions

We first consider the case when all cost functions $\gamma_i(x_i)$ are identical, i.e., for all $i \in Q$:

$$
\gamma_i(x_i) = \gamma(x_i) = \begin{cases} 
0 & \text{if } x_i = 0 \\
c + x_i f(x_i) & \text{if } 0 < x_i \leq 1 
\end{cases}
$$

(17)

In this case, the problem reduces to finding the optimal number $k \leq q$ of active resources.

**Proposition 1** If all cost functions are identical, and an optimal solution $x^* \in \mathbb{R}_+^q$ for $P$ uses $k^*$ resources, then $x_i^* = 1/k^*$ for $i$ corresponding to the chosen $k^*$ resources ($x_i^* = 0$ otherwise).

**Proof** Based on the convexity of $x_i f_i(x_i)$, for any $k^*$-uple of positive numbers $x_1, \ldots, x_{k^*}$ with $\sum_{i=1}^{k^*} x_i = 1$, we have:

$$
k^* \cdot \gamma \left( \frac{1}{k^*} \right) \leq \sum_{i=1}^{k^*} \gamma(x_i). \tag{18}
$$

Hence, we are only left with the question of determining $k^*$. Note that just trying out all values $k = 1, 2, \ldots, q$ would not result in a polynomial algorithm, given the compact input size in this case. However, a polynomial procedure can be easily devised, as follows. First, note that the cost of using $k$ resources is

$$
F(k) = k \left( \frac{1}{k} \right) f \left( \frac{1}{k} \right) + kc = f \left( \frac{1}{k} \right) + kc \tag{19}
$$

Since, by Assumption 1, $\gamma(x_i)$ is strictly convex when $x_i > 0$, we have that

$$
\frac{\partial^2 F}{\partial k^2} = \frac{\partial^2}{\partial k^2} \left( k \gamma \left( \frac{1}{k} \right) \right) = \left( \frac{1}{k^3} \right) \frac{\partial^2 \gamma}{\partial k^2} \left| \frac{1}{k} \right. > 0 \tag{20}
$$

i.e., also $F$ is strictly convex and hence $k^*$ must be such that

$$
F(1) \geq F(2) \geq \cdots \geq F(k^*) \quad \text{and} \quad F(k^*) \leq F(k^* + 1) \leq \cdots \leq F(q). \tag{21}
$$

Therefore, a binary search can be used to efficiently find $k^*$, and the following proposition holds.
Proposition 2 When the $q$ resources are identical with cost functions $\gamma_i(\cdot) = \gamma(\cdot)$ for all $i \in Q$, the problem $P$ is solvable in $O(C \log q)$ time, where $C$ is the maximum computational effort required to compute $\gamma(\frac{1}{n})$. □

3.1.2 Null fixed costs

When there are no fixed costs, the problem reduces to problem $P_S$ addressed in Sect. 2, in which, in particular, $S = Q$. In [3], in the context of a traffic equilibrium problem, the optimal solution is also referred to as a Nash equilibrium, because all marginal costs are identical (and equal to $\lambda^*$).

3.1.3 Constant latency functions

When the latency functions $f_i(x_i) = f_i$ are constant, Assumption 1 does not hold. However, it is easy to see that an optimal solution is simply $x_i^* = y_i^* = 1$, where $i^* = \arg \min \{c_i + f_i, i \in Q\}$ and $x_i^* = y_i^* = 0$ for all $i \neq i^*$.

3.2 $NP$-hardness of $P$

We next show that $P$ is $NP$-hard for any latency function of the form $f_i(x_i) = b_i x^{p_i}$, for any $p > 0$, even non integer. Note that this includes also concave latency functions (when $p \leq 1$).

Proposition 3 Problem $P$ is $NP$-hard, even when the latency functions are defined as $f_i(x_i) = b_i x^{p_i}$, for any $p > 0$.

Proof We use a reduction from the well known binary $NP$-complete problem:

PARTITION: given $q$ nonnegative integers $w_1, w_2, \ldots, w_q$, with $\sum_{i=1}^q w_i = W$, is there a subset $S \subseteq \{1, \ldots, q\}$ such that $\sum_{i \in S} w_i = \frac{W}{2}$?

First, recall from Sect. 2 that when we select a set $S$ of resources, the following convex subproblem $P_S$ is obtained:

$$z(S) = \min \left\{ \sum_{i \in S} (c_i + b_i x_i^{p+1}) : \sum_{i \in S} x_i \geq 1; x \geq 0 \right\}$$ (22)

which, from (16), is solved by

$$x_i^* = b_i^{\frac{1}{p}} \frac{1}{\sum_{j \in S} b_j^{\frac{1}{p}}}$$ (23)

yielding, after some algebra,

$$z(S) = \sum_{i \in S} c_i + \frac{1}{\left( \sum_{i \in S} b_i^{\frac{1}{p}} \right)^p}$$ (24)

Springer
and the optimal value $z^*$ of $P$ can be computed as

$$z^* = \min_{S \subseteq \{1, \ldots, q\}} z(S)$$  \hspace{1cm} (25)$$

Given an instance $I$ of \textsc{Partition}, let us define the following instance of $P$. There are $q$ resources, with $c_i = w_i$ and $b_i = \frac{1}{w_i^2} \left( \frac{W}{2} \right)^{p+1}$, $i = 1, \ldots, q$. Once a set $S$ of active resources is selected, from (24), the value of the optimal solution to $P_S$ is

$$z(S) = \sum_{i \in S} w_i + \frac{1}{p} \left( \frac{W}{2} \right)^{p+1} \left( \sum_{i \in S} w_i \right)^p. \hspace{1cm} (26)$$

Observe now that the function $\bar{z}(\xi) = \xi + a \xi^p$, in its nonnegative domain, has a minimum at $\xi^* = \left( \frac{1}{a^{1/p}} \right)^{1/p}$, giving $\bar{z}(\xi^*) = \frac{p+1}{p} \left( \frac{1}{a^{1/p}} \right)^{1/p}$. Therefore, since in our case $a = \frac{1}{p} \left( \frac{W}{2} \right)^{p+1}$, a lower bound for $z^*$ is given by $\frac{p+1}{p} \left( \frac{W}{2} \right)^{p+1}$, which is attained if and only if a set $S$ exists such that $\sum_{i \in S} w_i = \left( \frac{1}{a^{1/p}} \right)^{1/p} = \frac{W}{2}$. This in turn occurs if and only if $I$ is a yes-instance of \textsc{Partition}. \hfill \Box

4 Dual bounds

In this section we will study the Lagrangean relaxation of the problem, its continuous relaxation and the relation between them, in order to efficiently compute a dual bound.

4.1 Lagrangean bound

We now use Lagrangean relaxation to obtain a lower bound on $z^*$, the optimal solution value of Problem $P$. We first address the Lagrangean problem (Sect. on 4.1.1), and then the Lagrangean dual (Sect. 4.1.2).

4.1.1 Lagrangean problem

Relaxing the activation constraints (3) using nonnegative Lagrangean multipliers $\kappa_i$, $i = 1, \ldots, q$, we obtain the following Lagrangean problem $\text{LRP}(\kappa)$:

$$z_{\text{LRP}}(\kappa) = \min \sum_{i \in Q} [(c_i - \kappa_i) y_i + x_i f_i(x_i) + \kappa_i x_i]\hspace{1cm} (27)$$

s.t. $\sum_{i \in Q} x_i \geq 1$ \hspace{1cm} (28)

$x \in \mathbb{R}_+^q$ \hspace{1cm} (29)

$y \in \{0, 1\}^q$ \hspace{1cm} (30)
Problem \(LRP(\kappa)\) is a relaxation of the original problem for any \(\kappa \geq 0\), and it is decomposable since the optimal values for the \(y\) variables are independent of the values of the \(x\). Optimal solution values for the \(y\) variables are the following:

\[
y_i^* = \begin{cases} 
1 & \text{if } c_i < \kappa_i \\
0 & \text{if } c_i \geq \kappa_i
\end{cases} \quad \text{for all } i \in Q.
\]

We denote as \(LRP'(\kappa)\) the remaining problem on the \(x\) variables:

\[
z_{LRP'}(\kappa) = \min_{i \in Q} \sum_{i \in Q} x_i f_i(x_i) + \kappa_i x_i
\]

s.t. \(\sum_{i \in Q} x_i \geq 1\)

\(x \in \mathbb{R}_+^q\)

and therefore \(z_{LRP}(\kappa) = z_{LRP'}(\kappa) + \sum_{i \in Q} (c_i - \kappa_i) y_i^*\). \(LRP'(\kappa)\) is a convex program and it can be optimally solved via KKT conditions. Using multiplier \(\lambda \in \mathbb{R}_+\) for the covering constraint (32) and multipliers \(\mu \in \mathbb{R}_+^q\) for nonnegativity constraints (33), we obtain the following Lagrangean function:

\[
L_\kappa(\lambda, \mu) = \sum_{i \in Q} (x_i f_i(x_i) + \kappa_i x_i - \mu_i x_i) + \lambda \left( 1 - \sum_{i \in Q} x_i \right).
\]

Thus, recalling that \(g_i(x_i) = f_i(x_i) + x_i \dot{f}_i(x_i)\), the KKT conditions are

\[
\frac{\partial L_\kappa(x^*, \lambda^*)}{\partial x_i} = g_i(x_i^*) + \kappa_i - \lambda^* - \mu_i^* = 0 \quad \text{for all } i \in Q
\]

\(x_i^* \geq 1\)

\(\mu_i^* x_i^* = 0 \quad \text{for all } i \in Q\)

\(x_i^* \geq 0 \quad \text{for all } i \in Q\)

\(\mu_i^* \geq 0 \quad \text{for all } i \in Q\)

\(\lambda^* \geq 0\).

Now let \(S\) denote the set (so far unknown) of resources \(i\) such that \(x_i^* > 0\). If \(i \in S\), clearly \(\mu_i^* = 0\). Since, from Assumption 1, \(g_i(\cdot)\) is positive for positive arguments, from (34) we have

\[
\lambda^* > \kappa_i, \quad \text{for all } i \in S.
\]

Consider now \(i \notin S\). In this case, \(x_i^* = 0\) and hence from (34), \(g_i(x_i^*) = g_i(0) = \lambda^* + \mu_i^* - \kappa_i\). Since, by Assumption 1, \(g_i(0) = 0\), one has \(\mu_i^* = \kappa_i - \lambda^*\) and therefore, from (38)
\[ \lambda^* \leq \kappa_i, \quad \text{for all } i \notin S. \] (41)

Now, let us renumber the multipliers \( \kappa_i \) in nondecreasing order. Equations (40) and (41) imply that an optimal solution for \( \text{LRP}'(\kappa) \) has the following structure:

\[ \kappa_1 \leq \kappa_2 \leq \cdots \leq \kappa^*_h < \lambda^* \leq \kappa_{h+1}^* \leq \cdots \leq \kappa_q^* \quad \text{on } S \]
\[ \kappa_1^* \leq \cdots \leq \kappa_{q+1}^* \quad \text{on } Q \setminus S \] (42)

Note that if \( S = Q, \kappa_q < \lambda^* \). From (34), since in an optimal solution \( x_i \leq 1 \) and due to the fact that \( g_i(\cdot) \) is a nondecreasing function, we can bound \( \lambda^* \) as

\[ \lambda^* \leq \max_{i=1,\ldots,q} \{ g_i(1) + \kappa_i \} = \kappa_{q+1} \] (43)

and moreover, for all \( i \in S \)

\[ x_i^* = g_i^{-1}(\lambda^* - \kappa_i). \] (44)

Let us now define

\[ G_h(\lambda) = \sum_{i=1}^{h} g_i^{-1}(\lambda - \kappa_i) - 1 \] (45)

Recalling that \( S = \{1, \ldots, h^*\} \), since constraint (35) is active at optimality, \( \lambda^* \) can be computed through the expression

\[ G_{h^*}(\lambda^*) = 0 \] (46)

The next lemmas establish three useful properties of functions \( G_h(\lambda) \).

**Lemma 2** \( G_h(\lambda) \) is monotonically increasing in the interval \( \lambda \in (\kappa_h, \kappa_{h+1}] \), for all \( h = \{1, \ldots, q\} \).

**Proof** This immediately follows from the definition of \( G_h(\cdot) \) in Eq. (45) and from the fact that \( g_i^{-1}(\cdot) \) is monotonically increasing for positive arguments. \( \square \)

**Lemma 3** \( G_{h-1}(\kappa_h) = G_h(\kappa_h) \) for \( h = 2, \ldots, q \).

**Proof** From (45), one has

\[ G_h(\kappa_h) = G_{h-1}(\kappa_h) + g_h^{-1}(0) \]

Since \( g_h(0) = f_h(0) + 0 f'_h(0) = 0 \), the thesis follows. \( \square \)

Now consider the function \( G(\lambda) \) which takes values for \( \kappa_1 \leq \lambda \leq \kappa_{q+1} \), defined as:

\[ G(\lambda) = G_h(\lambda) \quad \text{for } \kappa_h \leq \lambda \leq \kappa_{h+1}, \quad h = 1, \ldots, q \] (47)
Fig. 1 Resource selection algorithm for $LRP'($κ$)$

| Lemma 4 | $G(κ_1) = -1$ and $G(κ_{q+1}) \geq 0$. |

**Proof** $G(κ_1) = -1$ follows from (45) with $h = 1$. From (43), consider $i^* = \arg\max_{i=1,...,q} \{g_i(1) + κ_i\}$ such that

$$g_i^{-1}(κ_{q+1} - κ_{i^*}) = 1$$

again from (45),

$$G(κ_{q+1}) = G_q(κ_{q+1}) = \sum_{i=1}^{q} g_i^{-1}(κ_{q+1} - κ_i) - 1 = \sum_{i\neq i^*} g_i^{-1}(κ_{q+1} - κ_i)$$

Since $κ_{q+1} \geq κ_i$, for any $i = 1, \ldots, q$ and $g_i^{-1}(\cdot) \geq 0$ for nonnegative arguments, one finally obtains $G(κ_{q+1}) \geq 0$. \hfill □

From Lemmas 2 and 3, $G(λ)$ is continuous and monotonically increasing. Hence, from Lemma 4, the following holds:

**Proposition 4** A unique $λ^*$ exists such that $G(λ^*) = 0$. \hfill □

In order to find $λ^*$ and therefore the set $S = \{1, \ldots, h^*\}$ of active resources, we first identify the interval $(κ_{h^*}, κ_{h^*+1})$ such that $G(λ^*) = G_h^*(λ^*) = 0$. This can be efficiently done through bisection, as described in the algorithm Resource Selection (Fig. 1). Note that since an optimal solution always exists, the algorithm is guaranteed to terminate.

### 4.1.2 Lagrangean dual

Let us now consider the Lagrangean dual problem

$$z_{LRP}(κ^*) = \max_κ \{z_{LRP}(κ)\}$$

(48)

It turns out that it can be solved by simply setting the Lagrangean multipliers to the values of the fixed costs.
Lemma 5 \[ \kappa_i^* = c_i, \text{ for all } i = 1, \ldots, q. \]

Proof Consider the optimal solution to \( LRP(c) \), and multipliers \( \kappa \) obtained by changing any coefficient of \( c \). We next show that the optimal value of the objective function cannot increase.

(i) Pick any \( i \), and let \( \kappa \) be defined as \( \kappa_i = c_i - \delta \) \((0 < \delta < c_i)\) and \( \kappa_j = c_j \) for \( j \neq i \). Then, \( y_i^* = 0 \) in the optimal solution to \( LRP(\kappa) \), and so there is no contribution of the \( y \) variables to \( z_{LRP}(\kappa) \). Since \( \kappa < c \), \( z_{LRP}(c) \geq z_{LRP}(\kappa) \).

(ii) Pick any \( i \), and let now \( \kappa \) be defined as \( \kappa_i = c_i + \delta \) \((\delta > 0)\) and \( \kappa_j = c_j \) for \( j \neq i \). This time, \( y_i^* = 1 \) in the optimal solution to \( LRP(\kappa) \), and this contributes \(-\delta\) to the value of the optimal solution to \( LRP(\kappa) \). Now, the remaining part of the objective function increases of no more than \( \delta x_i^* \), where \( x_i^* \) is the value of \( x_i \) in the optimal solution to \( LRP(c) \). Since \( x_i^* \leq 1 \), such increase does not exceed \( \delta \) and hence again \( z_{LRP}(c) \geq z_{LRP}(\kappa) \).

Since any \( \kappa \geq 0 \) can be obtained by applying the above argument to each component of \( c \), this proves that \( c \) is a local optimum. Due to the concavity of the dual function \( z_{LRP}(\cdot) \), \( c \) is also a global optimum. \( \square \)

Observe that in problem \( LRP(c) \), the objective function is not affected by the values of the \( y \) variables, so indeed \( LRP(c) \) and \( LRP'(c) \) are the same problem. Hence, \( z_{LRP}(c) = z_{LRP'}(c) \) and the Lagrangean dual is solved by a single run of algorithm Resource Selection in Figure 1.

4.2 Continuous relaxation

Consider now the continuous relaxation of the original problem \( P \), in which we relax the integrality constraint on the \( y \) variables. It is easy to see that in the optimal solution of the problem obtained, \( x_i^* = y_i^* \) for all \( i \in Q \) and thus we can replace each \( y_i \) with \( x_i \) in the objective function. We therefore get the problem \( CRP \):

\[
\begin{align*}
\text{z}_{CRP} &= \min \sum_{i \in Q} c_i x_i + x_i f_i(x_i) \quad (49) \\
\text{s.t.} \quad &\sum_{i \in Q} x_i \geq 1 \quad (50) \\
&x \in \mathbb{R}^q_+ \quad (51)
\end{align*}
\]

Note that setting \( c_i = \kappa_i \) makes \( CRP \) identical to \( LRP(c) \), that is \( z_{CRP} = z_{LRP'}(c) = z_{LRP}(c) \). Hence, Lagrangean dual and continuous relaxation are the same problem. Actually, this could have been established also using convexity arguments (by extending a well known result in linear programming, see [13] and [11]). A consequence of this observation is that the continuous relaxation is solvable using the Resource Selection algorithm with \( \kappa = c \) (Fig. 1).
5 Exact algorithm

An exact branch-and-bound algorithm can be devised, based on the results in the previous sections.

5.1 Bounding strategy

In the enumeration tree, each node $\nu$ represents a subproblem $P(\nu)$ that is defined by (i) a set $ON(\nu)$ of active resources, i.e., $ON(\nu) = \{i : y_i = 1; i \in \{1, \ldots, q\}\} \subseteq Q$, (ii) a set $OFF(\nu)$ of inactive resources (that cannot be used), i.e., $OFF(\nu) = \{i : y_i = 0; i \in \{1, \ldots, q\}\}$, and (iii) the set $Q \setminus (ON(\nu) \cup OFF(\nu))$ of resources that are not yet decided. At a node $\nu$, we force resources in $ON(\nu)$ to be selected and resources in $OFF(\nu)$ to be discarded. To this aim, in the computation of the Lagrangean bound, we use the Resource Selection algorithm with multipliers $\tilde{\kappa}_i$ defined as:

1. $\tilde{\kappa}_i = 0$ for all $i \in ON(\nu)$
2. $\tilde{\kappa}_i$ sufficiently large for all $i \in OFF(\nu)$
3. $\tilde{\kappa}_i = c_i$ for $i \in Q \setminus (ON(\nu) \cup OFF(\nu))$

Recalling (42), this choice for $\tilde{\kappa}$ ensures that, at node $\nu$, $ON(\nu) \subseteq S$. The value of the bound is given by $\sum_{i \in ON(\nu)} c_i + z_LRP'(\tilde{\kappa})$.

5.2 Branching strategy

Recall that two resources $u$ and $v$ are identical copies if $\gamma_u(\cdot) = \gamma_v(\cdot)$. At a node $\nu$, branching consists in deciding how many copies of a resource $i \in Q \setminus (ON(\nu) \cup OFF(\nu))$ are activated, while the other copies are turned off. So, we perform an $n$-ary branching scheme, as follows. If there are $n$ copies of a resource $i$, we generate $n + 1$ nodes (subproblems): in the $\ell$th subproblem, $\ell = 0, 1, \ldots, n$, $\ell$ copies of resource $i$ are activated and the other $n - \ell$ are turned off. Note that, compared to a binary branching strategy (that is, treating identical copies as different resources), we only generate $n + 1$ subproblems instead of $2^n$. Obviously, if there are no identical copies, the scheme boils down to a binary branching scheme.

5.3 Primal bound

The upper bound at the root node is computed using a heuristic method which employs the Resource Selection algorithm. Given an instance of $P$, let $S$ and $\lambda^*$ be the set and the value found by the Resource Selection algorithm with $\kappa = c$. Then use Eq. (14) to compute the values of the activated variables $x_i$ ($i \in S$) and set the others to zero ($x_i = 0, i \notin S$). The current objective function value (primal) is:

$$z(S) = \sum_{i \in S} (c_i + x_i f_i(x_i))$$

Now perform a local search as follows. Choose—by a suitable criterion—a resource $u \in S$ and compute $z(S \setminus \{u\})$. If $z(S \setminus \{u\}) < z(S)$, update the solution ($S := S \setminus \{u\}$).
Then, select a resource $v \not\in S$ and compute $z(S \cup \{v\})$. Again, if $z(S \cup \{v\}) < z(S)$, update the solution ($S := S \cup \{v\}$).

### 6 Computational experiments

The tests have been run on a PIV 3.2 GHz 1GB of RAM under Windows XP; our algorithms have been coded in C++. The computational experiments have been performed for the case of polynomial latency functions, that is $f_i(x_i) = b_i x_i^p$, i.e., the cost functions $\gamma_i$ are polynomials with degree from 1.5 to 3. A time limit of 1 h was set in all experiments.

Instances are randomly generated. Each instance class can be identified by the triple $(p, q, r)$ where $p \in \{0.5, 1, 1.5, 2\}$ is the exponent of the latency function, $q \in \{100, 200, 300, 500\}$ is the number of resources in the instance and $r \in \{H, L\}$ indicates whether quadratic costs are “high” ($H$) or “low” ($L$) compared with fixed costs. If $r = H$ ($r = L$), the coefficients $b_i$ are on the average up to 30 times larger (smaller) than fixed costs $c_i$. For each class, 20 instances are randomly generated.

In order to compare our branch and bound algorithm with alternative approaches, two popular mathematical programming solvers have also been used to solve the instances, whenever possible. Namely, Bonmin 1.1 was run on instances with $p = 0.5, 1.5, 2$, while Cplex 11 was run on instances with $p = 1$, for which $P$ is a quadratic mixed-integer problem and Cplex outperforms Bonmin. (It is worth to point out that, in order to use Bonmin for fractional values of $p$, we had to explicitly input gradient, Hessian of the Lagrangean function, as well as the Jacobian of the constraint matrix. Otherwise, Bonmin fails to solve any such instance).

In our approach, when $p = 1$, we use closed-form expressions for $\lambda^*$ to solve $LRP'$. Algorithm Resource Selection is used in all other cases.

### 6.1 Results and analysis

The results of the experiments are summarized in Tables 1, 2 and 3. Table 1 reports a comparison between our branch and bound and Bonmin, for $p = 0.5, 1.5, 2$. Each row refers to 40 instances with the same pair $(p, q)$ (20 with high and 20 with low cost coefficients). Average CPU times and enumeration nodes for instances solved to optimality within 1 h are reported, along with the number of optimally solved instances out of 40. Some remarks are worth to point out.

- Our branch and bound algorithm was able to solve to optimality all instances within the time limit.
- Bonmin encounters major numerical problems when $p$ is fractional. Actually, for $p = 0.5$ Bonmin solved to optimality within 1 h 24.3% of the instances, and only 1 out of 160 for $p = 1.5$. Its performance slightly improves for $p = 2$, solving 31.2% of the instances. However, for all $p$, the time required by Bonmin to find the optimal solution is always two orders of magnitude larger than our branch and bound. This is probably due to the fact that the computation of lower bounds at each node is much faster in our approach.
Table 1  Comparison with Bonmin. \( t \) average CPU time in seconds (on optimally solved instances), \( n \) average number of enumeration nodes (on optimally solved instances), # number of optimally solved instances out of 40 within the time limit of 1 h

| \( q \) | \( p = 0.5 \) | \( p = 1.5 \) | \( p = 2 \) |
|---|---|---|---|
| \( t \) | \( n \) | # | \( t \) | \( n \) | # | \( t \) | \( n \) | # |
| 100 | 0.18 | 128 | 40 | 159.78 | 11,079 | 12 | 3.28 | 6,730 | 40 | 847 | 62,305 | 1 |
| 200 | 0.66 | 247 | 40 | 116.17 | 900 | 13 | 52.69 | 50,848 | 40 | – | – | 0 |
| 300 | 1.86 | 442 | 40 | 83.43 | 1,234 | 9 | 21.46 | 9,845 | 40 | – | – | 0 |
| 500 | 8.01 | 930 | 40 | 912.04 | 4,988 | 5 | 80.21 | 14,081 | 40 | – | – | 0 |

Table 2  Comparison with Cplex

| \( q \) | All instances | Opt\(^a\) | Non opt\(^b\) |
|---|---|---|---|
| \( B&B \) | \( Cplex \) | \( B&B \) | \( Cplex \) | \( # \) | \( \text{gap} (%) \) |
| \( p = 1 \) | | | |
| 100 | 40 | 5.91 | 9,381 | 680.33 | 2,491,892 | 33 | 0.40 | 956 | 61.01 | 298,219 | 7 | 22 |
| 200 | 40 | 5.53 | 9,010 | 1,209.15 | 2,331,886 | 27 | 1.41 | 1,465 | 58.00 | 169,930 | 13 | 44 |
| 300 | 40 | 11.02 | 9,293 | 1,810.55 | 2,621,890 | 20 | 1.95 | 772 | 21.09 | 38,294 | 20 | 83 |
| 500 | 40 | 77.65 | 76,212 | 1,471.90 | 1,712,327 | 24 | 5.30 | 759 | 53.16 | 66,595 | 16 | 133 |

\(^a\) Only instances solved to optimality by Cplex within the time limit
\(^b\) Only instances not solved to optimality by Cplex within the time limit

# number of instances out of 40, \( t \) average CPU time in seconds, \( n \) average number of enumeration nodes

- The CPU time required to solve an instance tends to increase as \( q \) grows, although there are several exceptions. In any case, Bonmin solves a decreasing number of instances.
- Our approach works best for \( p = 0.5 \) and worst for \( p = 1.5 \). This is because the computation of \( g_i^{-1}(\cdot) \), which is needed in the Resource Selection algorithm, is simpler for \( p = 0.5 \) (requiring the computation of a square) than for \( p = 1.5 \) (requiring the computation of the cubic root of a square).

Table 2 reports a comparison with Cplex for instances with \( p = 1 \) (on which Cplex outperforms Bonmin). Here we present distinct results for: (i) all instances, (ii) only those instances on which Cplex reaches optimality within 1 h, (iii) only...
those instances for which Cplex fails to reach optimality within 1 h. For the two
former cases we report the number of instances as well as average CPU time and enu-
meration nodes. For the third case, we report the number of instances and the average
Cplex current gap \((UB - LB)/LB\) when the time limit is reached. As in Table 1, each
row refers to a set of 40 instances.

- Also for \(p = 1\) our branch and bound algorithm was able to solve to optimality all
instances within the time limit.
- Considering all instances (column “All instances”), again the average CPU time
and number of nodes required by Cplex (where we assumed a CPU time of 3600
seconds for unsolved instances) is roughly two orders larger than the corresponding
branch and bound figure.
- Very similar considerations apply even if we restrict only to those instances on
which Cplex finds the optimal solution (column “Opt”).
- As \(q\) grows, Cplex is able to solve a decreasing number of instances, and for
unsolved instances (column “Non opt”), the gap after 1 h increases dramatically.
Apparently, there is a very high variance in Cplex CPU times when \(q = 500\).
Actually, our approach never required more than 705 seconds on all instances with
\(p = 1\).

Table 3 highlights how the ratio between fixed \((c_i)\) and variable \((b_i)\) cost coeffi-
cients affects the performance of the various solution approaches. Data are aggregated
by different values of \(p\) and \(q\), and refer only to those instances which are solved to
optimality by the respective approach.

- Very often, instances with high variable cost coefficients \((r = H)\) are significantly
harder than those with \(r = L\). This is particularly true with our branch and bound
approach, but it is indeed verified also for the benchmark solvers. In particular,
recall that when \( p = 1 \) and \( r = L \), the quadratic contribution is small with respect to the fixed costs. Hence, in this case the problem is very close to a knapsack problem, for which the benchmark solver Cplex is extremely efficient (clearly this is not the case when \( r = H \)). However, we observe that Bonmin does not solve any instance with \( r = L \) for fractional \( p \).

- The performance gap between the branch and bound and the solvers is less apparent when data are grouped by \( q \). However, we point out that in Table 3, average figures for the solvers only consider optimally solved instances within 1 h, which are between 30 and 53% of the total number of instances for various \( q \).

Table 4 illustrates the performance of the heuristic which computes the first feasible solution in our branch and bound. We report the CPU time and the percentage gap between the heuristic (\( z^H \)) and the optimal (\( z^* \)) values, i.e., gap = \((z^H - z^*)/z^*\). We only show the data grouped by \( p \) and \( r \), since no significant dependency on \( q \) was observed.

- CPU times do not particularly depend on problem class, and they are always small.
- The heuristic gap is strongly affected by the cost coefficients. When \( r = L \), the gap is always quite small (and does not significantly depend on \( p \)). When \( r = H \), the gap can be very large, making the heuristic solution ineffective. This behavior may be partially due to local search settings.
- When \( r = H \), the performance of the heuristic is heavily affected by \( p \), degrading as \( p \) increases. The only case in which running the heuristic has a clear impact on the branch and bound is when \( p = 0.5 \), which also explains why our approach works best in this case (see Table 3). This is probably related to the fact that the term \( x_i^{1.5} \) in the objective function mitigates the effect of the large \( b_i \).

In order to investigate the size of the largest instances which can still be solved by our approach, we devised an additional set of experiments with \( q = 1,500 \). We
generated 20 instances (5 for each value of \( p \)), all of which were solved within 1 h (see average solution times in Table 5). None of these instances were solved by Cplex or Bonmin.

### 7 Conclusions

In this paper we studied the problem of allocating demand to resources when these have fixed and variable costs, and the variable costs are given by a strictly convex function \( xf(x) \), where \( x \) is the resource usage level and \( f(x) \) is a latency function. The problem is in general \( \mathcal{NP} \text{-hard} \). We pointed out some relevant cases in which it is polynomially solvable.

For the general case, we devised a branch-and-bound solution algorithm. A lower bound, based on Lagrangean relaxation, can be computed very efficiently at nodes in the enumeration tree, for any convex objective function. An extensive set of computational experiments on instances with polynomial latency functions shows that we can solve to optimality large instances (up to 1,500 resources) within reasonable CPU time, outperforming commercial solvers. Moreover, we observed that instances with high variable cost coefficients are typically harder than those with low coefficients.

Future research will address more general cases, such as when different resources may have different cost structures.

### Acknowledgments

The authors wish to thank two anonymous referees for their careful comments that greatly helped improving the presentation of the results. The first author has been partially supported by the PRIN grant of the Italian Ministry of Education, 2007ZMZK5T.

### References

1. Agnetis, A., Grande, E., Mirchandani, P.B., Pacifici, A.: Covering a line segment with variable radius discs. Comput. Oper. Res. 36(5), 1423–1436 (2009)
2. Aktürk, S., Atamtürk, A., Gürel, S.: A strong conic quadratic reformulation for machine-job assignment with controllable processing times. Oper. Res. Lett. 37(3), 187–191 (2009)
3. Beckmann, M., McGuire, C.B., Winsten, C.B.: Studies in the Economics of Transportation. Yale University Press, Yale (1956)
4. Braess, D.: Über ein paradoxon aus der verkehrsplanung. Unternehmensforschung 12, 258–268 (1969)
5. Button, K.J., Verhoef, E.T. (eds.): Traffic Congestion and the Environment: Issues of Efficiency and Social Feasibility. Edward Elgar, Chellenham (1998)
6. Ceria, S., Soares, J.: Convex programming for disjunctive convex optimization. Math. Programm. 86, 595–614 (1999)
7. Cole, R., Dodis, Y., Roughgarden, T.: Pricing network edges for heterogeneous selfish users. In: Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing, pp. 521–530 (2003)
8. Dafermos, S.C., Sparrow, F.T.: The traffic assignment problem for a general network. J. Res. Nat. Bureau Stand. Ser. B 73(2), 91–118 (1969)
9. Frangioni, A., Gentile, C.: Perspective cuts for 0–1 mixed integer programs. Math. Program. 106(2), 225–236 (2006)
10. Günlük, O., Linderoth, J.: Perspective relaxation of MINLPs with indicator variables. In: Lodi A., Panconesi A., Rinaldi G. (eds.) Proceedings 13th IPCO Volume 5035 of Lecture Notes in Computer Science, pp. 1–16 (2008)
11. Hiriart-Urruty, J.-B., Lemarchal, C.: Convex Analysis and Minimization Algorithms. Springer, Berlin (1993)
12. Maillé, P., Stier-Moses, N.: Eliciting coordination with Rebates. Transp. Sci. 43(4), 473–492 (2009)
13. Lemarechal, A.R.C.: A geometric study of duality gaps, with applications. Math. Program. Ser. A 90, 399–427 (2001)
14. Pigou, A.C.: The Economics of Welfare. Macmillan and Co, London (1920)
15. Roughgarden, T.: The Price of Anarchy and Selfish Routing. MIT Press, Cambridge (2005)
16. Roughgarden, T., Tardos, E.: How bad is selfish routing? J. ACM 49(2), 236–259 (2002)
17. Smith, M.J.: The existence, uniqueness and stability of traffic equilibria. Trans. Res. Part B Methodol. 13(4), 295–304 (1979)
18. Wardrop, J.G.: Some theoretical aspects of road traffic research. In: Proceedings, Institute of Civil Engineers, vol. 1, pp. 325–378 (1952)