Using Dual Feasible Functions to Construct Fast Lower Bounds for Routing and Location Problems

Daniel Porumbel*, Gilles Goncalves*
* Univ Lille Nord de France, F-59000 Lille, France
UArtois, LGI2A, F-62400, Béthune, France
Contact: daniel.porumbel@univ-artois.fr, gilles.goncalves@univ-artois.fr
tel: +333.21.63.72.78
fax: +333.21.63.71.21

Using Dual Feasible Functions to Construct Fast Lower Bounds for Routing and Location Problems

Daniel Porumbel, Gilles Goncalves

Abstract

In cutting and packing problems, Dual Feasible Functions (DFFs) represent a well established tool for deriving high quality lower bounds in very short times. A well-known DFF lower bounding approach consists of using DFFs to rapidly generate feasible values for the dual variables of a Column Generation (CG) program (*i.e.*, of an extended formulation commonly associated to CG). This paper presents a method for extending this approach to problems such as `Capacitated` p-`Median`, `Distance Constrained General Routing` and related variants (*e.g.*, `Capacitated Arc-Routing`). In contrast to classical DFF bounds for cutting and packing, our extended DFF bounds deal with *non-equal column costs*, *i.e.*, the column costs (primal objective function coefficients) have a more complex structure depending on some sums of distances. The general idea is to use a (reformulated) classical CG model in which a *feasible dual solution* is expressed as a linear combination of *both* DFF *and* non-DFF terms. In fact, one of the proposed approaches (the mixed CG-DFF bound for `Capacitated` p-`Median` in Section 2.2) still requires optimizing a *restricted CG program*, but with fewer variables and easier pricing sub-problems. The most refined bound version is the "*DFF warm-started CG*" from Section 2.2.2: it takes dual constraints generated while solving the above *restricted CG program* are re-uses them to warm-start a full CG phase. In the best cases, this can yield a speed-up between 2 and 3 relative to the pure CG. We present numerical experiments on `Capacitated` p-`Median`, `Capacitated Arc-Routing` (with fixed costs) and `Distance Constrained Arc Routing`; the numerical comparisons with the CG bound concern both the quality *and* the running time.

Keywords: Dual Feasible Functions, Column Generation, Fast Lower Bounds, p-Median, General Routing

1. Introduction

The *explicit or implicit* use of Dual Feasible Functions (DFFs) has a long tradition in optimization—see, chronologically, the work in [15, 19, 28, 11, 5, 12, 4, 2, 9, 6, 26, 3], or the survey [8]. The DFFs are most often used to generate: (i) valid inequalities in certain Integer Linear Programs (*e.g.*, for the knapsack polytope) or (ii) high-quality fast lower bounds (LBs) for cutting and packing problems. Indeed, a well-known DFF LB for these problems is determined by integrating in a Column Generation (CG) program the values returned by a DFF: these values produce a dual solution whose feasibility can be guaranteed without running the CG algorithm. Such guarantees arise from exploiting a similarity between the dual constraints and the main DFF property (see below). The goal of our study is to show that the above DFF approach can be extended to problems outside the cutting and packing field. We use similar CG LPs (Linear Programs), but with different dual constraints (coming from routes or patterns) that require more refined reformulations to integrate DFFs. For instance, while a cutting or packing problem usually requires minimizing a number of patterns of *equal* (column) cost, the proposed approach deals with configurations (clusters or routes) with more complex costs.

Given function $f : [0, 1] \rightarrow [0, 1]$, capacity Q and some quantities $q_1, q_2, \ldots q_n \in [0, Q]$, the *fundamental (D)DFF inequality* is the following:

$$\sum_{i=1}^{n} a_i f\left(\frac{q_i}{Q}\right) \leq 1, \forall\ a_1 \ldots a_n \in \mathbb{Z}_+ \text{ such that } \sum_{i=1}^{n} a_i q_i \leq Q \qquad (\star)$$

If this inequality holds for any input quantities $q_1, q_2, \dots q_n \in [0, Q]$, then f is a general DFF. If this inequality only holds for some given fixed values $\frac{q_i}{Q}$, with $i \in [1..n]$, then f is a *Data-Dependent* DFF (DDFF). Denoting $\mathbf{q} = [q_1 \dots q_n]^\top$, such a data-dependent function is also referred to as a $\mathbf{q}$-DDFF.

This fundamental inequality has strong similarities with a class of dual constraints of the following form: the sum of any selected dual values is at most 1 if the sum of their corresponding weights (see below) is less than or equal to Q. Such constrains often arise in cutting and packing formulations (such as the Gilmore-Gomory model) with capacity Q. More exactly, $(\star)$ is very similar to dual constraints of the form $\mathbf{a}^\top \mathbf{y} \leq 1$ (in dual variables $\mathbf{y}$) associated to patterns (columns) $\mathbf{a} = [a_1\ a_2 \dots a_n]^\top \in \mathbb{Z}_+^n$ with: (a) fixed constant pattern cost (accounting for the right-hand value of 1) and (b) total weight (size, length, supplied quantity) $\mathbf{a}^\top \mathbf{q}$ not exceeding capacity Q. The terms $f\left(\frac{q_i}{Q}\right)$ are integrated in such constraints by providing values for the dual variables $\mathbf{y}$, *i.e.*, the solution obtained via $y_i \leftarrow f(\frac{q_i}{Q})\ \forall i \in [1..n]$ satisfies $\mathbf{a}^\top \mathbf{y} \leq 1$. The above property (a) renders the classical DFF LB approach very well-suited to problems that minimize a *number* of selected patterns, *e.g.*, the number of bins in `Bin-Packing`, the number of rolls in `Cutting-Stock`, the number of independent sets in `Graph Coloring`, etc. A challenging aspect in this work is the need to address more complex costs (of routes or clusters) for which the above DFF integration is less straightforward. For instance, in p-`median` (or `Arc-Routing`), the goal is to minimize a total covering (traversal) distance and the dual constraints have fewer similarities with $(\star)$, *e.g.*, the dual constraints do no longer have 1 in the right-hand side.

The paper is organized as follows. Section 2 presents several DFF-based lower bounds for `Capacitated` $p-$`Median`, as well as the convergent DFF warm-started CG (Section 2.2.2). Section 3 is devoted to a study of (D)DFF applications to `General Routing` problem variants. Numerical experiments are carried out in Section 4, followed by conclusions in the last section. Appendix A presents a case study of the proposed approach for a different `Capacitated` p-`median` variant, showing how the DFFs can be (better) integrated to more problems.

2. DFF Bounds for `Capacitated` p-`Median`

2.1. Problem Introduction and a Basic Bound

The p-`Median` problem is a core topic in customer allocation and distribution system design, and so, it has been tackled with a large variety of algorithms [25], including Column Generation (CG) methods [18, 7, 1, 10, 13]. Besides location and distribution applications, p-`Median` problems can also arise in rather unexpected fields such as cluster analysis (see references in the introduction of [1]), processor scheduling or packet management in networks [14].

Consider a graph $G = (V, E)$ with $V = [1..n]$ and edges $\{i, j\} \in E$ weighted by distance (cost) values $d_{ij} \geq 0$; one can interpret d as a function acting on $\{\{i, j\} :\ i, j \in V\}$ with $d_{ij} = \infty$ if $\{i, j\} \neq E$. There are b_i clients placed in each vertex $i \in V$ (with $b_i = 1$ in classical instances); each of these clients have to be serviced from some median (facility) that can be placed anywhere on the vertices V. More exactly, each client in i requires a supply (weight, quantity) of q_i; a median can supply a *maximum quantity* of Q to a *cluster* of clients (subset of V). The goal is do choose exactly $p \leq n$ clusters and to assign all clients from each cluster to a cluster median so as to minimize the total distance from medians to clients. Remark that G is not oriented, and so, $d_{ji} = d_{ij}\ \forall i, j \in [1..n]$; also, we do *not* impose $d_{ii} = 0$ by default (even if this is often the case in p-`median` instances).

Let us start with an introductory binary formulation: we simply copy the model of problem "P" from [18, §2] and apply the notational translations: $N \to V$ and $x \to z$. The binary decision variables z_{ij} indicate if client $i \in V$ is serviced by a median placed in $j \in V$; z_{jj} is 1 if and only if a median is placed in j The objective function below simply minimizes the sum of the distances from each vertex to its servicing median. The first constraint states that each vertex $i \in V$ has a (unique) client that has to be serviced once (*i.e.*, we actually used $b_i = 1, \forall i \in V$) from one median. The second equality constraints fixes the number of medians to p. The third constraint imposes a limit of Q to the total quantity that can be serviced from one median. The last constraint $z_{ij} \in \{0, 1\}\ \forall i, j \in V$ reinforces the fact that each vertex can accept at maximum one client and at maximum one median.

$$
\begin{array}{ll}
\min \sum_{i\in V} \sum_{j\in V} d_{ij} z_{ij} & \\
\sum_{j\in V} z_{ij} = 1 & ,\forall i \in V \\
\sum_{j\in V} z_{jj} = p & \\
\sum_{i\in V} q_i z_{ij} \leq Q z_{jj} & ,\forall j \in V \\
z_{ij} \in \{0,1\} & \forall i,j \in V
\end{array}
$$

In the rest of the paper, we will actually work on extended formulations for p-`median`. Compared to above program, we introduce two novelties: (i) we allow non-equal numbers of clients (demand multiplicities) $b_i \geq 0$ to be placed at vertices $i \in V$, each client needing a supply of q_i, (ii) we lift the restriction of placing at maximum one median per vertex (such restrictions are not explicitly imposed in all set-covering formulations, *e.g.*, see also problem SCP in [18, §3]). However, property (i) is not relevant in practical instances, because $b_i = 1 \forall i \in V$. The restriction from property (ii) is also not very often needed in practice; when $b_i = 1 \forall i \in V$, practical (Euclidean) instances might not exhibit high-quality solutions with multiple medians per vertex.

2.1.1. A Set-Covering Extended Formulation with an Equality Cardinality Constraint

Using interpretations above, we formulate a first Integer Linear Program (ILP) based on a set-covering extended formulation with a prohibitively large number of variables. This ILP considers a variable for each cluster, *i.e.*, for each subset of V satisfying a knapsack constraint (see below) with capacity Q. The resulting program is :

$$
\begin{array}{ll}
\min \sum d_a x_a & \\
\sum a_i x_a \geq b_i, & \forall i \in V \\
\sum x_a = p & \\
x_a \in \mathbb{Z}_+ & \forall [d_a\ \mathbf{a}]^\top \in A,
\end{array}
\tag{2.1}
$$

where each sum is carried out over all configurations (clusters) $[d_a\ \mathbf{a}]^\top$ of set A, *i.e.*, $\mathbf{a} = [a_1\ a_2 \dots a_n]^\top \in \{0,1\}^n$ is a column and d_a is the objective function coefficient (cluster cost). To reduce clutter, please accept the following notational shorthands: (i) we write $[d_a\ \mathbf{a}]^\top$ to mean $[d_a\ \mathbf{a}^\top]^\top$, equivalent to $[\frac{d_a}{\mathbf{a}}]$; (ii) notation d_a with only one index is a (distance) cost of cluster, while notation $d_{i,j}$ is a distance between two vertices $i,j \in V$. This ILP formulation is constructed based on the following:

The objective function uses decision variables x_a to indicate the number of selections of column $\mathbf{a}$. All sums are carried out over a (prohibitively large) set A with elements $[d_a\ \mathbf{a}]^\top$. Any such element $[d_a\ \mathbf{a}]$ comes from a cluster $V_a = \{i \in V = [1..n] :\ a_i = 1\}$ with cost d_a. This representation does not explicitly indicate the median. However, the best median location for cluster V_a is at vertex $m(\mathbf{a}) = m(V_a) \in V$ such that $\sum_{i\in V_a} d_{i,m(\mathbf{a})} \leq \sum_{i\in V_a} d_{i,\hat{m}},\ \forall \hat{m} \in V$. Using this notation, the cost of the cluster V_a associated to column $\mathbf{a}$ is:

$$
d_a = \sum_{i\in V_a} d_{i,m(\mathbf{a})} = \sum_{i=1}^{n} a_i d_{i,m(\mathbf{a})} \tag{2.2}
$$

Since we only consider the *capacitated version* of the problem, a feasible column $\mathbf{a}$ needs to satisfy $\sum a_i q_i \leq Q$. There are no other restrictions on the composition of clusters: $m(\mathbf{a})$ does not necessarily belong to V_a, but it influences the cluster cost d_a.

the first primal constraint $\sum a_i x_a \geq b_i$ is a set-covering constraint; $b_i \in \mathbb{Z}$ can be seen as the number of clients that require service at vertex $i \in V$ (usually $b_i = 1$);

the second primal constraint $\sum x_a = p$ is an (equality) cardinality constraint imposing the selection of an exact number of p clusters.

This model is very similar to other ILPs arising in CG studies for `Capacitated` p-`Median`, see (10) in [18] or (6)-(8) in [7]. The former model is almost exactly the same as (2.1), up to the following notational equivalences: they index the columns with $k \in [1..m]$, the decision variables are noted x_k and the columns are indicated by vector A_k of cost c_k—this cost c_k is determined by solving n (*i.e.*, $|V|$) binary knapsack problems, *i.e.*, one for each possible median $\hat{m} \in V$. The latter model [7] is slightly different because it encodes the selection of the medians in the columns. As opposed to both above papers, our model does not require the median $m(a)$ to belong to the cluster V_a.

2.1.2. An Introductory Bound using a Modified Model

The above cardinality equality ($\sum x_a = p$) can be replaced with $\sum x_a \leq p$, as also discussed in [7]. Indeed, given a primal ILP solution $\mathbf{x}$ in (2.1) such that $\sum x_a \leq p-1$, one can always derive an equal-quality solution by repeating the following operation: split one of the selected clusters V_a (*i.e.*, with $x_a \neq 0$) in two and keep the same median location for both resulting clusters. This is always possible: as long as $\sum x_a \leq p-1 < n$, there exists a selected cluster V_a such that $|V_a| \geq 2$ that can be split into non-empty disjoint V_a^1 and V_a^2. As such, we can replace $\sum x_a = p$ with $\sum x_a \leq p$; we will use below the equivalent form $\sum -x_a \geq -p$, which is slightly more convenient to have only "$\geq$" inequalities in the primal. The initial set-covering ILP (2.1) leads after linear relaxation to the following CG primal-dual LPs, in which all sums are carried out over all $[d_a\ \mathbf{a}]^\top \in A$.

$$\begin{array}{lll} & \min \sum d_a x_a & \\ \mathbf{y}: & \sum a_i x_a \geq b_i, & \forall i \in [1..n] \\ \mu: & \sum -x_a \geq -p & \\ & x_a \in \mathbb{R}_+ & \forall [d_a\ \mathbf{a}]^\top \in A \end{array} \qquad (2.3a) \qquad \left.\begin{array}{lll} & \max \mathbf{b}^\top \mathbf{y} - p\mu & \\ \mathbf{x}: & \mathbf{a}^\top \mathbf{y} - \mu \leq d_a, & \forall [d_a\ a]^\top \in A \\ & \mathbf{y} \geq \mathbf{0}_n & \\ & \mu \geq 0 & \end{array}\right\} \mathcal{P} \qquad (2.3b)$$

Observe that we indicate in front of each constraint the corresponding dual variable. Given column $[d_a\ \mathbf{a}]^\top \in A$, recall that the (best) median $m(\mathbf{a})$ of $\mathbf{a}$ satisfies $\sum_{i \in V_a} d_{i,m(\mathbf{a})} \leq \sum_{i \in V_a} d_{i,\hat{m}},\ \forall \hat{m} \in V$. Using (2.2), the main constraint of the (dual) polytope $\mathcal{P}$ can be written:

$$\mathbf{a}^\top \mathbf{y} - \mu \leq d_a = \sum_{i=1}^{n} a_i d_{i,m(\mathbf{a})}$$

We will more often make use of this constraint in the form:

$$\sum_{i=1}^{n} a_i (y_i - d_{i,m(\mathbf{a})}) \leq \mu, \qquad (2.4)$$

Using notation $\mathbf{d}^{m(\mathbf{a})} = [d_{1,m(\mathbf{a})} d_{2,m(\mathbf{a})} \dots d_{n,m(\mathbf{a})}]^\top$, it can also be written $\mathbf{a}^\top \left(\mathbf{y} - \mathbf{d}^{m(\mathbf{a})}\right) \leq \mu$.

We now start out with an introductory proposition. It is mainly useful for getting a very general idea on the proposed DFF lower bounding approach.

Proposition 1. *The dual values below yield a dual feasible solution in* (2.3b)*, for any $\alpha \in [0, \mu]$ and for any (D)DFF f (one can use a general DFF f or a* $\mathbf{q}$*-DDFF).*

$$y_i = \alpha f\left(\frac{q_i}{Q}\right) + \min_{j \in [1..n]} d_{i,j} \qquad (2.5)$$

Proof. We need to show that the main (dual) constraint (2.4) holds for any feasible cluster $\mathbf{a}$; this constraint is expanded as:

$$\sum_{i=1}^{n} (a_i y_i - d_{i,m(\mathbf{a})}) = \sum_{i=1}^{n} a_i \left(\alpha f\left(\frac{q_i}{Q}\right) + \min_{j \in [1..n]} d_{i,j} - d_{i,m(\mathbf{a})}\right) \leq \mu,$$

Given that $\min_{j \in [1..n]} d_{i,j} - d_{i,m(\mathbf{a})} \leq 0\ \forall i \in V$, it is enough to show that:

$$\alpha \sum_{i=1}^{n} a_i f\left(\frac{q_i}{Q}\right) \leq \mu,$$

which is true because $\alpha \leq \mu$ and f satisfies the fundamental (D)DFF inequality ($\star$) with regards to quantities $q_i \leq Q,\ \forall i \in [1..n]$ – recall that all feasible clusters $\mathbf{a}$ verify the capacity constraint $\sum_{i=1}^{n} a_i q_i \leq Q$. □

This proposition actually allows one to derive a *class* of valid dual solutions, depending on variables $\alpha \in [0, \mu]$ *and* $\mu \geq 0$. The best LB value is reached by maximizing (under constraint $\alpha \leq \mu$) the dual objective function:

$$\mathbf{b}^\top \mathbf{y} - p\mu = \sum_{i=1}^{n} b_i \left(\alpha f\left(\frac{q_i}{Q}\right) + \min_{j \in [1..n]} d_{i,j}\right) - p\mu = \sum_{i=1}^{n} b_i \min_{j \in [1..n]} d_{i,j} + \alpha \sum_{i=1}^{n} b_i f\left(\frac{q_i}{Q}\right) - p\mu \qquad (2.6)$$

This actually leads to only two cases:

(A) $\sum_{i=1}^{n} b_i f(q_i) - p > 0$**:** the objective value can be made indefinitely large using an arbitrarily large $\alpha = \mu$. In other words, the instance is infeasible because it is impossible to put all requested quantities $\sum_{i=1}^{n} b_i q_i$ in p facilities with capacity Q. DFFs (and **q**-DDFFs) can actually serve as a very fast tool for detecting such situations;

(B) $\sum_{i=1}^{n} b_i f(q_i) - p \leq 0$**:** the best objective value is reached using $\alpha = \mu = 0$.

Both of these cases lead to $\alpha = \mu$. The reason for having introduced two separated variables α and μ will become obvious in the next section. Above proposition only aims at introducing the very general idea of our approach, *i.e.*, we combined DFF terms (like $a_i f\left(\frac{q_i}{Q}\right)$) with non-DFF terms $\left(\text{like } \min_{j\in[1..n]} d_i^j\right)$.

A drawback of the dual feasible solution from Proposition 1 is related to this non-DFF term $\min_{j\in[1..n]} d_i^j$ in (2.5). In most practical instances, d_i^i can be zero and this is not very helpful for obtaining high quality LB values using (2.6). To overcome this issue, the mixed CG-DFF approach from the next section replaces $\min_{j\in[1..n]} d_i^j$ with $\min_{j\in[1..n],j\neq i} d_i^j$. Furthermore, such minimizing terms might not even arise at all in other p-`median` versions: we provide an example in this sense in Appendix A.

2.2. A Mixed $\overline{V_k}$-Reduced CG-DFF Bound and DFF warm-started Full CG

Let us now decompose $V = [1..n]$ into $V = V_k \cup \overline{V_k}$ (small and large quantities), such that

$$V_k = \left\{ i \in V : \; q_i \leq \frac{Q}{k} \right\} \text{ and } \overline{V_k} = \left\{ j \in V : \; q_j > \frac{Q}{k} \right\}. \tag{2.7}$$

The dual variables of V_k will be expressed using (D)DFF terms, while $\overline{V_k}$ will be associated to a ($\overline{V_k}$-reduced) set of *"stand-alone"* variables. We recall that notation m is used (above) as a function $m : 2^V \to V$ such that $m(V_a)$ is the best median location for cluster $V_a \in 2^V$. Please accept now a slight notation abuse and let us use the same symbol m for a function $m : V \to V$ defined as follows: given $i \in V$, $m(i)$ indicates (one of) the closest vertices to i that is *different* from i itself, *i.e.*, $d_i^{m(i)} = \min_{j\in V-\{i\}} d_i^j$. The following convention can avoid any confusion: (i) by writing $m(i)$, we make reference to this latter function $m : V \to V$, and (ii) by writing $m(\{i\})$ or $m(V_a)$ (equivalent to $m(\mathbf{a})$), we make reference to the former function $m : 2^V \to V$.

Theorem 1. *Let us consider a (D)DFF f and a* `Capacitated` p-`median` *instance acting on vertex set $V = [1..n] = V_k \cup \overline{V_k}$, where V_k and $\overline{V_k}$ are defined by above formula* (2.7) *from weights q_i (with $i \in V$), capacity Q and a fixed $k \in \mathbb{Z}^+$. A dual solution $(\mathbf{y}, \mu)$ constructed from some given values of $\alpha, \mu \geq 0$ and $\overline{y}_j \geq 0, \forall j \in \overline{V_k}$ via:*

$$y_j = \overline{y}_j \qquad \forall j \in \overline{V_k} \tag{2.8}$$

$$y_i = \alpha f\left(\frac{q_i}{Q}\right) + \frac{k}{k+1}\, d_{i,m(i)} \tag{2.9}$$

$$= \alpha f\left(\frac{q_i}{Q}\right) + \frac{k}{k+1} \min_{\substack{j\in[1..n]\\ i\neq j}} d_{i,j} \qquad \forall i \in V_k$$

is (dual) feasible in (2.3b) *if the following holds:*

$$\alpha \leq \mu \tag{2.10}$$

$$\sum_{i\in U} \alpha f\left(\frac{q_i}{Q}\right) + \frac{k}{k+1} \sum_{i\in U} d_{i,m(i)} \leq \mu + \sum_{i\in U} d_{i,m(U)} \quad , \forall U \subseteq V_k \text{ with } |U| \leq k \tag{2.11}$$

$$\sum_{j\in\overline{V_k}} \overline{a}_j \overline{y}_j \leq \sum_{j\in\overline{V_k}} \overline{a}_j d_{j,m(\overline{\mathbf{a}})} \quad , \forall [d_{\overline{a}}\ \overline{\mathbf{a}}]^\top \in \overline{A}, \tag{2.12}$$

where $\overline{A}$ is the set of columns associated to clusters completely contained in $\overline{V_k}$:

$$\overline{A} = \{[d_{\overline{a}}\ \overline{\mathbf{a}}]^T \in A : \; \overline{a}_i = 0, \forall i \notin \overline{V_k}\}.$$

Proof. Consider any feasible $[d_a\ \mathbf{a}] \in A$. We need to show that the main dual constraint (2.4) holds; this constraint can be written:

$$\sum_{i\in V_k} a_i y_i + \sum_{j\in \overline{V_k}} a_j y_j \le \mu + \sum_{i\in V_k} a_i d_{i,m(\mathbf{a})} + \sum_{j\in \overline{V_k}} a_j d_{j,m(\mathbf{a})}$$

Replacing (2.8)-(2.9), this further reduces to:

$$\alpha \sum_{i\in V_k} a_i f\left(\frac{q_i}{Q}\right) + \sum_{i\in V_k} a_i \frac{k}{k+1} d_{i,m(i)} + \sum_{j\in \overline{V_k}} a_j \overline{y}_j \le \mu + \sum_{i\in V_k} a_i d_{i,m(\mathbf{a})} + \sum_{j\in \overline{V_k}} a_j d_{j,m(\mathbf{a})}$$

We now reduce the last sums on both sides. Consider cluster $\overline{\mathbf{a}}$ obtained from $\mathbf{a}$ by setting $\overline{a}_j = a_j, \forall j \in \overline{V_k}$ and $\overline{a}_i = 0, \forall i \in V_k$, *i.e.*, we reduce $\mathbf{a}$ to elements from $\overline{V_k}$ and we obtain $\overline{\mathbf{a}}$. Using $\sum_{j\in\overline{V_k}} \overline{a}_j d_{j,m(\overline{\mathbf{a}})} \le \sum_{j\in\overline{V_k}} \overline{a}_j d_{j,m(\mathbf{a})}$, (2.12) guarantees that $\sum_{j\in\overline{V_k}} \overline{a}_j \overline{y}_j \le \sum_{j\in\overline{V_k}} \overline{a}_j d^{m(\mathbf{a})}$. Since $\overline{a}_j = a_j$ over $j \in \overline{V_k}$, this is equivalent to $\sum_{j\in\overline{V_k}} a_j \overline{y}_j \le \sum_{j\in\overline{V_k}} a_j d^{m(\mathbf{a})}$, and so, it is enough to prove that the following holds:

$$\alpha \sum_{i\in V_k} a_i f\left(\frac{q_i}{Q}\right) + \sum_{i\in V_k} a_i \frac{k}{k+1} d_{i,m(i)} \le \mu + \sum_{i\in V_k} a_i d_{i,m(\mathbf{a})}$$

Denoting $V_a^k = \{i \in V_k :\ a_i = 1\}$, this can be written:

$$\alpha \sum_{i\in V_a^k} f\left(\frac{q_i}{Q}\right) + \sum_{i\in V_a^k} \frac{k}{k+1} d_{i,m(i)} \le \mu + \sum_{i\in V_a^k} d_{i,m(\mathbf{a})}$$

We show that the above constraint is true by distinguishing two cases:

(a) $|V_a^k| \le k$**.** The constraint becomes a particular case of (2.11) with $U = V_a^k$ (all terms outside V_k are already reduced);

(b) $|V_a^k| \ge k+1$**.** We reduce the first term on both sides. This is rather straightforward using $\alpha \le \mu$ (given by (2.10)) and the feasibility condition $\sum_{i\in V} q_i \le Q$, coupled with the fundamental DFF inequality ($\star$), as in Proposition 1. As such, the constraint to prove reduces to:

$$\sum_{i\in V_a^k} \frac{k}{k+1} d_{i,m(i)} \le \sum_{i\in V_a^k} d_{i,m(\mathbf{a})}$$

Let us focus on the value of $m(a)$. If $m(a) \notin V_a^k$, the inequality $d_{i,m(i)} \le d_{i,m(\mathbf{a})}$ holds for all $i \in V_a^k$ (recall $d_{i,m(i)} = \min_{j\in V-\{i\}} d_{i,j} \le d_{i,m(\mathbf{a})}$) and the above constraint simply results by summing this inequality over all $i \in V_a^k$. We still have to address the case $m(\mathbf{a}) = i_m$ for some $i_m \in V_a^k$. By separating the term corresponding to i_m in both sides of above constraint, we have to prove:

$$\frac{k}{k+1} d_{i_m,m(i_m)} + \sum_{i\in V_a^k - \{i_m\}} \frac{k}{k+1} d_{i,m(i)} \le d_{i_m,i_m} + \sum_{i\in V_a^k - \{i_m\}} d_{i,i_m}$$

Since $0 \le d_{i_m,i_m}$ (we only consider non-negative service costs), we reduce the first term in the right-hand side. Using $\frac{k}{k+1} d_{i,m(i)} \le \frac{k}{k+1} d_{i,i_m}, \forall i \in V_a^k - \{i_m\}$, it is enough to prove:

$$\frac{k}{k+1} d_{i_m,m(i_m)} \le \sum_{i\in V_a^k - \{i_m\}} \frac{1}{k+1} d_{i,i_m}, \text{ or}$$

$$k \cdot d_{i_m,m(i_m)} \le \sum_{i\in V_a^k - \{i_m\}} d_{i,i_m}.$$

This is true using $d_{i_m,m(i_m)} \le d_{i_m,i} = d_{i,i_m}, \forall i \in V_a^k - \{i_m\}$ and the fact that the sum in the right-hand side has exactly k terms—we are in case **(b)** corresponding to $|V_a^k| \ge k+1$, or $|V_a^k - \{i_m\}| \ge k$.

□

Using this theorem, we can now formulate a mixed CG-DFF model. The objective function of the initial dual LP (2.3b) is expanded by simply replacing (2.8)-(2.9):

$$\mathbf{b}^\top \mathbf{y} - p\mu = \sum_{i \in V_k} b_i \frac{k}{k+1} d_{i,m(i)} + \left(\sum_{i \in V_k} b_i f\left(\frac{q_i}{Q}\right) \right) \alpha + \sum_{j \in \overline{V_k}} b_j \overline{y}_j - p\mu \tag{2.13}$$

This leads us to a restricted ($\overline{V_k}$-reduced) LP with $|\overline{V_k}| + 2$ variables ($\alpha \geq 0,\ \mu \geq 0$ and $\overline{y}_j \geq 0,\ \forall j \in \overline{V_k}$) defined by (dual) objective function above and by the the constraints (2.10)-(2.12) from Theorem 1. We describe below how to solve this $\overline{V_k}$-reduced model by CG: the resulting restricted pricing is (considerably) simplified compared to the one for the initial dual LP (2.3b).

2.2.1. Simplified Pricing in the $\overline{V_k}$-Reduced CG-DFF Model

We now present the pricing algorithms for the constraints (2.10)-(2.12) of the above CG-DFF model; we pick out the key claims in boldface.

The pricing algorithm for constraints (2.11) **is very fast, as the corresponding sub-problem is not NP-hard**. We consider every possible median location $m(U)$ from (2.11) as a decision variable $\hat{m} \in V$. The most negative reduced cost of constraints (2.11) is determined by solving sub-problem below for the current values of the dual variables α and μ (observe other variables $\overline{y}_j \geq 0,\ \forall j \in \overline{V_k}$ do not arise in (2.11)).

$$\min \left\{ \mu + \sum_{i \in U} \left(d_{i,\hat{m}} - \alpha f\left(\frac{q_i}{Q}\right) - \frac{k}{k+1} d_{i,m(i)} \right) :\ U \subseteq V_k,\ \hat{m} \in V,\ |U| \leq k \right\}$$

The pricing algorithm solves this sub-problem as follows: for each $\hat{m} \in V$, it sorts the $|V_k|$ values $\left(d_{i,\hat{m}} - \alpha f\left(\frac{q_i}{Q}\right) - \frac{k}{k+1} d_{i,m(i)} \right)$ corresponding to all $i \in V_k$ and it selects the lowest (only negative and at most k) values of this form. The lowest sum of such values (*i.e.*, for the best $\hat{m}$) is the most negative reduced cost; the corresponding (maximum k) selected values indicate the best U and the new column. The total complexity is $O(n|V_k|\log(|V_k|))$, *i.e.*, this is the asymptotic running time of n sorting algorithms.

The pricing for (2.12) **represents a $|\overline{V_k}|$-reduced variant of the initial pricing** of (2.3b). Equation (2.12) makes use of a median location $m(\overline{\mathbf{a}})$ that can be seen as a decision variable ($\hat{m}$ below) in the sub-problem. After few transformations, the pricing sub-problem can be formulated as:

$$\min \left\{ \sum_{j \in \overline{U}} d_{j,\hat{m}} - \overline{y}_j :\ \overline{U} \subseteq \overline{V_k},\ \hat{m} \in V,\ \sum_{j \in \overline{U}} q_j \leq Q \right\}$$

The pricing algorithm needs to try all n possible median locations $\hat{m}$; for each one, it needs to solve a binary knapsack problem with $|\overline{V_k}|$ items. Solving n times the knapsack problem is a common approach for p-`median` pricing, already used in other algorithms from the literature [18]. However, the main advantage of our $|\overline{V_k}|$-reduced pricing is the reduction of the number of knapsack items from n to $|\overline{V_k}|$. Such reductions are more important than the number n of knapsack problems, because the latter factor n only leads to a linear coefficient n in the total pricing complexity. As long as $|\overline{V_k}|$ can be considered bounded (compared to n), solving these n knapsack problems is not NP-hard. Recall from (2.7) that $q_j > \frac{Q}{k}\ \forall j \in \overline{V_k}$, and so, the maximum number of selected items is $k-1$; such knapsack problem can be solved in $O(|\overline{V_k}|^{k-1})$. To construct a polynomial bound with this mixed CG-DFF approach, it is enough to keep k (and implicitly $\overline{V_k}$) at sufficiently low values, *e.g.*, if the separation (CG pricing) sub-problem is polynomial, so is the main problem (using the equivalence between separation and optimization).

Larger $\overline{V_k}$ sets can be useful in a slightly different manner: one could use k as a parameter controlling a trade-off between the quality of the CG-DFF bound and its running time. As k becomes larger, $\overline{V_k}$ converges to V and the CG-DFF bound converges towards the CG bound (the optimum of LP (2.3b)). However, for a tighter convergence, the following version of Theorem 1 could be more useful.

Corollary 1. *(large $\overline{V_k}$ version for Theorem 1) The same argumentation from Theorem 1 holds if one updates constraints* (2.11)-(2.12) *as follows: (a) in* (2.11)*, replace μ in with $\left(\sum_{i\in U} f\left(\frac{q_i}{Q}\right)\right)\mu$, and (b) in* (2.12)*, insert term $\left(\sum_{j\in\overline{V_k}} \overline{a}_j f\left(\frac{q_j}{Q}\right)\right)\mu$.*

Above replacements would make constraint (2.11) stronger (which could decrease the final DFF bound) and constraint (2.12) weaker (which could increase the final DFF bound). As such, the bound version from this corollary could be more useful when $\overline{V_k}$ (and implicitly k) is rather large and the latter constraint is more important. When $\overline{V_k}$ is close to V, it approximates very well the initial program (2.3b). However, we only use Theorem 1 in the rest of the paper, as our study is rather focused on fast bounds.

2.2.2. DFF Warm-started CG

As hinted above, the pricing for (2.12) is a $|\overline{V_k}|$-reduced variant of the pricing for the initial dual program (2.3b). The CG-DFF approach can be used to warm-start a full CG algorithm, resulting in a hybrid DFF-CG method of two stages:

1. Compute the above CG-DFF bound by iteratively solving the $|\overline{V_k}|$-reduced pricing problems described in Section 2.2.1. By pricing (2.12) in this stage, we generate clusters with all selected vertices in $\overline{V_k}$, *i.e.*, we generate elements from $\overline{A} \subset A$.

2. Apply the classical CG for the initial model (2.3a)-(2.3b), but in the beginning insert the columns corresponding to the clusters from $\overline{A} \subset A$ generated in Stage 1.

Such DFF warm-starting produces the following effect: the new CG method starts out with a number of columns using only elements from $\overline{V_k} = \left\{j \in V :\ q_j > \frac{Q}{k}\right\}$ of larger weight (supply), see also (2.7). In the best cases (usually associated to large sets $\overline{V_k}$), the columns associated to $\overline{A} \subset A$ might constitute a considerable proportion (even more than half) of the total number of columns required to fully converge. We will also discuss in Section 4.1.2 a slight stabilization effect. However, the general CG "trajectory" is always changed: the DFF warm-started version generates columns in a completely different order. When $\overline{V_k}$ is smaller, Stage 1 generates fewer constraints with a more limited impact; however, the trajectory change in itself leads to a slight variation in the total number of iterations. We propose to apply the new method on large $\overline{V_k}$ values ($|\overline{V_k}| = \frac{3}{4}|V|$), so as to exploit the fact that a larger number of columns can be generated in the first stage.

3. DFF-based Bounds for `Distance Constrained General Routing`

3.1. Problem Definition and Column Generation Model

`General Routing` is a generalization of `Arc Routing` and `Vehicle Routing` in which service can be required on *both edges and vertices*. The problem was first introduced in the 1970s [20] and many different variations have been presented over the time. In this paper, we take interest in a version that considers *two route feasibility constraints*: (classical) capacity constraints and distance (service time) constraints (see an example in [21]). The former constraints state that the total quantity (supply) delivered to clients (by a single route) needs to stay below a *vehicle capacity* Q. The latter constraints state that a single route can provide service over a *maximum distance* of D. To illustrate this latter *distance constraint*, we can refer to the formulation [21, §2], *i.e.*, each edge service requires a pick-up time (depending on the number of houses on a road, which is proportional to the edge length) and the distance constraint simply comes from a (legislative) limit on the daily (shift) working time.

We now formalize our `Distance Constrained General Routing` variant. Consider a graph $G = (V, E)$, with $V = \{1, 2, \ldots, n\}$ and edge set $E = \{e_{n+1}, e_{n+2}, \ldots e_{n+m}\} \in V^2$; to lighten the text, we will also refer to edge e_j only using its index j and say $j \in E$ whenever $j \in \{n+1, n+2, \ldots n+m\} = [n+1..n+m]$. For each vertex $i \in V$, there is a quantity (supply) demand q_i that needs to be serviced (delivered) b_i times (b_i is the covering demand of i). Similarly, each edge $j \in E$ has a quantity demand q_j and a covering demand b_j; its length (distance or time) is denoted by d_j. We are also given a fleet of p vehicles of capacity Q that can each provide service over a maximum total distance of D; the use of a

vehicle comes with a fixed cost F. A feasible route is a path in G that: (i) starts and ends at a special vertex $v_0 \in V$ (the depot), (ii) delivers a maximum quantity of Q, (iii) provides service over a maximum total distance of D.

The objective is to find at maximum p feasible routes with a *minimum* total cost that deliver all requested demands. The cost d_a of a route $\mathbf{a}$ is calculated as the sum of a fixed cost F (for using the vehicle) and a traversal cost (a sum over all *traversed* edges). CG models are often applied to such problems; we present below the primal-dual LPs used for integrating DFFs.

$$\begin{array}{lll} & \min \sum_{[d_a\ a]^\top \in A} d_a x_a & \\ \mathbf{y}: & \sum_{[d_a\ a]^\top \in A} a_\ell x_a & \geq b_\ell,\ \forall \ell \in [1..n+m] \\ \mu: & \sum_{[d_a\ a]^\top \in A} -x_a & \geq -p \\ & x_a \geq 0 & \forall \begin{bmatrix} d_a \\ \mathbf{a} \end{bmatrix} \in A \end{array} \quad (3.1a) \qquad \left.\begin{array}{lll} & \max \mathbf{b}^\top \mathbf{y} - p\mu & \\ \mathbf{x}: & \mathbf{a}^\top \mathbf{y} - \mu \leq d_a, & \forall \begin{bmatrix} d_a \\ \mathbf{a} \end{bmatrix} \in A \\ & \mathbf{y} \geq \mathbf{0}_{n+m} & \\ & \mu \geq 0 & \end{array}\right\} \mathcal{P} \quad (3.1b)$$

We first provide below the interpretation for the primal objective, constraints and notations, followed by their dual counterparts.

the primal objective function uses decision variables x_a that indicate the number of times that route $\mathbf{a}$ is used; x_a is integer in the ILP version of (3.1a) and fractional after relaxation. Set A contains columns $[d_a\ \mathbf{a}]^\top$ that encode feasible routes $\mathbf{a}$ of cost d_a. Vector $\mathbf{a} = [a_1\ a_2 \dots a_n,\ a_{n+1} \dots a_{n+m}]^T \in \mathbb{Z}^{n+m}$ contains n positions corresponding to vertices and m corresponding to edges; a_ℓ is the number of times (vertex or edge) ℓ is *serviced*, $\forall \ell \in [1..n+m]$. The cost of using route $\mathbf{a}$ is:

$$d_a = F + \sum_{j=n+1}^{n+m} a_j^{\mathtt{tr}} d_j, \quad (3.2)$$

where $a_j^{\mathtt{tr}}$ is the number of *traversals* of edge j. If $a_j^{\mathtt{tr}} > a_j$, then edge j is *deadheaded* (traversed without service) $a_j^{\mathtt{tr}} - a_j$ times. Dead-heading is well-known phenomenon in `Arc-Routing`: a route that traverses edge j can either choose to service j or to dead-head it (traverse it without service). The necessity of considering deadheaded edges is lifted in Section 3.3. Besides natural contiguity conditions, a feasible route $\mathbf{a}$ needs to verify the following:

$$\begin{array}{ll} \sum_{i=1}^{n} a_i q_i + \sum_{j=n+1}^{n+m} a_j q_j \leq Q & \text{(route capacity constraint)} \\ \sum_{j=n+1}^{n+m} a_j d_j \leq D & \text{(route distance constraint)} \end{array}$$

the first (set covering) primal constraint in (3.1a) requires each edge or vertex ℓ to be serviced b_ℓ times. Most routing instances use $\mathbf{b} = \mathbf{1}_{n+m}$, each service is to be provided only once. By setting $b_j = 0, \forall j \in E$, one obtains a `Vehicle Routing` problem; $b_i = 0,\ \forall i \in V$ leads to an `Arc-Routing` problem;

the second (fleet size) primal constraint states (using sign inversion) that one can use at maximum p vehicles (routes);

the dual objective function has $n + m + 1$ variables (see below) with coefficients taken from the right-hand sides of the primal constraints (above);

the dual constraints $\mathbf{a}^\top \mathbf{y} - \mu \leq d_a$ set the boundaries of the dual polytope $\mathcal{P}$ over the non-negative orthant. They are constructed from the primal LP using: (i) a variable μ for the primal constraint on the fleet size, (ii) vector $\mathbf{y}$ of variables for to the $n+m$ primal set-covering constraints, and (iii) a right-hand side coefficient d_a from the primal objective function.

3.2. Feasible Dual Solutions on the General Case

Theorem 2. *We consider a DFF (or a* $\mathbf{q}$*-DDFF)* f_q *and a DFF (or a* $\mathbf{d}$*-DDFF)* f_d*. For any* $\alpha, \beta, \mu \in \mathbb{R}_+$ *such that*

$$\alpha + \beta - \mu \leq F, \quad (3.3)$$

the formulas below determine a dual solution $(\mathbf{y}, \mu)$ *that is feasible in* (3.1b)*.*

1. $y_i = \alpha f_q\left(\frac{q_i}{Q}\right)$, *for all vertices* $i \in [1..n]$

2. $y_j = d_j + \alpha f_q\left(\frac{q_j}{Q}\right) + \beta f_d\left(\frac{d_j}{D}\right)$, *for all edges* $j \in [n+1..n+m]$

Proof. We need to prove that the main (dual) constraint of the dual polytope $\mathcal{P}$ in (3.1b) (*i.e.*, $\mathbf{a}^\top \mathbf{y} - \mu \leq d_a$) is verified by such solution $(\mathbf{y}, \mu)$ for *any* route $\mathbf{a}$ with cost d_a. Given the route cost equation (3.2), the main dual constraint can be written:

$$\sum_{i=1}^{n} a_i y_i + \sum_{j=n+1}^{n+m} a_j y_j - \mu \leq F + \sum_{j=n+1}^{n+m} a_j^{\mathtt{tr}} d_j \tag{3.4}$$

In the right-hand side sum, the cost (naturally) depends on the number $a_j^{\mathtt{tr}}$ of traversals of edge $j \in [n+1..n+m]$, and not on the number of services a_j provided to j. If $a_j^{\mathtt{tr}} - a_j > 0$, than edge j is dead-headed $a_j^{\mathtt{tr}} - a_j$ times. Any use of a dead-headed edge increases the right-hand side of above constraint without modifying the left-hand side, *i.e.*, the constraint is weakened by using more dead-heading. Conversely, routes with less (or zero) dead-heading generally correspond to stronger constraints (3.4). For instance, if a route performs no dead-heading at all, the right-hand side of (3.4) has no terms with no match in the left-hand side, *i.e.*, any right-hand term $a_j^{\mathtt{tr}} d_j > 0$ is matched (balanced) by a left-hand term $a_j^{\mathtt{tr}} y_j = a_j y_j$. The necessity of considering dead-heading is completely lifted in Section 3.3, where $a_j^{\mathtt{tr}} = a_j$ is always satisfied. However, a sufficient condition for ensuring above constraint (3.4) in general is:

$$\sum_{i=1}^{n} a_i y_i + \sum_{j=n+1}^{n+m} a_j y_j - \mu \leq F + \sum_{j=n+1}^{n+m} a_j d_j \tag{3.5}$$

Let us now write $\mathbf{y}$ in terms of f_q, f_d, α and β, using the formulas from the hypothesis.

$$\sum_{i=1}^{n} a_i \alpha f_q\left(\frac{q_i}{Q}\right) + \sum_{j=n+1}^{n+m} a_j \left(d_j + \alpha f_q\left(\frac{q_j}{Q}\right) + \beta f_d\left(\frac{d_j}{D}\right)\right) - \mu \leq F + \sum_{j=n+1}^{n+m} a_j d_j$$

After elementary simplifications, we need to prove that the constraint below holds.

$$\alpha \sum_{\ell=1}^{m+n} a_\ell f_q\left(\frac{q_\ell}{Q}\right) + \beta \sum_{j=n+1}^{n+m} a_j f_d\left(\frac{d_j}{D}\right) - \mu \leq F$$

Since f_q satisfies by definition the fundamental (D)DFF inequality ($\star$) with regard to $\mathbf{q}$ and Q (recall the route capacity constraint $\sum_{\ell=1}^{n+m} a_\ell q_\ell \leq Q$), we establish that $\sum_{\ell=1}^{m+n} a_\ell f_q\left(\frac{q_\ell}{Q}\right) \leq 1$. Similarly, f_d satisfies the fundamental (D)DFF inequality ($\star$) with regard to $\mathbf{d}$ and D, and so, using the route distance constraint, we also state that $\sum_{j=n+1}^{n+m} a_j f_d\left(\frac{d_j}{D}\right) \leq 1$. Replacing these sums with 1 in the above inequality, we only need to prove $\alpha 1 + \beta 1 - \mu \leq F$, which is true from the hypothesis condition (3.3). □

The best (Lower Bound) LB resulting from this theorem is determined by maximizing objective function below (re-written in terms of variables α, β, μ)

$$\mathbf{b}^\top \mathbf{y} - p\mu = \sum_{j=n+1}^{n+m} b_j d_j + \left(\sum_{\ell=1}^{n+m} b_\ell f_q\left(\frac{q_\ell}{Q}\right)\right)\alpha + \left(\sum_{j=n+1}^{n+m} b_j f_d\left(\frac{d_j}{D}\right)\right)\beta - p\mu \tag{3.6}$$

under constraints $\alpha, \beta, \mu \geq 0$ and $\alpha + \beta - \mu \leq F$. This leads to a very small-size LP, *i.e.*, the feasible area is a 3-dimensional polyhedron with four constraints. Very elementary LP arguments show that there are only two cases (analogously to Section 2.1.2): either the LP is unbounded, or the optimum is at a vertex of the above 3-dimensional polyhedron. In former case, the objective function can be arbitrarily large and the instance is infeasible (the fleet size is not large enough to perform all requested service). The latter case leads to $\mu = 0$, because the above polyhedron has *no* vertex with $\mu > 0$ (such point could only belong to maximum two of the four constraints).

3.2.1. A mixed CG-DFF lower bound

If F is very low (*e.g.*, $F = 0$), the constraint $\alpha+\beta-\mu \leq F$ imposes very strong limits on α and β (*e.g.*, if $\mu = 0$ as above, this can even lead to $\alpha = \beta = \mu = F = 0$), which could degrade the resulting LB value. To tackle such situations, one can actually use, as in Section 2.2, a mixed CG-DFF model that can be generally described as follows. We put aside a set of variables $\overline{Y} \subseteq [n+1..n+m]$ that are not expanded as (D)DFF terms. Formally, $y_{\bar{j}}, \forall \bar{j} \in \overline{Y}$ are *"stand-alone"* variables and only y_i with $i \in [1..n+m] - \overline{Y}$ are expanded as (D)DFF terms. Following the reasoning from Theorem 2, one develops (3.4) and the condition $\alpha + \beta - \mu \leq F$ reduces to:

$$\alpha + \beta - \mu + \sum_{\bar{j} \in \overline{Y}} a_{\bar{j}} y_{\bar{j}} \leq F + \sum_{\bar{j} \in \overline{Y}} a_{\bar{j}}^{\mathtt{tr}} d_{\bar{j}} \quad \forall \begin{bmatrix} d_a \\ \mathbf{a} \end{bmatrix} \in A \tag{3.7}$$

This mixed ($\overline{Y}$-restricted) CG-DFF model requires solving a dual LP with $|\overline{Y}|+3$ variables (α, β, μ and $y_{\bar{j}}$ $\forall \bar{j} \in \overline{Y}$) and dual constraints (3.7) above. These constraints (3.7) can be generated one by one by iteratively solving a reduced pricing sub-problem (as in Section 2.2.1). At each CG iteration, this pricing sub-problem requires constructing a route $\mathbf{a}$ that minimizes $F + \sum_{\bar{j} \in \overline{Y}} a_{\bar{j}}^{\mathtt{tr}} d_{\bar{j}} - \alpha - \beta + \mu - \sum_{\bar{j} \in \overline{Y}} a_{\bar{j}} y_{\bar{j}}$; observe that route $\mathbf{a}$ is here only evaluated with regards to the edges $\overline{Y}$, *i.e.*, it is a reduced-size sub-problem. In fact, if $\overline{Y}$ is sufficiently small, the constraints (3.7) could even be enumerated. However, the larger the set $\overline{Y}$, the closer one gets to the initial CG model and to the pure CG bound.

3.3. `Distance-Constrained Arc Routing`

The proof of Theorem 2 uses a reduction of inequality (3.4) to (3.5), *i.e.*, each $a_j^{\mathtt{tr}}$ term is replaced by a_j. Such reductions lead to ignoring deadheading costs, making (possibly large) approximations that could degrade the quality of the bound. This drawback does not necessarily arise in all `Distance-Constrained General Routing` variants. We show this on `Distance-Constrained Arc-Routing`, a problem simply obtained from `Arc-Routing` by replacing the capacity constraint with a distance (tour length) constraint.

3.3.1. Problem Definition

The problem is formally defined from `Distance-Constrained General Routing` as follows:

- no demand on vertices (`Arc-Routing` property), *i.e.*, $b_i = y_i = 0 \forall i \in [1..n]$ in the primal-dual LPs (3.1a)-(3.1b);
- feasible routes have no capacity constraint (only the distances matter), *i.e.*, use $Q = \infty$ and f_q is no longer needed (one could use $f_q = 0$ in Theorem 2);
- the distance constraint acts on travelled edges and not only on serviced edges, *i.e.*, route constraint $\sum_{j=n+1}^{n+m} a_j d_j \leq D$ is replaced by $\sum_{j=n+1}^{n+m} a_j^{\mathtt{tr}} d_j \leq D$. This is a natural interpretation: we limit the total route duration (distance) and not only the service duration, as we did in Section 3.2.

3.3.2. A pure DFF lower bound

As in Theorem 2, one can express the remaining variables y_j (with $j \in [n+1..n+m]$) as DFF terms. We focus on the main dual constraint (3.4) and we show that routes with non-zero deadheading are dominated, *i.e.*, any route involving some deadheading leads to a weaker dual constraint than some route with no deadheading. Consider a dual constraint (3.4) generated by a route $\mathbf{a}$ that deadheads edge j, *i.e.*, $a_j^{\mathtt{tr}} > a_j$. By replacing $a_j \leftarrow a_j^{\mathtt{tr}}$, the route remains feasible because no route feasibility constraint can be violated by increasing the number of services provided to edge j. Indeed, our model places *no* upper bound on the value of a_j (we did not apply any cycle-elimination in our model, *e.g.*, we allow $a_j > b_j$ even if this can degrade the final bound quality). As such, this replacement renders dual constraint (3.4) even tighter: its right-hand side is unchanged, but the initial left-hand term $a_j y_j$ is replaced by $a_j^{\mathtt{tr}} y_j > a_j y_j$. The tightest dual constraints correspond to routes with no dead-heading (with $a_j = a_j^{\mathtt{tr}}$, $\forall j \in [n+1..n+m]$), and so, (3.4) becomes

$$\sum_{j=n+1}^{n+m} a_j y_j - \mu \leq F + \sum_{j=n+1}^{n+m} a_j d_j, \quad \forall \begin{bmatrix} d_a \\ \mathbf{a} \end{bmatrix} \in A \tag{3.8}$$

Setting $y_j = d_j + \beta f_d\left(\frac{d_j}{D}\right)$, this leads (using $f_q = 0$ in Theorem 2) to: $\beta \sum_{j=n+1}^{n+m} a_j f_d\left(\frac{d_j}{D}\right) - \mu \leq F$; condition (3.3) reduces to $\beta - \mu \leq F$.

3.3.3. A mixed CG-DFF lower bound

Using a similar modelling as in Section 3.2.1, we separate a set of "*stand alone*" variables $y_{\bar{j}}$ (with $\bar{j} \in \overline{Y}$) that are not expressed as DFF values; (3.7) reduces to

$$\beta - \mu + \sum_{\bar{j} \in \overline{Y}} a_{\bar{j}} y_{\bar{j}} \leq F + \sum_{\bar{j} \in \overline{Y}} a_{\bar{j}} d_{\bar{j}} \quad \forall \begin{bmatrix} d_a \\ \mathbf{a} \end{bmatrix} \in A \tag{3.9}$$

The strongest mixed CG-DFF bound of this form can be obtained by maximizing a reduced CG LP: express dual objective function $\mathbf{b}^\top \mathbf{y} - p\mu$ as a linear function of dual variables $\beta, \mu \geq 0$ and $y_{\bar{j}} \geq 0$ (with $\bar{j} \in \overline{Y}$) and maximize it under constraints (3.9). These constraints can be generated one by one by solving a ($\overline{Y}$-reduced) pricing sub-problem as in Section 3.2.1.

4. Numerical Evaluation: `Capacitated` *p*-`Median` and two `Capacitated Arc Routing` Variants

The main goal of this evaluation is to compare the pure CG bound with the proposed DFF bounds, both in terms of quality and running time. We first present below the functions used to calculate these DFF bounds. Section 4.1 is devoted to *p*-`median` experiments (implementing Section 2.2) and Section 4.2 presents two `General-Routing` variants, *i.e.*, `Arc-Routing` with Fixed Costs (implementing Section 3.2) and `Distance-Constrained Arc-Routing` (implementing Section 3.3.2).

The most straightforward DFF is $f_{\text{iden}}(x) = x, \forall x \in [0,1]$; this trivial DFF is hereafter referred to as the identity. The first (non-trivial) DFF used in this work is the Fekete-Schepers function f_0^λ from [11] (see also [8, § 4.1]). Using parameter $\lambda \in [0, 0.5)$, it can be expressed as:

$$f_0^\lambda(x) = \begin{cases} 0 & \text{if } x \leq \lambda \\ x & \text{if } \lambda < x < 1 - \lambda \\ 1 & \text{if } x \geq 1 - \lambda \end{cases} \tag{4.1}$$

When all involved quantities q_i are less than $\frac{Q}{2}$, we only use $\lambda = 0$; in this case f_0^λ is reduced to the identity f_{iden}. Function f_0^λ can be more effective than f_{iden} *only when* it is useful to valuate large quantities $q_i > \frac{Q}{2}$ at 1.

The second function $f^k_{\texttt{VB},2}$ was proposed in [8, § 4.4], by generalizing $f^k_{\texttt{VB},1}$ from [28]. Using integer parameter $k \geq 2$, it can be written as:

$$f^k_{\texttt{VB},2}(x) = \begin{cases} \frac{\max(0, \lceil kx \rceil - 1)}{k-1} & \text{if } x < 0.5 \\ 0.5 & \text{if } x = 0.5 \\ f^k_{\texttt{VB},2}(1-x) & \text{if } x > 0.5 \end{cases} \tag{4.2}$$

This formula generates a stair-case function in which the length of each interval (stair) is usually $\frac{1}{k}$ (except in the proximity of $x = \frac{1}{2}$). Based on very limited preliminary experiments, it is reasonable to consider all values of k from 2 to 300. By trying more parameter values and more functions (see many examples in [8]), the quality of our DFF bounds could have been improved. However, the main goal of the paper is not to present very refined "competition" results, but to describe a general technique for applying (D)DFFs to new (capacitated) problems.

4.1. `Capacitated` *p*-`Median`

While less studied than its non-capacitated counterpart, `Capacitated` *p*-`Median` has already been tackled with CG methods, see model (10) in [18] or model (6)-(8) in [7]. The former model is more similar to our model (2.3a)–(2.3b). We made used several times of the work in [18], for three main reasons: (i) it provides a set of real-life `Capacitated` *p*-`Median` instances (see `www.lac.inpe.br/~lorena/instancias.`

`html`); (ii) it derives Lagrangeean/Surogate lower bounds (calculated with a limit of 300 iterations) that can also be used for comparisons; and (iii) it also present results on the best-known integer feasible solution (upper bound based on heuristics from [17]). Compared to our primal-dual LPs (2.3a)–(2.3b), their model differs in that they impose the median $m(\mathbf{a})$ to belong to the cluster $\mathbf{a}$. As such, our primal LP (2.3a) has actually more columns than their primal LP, and so, it could yield (slightly) lower optima. The provided instances usually have a capacity Q between 700 and 1000; the quantities (weights) only rarely exceed $\frac{Q}{2}$.

The DFF bounds from this section are obtained using Theorem 1 (§2, p. 6); the objective value is determined by (2.13). We recall that this bound resides on a reduced DFF-CG model that is optimized by a classical CG algorithm based on two pricing sub-problems, one for (2.11) and the other for (2.12). As described in Section 2.2.1, the former pricing is rather straightforward. The latter one is similar to the pricing of the initial CG model (of (2.3b)) and it requires solving certain binary knapsack problems. The speed-up of the DFF-CG model comes from reducing the number of knapsack items from $n = |V|$ to $|\overline{V_k}|$, where $\overline{V_k}$ identifies the largest items, see (2.7). We solved the binary knapsack problems with `minknap`, *i.e.*, one of the best knapsack algorithms tested in [22] (we used the implementation available at `www.diku.dk/~pisinger/minknap.c`).

We compare below the results of the DFF bound (Section 4.1.1) and of the DFF warm-started CG (Section 4.1.2) with the results of the pure CG. This pure CG bound is obtained by solving to optimality (2.3a) using a classical CG algorithm. As in other CG work on `Capacitated` p-`median`, the pricing sub-problem requires solving n knapsack problems with n items, once for each possible median location of the next column. We used the same `minknap` implementation as above for all these knapsack problems.

$\frac{\lvert\overline{V_k}\rvert}{\lvert V\rvert}$ goal		`Capacitated` p-`Median` instance					
	$n/p \rightarrow$	`sjc1` 100/10	`sjc2` 200/15	`sjc3a` 300/25	`sjc3b` 300/30	`sjc4a` 402/30	`sjc4b` 402/40
$\frac{1}{4}$	LB_{DFF}	10062	18954	27915	26003	36564	32625
	speed-up of DFF bound	$1/70$	$1/183$	$1/205$	$1/199$	$1/453$	$1/392$
$\frac{1}{2}$	LB_{DFF}	12385	24294	32649	29820	43842	38121
	speed-up of DFF bound	$1/17$	$1/18$	$1/29$	$1/29$	$1/47$	$1/43$
$\frac{3}{4}$	LB_{DFF}	14357	27568	38470	34651	51891	44460
	speed-up of DFF bound	$1/4$	$1/4$	$1/6$	$1/6$	$1/7$	$1/7$
	speed-up of DFF warm-started CG	$1/1.22$	$1/1.35$	$1/2.04$	$1/2.38$	$1/2.05$	$1/3.08$
	iters in DFF stage w.r.t. tot iters	56%	55%	56%	60%	57%	79%
	tot iters w.r.t. tot iters pure CG	100%	98%	91%	86%	87%	66%
1	LB_{CG}	17263	33232	45315	40635	61850	52403
	CPU Time[sec] for pure CG[a]	62	799	4232	3953	17305	14165
Lagr/Surr Bnd. [18, Tab. 1]		17150	33233	45245	40635	61851	52404
Best Known Feasible [17, 18]		17289	33396	45365	40636	62001	52642

Table 1: Results of two DFF-based lower bounding methods. After the heading, the first 6 rows indicate the DFF bound and its speed-up for several values of the (goal) ratio $\frac{|\overline{V_k}|}{|V|}$. Recall that $\overline{V_k}$ is the set of stand-alone variables not expressed via DFFs. The next 3 rows (starting at row "*DFF warm-started CG*") provide several indicators concerning the method from Section 2.2.2 (we use the $|\overline{V_k}|$-reduced clusters generated by the DFF method for $\frac{|\overline{V_k}|}{|V|} = \frac{3}{4}$ to warm-start a full CG stage). The speed-up is always expressed as a ratio of the CPU time of pure CG (the third row from the bottom). The last two rows provide other bounds from the literature (a Lagrangean/Surrogate bound obtained with a limit of 300 iterations, rounded up) and the best known integer feasible solution (based on heuristics from [17]).

[a] All running times are reported on a HP ProBook 4720 laptop clocked at 2.27GHz (Intel Core i3), using `gnu g++` with code optimization option `-O3` on Linux Ubuntu, kernel version 2.6. (Cplex version 12.3)

4.1.1. Analysing the results for the DFF bound of Theorem 1

Table 1 presents a summary of the results of the DFF bounds and of the DFF warm-started CG. Let us here focus on comparing the DFF bounds (denoted LB_{DFF}) with the pure CG bound (noted LB_{CG}) and to some upper bounds from the literature. For each "$\frac{|\overline{V_k}|}{|V|}$ goal" value (Column 1), we consider the lowest k that leads via (2.7) to a ratio $\frac{|\overline{V_k}|}{|V|}$ greater or equal to this goal. The "trade-off" between running time and quality can be controlled by varying k, *i.e.*, using a larger k, $\frac{|\overline{V_k}|}{|V|}$ becomes larger, and so, the DFF bound is improved at the cost of increasing the running time. Denoting the running time of the pure CG bound by t_{CG}, Table 1 shows the following general facts:

- within about $\frac{t_{\text{CG}}}{100}$ time (usually less), the quality of our mixed DFF-CG bound is about $LB_{\text{DFF}} \approx \frac{1}{2}LB_{\text{CG}}$, see the rows with a value of 0 or $\frac{1}{4}$ in Column 1;
- within about $\frac{t_{\text{CG}}}{20}$ time (minimum $\frac{t_{\text{CG}}}{43}$, maximum $\frac{t_{\text{CG}}}{17}$), LB_{DFF} raises to almost $\frac{3}{4}LB_{\text{CG}}$, see the rows with value $\frac{1}{2}$ in Column 1;
- within $\frac{t_{\text{CG}}}{4}$ time (or less), LB_{DFF} raises to almost $\frac{4}{5}LB_{\text{CG}}$, see the rows having $\frac{3}{4}$ in Column 1;

Since the objective value of the best-known feasible solution (see last row of Table 1) is always less than $101\%LB_{\text{CG}}$, so is the integer optimum value OPT_{IP}, *i.e.*, we can state $OPT_{\text{IP}} \leq 101\%LB_{\text{CG}}$. As such, the above quality remarks on LB_{DFF} can generally be stated even with regards to OPT_{IP}; for instance, one can safely say that our DFF approach can reach, using $\frac{1}{20}$ of the total CG convergence time, a LB value of about $\frac{3}{4}LB_{\text{CG}} \geq \frac{3}{4} \cdot \frac{100}{101}OPT_{\text{IP}} \approx \frac{3}{4}OPT_{\text{IP}}$. The DFFs used for this numerical experiment are the following: for the goal $\frac{|\overline{V_k}|}{|V|} = 0$, we tried f_0^0, $f_0^{0.1}, \ldots f_0^{0.4}$ and $f_{\texttt{VB},2}^2$, $f_{\texttt{VB},2}^3$, $\ldots f_{\texttt{VB},2}^{300}$; for the other goals, we only tried f_0^0 (equivalent to the identity).

4.1.2. Analysing the results of the DFF warm-started CG

The results of the DFF warm-started CG (Section 2.2.2) are also presented in Table 1, see the three rows starting from row "speed-up of DFF warm-started CG". These three rows provide the following information: (1) the general CPU time speed-up of the new warm-started method relative to pure CG, (2) the proportion of columns generated only by the pricing performed during the first DFF stage relative to the total number of iterations of the new CG, and (3) the ratio (in percents) of the number of columns generated by the new warm-started CG relative to the pure CG.

The main conclusion is that the speed-up is due to two reasons: (i) the fact that more than half of the total number of columns are actually generated in the first DFF stage, and (ii) a slight reduction (usually between 10% and 30%) of the number of iterations. The point (i) seems more important and let us first comment on it. The columns generated in the DFF stage come from clusters constructed by the $|\overline{V_k}|$-reduced pricing (2.12) in Section 2.2.1. This DFF stage is very fast, as also resulting from row "speed-up of the DFF bound" for $\frac{|\overline{V_k}|}{|V|} = \frac{3}{4}$. This acceleration is not only due to the fact that the number of variables is reduced by $\frac{1}{4}$; since we eliminate the smallest items, our knapsack algorithm (*i.e.*, `minknap`) is accelerated by more than $\frac{1}{4}$, as many dynamic programming states are eliminated. The second point above (ii) is related to a certain stabilization effect that seem to have a variable amplitude. However, the DFF-generated constraints provided to the new warm-started method allows it to start with a significantly better objective function relative to the pure CG.

4.2. `Arc Routing` *Problems*

4.2.1. `Capacitated Arc-Routing` *with Fixed Cost*

To our knowledge, there are four `Capacitated Arc Routing` Problem (CARP) instance sets that are publicly available (see `www.uv.es/~belengue/carp.html` for more information and references): standard-size instances `gdb`, `kshs`, `val` (also called `bccm`) and large-scale instances `egl` (with $|V|, |E| > 50$). The capacity Q ranges from from 5 to 305 (with very small capacities only for the `gdb` instances); the edge lengths d_j are in the same order of magnitude as the quantity demands q_j (a few hundreds at maximum). The fleet size p is always large enough to be able to provide all demanded service; no service is required more than once ($\nexists j \in E$ s.t. $b_j \geq 2$). For each CARP instance, one can construct a `Capacitated Arc Routing With Fixed Cost` counterpart by simply adding a fixed cost F in the route costs, as first

proposed in the literature of the 1980s [27]. We simply tried several F values ($F = 5C$, $F = C$, $F = 0.2C$ and $F = 0$—pure CARP) and we applied them on some random instances from the four CARP instance classes above.

The DFF bounds are obtained by applying a particular case of Theorem 2 (p. 10), in which we only use dual variables for edges $j \in E$. We report the best objective function value (3.6) under constraint (3.3), with $\beta = 0$—*i.e.*, we ignore f_d terms, because we do *not* (yet) consider distance constraints. The following DFF have been considered: f_0^0, $f_0^{0.1}$, $f_0^{0.2} \ldots f_0^{0.4}$ and $f_{\mathtt{VB},2}^1$, $f_{\mathtt{VB},2}^2$, $\ldots f_{\mathtt{VB},2}^{300}$.

The pure CG bound (denoted LB_{CG}) is determined using an implementation of the method from [23] that optimizes (3.1b). We did not try to generate only elementary routes, we did not apply refined k-cycle reductions or other fancy strengthening methods; one can check that, for $F = 0$ (pure CARP), our pure CG bound is similar to other pure CG bounds from the literature [16, 24].

The results are provided in Table 2. One first observes that the DFF running time is very low, *i.e.*, as expected, in most cases, it is a matter of milliseconds. Larger fixed costs leads to DFF bounds within $90\% - 99.9\%$ of the CG bound for all instances except `egl` (see below more discussions on these instances). Even for $F = 0.2C$, the DFF bound stays at more than $80\% LB_{\mathrm{CG}}$ in most cases. For $F = 0$ (pure CARP), the DFF bound is simply equivalent to summing $\sum_{j \in E} b_j d_j$; this case could be more effectively addressed by a *mixed* CG-DFF bound, as sketched in Section 3.2.1. However, even for instances for which the DFF bound is generally not very good, the quality ratio $LB_{\mathrm{DFF}}/LB_{\mathrm{CG}}$ improves as the fixed cost increases, *e.g.*, for `val1C`, it increases from $\approx 50\%$ (*i.e.*, $\frac{146}{229}$) at $F = 0$ to $\approx 95\%$ (*i.e.*, $\frac{1936}{2029}$) at $F = 5D$.

Fixed cost		Instance of `Arc-Routing With Fixed Costs`								
		`gdb05`	`gdb06`	`gdb12`	`gdb23`	`kshs5`	`val1C`	`val7C`	`egl-e1-C`	`egl-e4-C`
$F = 5C$	LB_{DFF}	446	370	1396	1553	11493	1936	3044	8808	14718
	speed-up	$1/1416$	$1/450$	$1/1735$	$1/2762$	$1/1021$	$<1/10000$	$<1/10000$	$<1/10000$	$<1/10000$
	LB_{CG}	489	392	1532	1563	12620	2029	3103	≈ 12600	≈ 23500
	time[s][a]	2.0	0.8	3.5	7.1	0.8	320	> 1000	> 1000	> 1000
$F = 1C$	LB_{DFF}	342	282	548	489	9721	504	808	2936	4906
	speed-up	$1/1565$	$1/712$	$1/1238$	$1/1505$	$1/843$	$<1/10000$	$<1/10000$	$<1/10000$	$<1/10000$
	LB_{CG}	385	304	668	499	10820	592	865	≈ 6700	≈ 13500
	time[s]	2.1	0.8	1.5	5	1.1	277	> 1000	> 1000	> 1000
$F = 0.2C$	LB_{DFF}	321	264	378	276	9366	217	360	1761	2943
	speed-up	$1/1614$	$1/328$	$1/582$	$1/974$	$1/697$	$<1/10000$	$<1/10000$	$<1/10000$	$<1/10000$
	LB_{CG}	364	287	487	286	10460	303	416	≈ 5550	≈ 11500
	time[s]	2.2	0.6	1.1	4.4	0.9	445	> 1000	> 1000	> 1000
$F = 0$	LB_{DFF}	316	260	336	233	9278	146	249	1468	2453
	speed-up	$1/1054$	$1/1990$	$1/679$	$1/1354$	$1/809$	$<1/10000$	$<1/10000$	$<1/10000$	$<1/10000$
	LB_{CG}	359	282	445	233	10370	229	301	≈ 5250	≈ 11000
	time[s]	2.2	2.3	1.4	3.7	1.0	490	> 1000	> 1000	> 1000
	ILP Opt.	377	298	458	233	10957	245	334	5595	11601

Table 2: DFF results and comparisons for several values of the fixed vehicle cost F in `Arc Routing` instances. The speed-up indicates the running time ratio between the DFF bound and the method from [23] for finding the optimum LB_{CG} of the CG model.[b] However, the essential idea is that most DFF bounds can be calculated in a time of milliseconds. The last row indicates the ILP optimum from [16] (see also `logistik.bwl.uni-mainz.de/benchmarks.php`) for $F = 0$ (pure CARP): it represents less than 115% of the DFF bound in half of the cases.

[a] All running times are reported on an ASUS UL laptop clocked at 1.30GHz (Intel Core Duo SU7300), using `gnu g++` with code optimization option `-O3` on Linux Ubuntu, kernel version 2.6.

[b] When our method took too long to fully converge, we provide approximate LB_{CG} values based on some intermediate bounds.

A disadvantage of the proposed DFF approach is that it can also provide low quality bounds for certain instances. The last two columns of Table 2 present the worst-case scenario we ever encountered

throughout all the tests we performed. In fact, these are the only CARP instances for which there exist some non-required edges $E' \subsetneq E$, *i.e.*, such that $b_{j'} = 0, \forall j' \in E'$. In this case, most feasible routes need to dead-head certain edges from E' only to reach the required edges $E - E'$. This generates large dead-heading costs that are ignored by the DFF bound from Theorem 2, see the arguments leading from (3.4) to (3.5).

4.2.2. `Distance-Constrained Arc-Routing`

To construct a `Distance-Constrained Arc-Routing` instance (see Section 3.3.1) from a classical CARP instance, one has to define: (i) the maximum distance route limit D, (ii) the fixed cost F, and (iii) a fleet size p (the original value of p from the CARP instance might not be enough to execute all required service, because of the new distance constraints). These values are respectively determined as follows: (i) D is 5 times the longest edge for standard-size instances (`gdb`, `val` or `kshs`) or 10 times the longest edge for large-scale instances (`egl`); (ii) we tested $F = \frac{D}{5}$, $F = D$ or $F = 5D$; (iii) p is rather large (*i.e.*, we used $p = 2\frac{\sum_{j \in E} b_j d_j}{D}$) to ensure that the instance is feasible.

Table 3 below provides the results obtained using the DFF approach from Section 3.3.2. This DFF bound is a particular case of the general DFF bound from Theorem 2, Section 3.2 (p. 10). The objective function is given by formula (3.6) (p. 11) with the terms using f_q removed. The corresponding dual polytope $\mathcal{P}$ has (3.8) as main constraints, *i.e.*, no dead-heading information intervene in describing $\mathcal{P}$.

Regarding the pure CG bound LB_{CG}, we calculate the optimum over above $\mathcal{P}$ by applying classical CG. The pricing for (3.8) requires finding cycle of length not exceeding D that maximizes the total profit (the edge profits at each iteration are given by the dual values); for this, we perform a classical deep-first traversal of G. The main conclusions are:

- for standard-size instances (first seven in Table 3, with less than 50 edges), the DFFs usually produce LBs within 90-99% of the CG bound and they require about $0.1 - 0.5\%$ of the CG time;
- for large-scale instances (last two instances in Table 3), the DFF bound is obtained in very small times, but its quality stands at about $3/4$ of the CG bound.

Fixed cost		Instance of `Distance-Constrained Arc-Routing`								
		`gdb05`	`gdb06`	`gdb12`	`gdb23`	`kshs5`	`val1C`	`val7C`	`egl-e1-C`	`egl-e4-C`
$F = 5D$	LB_{DFF}	19004	15668	20272	13473	556680	8833	15065	147180	147180
	speed-up	$1/427$	$1/271$	$1/902$	$1/366$	$1/415$	$1/460$	$1/477$	$<1/10000$	$<1/10000$
	LB_{CG}	19157	16350	20378	13452	606280	8931	16813	200733	200756
	time[s][a]	0.40	0.21	0.75	0.71	0.21	0.61	1.07	1145.95	987.55
$F = D$	LB_{DFF}	6328	5213	6742	4478	185560	2934	5005	49060	49060
	speed-up	$1/480$	$1/465$	$1/932$	$1/322$	$1/273$	$1/473$	$1/499$	$<1/10000$	$<1/10000$
	LB_{CG}	6384	5350	6782	4484	195479	2977	5444	66906	66902
	time[s]	0.43	0.37	0.74	0.62	0.15	0.65	1.11	1133.51	1301.83
$F = 0.2D$	LB_{DFF}	3793	3122	4036	2679	111336	1754	2993	29436	29436
	speed-up	$1/453$	$1/243$	$1/1128$	$1/531$	$1/460$	$1/591$	$1/568$	$<1/10000$	$<1/10000$
	LB_{CG}	3832	3150	4063	2690	113320	1786	3170	40140	40141
	time[s]	0.43	0.19	0.95	1.04	0.25	0.84	1.31	1326.10	1627.63

Table 3: Comparison between the DFF bound LB_{DFF} and the pure CG bound LB_{CG} for several values of the fixed cost F on `Distance-Constrained Arc-Routing`. The speed-up indicates the ratio between the running times of DFF and CG bounding methods.

[a] All running times are reported on an ASUS UL laptop clocked at 1.30GHz (Intel Core Duo SU7300), using `gnu g++` with code optimization option `-O3` on Linux Ubuntu, kernel version 2.6.

We finish by noting that the main advantage of all these `Arc-Routing` DFF bounds resides in their speed, *i.e.*, they require running times of milliseconds. This is not surprising given that the calculation of a DFF bound only consists of a loop that computes a weighted sum. While the DFF bound *can*

have a very high quality in certain cases (*e.g.*, it can even reach the optimal IP solution for pure CARP instance `gdb23` in Table 2), it can also produce results of mixed quality in other cases. However, pure CG methods do not seem to be able to compute a similar quality (Lagrangian) bound within such times of milliseconds. Indeed, some preliminary experiments confirm that it could be very hard (or impossible) to solve a single sub-problem within the time used to calculate a DFF bound.

5. Conclusions

(D)DFFs have often been used to generate very fast lower bounds for `Bin-Packing` and `Cutting-Stock` problems. For such problems, all columns (patterns) have the same cost and the DFF lower bound has a rather simple form (*e.g.*, $\sum_{i=1}^{n} b_i f(\frac{q_i}{Q})$, up to notational equivalences); the dual variable of i takes the value of a DFF function applied on $\frac{q_i}{Q}$. We showed that (D)DFFs can be applied to new different (capacitated) problems in which the cost of columns (routes or clusters) has a more complex structure, depending on a sum of distances. The new lower bound formulas are more complex—see (2.13) for `Capacitated` p-`Median` (p. 8) and (3.6) for `General Routing` (p. 11)—, but the (D)DFF bounds are usually still fast. Numerical experiments confirm that, generally speaking, the proposed approach produces good quality bounds, often using less than 1% of the total CG convergence time.

We also present a mixed CG-DFF lower bounding approach that can be slower, because it still needs to optimize a restricted model by CG (see Section 2.2 for `Capacitated` p-`Median` or Sections 3.2.1 and 3.3.3 for `General Routing`). However, this restricted model has a smaller pricing sub-problem, *i.e.*, the sub-problem has a reduced number of variables $|\overline{V_k}|$ (or $|\overline{Y}|$ for `General Routing`) instead of n, where $\overline{V_k}$ (or $\overline{Y}$) indicates a subset of dual values that are not expanded as DFF terms. Section 2.2.2 extended this approach even further: the constraints generated by the above CG-DFF approach are used as initial constraints in the beginning of a "DFF warm-started" CG. This new CG can find the CG bound 2-3 times more rapidly that the pure CG.

References

[1] P. Avella, A. Sassano, and I. Vasil'ev. Computational study of large-scale p-median problems. Mathematical Programming, 109(1):89–114, 2007.

[2] R. Baldacci and M. A. Boschetti. A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem. European Journal of Operational Research, 183(3):1136–1149, 2007.

[3] G. Belov, V. Kartak, H. Rohling, and G. Scheithauer. Conservative scales in packing problems. OR spectrum, 35(2):505–542, 2013.

[4] M. A. Boschetti. New lower bounds for the three-dimensional finite bin packing problem. Discrete Applied Mathematics, 140(1):241–258, 2004.

[5] M. A. Boschetti and A. Mingozzi. The two-dimensional finite bin packing problem. part i: New lower bounds for the oriented case. 4OR, 1(1):27–42, 2003.

[6] A. Caprara and M. Monaci. Bidimensional packing by bilinear programming. Mathematical programming, 118(1):75–108, 2009.

[7] A. Ceselli. Two exact algorithms for the capacitated p-median problem. Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 1(4):319–340, 2003.

[8] F. Clautiaux, C. Alves, and J. Carvalho. A survey of dual-feasible and superadditive functions. Annals of Operations Research, 179(1):317–342, 2009.

[9] F. Clautiaux, A. Moukrim, and J. Carlier. New data-dependent dual-feasible functions and lower bounds for a two-dimensional bin-packing problem. International Journal of Production Research, 47(2):537–560, 2009.

[10] O. Du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. Discrete Mathematics, 194(1):229–237, 1999.

[11] S. P. Fekete and J. Schepers. New classes of fast lower bounds for bin packing problems. Mathematical Programming, 91(1):11–31, 2001.

[12] S. P. Fekete and J. Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. Mathematical Methods of Operations Research, 60(2):311–329, 2004.

[13] R. Garfinkel, A. Neebe, and M. Rao. An algorithm for the m-median plant location problem. Transportation Science, 8(3):217–236, 1974.

[14] L. E. Jackson, G. N. Rouskas, and M. F. Stallmann. The directional p-median problem: Definition, complexity, and algorithms. European Journal of Operational Research, 179(3):1097 – 1108, 2007.

[15] D. S. Johnson. Near-optimal bin packing algorithms. PhD thesis, Massachusetts Institute of Technology, 1973.

[16] A. Letchford and A. Oukil. Exploiting sparsity in pricing routines for the capacitated arc routing problem. Computers & Operations Research, 36:2320–2327, 2009.

[17] L. A. N. Lorena and E. L. F. Senne. Local search heuristics for capacitated p-median problems. Networks and Spatial Economics, 3(4):407–419, 2003.

[18] L. A. N. Lorena and E. L. F. Senne. A column generation approach to capacitated p-median problems. Computers & Operations Research, 31(6):863–876, 2004.

[19] G. S. Lueker. Bin packing with items uniformly distributed over intervals $[a, b]$. In 24th Annual Symposium on Foundations of Computer Science, pages 289–297. IEEE, 1983.

[20] C. Orloff. A fundamental problem in vehicle routing. Networks, 4(1):35–64, 1974.

[21] R. Pandi and B. Muralidharan. A capacitated general routing problem on mixed networks. Computers & operations research, 22(5):465–478, 1995.

[22] D. Pissinger. Where are the hard knapsack problems? Computers & Operations Research, 32(9):2271 – 2284, 2005.

[23] D. Porumbel. The integer ray method: Optimizing polytopes with prohibitively many constraints in set-covering column generation models. Submitted Paper (preprint at `www.optimization-online.org/DB_HTML/2013/09/4056.html`).

[24] M. P. R. Martinelli, D. Pecin and H. Longo. Column generation bounds for the capacitated arc routing problem. In XLII SBPO, 2010.

[25] J. Reese. Solution methods for the p-median problem: An annotated bibliography. Networks, 48(3):125–142, 2006.

[26] J. Rietz, C. Alves, and J. V. de Carvalho. Worst-case analysis of maximal dual feasible functions. Optimization Letters, 6(8):1687–1705, 2012.

[27] G. Ulusoy. The fleet size and mix problem for capacitated arc routing. European Journal of Operational Research, 22(3):329–337, 1985.

[28] F. Vanderbeck. Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. Operations Research, 48(6):915–926, 2000.

Appendix A. `Capacitated` p-`Median` with Fixed Costs and Median Assignment Restrictions

A drawback of the `Capacitated` p-`Median` DFF approach from Section 2 is the fact that certain dual values are expressed using terms like $\min_{j\in[1..n]} d_{i,j}$ (Proposition 1, p. 5) or $\min_{j\in[1..n],i\neq j} d_{i,j}$ (Theorem 1, p. 6). It is enough to have only a few *very small* distances $d_{i,j}$ to generate many dual variables of very small value. This can degrade the quality of the resulting bound. The goal of this appendix is to show that certain p-`Median` versions do not have such issues; let us introduce two (natural) assumptions:

(a) Given $i, j \in V$, we bound to $\widetilde{c}_{ij}$ the maximum number of times median j can service i. This is a *median assignment restriction*. It can arise, for instance, if a client i does not want to have all demands b_i supplied from only one provider.

(b) Add a fixed cost F to the cost of each column (cluster). The resulting model is more similar to a `Facility Location` problem with fixed costs and a maximum number p of medians (locations).

To take the median assignment restriction (a) into consideration, we need to extend the column representation from the primal-dual LPs (2.3a)-(2.3b). Please accept a notational shorthand that extends columns $[d_a\ \mathbf{a}]^\top \in A$ to $[d_a\ \mathbf{a}\ \widetilde{\mathbf{a}}]^\top \in \widetilde{A}$ by appending a 2-index column vector $\widetilde{\mathbf{a}} = [\widetilde{a}_{11}\dots\widetilde{a}_{1n},\ \widetilde{a}_{12}\dots\widetilde{a}_{2n},\ \dots,\ \widetilde{a}_{n1}\dots\widetilde{a}_{nn}]^\top$ with elements $\widetilde{a}_{ij}$: i is a serviced vertex and j is the cluster median, *i.e.*, $a_{ij} = 1$ if $i \in V$ is serviced from median $j \in V$. All valid extended columns $[d_a\ \mathbf{a}\ \widetilde{\mathbf{a}}]^\top \in \widetilde{A}$ need to respect contiguity relations between $\mathbf{a}$ and $\widetilde{\mathbf{a}}$, *e.g.*, $\widetilde{a}_{ij} \leq a_i \forall i, j \in V$ (to service i by median j, i needs to be in the cluster). The new *median assignment restriction* acts on the primal solutions $\mathbf{x}$ and is added (after a sign inversion) to (2.3a) under the form:

$$-\sum_{[d_a\ \mathbf{a}\ \widetilde{\mathbf{a}}]^\top\in\widetilde{A}} x_a\widetilde{a}_{ij} \geq -\widetilde{c}_{ij}, \forall i, j \in [1..n],$$

This constraint is dualized using a 2-index column vector of dual variables $\widetilde{\mathbf{y}} = [\widetilde{y}_{11}\dots\widetilde{y}_{1n}, \widetilde{y}_{21}\dots\widetilde{y}_{2n}, \dots, \widetilde{y}_{n1}\dots\widetilde{y}_{nn}]^\top$. The values $\widetilde{c}_{ij},\ \forall i, j \in [1..n]$ represent right-hand side values in the *median assignment* constraints of the primal LP (2.3a); they are thus dualized into objective function coefficients in the dual LP (2.3b). Denoting $\widetilde{\mathbf{c}} = [\widetilde{c}_{11}\dots\widetilde{c}_{1n}, \widetilde{c}_{21}\dots\widetilde{c}_{2n}, \dots, \widetilde{c}_{n1}\dots\widetilde{c}_{nn}]^\top$, the dual program (2.3b) becomes:

$$\left.\begin{array}{ll} \multicolumn{2}{c}{\max \mathbf{b}^\top\mathbf{y} - \widetilde{\mathbf{c}}^\top\widetilde{\mathbf{y}} - p\mu} \\ \mathbf{a}^\top\mathbf{y} - \widetilde{\mathbf{a}}^\top\widetilde{\mathbf{y}} - \mu \leq d_a, & \forall [d_a\ \mathbf{a}\ \widetilde{\mathbf{a}}]^\top \in \widetilde{A} \\ \widetilde{\mathbf{y}} \geq \mathbf{0}_{n^2} & i \in [1..n] \\ \mathbf{y} \geq \mathbf{0}_n & i \in [1..n] \\ \mu \geq 0 & \end{array}\right\} \mathcal{P} \tag{A.1}$$

The fixed-cost assumption (b) is taken into consideration by incorporating a fixed cost F in the calculation of the cluster (column) cost d_a; (2.2) evolves to:

$$d_a = F + \sum_{i=1}^{n} a_i d_{i,m},$$

where m is the median of the cluster, *i.e.*, the unique index $m \in [1..n]$ for which $\exists i \in [1..n]$ such that $\widetilde{a}_{im} = 1$.

Proposition 2. *For any* $\mathbf{q}$*-DDFF* f*, the dual values below yield a valid dual solution in* (A.1)*.*

$$\begin{array}{ll} y_i = (\mu + F) f\left(\frac{q_i}{Q}\right) + d_{\max} & \forall i \in [1..n] \\ \widetilde{y}_{ij} = d_{\max} - d_{i,j} & \forall i, j \in [1..n] \end{array} \tag{A.2}$$

where $d_{\max} = \max_{i,j\in[1..n]} d_{i,j}$*.*

Proof. Consider a valid column (cluster) with the median located in m. We need to prove that the main (dual) constraint $\mathbf{a}^\top \mathbf{y} - \widetilde{\mathbf{a}}^\top \widetilde{\mathbf{y}} - \mu \leq d_a$ is verified in (A.1); this constraint can be written as:

$$\sum_{i=1}^{n} a_i \left((\mu + F)\, f\left(\frac{q_i}{Q}\right) + d_{\max} \right) - \sum_{j=1}^{n} \sum_{i=1}^{n} \widetilde{a}_{ij} \left(d_{\max} - d_{i,j} \right) - \mu \leq F + \sum_{i=1}^{n} a_i d_{i,m}$$

The above inequality can be proved using two rather straightforward reductions. First, for each right-hand side $a_i d_{i,m}$, there is a left-hand side term $\widetilde{a}_{ij} d_{i,j} = a_i d_{i,m}$ (recall that the definition of $\widetilde{\mathbf{a}}$ states that $\widetilde{a}_{ij} = a_i$ at the cluster median $j = m$ and $\widetilde{a}_{ij} = 0$ if $j \neq m$). As such, we only have to prove:

$$\sum_{i=1}^{n} a_i \left((\mu + F)\, f\left(\frac{q_i}{Q}\right) + d_{\max} \right) - \sum_{j=1}^{n} \sum_{i=1}^{n} \widetilde{a}_{ij} d_{\max} - \mu \leq F$$

The $d_{\max}$ terms are also reduced: they arise twice for each i in the cluster, once with a positive sign (first sum) and once with a negative sign (second sum). Indeed, each "$a_i d_{\max}$" term in the first sum is canceled by a "$-\widetilde{a}_{ij} d_{\max}$" term in the second sum when $j = m$ (other $\widetilde{a}_{ij}$ values are 0 at $j \neq m$). As such, we only have to prove:

$$\sum_{i=1}^{n} a_i \left((\mu + F)\, f\left(\frac{q_i}{Q}\right) \right) - \mu \leq F$$

This is simply equivalent to $(\mu + F) \sum_{i=1}^{n} a_i f\left(\frac{q_i}{Q}\right) \leq \mu + F$, which is true due to the fundamental D(DFF) property ($\star$) of the **q**-DDFF f (given cluster capacity condition $\sum_{i=1}^{n} a_i q_i \leq Q$). □

After replacing (A.2) in the objective function of (A.1), we obtain the same two cases of Proposition 1 (see the end of Section 2.1) regarding μ: either $\mu = \infty$ if the instance is infeasible ($\sum_{i=1}^{n} b_i f\left(\frac{q_i}{Q}\right) > p$), or $\mu = 0$ otherwise. The second case is more important an it leads to an objective function value

$$F \sum_{i=1}^{n} b_i f\left(\frac{q_i}{Q}\right) - \sum_{i=1}^{n} \sum_{j=1}^{n} \widetilde{c}_{ij} \left(d_{\max} - d_{i,j} \right)$$

A fast DFF bound can simply be calculated by integrating a (D)DFF f in the above formula, as in other examples throughout the paper. One could expect a higher LB quality for instances with rather homogeneous distance values. However, if some values $d_{i,j}$ are much larger than all others, it could be easy to detect that *no* optimal solution can service i from median j. In such cases, one could (artificially) set $\widetilde{c}_{ij} = 0$ so as to obtain a better (and still valid) DFF lower bound.