# Optimal Solution of Vehicle Routing Problems with Fractional Objective Function

Roberto Baldacci

Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi", University of Bologna, Via Venezia 52, 47521 Cesena, Italy

Andrew Lim

Department of Industrial & Systems Engineering, National University of Singapore, 1 Engineering Drive 2, Singapore 117576, Singapore

Emiliano Traversi

Laboratoire dInformatique de Paris Nord, Université de Paris 13; and Sorbonne Paris Cit, CNRS (UMR 7538), 93430 Villetaneuse, France

Roberto Wolfler Calvo

Laboratoire dInformatique de Paris Nord, Université de Paris 13; and Sorbonne Paris Cit, CNRS (UMR 7538), 93430 Villetaneuse, France

#### Abstract

This work proposes a first extensive analysis of the Vehicle Routing Problem with Fractional Objective Function (VRPFO). We investigate how the principal techniques used either in the context of fractional programming or in the context of vehicle routing problems interact. We present new dual and primal bounding procedures which have been incorporated in an exact method. The method proposed allows to extend specific variants of VRP to their counterpart with a fractional objective function. Extensive numerical experiments prove the validity of our approach.

## 1 Introduction

In this paper, we investigate how to solve to optimality the Vehicle Routing Problem (VRP) with optional customers and Fractional Objective Function (FO). Given a fleet of vehicles and a set of customers, the VRP aims at serving the customers with a set of feasible routes at minimal cost. VRPs represent a wide area of combinatorial optimization and mathematical programming. Most of the works in the literature address the case where the objective function is linear and all customers must be visited (see Toth and Vigo 2014). In the case considered in this paper, a subset of the customers is optional. Therefore, an optional customer is visited only if the objective function improves. With a linear objective function minimizing the overall cost, the optional customers would never be part of an optimal solution. However, this is not true when the objective function analyzed is fractional, i.e., it is of the form  $\frac{f(x)}{g(x)}$ . In this work we focus on the case where f(x) and g(x) are linear. Such type of objective function is used to model the so-called Logistic Ratio (LR), which is the ratio of the total cost to the overall resources spent to serve the customers.

In the past, the LR has been widely used in the context of inventory control and production planning (Bázsa et al. (2001) or Barros et al. (1997)). The LR has been also

applied to routing problems, more precisely, to the Inventory Routing Problem (IRP) (Morton (2011), Benoist et al. (2011), Garaix et al. (2011) and Archetti et al. (2016)).

The IRP can be described as the combination of a VRP with an inventory management problem, where a supplier has to deliver products to a number of customers over a given time horizon without running out of stock. Therefore, the IRP boils down to simultaneously decide the inventory management, the vehicle routing and the delivery scheduling (Coelho et al. (2014)).

When an inventory routing problem is used in practice, a time horizon needs to be fixed. Nevertheless, a fixed time horizon, neglects the fact that it would be necessary to solve another optimization problem over the next time horizon. This is due to the fact that a direct minimization of costs would lead to postponing as many deliveries as possible to later planning periods (Dror and Ball (1987)). It is difficult to define appropriate models able to provide solutions that are "robust" also beyond the time horizon considered. The LR uses more efficiently the resources available in the current planning period, since it provides a solution with better average cost per quantity of resources used. The needs of customers are better anticipates by minimizing the LR since it avoids myopic behaviours of the optimization models at the end of the time horizon. While it is not the case with the objective function that directly minimize the cost. What it is more, as observed by Campbell et al. (2001), real-life inventory routing problems are stochastic. Therefore, any distribution plan covering more than a couple of days will never be executed completely as planned. Actual volumes delivered differ from planned volumes because usage rates deviate from their forecasts, the planned driving time is off due to traffic congestion, and so forth. Therefore, any planning system needs to be flexible. In addition, it needs to take advantage of the latest changes in the data, such as last minute orders. Therefore, in practice, long-term plan are often implemented on a daily basis, and the daily planning needs to capture the costs and benefits of delivering to a customer earlier than strictly necessary.

The VRPFO presents the advantages of planning the current delivery by also taking into account the future demand, while optimizing only over a single day (allowing to keep the size of the problem under control). In this context, a customer whose inventory will reach the safety stock level at the end of the day is considered as a compulsory customer (also called *must-go* customer). On the other hand, a customer whose inventory is below the earliest delivery level but will not reach the safety stock level in the planning window becomes an optional customer (also called *may-go* customer), see Loes (2016) for more details. Therefore, this work answers to the research question of whether it is possible to find a model able to capture the multiperiod optimization aspect typical of an IRP with FO that is also able to handle instances of size of practical interest.

#### 1.1 Literature review

The use of the LR as objective function characterizes the problem to solve as a Fractional Programming (FP) problem, which is a generalization of linear programming. We refer to Radzik (1999), Frenk and Schaible (2005), Schaible and Shi (2004) or Stancu-Minasian (2012) for a brief introduction to FP.

Two main techniques are available in the literature to tackle FP: the variable substitution presented by Charnes and Cooper (Charnes and Cooper (1962)) and the Dinkelbach's algorithm (Dinkelbach (1967)). In the former method, the variable substitution allows to rewrite the problem as linear, if the divisor of the objective function is always greater than zero. The main drawbacks of this technique are that it can potentially lead to numerical instability and that it is difficult to efficiently use it when solving a fractional problem with discrete variables. Dinkelbach presents an algorithm for problems with (convex) fractional objective function. The algorithm is valid both in case of linear or nonlinear terms in the numerator and denominator. The basic idea of the algorithm is to iteratively solve a parametric linearization of the problem where the parameter represents an estimation of the original objective function. The author also shows that the algorithm terminates in a finite number of iterations.

A generalization of Dinkelback's algorithm to nonconvex (continuous) objective functions is presented in Ródenas et al. (1999). Ródenas et al. also extended the Dinkelbach approach to integer linear fractional programming and observed that the algorithm converges on a finite number of iterations. In Espinoza et al. (2010), the authors show how lifting, tilting and fractional programming can be viewed as the same optimization problem. Espinoza et al. described an exact algorithm for the case of mixed integer linear fractional programming and, by combining results of Dinkelbach (1967) and Schaible (1976), provided a proof of its superlinear convergence rate. Methods for mixed integer linear fractional programming with application in cyclic process scheduling problems were also considered by You et al. (2009). In particular, You et al. extended Dinkelbach approach to discrete problems by also shoving its superlinear convergence rate.

VRPs with fractional objective functions have been studied by Benoist et al. (2011), Garaix et al. (2011) and Archetti et al. (2016). Archetti et al. (2016) presented a study on IRP with LR and an exact method for its solution. One of the main contributions of their work is a comparison of the optimal solutions obtained when minimizing the LR to the ones obtained when minimizing the total cost. The solution technique used is an adaptation of the one proposed in Dinkelbach (1967) to discrete problems. The authors were able to solve to optimality instances involving up to 5 vehicles, 15 customers and 3 periods. Garaix et al. (2011) investigated the maximization of passenger occupancy rate in a dial-a-ride problem. The specific objective function considered is to maximize the rate defined as the sum of the passenger travel times divided by the total travel time of vehicles. Garaix et al. (2011) proposed two approaches for solving the continuous relaxation of a set partitioning model based on the Charnes and Cooper's transformation and on Dinkelback's algorithm. Benoist et al. (2011) described a randomized local search algorithm for solving a real-life routing and scheduling problem arising in optimizing the distribution of fluids by tank trucks in the long run, with the objective to minimize the LR.

The book edited by Toth and Vigo (2014) provides a comprehensive overview of exact and heuristic methods for VRPs. In particular, the chapter by Archetti et al. (2014) reviews VRPs where the set of customers to serve is not given and a profit is associated with each customer that makes such a customer more or less attractive. In this case, the difference between route profit and cost may be maximized, or the profit or the cost optimized with the other measure bounded in a constraint.

A well know technique used for solving routing problems is column generation (CG). Column generation exploits the Dantzig-Wolfe Reformulation of the flow formulation of the original problem and leads to a formulation with an exponential number of variables (we refer the reader to Desaulniers et al. (2006) for a extensive analysis of such a method).

### 1.2 Contributions of this paper

In this paper, we describe an exact method to solve the VRPFO. We formulate the VRPFO using a Set Partitioning (SP) like formulation with a fractional objective function. The VRPFO formulation adopted can be used to model alternative objective functions, such as the LR and the maximization of profit over time.

The exact method combines two bounding procedures, derived from the SP like formulation, with an extension of Dinkelbach's algorithm for fractional programming to integer programs. More precisely, the bounding procedures are used within a route enumeration scheme (see Baldacci et al. 2008) to reduce the number of variables of the integer problems solved at each iterations of the Dinkelbach approach.

The extension of the procedure described is possible thanks to the two following novelties presented in this paper:

- We present a new linear transformation which allows to use a dual ascent heuristic to solve the master problem. It is alternative to the one presented by Charnes and Cooper (Charnes and Cooper (1962)).
- We show how the final dual solution of the new linearization can be used to generate a reduced problem containing only the routes whose reduced costs are smaller than a given threshold.

We perform extensive computational results on instances derived from the VRP literature with different fractional objective functions. The results obtained show that the proposed method is able to solve instances involving up to 79 customers.

The remainder of this paper is organized as follows. In Section 2, we formally describe the problem addressed in this paper and we present the SP model. In Section 3, we present dual bounds based on both continuous and integer relaxations of the SP model. This section also describes the bounding procedures used to compute the dual bounds. Section 4 describes the exact method. Dynamic programming algorithms for generating nonelementary routes and feasible and elementary routes are described in Section 5. We provide the computational studies in Section 6 and concluding remarks in Section 7.

# 2 Problem description and mathematical formulation

The Vehicle Routing Problem with Fractional Objective Function (VRPFO) considered in this paper can be described as follows.

A complete digraph G = (V, A) is given, where the vertex set V is partitioned as  $V = \{0\} \cup V_1 \cup V_2$ . Vertex 0 represents the depot, vertex set  $V_1 = \{1, 2, \ldots, n_1\}$  represents  $n_1$  mandatory customers, and vertex set  $V_2 = \{n_1 + 1, \ldots, n_1 + n_2\}$  represents  $n_2$  optional customers. We denote with  $V_c = V_1 \cup V_2$ ,  $n = n_1 + n_2$  and we assume that  $n_1 > 0$ . With each vertex  $i \in V$  is associated a service time  $s_i > 0$  (we assume  $s_0 = 0$ ). With each arc  $(i, j) \in A$  are associated a travel or routing cost  $d_{ij}$  and a travel time  $t_{ij} \ge 0$ . At the depot it is based a vehicle fleet composed of a set of m identical vehicles. To each vehicle is associated a maximum working time equal to T.

A vehicle route  $R = (0, i_1, \ldots, i_r, 0)$ , with  $r \ge 1$ , is a simple circuit in G passing through the depot, visiting vertices  $V(R) = \{i_1, \ldots, i_r\}$ ,  $V(R) \subseteq V_c$ , and such that the total working time computed as the sum of the total service time of the customers visited and the total travel time of the arcs traversed by the route is less than or equal to T, i.e.,  $\sum_{i \in V(R)} s_i + \sum_{(i,j) \in A(R)} t_{ij} \le T$ , where A(R) is the set of arcs traversed by route R. The cost of route R is equal to the sum of the travel costs of the arc set traversed by route R, i.e.,  $\sum_{(i,j) \in A(R)} d_{ij}$ .

We consider the problem of visiting the mandatory customers and of choosing a subset of the optional customers to visit using vehicles based at the depot. More precisely, the VRPFO consists of designing at most m routes such that (i) each vehicle is used at most once, (ii) each mandatory customer is visited once, and (iii) each optional customer is visited at most once.

The VRPFO models the following fractional linear objective functions of practical interest:

(i) Minimization of Cost/Load. Let q<sub>i</sub> be the demand associated with each customer i ∈ V<sub>c</sub> (we assume q<sub>0</sub> = 0). In addition, a fleet of m identical vehicles of capacity Q is stationed at the depot. The load of a route R = (0, i<sub>1</sub>,..., i<sub>r</sub>, 0) is equal to the total demand of visited customers, i.e. ∑<sub>i∈V(R)</sub> q<sub>i</sub>. The objective is to minimize the ratio of the total travel or routing cost divided by the total load of the routes selected in solution (i.e., the LR). This VRP can be solved as a VRPFO by setting T = Q, s<sub>i</sub> = q<sub>i</sub>, ∀i ∈ V<sub>c</sub>, and t<sub>ij</sub> = 0, ∀(i, j) ∈ A.

(ii) Maximization of Profit/Time. Let  $p_i$  be a nonnegative profit associated with each customer  $i \in V_c$  (we assume  $p_0 = 0$ ). The profit of a route is equal to the total profit of the visited customers, i.e.,  $\sum_{i \in V(R)} p_i$ . The objective is to maximize the ratio of the total profit divided by the total working time of the routes selected in solution. This VRP can be solved as a VRPFO by setting  $d_{ij} = -p_j$ ,  $\forall (i, j) \in A$ .

It is worthwhile to mention that the special case of the VRPFO where all customers are mandatory (i.e.,  $n_2 = 0$ ) and the objective function is the minimization of Cost/Load, is the Capacitated VRP (CVRP), which is in turn a special case of the VRP with Time Windows (VRPTW).

For the state-of-the-art exact algorithms for deterministic VRP, we refer readers to Baldacci et al. (2012), Toth and Vigo (2014), Pecin et al. (2017a,b), among others.

#### 2.1 Mathematical formulation

In this section, we model the VRPFO as a SP problem with side constraints and a linear fractional objective function.

Let  $\mathscr{R}$  be the index set of all routes. Given a route  $\ell \in \mathscr{R}$ , we denote with  $R_{\ell}$  the sequence  $(i_1 = 0, i_2, \ldots, i_r = 0)$  of the vertices visited by the route and with  $V_1(R_{\ell})$  and  $V_2(R_{\ell})$  the sets  $V_1 \cap V(R_{\ell})$  and  $V_2 \cap V(R_{\ell})$ , respectively. Let  $a_{i\ell}$  be a (0-1) binary coefficient equal to 1 if node  $i \in V(R_{\ell})$ , 0 otherwise. Given a route  $\ell$ , we denote with  $c_{\ell}$  and  $w_{\ell}$  the routing cost and the working time of route  $\ell$ , respectively, computed as  $c_{\ell} = \sum_{(i,j) \in A(R_{\ell})} d_{ij}$  and  $w_{\ell} = \sum_{i \in V(R_{\ell})} s_i + \sum_{(i,j) \in A(R_{\ell})} t_{ij}$ .

Let  $x_{\ell}, \ell \in \mathscr{R}$ , be a (0-1) binary variable equal to 1 if and only if route  $\ell$  is in the optimal solution. The VRPFO formulation based on the SP model, hereafter called F, is

$$(F) \quad z(F) = \min \frac{\sum_{\ell \in \mathscr{R}} c_\ell x_\ell}{\sum_{\ell \in \mathscr{R}} w_\ell x_\ell}$$
(1)

$$s.t.\sum_{\ell \in \mathscr{R}} a_{i\ell} x_{\ell} = 1, \quad \forall i \in V_1$$

$$\tag{2}$$

$$\sum_{\ell \in \mathscr{R}} a_{i\ell} x_{\ell} \le 1, \quad \forall i \in V_2 \tag{3}$$

$$\sum_{\ell \in \mathscr{R}} x_{\ell} \le m,\tag{4}$$

$$x_{\ell} \in \{0, 1\}, \qquad \forall \ell \in \mathscr{R}.$$
(5)

In the formulation, the objective function states either to minimize the Cost/Load ratio or to maximize the Profit/Time ratio.

Constraints (2) and (3) impose that each mandatory customer has to be visited by exactly one route and each optional customer has to be visited at most once by the routes selected in the solution, respectively. Constraint (4) requires that at most m routes are selected in the solution.

# **3** Dual bounds for the VRPFO

In this section, we describe relaxations and bounding procedures for the VRPFO. We first present two basic dual bounding techniques, the first one, called DK, solves directly the continuous relaxation of F. The second one, called CG, solves a reformulation of the continuous relaxation of F, proposed for the first time in Charnes and Cooper (1962). In section 3.1, we describe an alternative transformation to the one presented in Charnes and Cooper (1962), such new transformation is used in an advanced dual ascent bounding procedure called DA. Finally, in Section 3.2, we describe a dual bounding procedure CB based on an integer relaxation of formulation F.

Let  $X = \{\mathbf{x} \in \mathbb{R}^{|\mathscr{R}|}_+ : (2), (3), \text{ and } (4)\}$ . We define  $z(CF) = \min\{\sum_{\ell \in \mathscr{R}} c_\ell x_\ell / \sum_{\ell \in \mathscr{R}} w_\ell x_\ell : \mathbf{x} \in X\}$  as the optimal solution cost of the continuous relaxation of formulation F, called CF. Since the objective function of formulation CF is the quotient of linear functions and X is a convex feasible set, the algorithm proposed by Dinkelbach (1967) can also be used to compute z(CF) by means of the solution of a sequence of linear programming problems. In the following, we identify with DK the bounding procedure corresponding to solve CF with the algorithm proposed by Dinkelbach (1967) where each linear programming problem is solved by column generation.

Under the assumption that X is non-empty and bounded, the following transformation, proposed by Charnes and Cooper (1962):

$$u = \frac{1}{\sum_{\ell \in \mathscr{R}} w_{\ell} x_{\ell}}$$
$$y_{\ell} = u x_{\ell}, \quad \forall \ell \in \mathscr{R}$$

translates formulation CF into the equivalent linear program:

$$(CCF) \quad z(CCF) = \min \sum_{\ell \in \mathscr{R}} c_{\ell} y_{\ell}$$
  
s.t.  $\sum a_{i\ell} y_{\ell} = u, \quad \forall i \in V_1$  (6)

$$\sum_{\ell \in \mathscr{R}}^{i \in \mathscr{R}} a_{i\ell} y_{\ell} \le u, \quad \forall i \in V_2$$

$$\tag{7}$$

$$\sum_{\ell \in \mathscr{R}} y_{\ell} \le mu,\tag{8}$$

$$\sum_{\ell \in \mathscr{R}} w_{\ell} y_{\ell} = 1, \tag{9}$$

$$\begin{split} & u \geq 0, \\ & y_{\ell} \geq 0, \\ & \forall \ell \in \mathscr{R} \end{split}$$

CCF contains an exponential number of variables. In practice, such problems are solved

with a column generation procedure (See Desaulniers et al. (2006)). In the following, we identify with CG the dual bounding procedure that solves (CCF) via column generation. The computational results of Section 6 reports a comparison between a DK and CG for computing the dual bound z(CF).

# 3.1 Dual bounding procedure based on an alternative transformation of CF - DA

The alternative transformation is based on the observation that since  $\beta = \sum_{i \in V_1} s_i > 0$ , then the term at the denominator of objective function (1) can be rewritten as  $\beta + \sum_{\ell \in \mathscr{R}} \overline{w}_{\ell} x_{\ell}$ , where  $\overline{w}_{\ell} = w_{\ell} - \sum_{i \in V_1(R_{\ell})} s_i$ . The following lemma holds.

**Lemma 1** Let  $\mathbf{y} = \mathbf{x}/(\beta + \overline{\mathbf{w}}^T \mathbf{x})$ . Then the objective function of problem CF can be rewritten as  $z(CF) = \min \mathbf{c}^T \mathbf{y}$ . In addition, any inequality  $\mathbf{\alpha}^T \mathbf{x} \leq \alpha_0$ ,  $\mathbf{\alpha} \in \mathbb{R}^{|\mathscr{R}|}$ ,  $\alpha_0 \in \mathbb{R}$ , can be rewritten as  $\overline{\mathbf{\alpha}}^T \mathbf{y} \leq \alpha_0$  where  $\overline{\mathbf{\alpha}} = (\beta \alpha_1 + \alpha_0 \overline{w}_1, \beta \alpha_2 + \alpha_0 \overline{w}_2, \dots, \beta \alpha_{|\mathscr{R}|} + \alpha_0 \overline{w}_{|\mathscr{R}|})$ .

*Proof.* It is easy to see that  $z(CF) = \mathbf{c}^T \mathbf{x}/(\beta + \overline{\mathbf{w}}^T \mathbf{x}) = \mathbf{c}^T \mathbf{y}$ . Sice  $\beta > 0$ , we have

$$\boldsymbol{\alpha}^{T}\mathbf{x} \leq \alpha_{0} \Rightarrow \boldsymbol{\alpha}^{T}\mathbf{x} + \frac{\alpha_{0}}{\beta}\overline{\mathbf{w}}^{T}\mathbf{x} \leq \frac{\alpha_{0}}{\beta}(\overline{\mathbf{w}}^{T}\mathbf{x} + \beta) \Rightarrow$$
$$\frac{\boldsymbol{\alpha}^{T}\mathbf{x}}{\overline{\mathbf{w}}^{T}\mathbf{x} + \beta} + \frac{\alpha_{0}}{\beta}\frac{\overline{\mathbf{w}}^{T}\mathbf{x}}{\overline{\mathbf{w}}^{T}\mathbf{x} + \beta} \leq \frac{\alpha_{0}}{\beta} \Rightarrow$$
$$\beta\boldsymbol{\alpha}^{T}\mathbf{y} + \alpha_{0}\overline{\mathbf{w}}^{T}\mathbf{y} \leq \alpha_{0} \Rightarrow (\beta\boldsymbol{\alpha}^{T} + \alpha_{0}\overline{\mathbf{w}}^{T})\mathbf{y} \leq \alpha_{0}. \Box$$

Formulation CF can now be transformed into the following equivalent linear program:

$$(NCF) \quad z(NCF) = \min \sum_{\ell \in \mathscr{R}} c_{\ell} y_{\ell}$$
$$s.t. \sum \overline{a}_{i\ell} y_{\ell} = 1, \quad \forall i \in V_{1}$$
(10)

$$\sum_{\ell \in \mathscr{R}} \overline{a}_{i\ell} y_{\ell} \le 1, \quad \forall i \in V_2$$
(11)

$$\sum_{\ell \in \mathscr{R}} \overline{b}_{\ell} y_{\ell} \le m, \tag{12}$$

$$y_{\ell} \ge 0, \qquad \forall \ell \in \mathscr{R},$$

where  $\overline{a}_{i\ell} = \beta a_{i\ell} + \overline{w}_{\ell}$  and  $\overline{b}_{\ell} = \beta + m\overline{w}_{\ell}$ ,  $\ell \in \mathscr{R}$ . Notice that formulation NCF has the same number of variables and constraints of formulation CF. Clearly, z(NCF) = z(CF) = z(CF).

We denote with DNCF the dual of NCF. The variables of DNCF are given by the vector  $\mathbf{v} = (v_0, v_1, \ldots, v_n)$ , where  $v_1, \ldots, v_{n_1} \in \mathbb{R}$  are associated with constraints (10),  $v_{n_1+1}, \ldots, v_n \leq 0$ , with constraints (11), and  $v_0 \leq 0$  with constraint (12).

Let DCCF be the dual of problem CCF. The variables of DCCF are given by vector  $\boldsymbol{\mu} = (\mu_0, \ldots, \mu_n)$  and variable  $\omega$ , where  $\mu_1, \ldots, \mu_{n_1} \in \mathbb{R}$  are associated with constraints (6),  $\mu_{n_1+1}, \ldots, \mu_n \leq 0$  with constraints (7),  $\mu_0 \leq 0$  with constraint (8), and  $\omega \in \mathbb{R}$  with constraint (9). The following theorem shows how to compute a solution of DCCF given a solution of DNCF.

**Theorem 1** Let **v** be a feasible solution of problem DNCF of cost z(DNCF). A feasible solution  $(\boldsymbol{\mu}, \omega)$  of DCCF of cost z(DCCF) = z(DNCF) can be obtained by setting:

$$\omega = \sum_{i \in V_c} v_i + mv_0, \quad \mu_0 = \beta v_0, \quad \mu_i = \beta v_i - s_i \omega, \forall i \in V_1, \quad \mu_i = \beta v_i, \forall i \in V_2.$$
(13)

*Proof.* The proof is provided in the e-companion to this paper.  $\Box$ 

#### 3.1.1 Dual Ascent Procedure DA

The structure of the new formulation NCF allows to use a bounding procedure, called DA, that is used to compute a near-optimal dual solution of problem NCF. Procedure DA differs from standard column generation methods based on the simplex algorithm as it uses a dual ascent heuristic to solve the master problem (see Baldacci et al. 2010).

The bounding procedure is based on the following theorem.

**Theorem 2** Let us associate penalties  $\lambda_i \in \mathbb{R}$ ,  $\forall i \in V_1$ , with constraints (10),  $\lambda_i \leq 0$ ,  $\forall i \in V_2$ , with constraints (11), and  $\lambda_0 \leq 0$ , with constraint (12). Let  $\mathscr{R}_i = \{\ell \in \mathscr{R} : \overline{a}_{i\ell} > 0\}$ . For each  $i \in V_1$  compute:

$$\phi_i = \pi_i \min_{\ell \in \mathscr{R}_i} \left\{ \frac{c_\ell - \lambda(R_\ell) - \overline{b}_\ell \lambda_0}{\pi(R_\ell)} \right\}$$

where  $\pi_i > 0$  is a weight assigned to customer  $i \in V_1$ ,  $\lambda(R_\ell) = \sum_{i \in V_c} \overline{a}_{i\ell} \lambda_i$  and  $\pi(R_\ell) = \sum_{i \in V_1} \overline{a}_{i\ell} \pi_i$ . A feasible DNCF solution **v** of cost  $z(DNCF(\lambda))$  is given by the following expressions:

$$v_i = \phi_i + \lambda_i, \forall i \in V_1, \quad v_i = \lambda_i, \forall i \in V_2, \quad v_0 = \lambda_0.$$
(14)

*Proof.* See Baldacci et al. (2010).  $\Box$ 

The optimal solution cost of the following problem

$$\max_{\lambda} \{ z(DNCF(\boldsymbol{\lambda})) \}$$
(15)

provides the best possible dual bound which can be computed by means of Theorem 2. In practice, problem (15) cannot be solved even by means of subgradient optimization as the computation of solution  $\mathbf{v}$ , for given vector  $\boldsymbol{\lambda}$  requires the a priori generation of the set  $\mathscr{R}$ . Method DA is an iterative algorithm which computes a dual bound as the cost of a suboptimal solution of problem (15) by using a limited subset  $\overline{\mathscr{R}} \subseteq \mathscr{R}$  and by changing the values of vector  $\lambda$ . At each iteration, DA uses expressions (14) to find a solution  $\mathbf{v}$ , for given  $\lambda$ , of the reduced *DNCF* problem defined on route subset  $\overline{\mathscr{R}}$  instead of  $\mathscr{R}$ . A pricing procedure is used to identify the route subset  $\mathscr{N} \subset \mathscr{R} \setminus \overline{\mathscr{R}}$  whose dual constraints are violated by the current solution  $\mathbf{v}$ . In case  $\mathscr{N} \neq \emptyset$ , then  $\mathbf{v}$  is not a feasible *DNCF* solution, and  $\mathscr{N}$  is added to the current core problem  $\overline{\mathscr{R}}$ . At each iteration, subgradient vectors are computed and used to change vector  $\lambda$  to maximize the value of the dual bound.

Section 5.1 describes the method used to compute set  $\mathcal{N}$  - for the details of method DA the reader is referred to Baldacci et al. (2010).

## 3.2 Dual bounding procedure based on an integer relaxation of F - CB

In this section, we describe an alternative dual bound derived from an integer relaxation of formulation F. The relaxation is based on the observation that the optimal solution  $\cosh z(F)$  of formulation F can be computed as  $z(F) = \min_{\substack{\overline{T} \leq \overline{t} \leq mT \\ |\overline{t}/T| \leq \overline{m} \leq m}} \left\{ z(F(\overline{t}, \overline{m}))/\overline{t} \right\}$ ,

where

$$(F(\overline{t},\overline{m})) \quad z(F(\overline{t},\overline{m})) = \min \sum_{\ell \in \mathscr{R}} c_{\ell} x_{\ell}$$
  
s.t.(2), (3) and  
$$\sum x_{\ell} = \overline{m}, \qquad (16)$$

$$\sum_{\ell \in \mathscr{R}} w_{\ell} x_{\ell} = \overline{t} \tag{17}$$

$$x_{\ell} \in \{0, 1\}, \quad \forall \ell \in \mathscr{R}, \tag{18}$$

and  $\overline{T}$  is a valid dual bound on the total working time of any feasible solution of formulation F. In practice, problem  $F(\overline{t}, \overline{m})$  cannot be solved directly but a valid dual bound on its optimal solution cost  $z(F(\overline{t}, \overline{m}))$  can be obtained as follows.

Let  $\lambda_i, \forall i \in V_c$ , be a set of Lagrangian penalties associated with constraints (2) and (3), where  $\lambda_i \in \mathbb{R}, \forall i \in V_1$ , and  $\lambda_i \leq 0, \forall i \in V_2$ . The Lagrangian relaxation of formulation  $F(\overline{t}, \overline{m})$  by means of penalty vector  $\boldsymbol{\lambda}$ , is as follows:

$$(RF(\overline{t},\overline{m},\boldsymbol{\lambda})) \quad z(F(\overline{t},\overline{m},\boldsymbol{\lambda})) = \min\sum_{\ell \in \mathscr{R}} \overline{c}_{\ell} x_{\ell} + \sum_{i \in V_{c}} \lambda_{i}$$
  
s.t.(16), (17) and (18),

where  $\overline{c}_{\ell} = c_{\ell} - \lambda(R_{\ell})$  with  $\lambda(R_{\ell}) = \sum_{i \in V_c} a_{i\ell} \lambda_i$ . Let  $\mathscr{R}_i^t \subseteq \mathscr{R}$  be the index set of all routes in  $\mathscr{R}$  ending in  $i \in V_c$  and with a total working time t. We have  $\mathscr{R} = \bigcup_{\substack{i \in V_c \\ 1 \leq t \leq T}} \mathscr{R}_i^t$  and  $\mathscr{R}_i^t \cap \mathscr{R}_j^{t'} = \emptyset, \forall i, j \in V_c, i \neq j, t, t' \in [1, T].$  Let  $\varphi_i^t$ ,  $i \in V_c$ ,  $1 \le t \le T$ , be a dual bound on the modified cost  $\overline{c}_\ell$  of any route  $\ell \in \mathscr{R}_i^t$ , i.e.,  $\varphi_i^t \le \overline{c}_\ell$ ,  $\forall \ell \in \mathscr{R}_i^t$ . We have

$$\sum_{\ell \in \mathscr{R}} \overline{c}_{\ell} x_{\ell} = \sum_{i \in V_c} \sum_{t=1}^{T} \sum_{\ell \in \mathscr{R}_i^t} \overline{c}_{\ell} x_{\ell} \ge \sum_{i \in V_c} \sum_{t=1}^{T} \varphi_i^t \sum_{\ell \in \mathscr{R}_i^t} x_{\ell}.$$

We assume  $\varphi_i^t = \infty$ ,  $i \in V_c$ , if no feasible routes ending in  $i \in V_c$  with a total working time equal to t exists. By setting  $z_i^t = \sum_{\ell \in \mathscr{R}_i^t} x_\ell$ , from relaxation  $RF(\overline{t}, \overline{m}, \lambda)$  we obtain the following relaxation:

$$(\overline{RF}(\overline{t},\overline{m},\boldsymbol{\lambda})) \quad z(\overline{RF}(\overline{t},\overline{m},\boldsymbol{\lambda})) = \min\sum_{i\in V_c} \sum_{t=1}^{T} \varphi_i^t z_i^t$$
$$s.t. \sum_{i\in V_c} \sum_{t=1}^{T} t z_i^t = \overline{t},$$
(19)

$$\sum_{i \in V_c} \sum_{t=1}^{T} z_i^t = \overline{m},\tag{20}$$

$$\sum_{t=1}^{T} z_i^t \le 1, \qquad \forall i \in V_c$$
$$z_i^t \in \{0, 1\}, \qquad \forall i \in V_c, \forall t, 1 \le t \le T.$$

Problem  $RF(\overline{t}, \overline{m}, \lambda)$  can be solved by DP as follows. Let  $g_i(t, k)$  be the optimal solution to problem  $RF(\overline{t}, \overline{m}, \lambda)$  with the right-hand-side of equation (19) replaced by t, the righthand-side of equation (20) replaced by k, and with  $z_j^t = 0$  for j > i, t = 1, ..., T. Function  $g_i(t, k)$  can be computed as follows:

$$g_i(t,k) = \min\left\{g_{i-1}(t,k), \min_{1 \le t' \le \min\{t-1,T\}} \{g_{i-1}(t-t',k-1) + \varphi_i^{t'}\}\right\}$$
(21)

for  $k = 2, ..., \overline{m}, i = k, ..., n - \overline{m} + k$ , and  $\forall t, \max\{1, \overline{T} - (\overline{m} - k)T\} \leq t \leq \min\{kT, \overline{t}\}$ . The following initialization is required:  $g_i(t, 1) = \min\{g_{i-1}(t, 1), \varphi_i^t\}, i = 2, ..., n, \max\{1, \overline{T} - (\overline{m} - 1)T\} \leq t \leq \min\{T, \overline{t}\}$  and  $g_1(t, 1) = \varphi_i^t, \max\{1, \overline{T} - (\overline{m} - 1)T\} \leq t \leq \min\{T, \overline{t}\}, g_i(t, i + 1) = \infty, i = 1, ..., \overline{m} - 1, \forall t \in [0, \overline{t}]$ . The optimal value  $z(\overline{RF}(\overline{t}, \overline{m}, \lambda))$  can then be computed as  $z(\overline{RF}(\overline{t}, \overline{m}, \lambda)) = g_n(\overline{t}, \overline{m})$ .

### 3.2.1 Bounding procedure CB

Based on relaxation  $RF(\overline{t}, \overline{m}, \lambda)$  we designed a bounding procedure, called CB. Procedure CB is based on the observation that all functions  $g_n(\overline{t}, \overline{m})$  can be computed using the DP recursion (21) with  $\overline{t} = mT$  and  $\overline{m} = m$  and for  $k = 2, \ldots, m$ ,  $i = k, \ldots, n$ , and  $\forall t$ ,  $\max\{1, \overline{T} - (m - k)T\} \leq t \leq \min\{kT, mT\}$ . Further, Procedure CB uses subgradient optimization to maximize the value of dual bound  $z(RF(\overline{t}, \overline{m}, \lambda))$ . Bounding procedure CB works as follows:

- Step 1. Initialization. Initialize the penalty vector  $\boldsymbol{\lambda} = \boldsymbol{0}$ . Set  $DB_{CB} = -\infty$ , i = 1,  $DB(\overline{t},\overline{m}) = -\infty$ ,  $\forall \overline{t}, 0 \leq \overline{t} \leq T$ ,  $\forall \overline{m}, 0 \leq \overline{m} \leq m$ .
- Step 2. Compute functions  $\varphi_i^t$ . Compute functions  $\varphi_i^t$ ,  $\forall i \in V_c$ ,  $\forall t, \max\{1, \overline{T} (m-1)T\} \le t \le T$  (see Section 5.1) and set  $\varphi_i^t = \infty \ \forall i \in V_c$ ,  $\forall t, t < \max\{1, \overline{T} (m-1)T\}$ .
- Step 3. Dual bound computation. Compute function  $g_i(t,k)$  using DP recursion (21) with  $\overline{t} = mT$  and  $\overline{m} = m$ . Compute  $DB(\overline{t},\overline{m}) = \max\left\{DB(\overline{t},\overline{m}), \frac{g_n(\overline{t},\overline{m})}{\overline{t}}\right\}, \forall \overline{t}, 1 \leq \overline{t} \leq T, \forall \overline{m}, 1 \leq \overline{m} \leq m$ . Compute

$$z^* = \min_{\substack{\overline{T} \le \overline{t} \le mT \\ [\overline{t}/T] \le \overline{m} \le m}} \left\{ DB(\overline{t}, \overline{m}) \right\}$$
(22)

and let  $t^*$  and  $m^*$  be the values producing  $z^*$  in expression (22). If  $z^* > DB_{CB}$ , set  $DB_{CB} = z^*$ .

Step 4. Update the penalty vector  $\lambda$ . Compute the solution corresponding to dual bound  $z^*$  by backtracking using recursions (21) and (29) and values  $t^*$  and  $m^*$ . Let  $\overline{\mathscr{R}}$  be the set of (NG, t, i)-route selected in solution.

Let  $\theta_i$  be the number of times that customer  $i \in V_c$  is visited by the routes in  $\overline{\mathscr{R}}$ , i.e.  $\theta_i = \sum_{\ell \in \overline{\mathscr{R}}} a_{il}$ . The value of  $\lambda$  is modified as follows:  $\lambda_i = \lambda_i - \epsilon \gamma(\theta_i - 1)$ ,  $\forall i \in V_1, \lambda_i = \min\{0, \lambda_i - \epsilon \gamma(\theta_i - 1)\}, \forall i \in V_2$ , where  $\epsilon$  is a positive constant and  $\gamma = |0.2z^*|/(\sum_{i \in V_c} (\theta_i - 1)^2)$ .

Step 5. Termination criteria. Set i = i + 1. If i = Maxit3, stop, otherwise go to Step 2.

At the end of the procedure, the value of  $DB_{CB}$  represent the best dual bound computed by the procedure. Let  $m_{min}$  and  $m_{max}$  be the minimum and maximum values of  $\overline{m}$ ,  $\lceil \overline{T}/T \rceil \leq \overline{m} \leq m$ , such that  $\min_{\overline{T} \leq \overline{t} \leq mT} \{DB(\overline{t}, \overline{m})\} < UB$ , where UB is a given primal bound on the optimal VRPFO solution cost. Values  $m_{min}$  and  $m_{max}$  represent valid dual and primal bounds on the number of vehicles used in any optimal VRPFO solution.

# 4 An exact method for solving the VRPFO

The VRPFO can be solved to optimality using the extension of the method proposed by Dinkelbach (1967) to integer problem once a generic exact method for solving problem (3)-(5) with a linear objective function is available; to our knowledge, this problem has never been addressed in the literature. A main drawback of this procedure is that the generic exact method must be applied from scratch at each iteration of the Dinkelbach's algorithm.

In this section, we describe an exact method that combines the bounding procedures DA and CB described in Section 3.1 and Section 3.2, respectively, to a priori generate a reduced F problem containing all routes of any optimal solution, thus reducing the dimensions of the integer problems solved at each iteration of the Dinkelbach's algorithm. The exact method relies on a technique, called *route enumeration*, used by Baldacci et al. (2008) to solve the CVRP and by Baldacci et al. (2011) to solve both the CVRP and the VRPTW. In particular, in Section 4.1, we revisit the technique for fractional linear objectives.

More precisely, in the exact method we use bounding procedure DA to obtain a nearoptimal solution  $(\boldsymbol{\mu}, \omega)$  of the dual problem DCCF of cost z(DCCF). This solution allows us to compute the reduced costs  $\overline{c}_{\ell} = c_{\ell} - \sum_{i \in V_c} a_{i\ell}\mu_i - \mu_0 - w_{\ell}\omega$  of each route  $\ell \in \mathscr{R}$ . Whenever the reduced cost  $c_{\ell}$  of a route  $\ell \in \mathscr{R}$  exceeds a given threshold, computed as a function of a known primal bound UB and the dual bound z(DCCF), it is possible to eliminate route  $\ell$  from  $\mathscr{R}$ . Nevertheless, the resulting F might still be too large to be solved exactly. We propose an iterative procedure for solving the VRPFO where at each iteration a reduced F problem is solved. Further, each reduced F is solved to optimality using an exact method based on the extension of the method proposed by Dinkelbach (1967) to integer problem. The procedure terminates when either an optimal F solution is achieved or the distance of the solution cost of the reduced F problem from the dual bound is less than a user-defined value or a maximum number of iterations is reached.

The bounding procedures DA and CB described in Section 3.1 and Section 3.2 respectively, are interwoven with a *Lagrangean heuristic* that produces a feasible VRPFO solution. More specifically, whenever an improved dual bound has been computed (see Step 3 of both procedure DA and CB), the procedure calls an algorithm that produces a feasible VRPFO solution using the route set  $\overline{\mathscr{R}}$ .

In the following, we describe the details of the exact method.

### 4.1 Variable reduction of formulation F

The aim of this section is to identify a criterion to restrict ourself to only the column that can be potentially part of the optimal solution, we call such procedure variable reduction. Let  $(\boldsymbol{\mu}, \omega)$  be a feasible solution of DCCF of cost z(DCCF) equal to  $\omega$ , since  $\omega$  is the only variable in the objective function with coefficient one. Let  $\overline{\mathbf{x}}$  be a feasible solution of F of cost  $\overline{z}(F)$  and let  $\overline{c}_{\ell}$  be the reduced cost of route  $\ell \in \mathscr{R}$  with respect to the dual solution  $(\boldsymbol{\mu}, \omega)$ , that is  $\overline{c}_{\ell} = c_{\ell} - \sum_{i \in V_c} a_{i\ell}\mu_i - \mu_0 - w_{\ell}\omega$ , and let  $\overline{c}_0 = \sum_{i \in V_c} \mu_i + m\mu_0$ .

The variable reduction of formulation F is based on the following Theorem 5:

**Theorem 3** Let  $\overline{\mathscr{R}} = \{\ell : \overline{x}_{\ell} = 1, \ell \in \mathscr{R}\}$ . The following inequality holds:

$$\overline{z}(F) \ge z(DCCF) + \frac{\sum_{\ell \in \overline{\mathscr{R}}} \overline{c}_{\ell} + \overline{c}_0}{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}}.$$
(23)

*Proof.* From the definition of  $\overline{z}(F)$  we have:

$$\overline{z}(F) = \frac{\sum_{\ell \in \overline{\mathscr{R}}} c_{\ell}}{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}} = \frac{\sum_{\ell \in \overline{\mathscr{R}}} (\sum_{i \in V_c} a_{i\ell} \mu_i + \mu_0 + w_{\ell} \omega + \overline{c}_{\ell}) - \sum_{i \in V_c} \mu_i - m\mu_0 + \overline{c}_0}{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}} = \omega \frac{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}}{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}} + \frac{\sum_{i \in V_c} (\sum_{\ell \in \overline{\mathscr{R}}} a_{i\ell} - 1)\mu_i + \sum_{\ell \in \overline{\mathscr{R}}} (1 - m)\mu_0 + \sum_{\ell \in \overline{\mathscr{R}}} \overline{c}_{\ell} + \overline{c}_0}{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}}.$$
(24)

Since  $\overline{x}$  represents a feasible VRPFO solution, we have (i)  $\sum_{\ell \in \overline{\mathscr{R}}} (a_{i\ell} - 1) = 0, \forall i \in V_1$ , (ii)  $\sum_{\ell \in \overline{\mathscr{R}}} (a_{i\ell} - 1) \leq 0, \forall i \in V_2$ , and (iii)  $\sum_{\ell \in \overline{\mathscr{R}}} (1 - m) \leq 0$ , therefore as  $\mu_i \leq 0, \forall i \in V_2$ , and  $\mu_0 \leq 0$ 

$$\sum_{i \in V_c} (\sum_{\ell \in \overline{\mathscr{R}}} a_{i\ell} - 1) \mu_i + \sum_{\ell \in \overline{\mathscr{R}}} (1 - m) \mu_0 = \sum_{i \in V_c} (\sum_{\ell \in \overline{\mathscr{R}}} a_{i\ell} - 1) \mu_i + \sum_{i \in V_c} (\sum_{\ell \in \overline{\mathscr{R}}} a_{i\ell} - 1) \mu_i + \sum_{\ell \in \overline{\mathscr{R}}} (1 - m) \mu_0 \ge 0.$$

$$(25)$$

From equation (24) and inequality (25) we obtain (23).  $\Box$ 

**Corollary 1** Let UB be the cost of a feasible VRPFO solution and let z(DCCF) be the cost of a feasible dual solution  $(\boldsymbol{\mu}, \omega)$  of DCCF. Any optimal solution x of cost z(F) less than UB cannot contain any route  $\ell \in \mathscr{R}$  such that  $\overline{c}_{\ell} \geq \alpha_{\ell} UB - (\alpha_{\ell} z(DCCF) + \overline{c}_0)$ , where  $\alpha_{\ell} = w_{\ell} + (m-1)T$  is an upper bound on the total working time of any solution containing route  $\ell$ , since m is the maximum number of vehicles in solution.

*Proof.*(By contradiction) Let  $\overline{\mathscr{R}}$  be the index set of the routes of the feasible solution x of cost z(F) < UB and suppose that exists  $\ell' \in \overline{\mathscr{R}}$  such that  $\overline{c}_{\ell'} \ge \alpha_{\ell'}UB - (\alpha_{\ell'}z(DCCF) + \overline{c}_0)$ .

From Theorem 5 and as  $\overline{c}_{\ell} \geq 0$ ,  $\forall \ell \in \overline{\mathscr{R}}$ , and since  $\alpha_{\ell'} \geq \sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}$  we have:

$$\begin{aligned} z(F) \geq z(DCCF) + \frac{\sum_{\ell \in \overline{\mathscr{R}}} \overline{c}_{\ell} + \overline{c}_{0}}{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}} \geq z(DCCF) + \frac{\overline{c}_{\ell'} + \overline{c}_{0}}{\sum_{\ell \in \overline{\mathscr{R}}} w_{\ell}} \geq z(DCCF) + \frac{\overline{c}_{\ell'} + \overline{c}_{0}}{\alpha_{\ell'}} \geq z(DCCF) + \frac{\alpha_{\ell'}UB - (\alpha_{\ell'}z(DCCF) + \overline{c}_{0}) + \overline{c}_{0}}{\alpha_{\ell'}} \geq UB. \ \Box \end{aligned}$$

Corollary 1 will be used in the exact procedure presented in Section 4.2 to generate only the columns with a reduced cost lower than the gap  $\alpha_{\ell}UB - (\alpha_{\ell}z(DCCF) + \overline{c}_0)$ .

#### 4.2 Description of the exact method

The method is based on an iterative procedure where at each iteration a reduced version of F involving at most  $\Delta^{max}$  routes ( $\Delta^{max}$  is a user-defined parameter) is solved. This procedure terminates when one of the following three conditions is encountered: (i) an optimal F solution is achieved, (ii) the distance of the solution cost of the reduced Fproblem from the dual bound is less than the user defined value, gapmax, and (iii) the maximum number of iterations, *itermax*, is reached. The scheme of the proposed exact method for solving F is as follows.

- Step 1. Initialization. Set i = 0 and  $z^* = \infty$ . Initialize itermax,  $\Delta^{max}$  and the time.
- Step 2. Execute bounding procedure CB. Let  $DB_{CB}$  and  $PB_{CB}$  be the final dual and primal bounds computed, respectively. Let  $m_{min}$  and  $m_{max}$  be computed as described in Section 3.2.1.
- Step 3. Execute bounding procedure DA. Let  $DB_{DA}$  and  $PB_{DA}$  be the final dual and primal bounds computed and let  $(\boldsymbol{\mu}, \omega)$  be the solution of problem DCCF computed by means of Theorem 1 using solution  $\overline{\mathbf{v}}$  corresponding to  $DB_{DA}$ . Let  $\overline{c}_{\ell}$  be the reduced cost of route  $\ell \in \mathscr{R}$  with respect to the dual solution  $(\boldsymbol{\mu}, \omega)$ , that is  $\overline{c}_{\ell} = c_{\ell} - \sum_{i \in V_c} a_{i\ell} \mu_i - \mu_0 - w_{\ell} \omega$ , and let  $\overline{c}_0 = \sum_{i \in V_c} \mu_i + m\mu_0$ . Set  $z^* = \min\{PB_{CB}, PB_{DA}\}$ and let  $\overline{\mathbf{x}}$  be the corresponding solution.
- Step 4. Define a reduced problem  $F(\overline{\mathscr{R}})$  from F. Set i = i + 1. Generate the largest route set  $\overline{\mathscr{R}} \subseteq \mathscr{R}$  such that:

$$\begin{array}{l} a) \quad |\overline{\mathscr{R}}| \leq \Delta^{max}, \\ b) \quad \overline{c}_{\ell} < \gamma_{\ell}, \forall \ell \in \overline{\mathscr{R}}, \end{array}$$

$$(26)$$

where  $\gamma_{\ell} = \alpha_{\ell} z^* - (\alpha_{\ell} DB_{\mathsf{DA}} + \overline{c}_0)$  and  $\alpha_{\ell} = w_{\ell} + (m_{max} - 1)T$ . Based on Corollary 1, if  $|\overline{\mathscr{R}}| < \Delta^{max}$ , then  $\overline{\mathscr{R}}$  contains the routes of any optimal solution and is defined *optimal*. Problem  $F(\overline{\mathscr{R}})$  is obtained from problem F where the route set  $\mathscr{R}$  is substituted with  $\overline{\mathscr{R}}$ .

- Step 5. Solve problem  $F(\overline{\mathscr{R}})$ . Let  $\overline{\mathbf{x}}$  be the solution obtained and let  $\overline{z}(F(\overline{\mathscr{R}}))$  be its cost (we assume that  $\overline{\mathscr{R}}$  contains also the routes corresponding to the current solution  $\mathbf{x}^*$ ); we impose a time limit of *tlim* seconds in solving  $F(\overline{\mathscr{R}})$ . Problem  $F(\overline{\mathscr{R}})$ is defined *optimal* if it has been solved to optimality within the imposed time limit; otherwise it is defined *not optimal* (see Section 4.3). If  $z^* > \overline{z}(F(\overline{\mathscr{R}}))$  set  $z^* = \overline{z}(F(\overline{\mathscr{R}}))$  and  $\mathbf{x}^* = \overline{\mathbf{x}}$ .
- Step 6. Test if the  $F(\overline{\mathscr{R}})$  solution obtained is an optimal VRPFO solution. Let gapmin be a dual bound on the reduced cost of any route that has not been generated, i.e.,  $\overline{c}_{\ell} \geq gapmin$ ,  $\forall \ell \in \mathscr{R} \setminus \overline{\mathscr{R}}$ , and let  $DB_{new}$  be a dual bound on the cost of any solution to F involving one or more routes of the set  $\mathscr{R} \setminus \overline{\mathscr{R}}$ , computed as  $DB_{new} = DB_{DA} + (gapmin + \overline{c}_0)/(m_{max}T)$ . If one of the following two conditions applies, then  $\mathbf{x}^*$  is guarantee to be an optimal VRPFO solution and the algorithm terminates:
  - (i) Both  $\overline{\mathscr{R}}$  and  $F(\overline{\mathscr{R}})$  are *optimal*, or

(ii)  $F(\overline{\mathscr{R}})$  is optimal and  $z^* \leq DB_{new}$ .

- Step 7. Termination condition. If  $i \ge itermax$  or  $(z^* DB_{new})/DB_{new} \le gapmax$ , then Stop.
- Step 8. Updating  $\Delta^{max}$  and then. Set  $\Delta^{max} = \varepsilon_1 \Delta^{max}$  ( $\varepsilon_1 > 1$ ) and the thin the theorem  $(\varepsilon_2 > 0)$  and go to Step 4 ( $\varepsilon_1$  and  $\varepsilon_2$  are two user-defined parameters).

The exact method starts with the use of both bounding procedures CB and DA to compute valid primal and dual bounds for our problem (Steps 2 and 3). As explained in Section 3.1, procedure DA provides a good approximation of problem DNCF. Thanks to Theorem 1 it is hence possible to obtain  $(\mu, \omega)$ , the corresponding dual variables of DCCF. These new duals allow to compute the reduces costs  $\overline{c}_{\ell}, \forall \ell \in \mathscr{R}$ .

Step 4 selects the  $\Delta^{max}$  columns to be used in the solution of  $F(\overline{\mathscr{R}})$ , Corollary 1 allows to restrict set  $\overline{\mathscr{R}}$  to only useful columns. It is worth noting that the primal bounds computed in Steps 1 and 2 play an important role in the identification of such columns. In Step 5, problem  $F(\overline{\mathscr{R}})$  is solved to optimality (up to a given time limit). The exact procedure used to solve it is explained in Section 4.3. If the primal solution is better than the best primal solution found so far,  $z^*$  is updated.

In Step 6, the optimality of the solution of  $F(\overline{\mathscr{R}})$  is checked. If the number of columns added to  $\overline{\mathscr{R}}$  in Step 5 was lower than  $\Delta^{max}$  (i.e.,  $\overline{\mathscr{R}}$  is optimal) and we were able to solve  $F(\overline{\mathscr{R}})$  within the given time limit, then the algorithm terminates. On the other hand, if  $F(\overline{\mathscr{R}})$  is solved to optimality but some of the columns with  $\overline{c}_{\ell} < \gamma_{\ell}$  has been excluded from  $\overline{\mathscr{R}}$ , a new dual bound  $DB_{new}$  is computed, based on the columns that have not been added to  $\overline{\mathscr{R}}$ . If  $z^* \leq DB_{new}$ , the optimal solution of  $F(\overline{\mathscr{R}})$  is also an optimal solution of  $F(\mathscr{R})$ .

Step 7 terminates the algorithm if either the number of maximum iterations is reached or the relative gap between the best primal and dual bounds is sufficiently small. Finally, in Step 8,  $\Delta^{max}$  and the time limit are updated.

Notice that whenever the algorithm terminates at Step 7, problem F has not been solved to optimality and value  $z^*$  represents the cost of the best solution found. The next section describes the method used to solve problem  $F(\overline{\mathscr{R}})$  whereas the procedure used to generate the reduced set of routes  $\overline{\mathscr{R}}$  is described in Section 5.2.

## 4.3 Solving problem $F(\overline{\mathscr{R}})$ to optimality

Problem  $F(\mathscr{R})$  can be rewritten as the following mixed integer linear programming problem using the transformation proposed by Charnes and Cooper (1962):

$$(F(\overline{\mathscr{R}})) \quad z(F(\overline{\mathscr{R}})) = \min \sum_{\ell \in \overline{\mathscr{R}}} c_{\ell} y_{\ell}$$
$$s.t.(6) - (9)$$

$$y_{\ell} \le u, \qquad \qquad \forall \ell \in \overline{\mathscr{R}} \tag{27}$$

$$u - M(1 - x_{\ell}) \le y_{\ell} \le M x_{\ell}, \quad \forall \ell \in \overline{\mathscr{R}}$$

$$u \ge 0,$$
(28)

$$y_{\ell} \ge 0, x_{\ell} \in \{0, 1\} \qquad \qquad \forall \ell \in \overline{\mathscr{R}},$$

where in constraints (6)-(9) set  $\mathscr{R}$  is substituted with  $\overline{\mathscr{R}}$  and M is a very large positive number. The above formulation is impractical to solve, even for moderate size VRPFO instances, since the numbers of variables and of the additional constraints (27) and (28) can be huge.

We now describe an exact method for solving problem  $F(\overline{\mathscr{R}})$  based on the extension of the method proposed by Dinkelbach (1967) to integer problems. For sake of notation, we rewrite problem  $F(\overline{\mathscr{R}})$  as  $(F(\overline{\mathscr{R}})) = z(F(\overline{\mathscr{R}})) = \min\{n(\mathbf{x})/d(\mathbf{x}) : \mathbf{x} \in P\}$ , where  $n(\mathbf{x}) = \sum_{\ell \in \overline{\mathscr{R}}} c_{\ell} x_{\ell}, d(\mathbf{x}) = \sum_{\ell \in \overline{\mathscr{R}}} w_{\ell} x_{\ell}, X = \{\mathbf{x} \in \mathbb{R}^{|\overline{\mathscr{R}}|}_{+} : (2), (3), \text{ and } m_{min} \leq \sum_{\ell \in \overline{\mathscr{R}}} x_{\ell} \leq m_{max}\}$ and  $P = X \cap \{0, 1\}^{|\overline{\mathscr{R}}|}$ . We assume that  $d(\mathbf{x}) > 0, \forall \mathbf{x} \in P$ , and that  $F(\overline{\mathscr{R}})$  admits a finite optimal solution.

The exact method is based on the following theorem (see Dinkelbach 1967).

**Theorem 4**  $\overline{r} = n(\overline{\mathbf{x}})/d(\overline{\mathbf{x}}) = z(F(\overline{\mathscr{R}}))$  if, and only if, for the parametric problem FP(r),  $z(FP(r)) = \min\{n(\mathbf{x}) - rd(\mathbf{x}) : \mathbf{x} \in P\}, \ z(FP(\overline{r})) = n(\overline{\mathbf{x}}) - \overline{r}d(\overline{\mathbf{x}}) = 0.$ 

*Proof.* The proof is provided in the e-companion to this paper.  $\Box$ 

The scheme of the proposed exact method for solving  $F(\overline{\mathscr{R}})$  is as follows.

- Step 1. Initialization. Set  $\mathbf{x}^0 = \mathbf{x}^*$ , where  $\mathbf{x}^*$  is the current best know solution (see Step 5 of the exact method). Set i = 0.
- Step 2. Compute the current ratio  $r_{i+1}$ . Set  $r_{i+1} = n(\mathbf{x}^i)/d(\mathbf{x}^i)$  and set i = i+1.
- Step 3. Solve the parametric problem. Solve problem  $FP(r_i)$ ,  $z_i(FP(r_i)) = \min\{n(\mathbf{x}) r_i d(\mathbf{x}) : \mathbf{x} \in P\}$ , and let  $\mathbf{x}^i$  be the solution obtained.
- Step 4. Termination condition. If  $z_i(FP(r_i)) < 0$ , go to Step 2; otherwise, set  $\overline{z}(F(\overline{\mathscr{R}})) = r_i, \, \overline{\mathbf{x}} = \mathbf{x}^i, \, \overline{r} = r_i$  and if solution  $\mathbf{x}^i$  is an optimal solution of problem  $FP(r_i)$ , define  $F(\overline{\mathscr{R}})$  optimal; otherwise define  $F(\overline{\mathscr{R}})$  not optimal.

In solving problem  $FP(r_i)$ , the time limit of *tlim* seconds introduced at Step 5 of the exact method is imposed, therefore solution  $\mathbf{x}^i$  is a proven optimal solution of  $FP(r_i)$  if the problem has been solved within the imposed time limit.

The following Lemma 2 revisits a result from Espinoza et al. (2010) to show the correctness of the above iterative algorithm by also considering the termination conditions of Step 4.

**Lemma 2** The following properties hold about the exact algorithm for solving  $F(\overline{\mathscr{R}})$ .

- (1) The sequence  $\{r_i\}$  is monotone decreasing, i.e.,  $r_i > r_{i+1}$ , for all i such that  $z_i(FP(r_i)) < 0$ .
- (2) If  $z_i(FP(r_i)) \ge 0$  and solution  $\mathbf{x}^i$  is optimal, the value  $\overline{z}(F(\overline{\mathscr{R}}))$  corresponds to the optimal solution value and  $\overline{r} = n(\overline{\mathbf{x}})/d(\overline{\mathbf{x}})$ .
- (3) If  $z_i(FP(r_i)) < 0$ , we have  $d(\mathbf{x}^i) > d(\mathbf{x}^{i+1})$ .

*Proof.* The proof is provided in the e-companion to this paper.  $\Box$ 

Based on the lemma above, the following theorem shows that the convergence rate of the algorithm is superlinear (Espinoza et al. 2010).

**Theorem 5** For all  $r_i \neq \overline{r}$  we have

$$\frac{\overline{r} - r_{i+1}}{\overline{r} - r_i} \le 1 - \frac{d(\overline{\mathbf{x}})}{d(\mathbf{x}^i)} < 1.$$

*Proof.* The proof is provided in the e-companion to this paper.  $\Box$ 

# 5 Pricing problem and generation of sets $\mathscr{R}$

In this section, we describe the details of the pricing problem associated with bounding procedures DK, CG and DA and the procedure used to compute functions  $\varphi_i^t$  in bounding procedure CB. Moreover, we describe the details of the procedure used to generate sets  $\overline{\mathcal{R}}$  in the exact method.

## 5.1 Route relaxation ng-routes

The pricing problem associated with procedure DK, CG and DA requires to find minimum cost elementary routes over a graph with both positive and negative edge and arc costs, a strongly  $\mathcal{NP}$ -hard problem. Therefore, in practice we enlarge the set of routes  $\mathscr{R}$  to contain also *non-necessarily elementary* routes, i.e., coefficients  $a_{i\ell}$  are general nonnegative integers. Although non-elementary routes are infeasible, this relaxation has the advantage that the pricing subproblem becomes solvable efficiently (by DP). Moreover, Theorem 2 remains valid if the set of routes  $\mathscr{R}$  is enlarged to contain also non-necessarily elementary routes. The relaxation we used is based on the route relaxation proposed by Baldacci et al. (2011) for the VRPTW and can be described as follows.

Let  $N_i \subseteq V_c$  be a set of selected customers for vertex i (according to some criterion) such that  $N_i \ni i$  and  $|N_i| \le \Delta(N_i)$ , where  $\Delta(N_i)$  is a parameter (e.g.,  $\Delta(N_i) = 5$ ,  $\forall i \in V_c$ , and  $N_i$  contains i and the four nearest customers to i). The sets  $N_i$  allow us to associate with each forward path  $P = (0, i_1, \ldots, i_k)$  the subset  $\Pi(P) \subseteq V(P)$ ,  $V(P) = \{0, i_1, \ldots, i_{k-1}, i_k\}$ , containing customer  $i_k$  and every customer  $i_r$ ,  $r = 1, \ldots, k-1$ , of P that belongs to all sets 
$$\begin{split} N_{i_{r+1}},\ldots,N_{i_k} \text{ associated with the customers } i_{r+1},\ldots,i_k \text{ visited after } i_r. \text{ The set } \Pi(P) \text{ is defined as: } \Pi(P) = \{i_r:i_r\in\bigcap_{s=r+1}^kN_{i_s},r=1,\ldots,k-1\}\cup\{i_k\}. \text{ A } ng\text{-}path\ (NG,t,i) \text{ is a non-necessarily elementary path } P = (0,i_1,\ldots,i_{k-1},i_k=i) \text{ starting from the depot at time } 0, \text{ visiting a subset of customers (even more than once) such that } NG = \Pi(P), \text{ ending at customer } i \text{ at time } t, \text{ and such that } i\notin\Pi(P'), \text{ where } P' = (0,i_1,\ldots,i_{k-1}) \text{ is a n} ng\text{-path. We denote by } f(NG,t,i) \text{ the cost of the least cost } ng\text{-path}\ (NG,t,i). \text{ An } (NG,t,i)\text{-route is an } (NG,t,0)\text{-path visiting at time } t \text{ the last customer } i \text{ before arriving at the depot. The cost of the least cost}\ (NG,t,i)\text{-route is given by } f(NG,t,i) + d_{i0}. \text{ Functions } f(NG,t,i) \text{ can be computed using DP as follows. The state space graph } \mathcal{H} = (\mathscr{E},\Psi) \text{ is defined as follows: } \mathscr{E} = \{(NG,t,i): \forall NG \subseteq N_i \text{ s.t. } NG \ni i, \forall t, 0 \leq t \leq T, \forall i \in V\}, \Psi = \{((NG',t',j),(NG,t,i)): \forall (NG',t',j) \in \Psi^{-1}(NG,t,i), \forall (NG,t,i) \in \mathscr{E}\}, \text{ where } \Psi^{-1}(NG,t,i) = \{(NG',t-s_i-t_{ji},j): \forall NG' \subseteq N_j \text{ s.t. } NG' \ni j \text{ and } NG' \cap N_i = NG \setminus \{i\}, t-s_i-t_{ji} \geq 0, \forall j \in V \setminus \{i\}\}. \end{split}$$

The DP recursion for computing f(NG, t, i) is as follows:

$$f(NG,t,i) = \min_{(NG',t',j)\in\Psi^{-1}(NG,t,i)} \{f(NG',t',j) + d_{ji}\}, \quad \forall (NG,t,i) \in \mathscr{E}.$$
 (29)

The following initialization is required:  $f(\{0\}, 0, 0) = 0$  and  $f(\{0\}, t, 0) = \infty$ ,  $\forall t$  such that  $0 < t \leq T$ .

## Computing functions $\varphi_i^t$ at Step 2 of algorithm CB

We first compute functions f(NG, t, i) using DP recursion (29) and the modified costs  $\overline{d}_{ij}$ instead of  $d_{ij}$ , where  $\overline{d}_{ij} = d_{ij} - \lambda_j$ . Then we compute functions  $\varphi_i^t$ ,  $\forall i \in V_c$ ,  $\forall t, \max\{1, \overline{T} - (m-1)T\} \leq t \leq T$ , as  $\varphi_i^t = \min_{(NG, t-t_{i0}, i) \in \mathscr{E}} \{f(NG, t-t_{i0}, i) + \overline{d}_{i0}\}$ .

## 5.2 Generating set $\overline{\mathscr{R}}$ : procedure genr

The generation of the reduced route set  $\overline{\mathscr{R}}$  performed at Step 4 is based on a similar procedure proposed by Baldacci et al. (2011) for the VRP with Time Windows, used to generate elementary and feasible routes. Given a dual solution  $(\boldsymbol{\mu}, \omega)$  of *DCCF* and a user defined parameter  $\Delta^{max}$ , the procedure generates the largest subset  $\overline{\mathscr{R}} \subseteq \mathscr{R}$  satisfying conditions (26)-a and (26)-b. Procedure genr is a DP programming that is analogous to Dijkstra's algorithm on an expanded state-space graph dynamically generated.

Associate with each arc  $(i, j) \in A$  modified arc cost  $\overline{d}_{ij}$  defined as  $\overline{d}_{ij} = d_{ij} - \mu_j - (t_{ij} + s_j)\omega$ . It is easy to see that the reduced cost with respect to the dual vector  $\boldsymbol{\mu}$  and value  $\omega$  can be computed as  $\overline{c}_{\ell} = \sum_{(i,j)\in A(R_{\ell})} \overline{d}_{ij}$ . Procedure genr dynamically generates a state-space graph where each state corresponds to a feasible forward path.

A forward path  $P = (0, i_1, \ldots, i_{k-1}, i_k)$  is an elementary path starting from depot 0 at time 0, visiting vertices  $V(P) = \{0, i_1, \ldots, i_{k-1}, i_k\}$  and ending at customer  $i_k = \sigma(P)$ at time t(P) with  $t(P) \leq T$ . We denote by A(P) the set of arcs traversed by path P and by  $c(P) = \sum_{(i,j)\in A(P)} d_{ij}$  the cost of path P. Let DB(P) be a dual bound on the reduced cost of any route that contains a forward path P. Any forward path P such that  $DB(P) \ge \gamma$  cannot be part of a route in the set  $\overline{\mathscr{R}}$  that satisfies conditions (26), where  $\gamma = \alpha z^* - (\alpha DB_{\mathsf{DA}} + \overline{c}_0)$  and  $\alpha = m_{max}T$ .

Let  $\tau$  be a set of temporary feasible forward paths that is initialized by setting  $\tau = \{P_0\}$ , where  $P_0$  represents the initial empty path such that  $\sigma(P_0) = 0$  and  $t(P_0) = 0$ . The route set  $\overline{\mathscr{R}}$  is initialized by setting  $\overline{\mathscr{R}} = \emptyset$ . At each iteration of algorithm genr the forward path  $P \in \tau$  having the smallest dual bound value (i.e., such that  $DB(P) = \min\{DB(P) :$  $P \in \tau\}$ ) is extracted from the set  $\tau$ . The expansion of a forward path P are derived by extending P with arc  $(\sigma(P), j) \in A, \forall j \notin V(P) \setminus \{0\}$ . We have two cases:

- i) j = 0. The expansion of forward path P creates a route; if the route is feasible and satisfy condition (26)-b it is inserted in the set  $\overline{\mathscr{R}}$ ;
- ii)  $j \neq 0$ . The expansion of path forward path P creates a path P'; if the path is feasible and  $DB(P') < \gamma$  (see above) it is added to the set  $\tau$ .

Procedure genr terminates when either  $\tau = \emptyset$  or  $|\overline{\mathscr{R}}| = \Delta^{max}$ . Since the size of the set  $\tau$  is exponential, we impose that the size of the set  $\tau$  cannot exceed an a-priori defined limit *NSTATB*. If  $|\tau|$  becomes greater than *NSTATB*, procedure genr terminates prematurely. At the end of the procedure, value gapmin is set equal to  $\max_{P \in \tau} \{DB(P)\}$  if  $|\overline{\mathscr{R}}| = \Delta^{max}$  or  $|\tau| = NSTATB$ , and  $\infty$  otherwise.

## **5.2.1** Computing DB(P)

We define a backward path  $\overline{P} = (\sigma(\overline{P}) = i_k, i_{k+1}, \dots, i_h, 0)$  as a path starting from vertex  $\sigma(\overline{P})$  at time  $t(\overline{P})$ , visiting vertices in  $V(\overline{P}) = \{i_k, i_{k+1}, \dots, i_h, 0\}$  and ending at the depot before time T.

A dual bounds on the cost  $c(\overline{P})$  of  $\overline{P}$  can be computed using the ng-path by defining nonnecessarily elementary backward ng - path (NG, t, i) similarly to the forward ng-path (NG, t, i) defined in Section 5.1. Let  $f^{-1}(NG, t, i)$  be the cost of the least-cost backward ng-path (NG, t, i). Functions  $f^{-1}(NG, t, i)$  can be computed with the same DP recursions used to compute f(NG, t, i) by replacing the cost and time matrices  $[d_{ij}]$  and  $[t_{ij}]$  with their transposed matrices  $[d_{ij}]^T$  and  $[t_{ij}]^T$ . We have that the cost  $c(\overline{P})$  of any elementary backward path  $\overline{P}$  satisfies the following inequality:

$$c(\overline{P}) \geq \min_{NG \subseteq V(\overline{P}) \cap N_{\sigma(\overline{P})}} \{ f^{-1}(NG, t(\overline{P}), \sigma(\overline{P})) \}.$$

To compute dual bound DB(P), function  $f^{-1}(NG, t, i)$  are computed with the modified costs  $[\overline{d}_{ij}]$ , and the subsets  $N_i$ ,  $i \in V_c$ , contain the  $\Delta(N_i)$  nearest customers to i according to  $\overline{d}_{ij}$ . Dual bound DB(P) is computed as follows:

$$DB(P) = \sum_{(i,j)\in A(P)} \overline{d}_{ij} + \min_{\substack{NG\subseteq N_{\sigma(P)} \text{ s.t. } NG\cap V(P) = \{\sigma(P)\}}} \{f^{-1}(NG, t', \sigma(P))\}.$$

#### 5.2.2 Dominance rules

A speed-up in procedure **genr** can be obtained by removing dominated paths from the set  $\tau$ . A dominated path is either a path that cannot lead to a feasible route or a path such that any route containing it cannot be part of any optimal solution. Dominance rules are defined based on the type of fractional objective function considered as follows.

**Dominance 1** Minimization of Cost/Load A forward path  $P_1$  dominates a forward path  $P_2$  if  $\sigma(P_1) = \sigma(P_2)$ ,  $V(P_1) = V(P_2)$  and  $c(P_1) \le c(P_2)$ .

**Dominance 2** Maximization of Profit/Time A forward path  $P_1$  dominates a forward path  $P_2$  if  $\sigma(P_1) = \sigma(P_2)$ ,  $V(P_1) = V(P_2)$  and  $t(P_1) \le t(P_2)$ .

## 6 Computational results

This section reports on the computational results of the dual and primal bounds and the exact method described in this paper. All algorithms were coded in C++ and compiled with Microsoft Visual Studio 2013 compiler. The IBM ILOG CPLEX 12.6.4 callable library (IBM CPLEX (2016)) was used as the integer programming solver for solving the parametric problem  $FP(r_i)$  in the exact method (see Section 4.3). All tests were run on a Lenovo ThinkStation P300 (i7-4790 CPU @ 3.6 GHz - 32 GB of RAM) running under Microsoft Windows 7 Professional operating system.

## 6.1 Instances description

The bounding procedures and the exact method were tested on two classes of instances, namely  $[\min |c/l]$  and  $[\max |p/t]$ , corresponding to VRPFO instances with the two objective functions (described in Section 2) minimization of Cost/Load and maximization of Profit/Time respectively.

The instances of the two classes were derived from instances proposed in the literature for the Capacitated VRP (CVRP). More precisely, we considered the 27 instances, and corresponding optimal solutions, of class A generated by Augerat (1995) and available at http://vrp.galgos.inf.puc-rio.br/index.php/en/.

Let  $\text{CVRP}(\overline{n}, [q_i], K, Q, [c_{ij}])$  be a CVRP instance, where  $\overline{n}$  represents the number of vertices (including the depot),  $[q_i]$  the customer demands, K the number of vehicles, Q the vehicle capacity, and  $[c_{ij}]$  the cost matrix; cost matrix  $[c_{ij}]$  is computed according to the

Procedure	Parameters
DA	$\pi_i = s_i, \forall i \in V_c$
Pricing $(NG, t, i)$ -routes	$\Delta(N_i) = 12$ nearest nodes to <i>i</i> according to cost matrix $[d_{ij}]$
CB	$\epsilon = 1.0, Maxit3 = 200$
genr	NSTATB = 200E + 6
Exact method	$itermax = 3, \ \Delta^{max} = 300,000, \ tlim = 3,600, \ gapmax = \infty, \ \varepsilon_1 = 5, \ \varepsilon_2 = 3,600$

Table 1: Parameters used by the different procedures

TSPLIB EUC\_2D standard (see Reinelt (1991)). For each  $CVRP(\overline{n}, [q_i], K, Q, [c_{ij}])$  instance of class A, we generate an instance for each of two classes as follows.

- i) The depot and the set of customers correspond to the depot and the set of customers of the original CVRP instance. We set  $n_1 = \lfloor \alpha(\overline{n} 1) \rfloor$ ,  $\alpha < 1$ ;
- ii) Class  $[\min |c/l]$ . We set  $d_{ij} = c_{ij}$ ,  $\forall (i,j) \in A$ ,  $t_{ij} = 0$ ,  $\forall (i,j) \in A$ , T = Q,  $s_i = q_i$ ,  $\forall i \in V_c$ , and  $m = \min\{BPP(\overline{n} - 1, [q_i], Q) + 1, K\}$ , where  $BPP(\overline{n} - 1, [q_i], Q)$  is the cost of the optimal solution of the Bin Packing Problem instance with  $\overline{n} - 1$  items, weights  $[q_i]$  and bin capacity equal to Q.
- iii) Class  $[\max |p/t]$ . We set  $d_{ij} = -q_i$ ,  $\forall (i, j) \in A$ ,  $t_{ij} = c_{ij}$ ,  $\forall (i, j) \in A$ . Service times  $\{s_i\}$ , maximum working time T and maximum number of vehicles m are computed using the best solution found for the corresponding CVRP instance and a simple heuristic algorithm for the VRPFO, used to guarantee the feasibility of the corresponding instance (details are omitted for sake of brevity).

In generating the instances, we used  $\alpha \in \{0.5, 0.75\}$ , therefore two instances per class are generated for each CVRP instance. Given the original CVRP instance name "< name >", the instance with  $\alpha = 0.5$  is denoted with < name > a whereas the instance with  $\alpha = 0.75$  is denoted with < name > b. A total number of 162 instances were generated, 54 instances per class. All the instances are available upon request to the authors as text files.

Based on the results of preliminary experiments to identify good parameter settings for our algorithms, we decided to use the settings reported in Table 1. The following section reports on the results about the dual and primal bounds computed by procedures DA and CB whereas Section 6.3 shows the results obtained by the exact method.

#### 6.2 Computational results on the dual and primal bounds

To compare dual bounds  $DB_{CB}$  and  $DB_{DA}$  computed by procedures CB and and DA, respectively, we implemented a standard column generation algorithm (called CG) based on formulation CCF and Dinkelbach's algorithm (called DK) applied to formulation CF. Both algorithms CG and DK are based on the (NG, t, i)-routes relaxation and the linear

Class	Pro	cedure	e CB	Procedure DA				Pre	ocedure	CG	Pr	e DK	
	%B	% PB	Time	%B	% PB	$ \mathcal{R} $	Time	%B	$ \mathscr{R} $	Time	Iter	$ \mathscr{R} $	Time
$[\min  c/l]$	98.5	107.0	18.8	98.7	106.3	374.2	0.4	98.8	7683.7	1.2	2.8	7254.2	1.1
$\left[\max  p/t\right]$	106.8	96.3	88.9	106.1	97.7	610.1	0.7	105.9	2282.0	0.7	2.6	2377.5	0.8

Table 2: Summary results on the dual bounds

programming solver of IBM CPLEX is used to solve the master problem at each iteration of algorithm CG.

Table 2 summarises the results obtained about the different dual and primal bounds on the two classes of instances. For bounding procedures CB, DA and CG, the table reports the average percentage deviation of the dual bound (column %B) and the average computing time in seconds (column "Time"). The percentage deviation is computed as  $100.0 \times B/z^*$ , where  $z^*$  is the cost of the best solution found and B is the value of the dual bound; for class [min |c/l], B refers to a lower bound whereas for class [max |p/t], B refers to a upper bound. For bounding procedure DK, the table reports only the average computing time in seconds, being the value of the dual bound computed by the procedure equal to the one computed by procedure CG.

For bounding procedures CB and DA, the table also reports the average percentage deviation of the primal bound obtained by the heuristic procedure (column %PB), computed as  $100.0 \times PB/z^*$ , where  $z^*$  is the cost of the best solution found and PB is the value of the primal bound; column "Time" also includes the time spent for computing the primal bound and, in the case of procedures CB, the time spent for computing value  $\overline{T}$ .

For procedures DA, CG, and DK, the table also shows the average number of columns or variables of the final master problem. In particular, for procedure DK the number is computed as the sum over all algorithm iterations, whose average number is reported under column "Iter" in the table.

Complete computational results about the dual and primal bounds are reported in the e-companion to this paper. Moreover, the e-companion also reports statistics about the best solutions found by the heuristic procedures and the exact method, including the values  $z^*$  used to compute the different percentage deviations.

Table 2 shows that the dual bounds computed are on average quite tight for instances belonging to the class  $[\min |c/l]$ . Clearly, the dual bound computed by procedure CG dominates the dual bound produced by procedure DA and it is equivalent to the one computed by procedure DK. Instances of class  $[\max |p/t]$  are more difficult for our bounding procedures, as shown by the average percentage deviation reported in the table; this is probably due to cost structure of the class. The dual bounds computed by procedure DA are on average very close to the ones produced by CG and can be computed faster. Column  $|\overline{\mathscr{R}}|$  (i.e., the average number of columns of the final master problem) shows that procedure DA is not affected by the typical degeneracy of standard column generation generation based on the simplex, like CG. The detailed results reported in the e-companion show that the dual bound produced by CB is always inferior with respect to one produced by CG. Further, its computation is more time consuming due to the complexity of the corresponding DP recursion - it is worth mentioning that the computation of value  $\overline{T}$  is negligible.

Concerning the primal bounds computed by procedures CB and DA, the table show that both the two procedures can compute good quality solutions. The detailed results reported in the e-companion also show that it is convenient to compute both primal bounds. The detailed results show that the heuristic algorithm applied during the execution of procedure DA failed to compute feasible primal bounds for three instances of class  $[\max |p/t]$ .

#### 6.3 Computational results on the exact method

Tables 3-4 show the results about the exact method. For each instance, we report a symbol to denote if the instance was solved to optimality ("(a)"), and the corresponding total computing time in seconds, that also includes the time spent by the bounding procedures executed at steps 2 and 3 of the exact method (see Section 4) and the time spent by procedure genr.

The next three blocks of columns show the details of the iterations of the exact method. For each iteration, we report the cardinality of the reduced set  $\overline{\mathscr{R}}$  ( $|\overline{\mathscr{R}}|$ ) generated by procedure genr, the percentage deviation of the value of the optimal solution of the reduced integer program  $F(\overline{\mathscr{R}})$  ( $\% z^*$ ), a symbol ("(d)") to denote if the time limit imposed was reached in solving  $F(\overline{\mathscr{R}})$  (IP), the number of iterations executed to solve problem  $F(\overline{\mathscr{R}})$ with the procedure described in Section 4.3, the percentage deviation of bound  $DB_{new}$ with respect to  $\% z^*$  (% B) (see Step 6 of the exact method) and the computing time in seconds spent to solve problem  $F(\overline{\mathscr{R}})$ .

In the tables, under column  $|\overline{\mathscr{R}}|$ , symbols "<sub>(b)</sub>" and "<sub>(c)</sub>" are reported whenever limit NSTATB or  $\Delta^{max}$  of procedure **genr** has been reached, respectively. Further, the heading of each table shows, for each iteration of the method, the value of parameters  $\Delta^{max}$  and *tlim*.

Details about the best solution found by the heuristic procedures and the exact method are reported in the e-companion. In particular, for each instances, it is shown the values  $z^*$ of the best solution found (including the corresponding numerator and denominator), the number of routes of the solution, the number of optional customers selected in solution, the percentage of the working time utilization and the average number of customers per route.

Tables 3-4 show that 53 and 30 out of the 54 instances per class were solved to optimality by the exact method for classes  $[\min |c/l]$  and  $[\max |p/t]$ , respectively. For classes  $[\min |c/l]$ , instances with up to 79 customers were solved to optimality whereas the largest instance solved to optimality for class  $[\max | p/t]$  involves 62 customers. Instances of class  $\left[\max | p/t \right]$  are clearly more difficult for our method, as also shown by the quality of the corresponding dual bounds, and by the fact that the limit  $\Delta^{max}$  imposed at the different iterations to the exact method is generally reached. For the instances not solved to optimality for classes  $[\min |c/l]$ , the final bound %B is very tight, thus showing that nearoptimal solutions are also computed for the corresponding instances. Notice that in the tables, whenever under column  $|\overline{\mathscr{R}}|$  of the last iteration appears symbol "<sub>(c)</sub>" (i.e.,  $\Delta^{max}$ limit reached) and the instance has been solved to optimality, the condition on the dual bound  $DB_{new}$  (see Step 6 of the exact method) is used as optimality condition. Most of the instances of class  $[\min |c/l]$  can be solved to optimality within the first two iterations of the exact method and the limit generally attained during the different iterations is the maximum number of columns or routes  $\Delta^{max}$ . In particular, 46 and 12 instances were solved to optimality at the first iteration of the exact algorithm for the two classes of instances, respectively, thus showing the effectiveness of our iterative exact procedure in reducing the size of the integer problems solved at each iteration of the Dinkelbach's algorithm.

In order to have some insights about the type of instances used and the solutions computed, the e-companion reports some statistics about the best solutions found for the two classes of instances. In particular, the tables show that the maximum number of vehicles m is generally tight and that the percentage of the working time utilization is on average superior to 90%; optional customers are generally included in the best solutions found.

			Iter = 1, $\Delta^{max} = 300,000, tlim = 3,600$						Iter = 2, $\Delta^{max} = 1,500,000, tlim = 7,200$						Iter = 3	$\Delta^{max}$	nax = 7,500,000, tlim = 10,800				
Name	Opt	Time	$ \mathcal{R} $	$\%z^*$	IP	Iter	%B	Time	$ \mathcal{R} $	$\%z^*$	IP	Iter	%B	Time	$ \mathcal{R} $	$\%z^*$	IP	Iter	%B	Time	
A-n32-k5a	(a)	8.4	24,022	100.0		2		1.4													
A-n32-k5b	(a)	21.0	35,174	100.0		2		2.9													
A-n33-k5a	(a)	25.2	(c)	100.0		3		7.7													
A-n33-k5b	(a)	20.7	632	100.0		2		0.0													
A-n33-k6a	(a)	12.4	(c)	100.0		3		6.0													
A-n33-k6b	(a)	24.2	(c)	100.0		3		11.2													
A-n34-k5a	(a)	28.1	(c)	100.0		2		18.5													
A-n34-k5b	(a)	37.5	(c)	100.0		2		21.3													
A-n36-k5a	(a)	38.4	(c)	100.0		3		11.1													
A-n36-k5b	(a)	39.5	(c)	100.0		2		19.3													
A-n37-k5a	(a)	156.8	206,203	100.0		2		21.3													
A-n37-k5b	(a)	75.1	(c)	100.0		2		52.0													
A-n37-k6a	(a)	12.1	50,360	100.0		2		4.1													
A-n37-k6b	(a)	25.3	(c)	100.0		3		14.0													
A-n38-k5a	(a)	137.7	(c)	100.0		2		132.4													
A-n38-k5b	(a)	24.7	(c)	100.0		2		14.0													
A-n39-k5a	(a)	79.4	(c)	100.0		2		41.9													
A-n39-k5b	(a)	35.8	8,135	100.0		2		0.4													
A-n39-k6a	(a)	20.7	(c)	100.0		2		9.3													
A-n39-k6b	(a)	18.7	24,621	100.0		2		2.1													
A-n44-kba	(a)	18.8	(c)	100.0		2		7.2													
A-n44-k6b	(a)	159.3	(c)	100.0		2		142.8													
A-n45-kba	(a)	14.9	12,138	100.0		2		0.4													
A-n45-K6D	(a)	91.1	(C)	100.0		2		69.6													
A-n45-K7a	(a)	40.7	262,658	100.0		2		27.9													
A-n45-K7D	(a)	28.8	(C)	100.0		3		12.0													
A-n40-K/a		10.4	6,492	100.0		2		112.1													
A-1140-K7D	(a)	120.9	(C)	100.0		3		113.1													
A-1140-K/a	(a)	00.1	(C)	100.0		2		54.5 70.7													
A-1146-K7D		91.0		100.0		2		10.7													
A-1155-K7a	(a)	29.0	(0)	100.0		2		4.7													
A-n54 k7a	(a)	27.1	422	100.0		2		0.0													
A-1154-K7a		20.0		100.0		2	00.0	27.0	149 554	100.0		1		85							
A n55 k0a	(a)	21.7		100.0		2	33.3	21.3	143,004	100.0		T		0.0							
A-n55-k9b	(a)	30.6		100.0		3		24.4													
A-n60-k9a		67.2		100.0		2		40.7													
A-n60-k9b	(a)	68.9		100.0		2		48.2													
A-n61-k9a	(a)	34.0	(c)	100.0		3		16.8													
A-n61-k9b	(a)	32.6		100.0		2		10.9													
A-n62-k8a	(a)	368.8	99.422	100.0		2		6.6													
A-n62-k8b	(a)	948.4	(c)	100.0		2	99.1	138.5	(c)	100.0		1	99.6	245.5	5,681,865	100		1		507.0	
A-n63-k10a	(a)	30.4	(c)	100.0		2		9.8	( )						- , ,						
A-n63-k10b	(a)	95.0	(c)	100.0		2		68.6													
A-n63-k9a	(a)	61.6	(c)	100.0		2	99.4	15.2	35,241	100.0		1		0.4							
A-n63-k9b	(a)	47.3	(c)	100.0		2		15.5													
A-n64-k9a	(a)	118.2	(b)	100.0		2	99.6	13.7	2,736	100.0		1		0.0							
A-n64-k9b	(a)	118.9	(c)	100.0		2		106.8	-												
A-n65-k9a	(a)	29.4	(c)	100.0		2		9.6													
A-n65-k9b	(a)	71.3	(c)	100.0		2		41.7													
A-n69-k9a	(a)	120.6	(c)	100.0		2	99.8	75.0	6,329	100.0		1		0.0							
A-n69-k9b	(a)	141.9	(c)	100.0		2	99.2	37.1	565,327	100.0		1		51.5							
A-n80-k10a	(a)	961.2	(b)	101.0		2	99.5	37.7	(c)	100.0		2		431.7							
A-n80-k10b		18338.3	(c)	100.0		2	99.0	153.7	(c)	100.0	(d)	1	99.6	7204.3	(c)	100	(d)	1	99.9	10859.8	
	I		(a): solve	ed to opt	imality	(	b): <i>NS</i>	TATB li	mit reached	(c)	: $\Delta^{ma}$	x limit	reached	(d):	<i>tlim</i> limit rea	ached					

Table 3: Instances of class  $[\min |c/l]$ : exact method

	Iter = 1, $\Delta^{max} = 300,000, tlim = 3,600$					600	Iter = 2, $\Delta^{max} = 1,500,000, tlim = 7,200$					Iter = 3, $\Delta^{max}$ = 7, 500, 000, tlim = 10, 800								
Name	Opt	Time	$ \mathcal{R} $	$\%z^*$	IP	Iter	% B	Time	$ \overline{\mathscr{R}} $	$\%z^*$	IP	Iter	% B	Time	$ \mathcal{R} $	$\%z^*$	IP	Iter	%B	Time
A-n32-k5a	(a)	134.6	605	100.0		2		0.0												
A-n32-k5b	(a)	53.5	176,046	100.0		2		2.8												
A-n33-k5a	(a)	74.0	283	100.0		$^{2}$		0.0												
A-n33-k5b	(a)	77.3	2,285	100.0		1		0.0												
A-n33-k6a	(a)	550.0	(c)	100.0		2	105.4	33.7	(c)	100.0		1	102.7	69.3	4,433,401	100.0		1		387.1
A-n33-k6b	(a)	455.9	(c)	100.0		2	105.7	40.7	(c)	100.0		1	103.6	183.7	6,732,081	100.0		1		168.9
A-n34-k5a	(a)	39.7	60,015	100.0		1		1.5												
A-n34-k5b	(a)	35.5	460	100.0		2		0.0												
A-n36-k5a	(a)	151.8	(c)	100.0		2		7.6												
A-n36-k5b	(a)	165.7	74,972	100.0		2		3.2												
A-n37-k5a	(a)	98.9	290,301	100.0		2		37.0												
A-n37-k5b	(a)	166.0	(c)	100.0		$^{2}$	102.0	31.7	1,401,121	100.0		1		48.7						
A-n37-k6a	(a)	27.0	171,702	100.0		2		3.5												
A-n37-k6b	(a)	31.2	(c)	100.0		2	100.2	5.5	259,368	100.0		1		2.2						
A-n38-k5a	(a)	99.4	(c)	100.0		2	101.1	5.7	352,959	100.0		1		32.4						
A-n38-k5b	(a)	64.8	(c)	100.0		2	103.0	4.9	1,152,405	100.0		1		8.2						
A-n39-k5a	(a)	155.8	(c)	100.0		2	102.4	7.2	(c)	100.0		1	100.8	12.1	2,701,913	100.0		1		21.6
A-n39-k5b	(a)	182.7	(c)	100.0		$^{2}$	101.9	9.5	(c)	100.0		1	100.3	15.0	1,909,058	100.0		1		19.1
A-n39-k6a	(a)	222.9	(c)	99.7		2	103.3	8.8	(c)	100.0		2	100.8	65.3	1,997,245	100.0		1		41.8
A-n39-k6b	(a)	100.6	(c)	100.0		2	102.7	6.8	(c)	100.0		1	100.7	12.9	2,070,281	100.0		1		16.7
A-n44-k6a	(a)	90.0	(c)	100.0		2	102.4	8.6	1,088,748	100.0		1		11.7						
A-n44-k6b		286.3	(c)	100.0		$^{2}$	102.9	29.5	(c)	100.0		1	102.0	33.1	(c)	100.0		1	100.7	126.9
A-n45-k6a	(a)	202.9	(c)	100.0		2		132.9												
A-n45-k6b	(a)	153.5	(c)	100.0		2	101.3	18.1	1,220,510	100.0		1		70.7						
A-n45-k7a	(a)	73.5	(c)	100.0		2	105.3	6.9	(c)	100.0		1	103.0	12.9	3,416,812	100.0		1		26.6
A-n45-k7b	(a)	75.2	(c)	100.0		2	104.1	6.8	(c)	100.0		1	102.4	12.4	3,568,626	100.0		1		27.1
A-n46-k7a	(a)	102.8	(c)	100.0		2	101.8	39.9	970,793	100.0		1		12.0						
A-n46-k7b		235.0	(c)	100.0		2	104.0	15.8	(c)	100.0		1	102.5	24.7	(c)	100.0		1	100.5	131.2
A-n48-k7a	(a)	82.2	226,299	100.0		2		22.9												
A-n48-k7b	(a)	87.0	(c)	100.0		$^{2}$	100.3	7.4	278,742	100.0		1		17.3						
A-n53-k7a	(a)	503.7	(c)	98.8		2	103.5	40.3	(c)	100.0		2	101.1	225.1	2,185,209	100.0		1		166.9
A-n53-k7b		355.6	(c)	99.8		2	104.1	21.8	(c)	100.0		2	103.2	66.6	(c)	100.0		1	102.1	183.1
A-n54-k7a		265.7	(c)	100.0		2	102.7	8.0	(c)	100.0		1	101.7	15.7	(c)	100.0		1	100.4	59.4
A-n54-k7b		1065.3	(c)	99.7		2	104.8	146.6	(c)	100.0		2	104.1	414.6	(c)	100.0		1	103.2	412.3
A-n55-k9a		599.1	(c)	96.4		2	109.6	78.4	(c)	97.7		2	108.8	41.0	(c)	100.0		3	107.8	372.5
A-n55-k9b		377.7	(c)	97.5		1	106.5	4.3	(c)	100.0		2	105.7	57.3	(c)	100.0		1	104.9	204.3
A-n60-k9a		948.9	(c)	98.1		2	106.6	54.8	(c)	99.8		2	105.7	336.2	(c)	100.0		2	104.5	444.7
A-n60-k9b		21619.5	(c)	96.9		2	104.9	100.8	(c)	96.9		1	104.3	2269.1	(c)	100.0		3	103.5	19142.3
A-n61-k9a		592.5	(c)	99.3		1	108.0	3.9	(c)	99.3		1	107.1	24.5	(c)	100.0		2	106.1	288.9
A-n61-k9b		1020.9	(c)	98.2		2	106.2	32.6	(c)	98.3		2	105.6	366.7	(c)	100.0		2	104.9	469.8
A-n62-k8a		446.2	(c)	95.5		2	104.9	9.6	(c)	100.0		2	104.0	54.5	(c)	100.0		1	103.1	122.0
A-n62-k8b		1432.0	(c)	99.5		2	104.8	21.4	(c)	100.0		2	104.3	572.1	(c)	100.0		1	103.7	562.1
A-n63-k10a		1446.4	(c)	98.8		2	107.0	16.7	(c)	99.9		2	106.0	1043.6	(c)	100.0		2	105.1	313.5
A-n63-k10b	( )	348.7	(c)	99.1		2	104.7	61.8	(c)	100.0		2	104.1	61.1	(c)	100.0		1	103.4	132.7
A-n63-k9a	(a)	77.7	(c)	100.0		2	101.6	6.9	700,337	100.0		1		6.6					100.0	
A-n63-k9b		242.9	(c)	100.0		2	103.2	12.4	(c)	100.0		1	102.3	27.6	(c)	100.0		1	100.6	106.0
A-n64-k9a		894.1	(c)	98.6		2	103.9	73.0	(c)	99.6		2	103.2	87.4	(c)	100.0		2	102.1	624.6
A-n64-k9b		1120.9	(c)	98.0		2	104.7	37.7	(c)	99.7		2	104.2	229.1	(c)	100.0		2	103.7	677.5
A-n65-k9a		340.5	(c)	99.4		2	105.2	8.0	(c)	99.4		1	104.3	20.5	(c)	100.0		2	103.0	209.2
A-nbb-k9b		298.4	(c)	99.1		1	106.0	3.9	(c)	100.0		2	105.0	67.8	(c)	100.0		1	103.8	127.1
A-n69-k9a		3493.2	(c)	98.8		1	107.5	1.3	(c)	99.7		2	105.4	754.6	(c)	100.0		2	103.9	2579.9
A-n69-k9b		1833.4	(c)	96.8		2	107.2	11.2	(c)	97.2		2	106.4	161.4	(c)	100.0		2	105.7	1559.4
A-n80-k10a		599.5	(c)	98.4		2	104.1	36.1	(c)	100.0		2	103.6	100.7	(c)	100.0		1	102.9	263.9
A-n80-k10b		1240.5	(c)	99.3		2	102.8	30.2	(c)	99.8		2	102.4	334.2	(c)	100.0		2	102.0	0/1.4
	I																			

Table 4: Instances of class  $[\max | p/t]$ : exact method

(a): solved to optimality

ptimality (b): NSTATB limit reached

ed (c):  $\Delta^{max}$  limit reached

(d): *tlim* limit reached

27

# 7 Conclusions

In this paper, we considered vehicle routing problems that can be modelled as a Set Partitioning (SP) problem with a linear fractional objective function. More precisely, we considered two objective functions: minimization of cost over load (also known as logistic ratio) and maximization of profit over time.

We investigated both continuous and integer relaxations of the SP model. In particular, we proposed an alternative transformation to the transformation proposed by Charnes and Cooper (1962) for linear fractional programming and a dual ascent heuristic used to compute both dual and primal bounds. The dual and primal bounds computed are embedded in an iterative exact procedure where at each iteration a reduced SP problem is solved by the extension of Dinkelbach's algorithm for fractional programming to integer programs.

We reported computational results showing that the proposed method solves to optimality instances involving up to 79 customers. The method can be easily adapted to deal with other routing constraints, simply by taking into account of such constraints in the route generation phase.

## References

- Archetti, C., G. Desaulniers, M. G. Speranza. 2016. Minimizing the logistic ratio in the inventory routing problem. EURO Journal on Transportation and Logistics 1–18.
- Archetti, C., M. Grazia Speranza, Daniele Vigo. 2014. Vehicle routing problems with profits. MOS-SIAM Series on Optimization.
- Augerat, P. 1995. Approche polyèdrale du problème de tournées de véhicules. Ph.D. thesis, Institut National Polytechnique de Grenoble.
- Baldacci, R., E. Bartolini, A. Mingozzi, R. Roberti. 2010. An exact solution framework for a broad class of vehicle routing problems. *Computational Management Science* 7 229–268.
- Baldacci, R., N. Christofides, A. Mingozzi. 2008. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming* Ser. A 115(2) 351–385.
- Baldacci, R., A. Mingozzi, R. Roberti. 2011. New route relaxation and pricing strategies for the vehicle routing problem. Operations Research 59(5) 1269–1283.
- Baldacci, R., A. Mingozzi, R. Roberti. 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research* 218(1) 1–6.
- Barros, A. I., R. Dekker, J. B. G. Frenk, S. van Weeren. 1997. Optimizing a General Optimal Replacement Model by Fractional Programming Techniques. *Journal of Global Optimization* 10(4) 405–423.
- Bázsa, E.M, J.B.G Frenk, P.W den Iseger. 2001. Modeling of inventory control with regenerative processes. International Journal of Production Economics 71(1–3) 263–276.

- Benoist, T., F. Gardi, A. Jeanjean, B. Estellon. 2011. Randomized local search for real-life inventory routing. *Transportation Science* 45(3) 381–398.
- Campbell, Ann M., Lloyd W. Clarke, Martin W. P. Savelsbergh. 2001. Inventory routing in practice. Paolo Toth, Daniele Vigo, eds., *The Vehicle Routing Problem*, chap. Inventory Routing in Practice. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 309–330.
- Charnes, A., W. W. Cooper. 1962. Programming with linear fractional functionals. Naval Research Logistics Quarterly 9(3-4) 181–186.
- Coelho, L. C., J.-F. Cordeau, G. Laporte. 2014. Thirty years of inventory routing. Transportation Science 48(1) 1–19.
- Desaulniers, G., J. Desrosiers, M. M. Solomon. 2006. *Column generation*, vol. 5. Springer Science & Business Media.
- Dinkelbach, W. 1967. On nonlinear fractional programming. Management Science 13(7) 492-498.
- Dror, M., M. Ball. 1987. Inventory/routing: Reduction from an annual to a short-period problem. Naval Research Logistics (NRL) **34**(6) 891–905.
- Espinoza, D., R. Fukasawa, M. Goycoolea. 2010. Lifting, tilting and fractional programming revisited. Oper. Res. Lett. 38(6) 559–563.
- Frenk, J. B. G., S. Schaible. 2005. Fractional Programming. Springer New York.
- Garaix, T., C. Artigues, D. Feillet, D. Josselin. 2011. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. *Computers & OR* 38(10) 1435–1442.
- IBM CPLEX. 2016. IBM ILOG CPLEX 12.6.4 callable library.
- Loes, Knoben. 2016. Optimizing the Moment of Customer Delivery in ORTEC Inventory Routing. Master Thesis .
- Morton, S. 2011. Pioneering solutions in supply chain management. a comprehensive insight into current management approaches, edited by w. kersten, t. blecker, and c. luthje. *International Journal of Production Research* 49(24) 7517–7518.
- Pecin, Diego, Claudio Contardo, Guy Desaulniers, Eduardo Uchoa. 2017a. New enhancements for the exact solution of the vehicle routing problem with time windows. *INFORMS Journal on Computing* 29(3) 489–502.
- Pecin, Diego, Artur Pessoa, Marcus Poggi, Eduardo Uchoa. 2017b. Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation* 9(1) 61–100.
- Radzik, T. 1999. Fractional combinatorial optimization. Panos M. Pardalos Ding-Zhu Du, ed., Handbook of Combinatorial Optimization: Volume1-3. Springer US, Boston, MA, 429–478.
- Reinelt, G. 1991. Tsplib–a traveling salesman problem library. ORSA Journal on Computing **3**(4) 376–384.
- Ródenas, R. G., M. L. López, D. Verastegui. 1999. Extensions of dinkelbach's algorithm for solving non-linear fractional programming problems. Top 7(1) 33–70.
- Schaible, S. 1976. Fractional Programming. II, on Dinkelbach's Algorithm. Management Science 22(8) 868–873.
- Schaible, S., J. Shi. 2004. Recent developments in fractional programming: single-ratio and maxmin case. Nonlinear analysis and convex analysis 493–506.

- Stancu-Minasian, IM. 2012. Fractional programming: theory, methods and applications, vol. 409. Springer Science & Business Media.
- Toth, P., D. Vigo. 2014. Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization, SIAM, Philadelphia.
- You, F., P. M. Castro, I. E. Grossmann. 2009. Dinkelbach's algorithm as an efficient method to solve a class of minlp models for large-scale cyclic scheduling problems. *Computers and Chemical Engineering* 33(11) 1879–1889.

# Appendix

## **Proofs of statements**

**Theorem 2.** Let **v** be a feasible solution of problem DNCF of cost z(DNCF). A feasible solution  $(\boldsymbol{\mu}, \omega)$  of DCCF of cost z(DCCF) = z(DNCF) can be obtained by setting:

$$\omega = \sum_{i \in V_c} v_i + mv_0, \quad \mu_0 = \beta v_0, \quad \mu_i = \beta v_i - s_i \omega, \forall i \in V_1, \quad \mu_i = \beta v_i, \forall i \in V_2.$$
(13)

*Proof.* It is easy to see that z(DCCF) = z(DNCF) due to the definition of  $\omega$  in expressions (13). We have

$$-\sum_{i \in V_c} \mu_i - m\mu_0 = -\sum_{i \in V_1} \beta v_i + \sum_{i \in V_1} s_i \omega - \sum_{i \in V_2} \beta v_i - m\beta v_0 = -\beta (\sum_{i \in V_c} v_i + mv_0) + \beta (\sum_{i \in V_c} v_i + mv_0) = 0,$$

showing that the dual constraint associated with variable u of CCF is satisfied. Further,

$$\sum_{i \in V_c} a_{i\ell} \mu_i + \mu_0 + w_\ell \omega = \sum_{i \in V_1} a_{il} \beta v_i - \sum_{i \in V_1} a_{il} s_i (\sum_{i \in V_c} v_i + mv_0) + \sum_{i \in V_2} a_{i\ell} \beta v_i + \beta v_0 + w_\ell (\sum_{i \in V_c} v_i + mv_0) = \sum_{i \in V_1} a_{il} \beta v_i + \sum_{i \in V_2} a_{i\ell} \beta v_i + \beta v_0 + (w_\ell - \sum_{i \in V_1} a_{il} s_i) (\sum_{i \in V_c} v_i + mv_0) = \sum_{i \in V_c} (\beta a_{il} + \overline{w}_\ell) v_i + (\beta + m\overline{w}_\ell) v_0 \le c_\ell,$$

showing that the dual constraint associated with variable  $y_{\ell}, \ell \in \mathscr{R}$ , of CCF is satisfied.

**Theorem 4.**  $\overline{r} = n(\overline{\mathbf{x}})/d(\overline{\mathbf{x}}) = z(F(\overline{\mathscr{R}}))$  if, and only if, for the parametric problem FP(r),  $z(FP(r)) = \min\{n(\mathbf{x}) - rd(\mathbf{x}) : \mathbf{x} \in P\}, \ z(FP(\overline{r})) = n(\overline{\mathbf{x}}) - \overline{r}d(\overline{\mathbf{x}}) = 0$ . *Proof.* 

- (a) Let  $\overline{\mathbf{x}}$  be an optimal solution of problem  $F(\overline{\mathscr{R}})$ . We have  $\overline{r} = n(\overline{\mathbf{x}})/d(\overline{\mathbf{x}}) \leq n(\mathbf{x})/d(\mathbf{x}), \forall \mathbf{x} \in P$ . Hence
  - (i)  $n(\mathbf{x}) \overline{r}d(\mathbf{x}) \ge 0, \forall \mathbf{x} \in P$ , and
  - (ii)  $n(\overline{\mathbf{x}}) \overline{r}d(\overline{\mathbf{x}}) = 0.$

This implies that  $\overline{\mathbf{x}}$  is an optimal solution of  $FP(\overline{r})$  of value  $z(FP(\overline{r})) = 0$ .

(b) Let  $\overline{\mathbf{x}}$  be an optimal solution of  $FP(\overline{r})$  such that  $z(FP(\overline{r})) = n(\overline{\mathbf{x}}) - \overline{r}d(\overline{\mathbf{x}}) = 0$ . We have  $n(\mathbf{x}) - \overline{r}d(\mathbf{x}) \ge 0, \forall \mathbf{x} \in P$ , and  $\overline{r} \le n(\mathbf{x})/d(\mathbf{x}), \forall \mathbf{x} \in P$ , therefore  $\overline{r}$  is the minimum of problem  $F(\overline{\mathscr{R}})$  that is taken at  $\overline{\mathbf{x}}.\Box$ 

**Lemma 2.** The following properties hold about the exact algorithm for solving  $F(\overline{\mathscr{R}})$ .

- (1) The sequence  $\{r_i\}$  is monotone decreasing, i.e.,  $r_i > r_{i+1}$ , for all i such that  $z_i(FP(r_i)) < 0$ .
- (2) If  $z_i(FP(r_i)) \ge 0$  and solution  $\mathbf{x}^i$  is optimal, the value  $\overline{z}(F(\overline{\mathscr{R}}))$  corresponds to the optimal solution value and  $\overline{r} = n(\overline{\mathbf{x}})/d(\overline{\mathbf{x}})$ .
- (3) If  $z_i(FP(r_i)) < 0$ , we have  $d(\mathbf{x}^i) > d(\mathbf{x}^{i+1})$ .

## Proof.

(1) Since  $d(\mathbf{x}^i) > 0$  and  $z_i(FP(r_i)) = n(\mathbf{x}^i) - r_i d(\mathbf{x}^i)$  we have

$$r_i > n(\mathbf{x}^i)/d(\mathbf{x}^i) = r_{i+1}.$$
(30)

- (2) Assume that the algorithm terminates on the *i*-th iteration. We have  $z_i(FP(r_i)) \ge 0$ . Consider a value  $\hat{r} > \overline{r} = r_i$ , we have:
  - (i)  $\overline{r} = n(\mathbf{x}^{i-1})/d(\mathbf{x}^{i-1})$  and  $n(\mathbf{x}^{i-1}) \overline{r}d(\mathbf{x}^{i-1}) = 0$
  - (ii) As  $d(\mathbf{x}^{i-1}) > 0$ , we have  $n(\mathbf{x}^{i-1}) \hat{r}d(\mathbf{x}^{i-1}) < 0$ , therefore value  $\hat{r}$  does not satisfies the termination condition.

(3) We have:

- (a)  $n(\mathbf{x}^i) r_{i+1}d(\mathbf{x}^i) = 0$  (since the definition of  $r_{i+1}$ ),
- (b)  $n(\mathbf{x}^{i+1}) r_{i+1}d(\mathbf{x}^{i+1}) < 0 \ (z_i(FP(r_{i+1})) < 0),$
- (c)  $n(\mathbf{x}^{i+1}) r_i d(\mathbf{x}^{i+1}) \ge n(\mathbf{x}^i) r_i d(\mathbf{x}^i) (\mathbf{x}^{i+1} \text{ is a feasible solution of } FP(r_i)).$  (31)

From (31)-a and (31)-b we have

$$n(\mathbf{x}^{i+1}) - n(\mathbf{x}^{i}) \le r_{i+1}(d(\mathbf{x}^{i+1}) - d(\mathbf{x}^{i}))$$
(32)

and by considering (31)-c from (32) we have  $(r_{i+1} - r_i)(d(\mathbf{x}^{i+1}) - d(\mathbf{x}^i)) \ge 0$ . From the above inequality and inequality (30) we have  $d(\mathbf{x}^{i+1}) \le d(\mathbf{x}^i)$ . Below we show that  $d(\mathbf{x}^{i+1}) \ne d(\mathbf{x}^i)$ . Let  $P_i = {\mathbf{x} \in P : d(\mathbf{x}) = d(\mathbf{x}^i)}$  - we have  $\mathbf{x}^i = \operatorname{argmin}\{n(\mathbf{x}) - r_i d(\mathbf{x}) : \mathbf{x} \in P_i\}$  =  $\operatorname{argmin}\{n(\mathbf{x}) : \mathbf{x} \in P_i\}$ . If  $\mathbf{x} \in P_i$ , we have  $n(\mathbf{x}) - r_{i+1}d(\mathbf{x}) =$  $n(\mathbf{x}) - r_{i+1}d(\mathbf{x}^i) \ge n(\mathbf{x}^i) - r_{i+1}d(\mathbf{x}^i) = 0$ , therefore  $n(\mathbf{x}) - r_{i+1}d(\mathbf{x}) \ge 0$ ,  $\forall \mathbf{x} \in P_i$ . Since  $r_{i+1}$  is not optimal, we have  $z_{i+1}(FP(r_{i+1})) = n(\mathbf{x}^{i+1}) - r_{i+1}d(\mathbf{x}^{i+1}) < 0$ , therefore  $\mathbf{x}^{i+1}$  is not in  $P_i$ .  $\Box$ 

**Theorem 5.** For all  $r_i \neq \overline{r}$  we have

$$\frac{\overline{r} - r_{i+1}}{\overline{r} - r_i} \le 1 - \frac{d(\overline{\mathbf{x}})}{d(\mathbf{x}^i)} < 1.$$

*Proof.* Inequality  $n(\mathbf{x}^i) - r_i d(\mathbf{x}^i) \le n(\overline{\mathbf{x}}) - r_i d(\overline{\mathbf{x}})$  implies

$$\frac{n(\mathbf{x}^i)}{d(\mathbf{x}^i)} - r_i \le \frac{n(\overline{\mathbf{x}})}{d(\mathbf{x}^i)} - r_i \frac{d(\overline{\mathbf{x}})}{d(\mathbf{x}^i)},$$

therefore

$$r_{i+1} - \overline{r} = \frac{n(\mathbf{x}^i)}{d(\mathbf{x}^i)} - \frac{n(\overline{\mathbf{x}})}{d(\overline{\mathbf{x}})} \le \frac{n(\overline{\mathbf{x}})}{d(\mathbf{x}^i)} - \frac{n(\overline{\mathbf{x}})}{d(\overline{\mathbf{x}})} + r_i \left(1 - \frac{d(\overline{\mathbf{x}})}{d(\mathbf{x}^i)}\right) = \left(\frac{1}{d(\mathbf{x}^i)} - \frac{1}{d(\overline{\mathbf{x}})}\right) \left(n(\overline{\mathbf{x}}) - r_i d(\overline{\mathbf{x}})\right) = \left(\frac{1}{d(\mathbf{x}^i)} - \frac{1}{d(\overline{\mathbf{x}})}\right) (\overline{r} - r_i) d(\overline{\mathbf{x}}),$$

since  $\overline{r} = n(\overline{\mathbf{x}})/d(\overline{\mathbf{x}})$ . Since from Lemma 2-(1) we have  $\overline{r} < r_i$  and dividing by  $(r_i - \overline{r}) > 0$ and since  $d(\mathbf{x}^i) > d(\overline{\mathbf{x}})$  (Lemma 2-(3)) we obtain

$$\frac{r_{i+1} - \overline{r}}{r_i - \overline{r}} \le \left(\frac{1}{d(\mathbf{x}^i)} - \frac{1}{d(\overline{\mathbf{x}})}\right) \frac{\overline{r} - r_i}{r_i - \overline{r}} d(\overline{\mathbf{x}}) = \left(-\frac{d(\overline{\mathbf{x}})}{d(\mathbf{x}^i)} + 1\right) < 1. \square$$

#### 7.1 Details about the computational results

Tables 5-6 report the details about the bounding procedures. In particular, for bounding procedures CB, DA and CG, the table reports the percentage deviation of the dual bound (column %B) and the total computing time in seconds (column "Time"). The percentage deviation is computed as  $100.0 \times B/z^*$ , where  $z^*$  is the cost of the best solution found and B is the value of the dual bound; for classes [min |c/l|, B refers to a lower bound whereas for class [max |p/t|, B refers to a upper bound. For bounding procedure DK, the table reports only the computing time in seconds, being the value of the dual bound computed by the procedure equal to the one computed by procedure CG.

For bounding procedures CB and DA, the table also reports the percentage deviation of the primal bound obtained by the heuristic procedure (column %PB), computed as  $100.0 \times PB/z^*$ , where  $z^*$  is the cost of the best solution found and PB is the value of the primal bound - column "Time" also includes the time spent for computing the primal bound and, in the case of procedures CB, the time spent for computing value  $\overline{T}$ .

For procedures DA, CG, and DK, the table also shows the number of columns or variables of the final master problem. In particular, for procedure DK the number is computed as the sum over all algorithm iterations, whose average number is reported under column "Iter" in the table.

The percentage deviations are computed with respect to the values of the best solutions found which are reported in Tables 7-8.

Tables 7-8 show details about the best solutions found. More precisely, for each instances, the tables show the values of the best solution found  $(z^*)$  (including the corresponding numerator "Cost" or "Profit" and denominator "Load" or "Time"), the number of routes of the solution (#r), the number of optional customers selected in solution (#o), the percentage of the working time utilization (%ut) and the average number of customers per route (ac).

	Procedure CB		т	Procedu	Iro D	٨	Dr	ocoduro	D	rocodure	אמ		
Nama	07 D		Time	07 D			н Т:та	07 D		Time	Iton		Time e
Ivame	70 <i>D</i>	70 <i>P</i> D	1 line	70 <i>D</i>	70 <i>P</i> D	$ \mathcal{A} $	1 ime	70 <i>D</i>	$ \mathcal{A} $	1 ime	ner	$ \mathcal{A} $	1 line
A-n32-k5a	98.6	103.1	7.2	99.3	103.1	160	0.1	99.3	2,690	0.2	3	2,394	0.2
A-n32-k5b	98.1	102.2	18.8	98.2	102.2	232	0.1	98.2	3,340	0.2	3	3,439	0.3
A-n33-k5a	98.1	112.5	18.4	98.1	112.5	252	0.1	98.1	2,689	0.1	2	2,503	0.2
A-n33-k5b	99.4	100.4	22.3	99.5	100.4	150	0.1	99.5	2,765	0.1	3	3,090	0.2
A-n33-k6a	99.5	119.4	6.3	100.0	119.4	147	0.1	100.0	2,079	0.1	3	1,787	0.1
A-n33-k6b	97.9	103.0	13.3	97.7	103.0	261	0.1	97.9	2,444	0.1	3	2,379	0.1
A-n34-k5a	99.4	107.5	19.6	99.4	107.5	207	0.1	99.4	2,731	0.1	3	3,237	0.2
A-n34-k5b	98.8	106.6	16.5	98.9	106.6	314	0.1	98.9	3,222	0.2	3	2,777	0.2
A-n36-k5a	99.5	115.4	25.2	99.7	113.4	315	0.5	99.7	4,103	0.4	3	4,356	0.8
A-n36-k5b	99.0	106.6	25.3	98.8	106.5	411	0.4	99.1	5,104	0.6	3	5,691	0.9
A-n37-k5a	96.1	102.8	17.0	97.5	102.8	200	0.2	97.5	4,020	0.4	3	4,126	0.4
A-n37-k5b	98.4	104.8	23.5	98.3	103.8	396	0.2	98.6	7,444	1.2	3	6,008	0.8
A-n37-k6a	99.2	105.3	8.1	99.3	103.6	313	0.1	99.3	2,899	0.1	3	2,595	0.2
A-n37-k6b	97.4	110.6	11.1	97.5	112.5	396	0.2	97.5	3,795	0.3	3	3,763	0.4
A-n38-k5a	99.8	112.5	4.7	100.0	112.5	299	0.1	100.0	3,915	0.3	2	4,434	0.3
A-n38-k5b	95.7	108.1	10.1	95.9	107.4	328	0.2	95.9	4,434	0.3	2	4,204	0.3
A-n39-k5a	97.8	106.4	33.5	97.8	109.2	313	0.2	97.8	3,646	0.3	3	3,799	0.5
A-n39-k5b	99.2	100.8	36.4	99.3	102.9	329	0.3	99.3	5,449	0.5	2	4,850	0.5
A-n39-k6a	97.8	107.9	11.0	98.2	107.9	197	0.2	98.2	3,505	0.2	3	3,133	0.2
A-n39-k6b	98.1	100.9	16.7	98.4	100.9	278	0.3	98.4	4,878	0.4	3	3,788	0.3
A-n44-k6a	100.0	118.9	10.0	100.0	118.9	349	0.2	100.0	4,661	0.4	2	5,211	0.5
A-n44-k6b	97.7	102.8	15.9	97.8	102.7	484	0.3	97.8	5,620	0.6	3	5,744	0.7
A-n45-k6a	99.0	101.7	13.7	99.1	102.5	220	0.4	99.1	4,738	0.4	3	4,647	0.5
A-n45-k6b	98.5	104.6	21.0	98.6	104.1	354	0.2	98.6	6,183	0.6	3	5,769	0.7
A-n45-k7a	98.7	104.0	12.2	98.8	102.7	304	0.2	98.8	4,597	0.4	3	5,268	0.5
A-n45-k7b	99.4	106.4	16.1	99.6	105.9	350	0.3	99.6	$5,\!647$	0.5	3	5,043	0.5
A-n46-k7a	98.3	103.5	9.6	98.7	100.7	221	0.3	98.7	5,387	0.5	3	4,206	0.4
A-n46-k7b	96.8	102.4	6.8	98.5	102.4	352	0.4	98.5	6,694	0.7	3	6,927	0.8
A-n48-k7a	97.6	106.1	12.7	97.7	106.1	214	0.2	97.7	4,831	0.4	3	5,146	0.5
A-n48-k7b	98.5	104.5	20.2	98.7	104.5	413	0.3	98.7	7,221	1.1	2	7,214	1.1
A-n53-k7a	100.0	116.0	21.7	100.0	105.7	405	0.2	100.0	9,810	1.4	2	9,232	1.4
A-n53-k7b	99.8	100.8	26.9	100.0	100.2	261	0.3	100.0	13,143	2.7	3	13,137	2.9
A-n54-k7a	99.7	111.1	15.1	99.9	111.1	426	0.3	99.9	7,265	0.9	3	6,080	0.8
A-n54-k7b	98.1	105.2	18.3	98.0	104.4	574	0.5	98.2	9,760	1.7	3	11,170	2.3
A-n55-k9a	98.5	108.5	11.6	98.8	107.9	379	0.4	98.8	4,514	0.3	3	4,487	0.4
A-n55-k9b	98.8	105.2	15.0	98.9	105.1	456	0.3	98.9	7,061	0.7	3	$6,\!687$	0.7
A-n60-k9a	98.2	110.2	23.6	98.4	102.7	264	0.6	98.4	9,161	1.2	2	8,787	0.9
A-n60-k9b	98.5	109.0	35.0	98.6	106.1	677	1.0	98.6	10,938	1.9	3	11,252	2.3
A-n61-k9a	98.0	109.8	14.9	98.2	109.9	468	0.4	98.2	6,989	0.7	3	6,510	0.7
A-n61-k9b	99.5	106.0	19.0	99.8	106.0	441	0.5	99.8	9,001	1.3	3	$^{8,235}$	1.2
A-n62-k8a	97.7	103.3	13.6	98.6	100.5	604	1.0	98.6	13,514	2.2	3	12,301	2.1
A-n62-k8b	98.0	102.9	27.1	98.2	102.9	502	1.0	98.2	15,392	3.4	3	18,403	4.4
A-n63-k10a	98.8	111.3	18.0	98.9	110.3	296	0.5	98.9	8,630	1.1	2	7,417	0.8
A-n63-k10b	98.1	102.6	23.0	98.7	101.6	409	1.1	98.7	11,353	1.7	3	10,505	1.6
A-n63-k9a	98.6	109.7	19.0	98.7	108.1	369	0.4	98.7	11,333	1.6	2	9,417	1.3
A-n63-k9b	98.6	104.2	30.2	98.8	104.2	494	0.7	98.8	11,169	1.7	3	9,443	1.7
A-n64-k9a	99.1	108.6	12.9	99.3	109.3	368	0.6	99.3	10,926	1.5	3	10,341	1.4
A-n64-k9b	97.5	112.0	8.6	99.1	106.8	719	1.0	99.1	15,511	3.2	2	14,823	3.4
A-n65-k9a	98.4	104.3	17.0	98.8	111.6	344	0.3	98.8	8,175	1.0	3	7,978	1.0
A-n65-k9b	98.5	104.1	27.4	98.6	106.2	404	0.7	98.6	14,128	2.5	3	11,708	2.2
A-n69-k9a	98.9	112.4	34.1	99.1	110.9	370	0.8	99.1	15,303	3.0	3	13,363	2.4
A-n69-k9b	97.5	110.7	49.3	97.8	107.8	675	1.0	97.8	17,575	4.3	3	15,376	3.4
A-n80-k10a	99.2	108.7	18.3	99.4	107.9	775	1.3	99.4	16,459	2.8	3	15,720	3.0
A-n80-k10b	98.4	107.6	30.7	98.5	106.5	769	1.9	98.5	$27,\!078$	8.5	3	$21,\!825$	5.5
	98.5	107.0	18.8	98.7	106.3	374	0.4	98.8	$7,\!684$	1.2	2.8	$7,\!254$	1.1

Table 5: Dual and primal bounds on instances of class  $[\min |c/l]$ 

	Procedure CB			Procedure DA					ocedure	CG	Procedure DK			
Name	%B	% PB	Time	%B	% PB	$ \overline{\mathscr{R}} $	Time	%B	$ \overline{\mathscr{R}} $	Time	Iter	$ \overline{\mathscr{R}} $	Time	
A-n32-k5a	105.8	99.8	134.8	103.7	99.8	228	0.1	103.5	719	0.6	2	597	1.0	
A-n32-k5b	106.5	96.3	49.8	106.2	98.3	286	0.1	106.0	1,009	0.3	2	950	0.2	
A-n33-k5a	100.5	99.3	69.0	100.5	99.9	413	0.2	100.3	1,367	0.4	2	$1,\!874$	1.9	
A-n33-k5b	101.5	99.6	72.5	101.2	100.0	511	0.3	101.1	1,589	0.4	3	$1,\!631$	0.3	
A-n33-k6a	110.7	97.7	47.9	110.6	98.3	586	0.1	110.4	1,412	0.2	3	1,754	1.0	
A-n33-k6b	109.6	99.1	50.1	109.6	99.6	566	0.2	109.4	1,713	0.2	3	$1,\!609$	0.1	
A-n34-k5a	107.5	98.2	37.5	106.8	100.0	323	0.1	106.8	1,366	0.3	3	1,292	0.6	
A-n34-k5b	100.9	99.7	34.3	100.8	99.7	317	0.1	100.6	$1,\!128$	0.3	3	$1,\!457$	0.3	
A-n36-k5a	105.2	94.2	141.2	104.7	95.7	458	0.4	104.6	1,133	0.4	2	1,051	0.8	
A-n36-k5b	103.7	96.0	159.6	103.5	97.2	516	0.5	103.5	1,469	1.1	2	1,127	0.6	
A-n37-k5a	111.1	98.8	61.2	110.2	98.0	464	0.1	110.0	1,292	0.2	2	1,150	0.4	
A-n37-k5b	105.3	99.5	81.6	105.1	99.6	441	0.2	105.1	$1,\!684$	0.2	3	1,517	0.2	
A-n37-k6a	104.1	97.8	22.4	103.6	97.8	522	0.3	103.4	1,281	0.1	2	1,377	0.2	
A-n37-k6b	105.2	96.0	20.7	104.2	96.8	540	0.3	104.0	1,210	0.2	2	1,333	0.1	
A-n38-k5a	106.5	95.3	61.5	106.3	96.4	395	0.1	106.2	1,406	0.2	3	1,641	0.2	
A-n38-k5b	107.6	96.4	49.2	107.7	99.0	394	0.2	107.4	1,545	0.2	3	1,597	0.1	
A-n39-k5a	105.9	98.0	106.7	106.0	96.1	371	0.3	105.8	1,349	0.5	3	1,970	1.0	
A-n39-k5b	105.2	95.8	131.7	104.8	97.7	342	0.4	104.7	1,458	0.5	3	1,628	0.3	
A-n39-k6a	107.6	98.6	101.6	107.4	98.6	450	0.2	107.4	1,140	0.6	3	1,584	0.6	
A-n39-k6b	106.3	94.5	58.1	106.3	98.3	402	0.3	106.0	1,282	0.2	3	1,211	0.3	
A-n44-k0a	109.5	94.4	05.8	107.0	97.0	545	0.4	107.5	1,008	0.0	2	1,485	0.5	
A-1144-K0D	100.4 102.5	95.0	79.9 67 1	105.8	90.7	611	0.0	100.7	1,999	0.8	ა ა	1,001	0.5	
A-1140-K0a	102.0 104.0	95.9	60.0	102.5	97.5	624	0.5	102.3 102.7	1,980	0.5	ა ა	2,000	0.5	
A-1145-K00	104.0 110.0	90.2	10.7	105.9	99.0	461	0.0	105.7	2,127	0.4	ა ე	2,272	0.4	
A-1145-K7a	110.9	91.1	19.7	107.0	90.0 07.6	401	0.3	109.8	1,100	0.1	2	1,142 1,406	0.1	
A - n46 - k70	10.5 106.5	90.7	19.9	107.0	97.0	430 547	0.4	107.0	2,300	0.1	5 9	1,400	0.1	
A-n46-k7b	100.5 107.4	96.6	46.7	107.0	99.2 97.4	601	0.5	105.8	2,140 2 108	0.5	2	2,043 2,471	0.7	
A-n48-k7a	107.4 103.4	98.6	56 7	107.0	99.0	411	0.4	100.0 103.3	1473	0.0	3	1,411	0.2	
A-n48-k7b	103.7	97.4	58.3	103.5	97.8	510	0.6	103.4	1.984	0.3	2	1.931	0.3	
A-n53-k7a	107.7	90.4	63.1	107.0	96.5	523	0.3	107.0	2.529	0.5	2	2.049	1.1	
A-n53-k7b	106.4	97.3	65.5	106.0	98.6	604	0.7	105.8	2.504	0.4	- 3	2.688	0.2	
A-n54-k7a	105.7	95.2	163.6	105.4	97.2	596	0.8	105.2	2.139	1.6	3	2.281	2.2	
A-n54-k7b	107.2	98.3	72.2	106.3	96.7	597	0.8	106.2	2,463	0.5	3	2,375	0.4	
A-n55-k9a	111.5	91.5	91.6	110.7	95.3	513	0.4	110.5	2,421	0.7	3	3,156	0.6	
A-n55-k9b	108.5	92.9	93.3	107.3	97.5	799	0.7	107.1	3,726	1.9	3	3,595	0.6	
A-n60-k9a	108.9	96.1	95.7	108.5	98.6	681	1.2	108.4	2,892	1.1	3	3,157	2.2	
A-n60-k9b	106.8	93.5	85.1	105.9	96.3	703	1.3	105.7	3,375	1.3	2	3,680	2.2	
A-n61-k9a	110.9	98.2	258.5	109.4	99.8	751	1.1	109.2	2,854	3.4	3	4,223	2.0	
A-n61-k9b	108.7	95.9	131.9	107.4	95.5	993	1.2	107.3	4,317	1.4	3	4,053	0.9	
A-n62-k8a	106.7	91.1	240.2	106.0	94.2	813	1.4	105.9	3,140	1.2	2	3,101	1.9	
A-n62-k8b	106.0	94.4	242.3	105.6	95.1	847	2.0	105.6	$4,\!475$	1.4	3	4,020	0.7	
A-n63-k10a	109.2	96.1	51.9	108.7	98.4	764	0.6	108.6	3,072	0.8	3	3,730	1.2	
A-n63-k10b	107.0	98.6	68.7	106.2	98.7	1,026	1.3	106.1	$3,\!054$	0.7	3	$3,\!686$	0.4	
A-n63-k9a	105.8	98.0	56.4	105.8	-	560	0.8	105.6	$1,\!896$	0.4	2	1,719	0.3	
A-n63-k9b	109.7	96.5	72.7	104.9	-	650	1.2	104.8	$2,\!110$	0.4	3	2,132	0.3	
A-n64-k9a	107.6	95.1	88.9	105.6	95.6	812	1.5	105.5	$2,\!176$	0.7	3	$2,\!423$	0.8	
A-n64-k9b	106.5	93.0	152.6	105.8	-	$1,\!325$	1.5	105.7	$4,\!154$	1.6	3	$4,\!439$	2.2	
A-n65-k9a	107.5	93.9	82.7	106.8	97.4	624	0.7	106.8	2,861	0.6	3	$3,\!671$	0.5	
A-n65-k9b	107.7	93.6	76.9	106.9	96.0	746	1.2	106.8	3,185	0.6	3	3,283	0.4	
A-n69-k9a	108.5	95.9	104.8	107.7	98.8	814	0.9	107.5	4,513	2.1	2	3,863	2.5	
A-n69-k9b	109.1	94.5	78.1	108.0	96.0	951	1.3	107.9	5,028	1.4	3	4,633	0.7	
A-n80-k10a	107.8	97.5	173.5	105.3	98.0	1,096	1.5	105.2	4,087	1.8	2	4,408	2.3	
A-n80-k10b	104.9	95.9	175.2	103.6	97.4	1,250	2.6	103.6	5,697	2.6	2	5,535	2.4	
	106.8	96.3	88.9	106.1	97.7	610	0.7	105.9	2,282	0.7	2.6	2,377	0.8	

Table 6: Dual and primal bounds on instances of class  $[\max |p/t]$ 

Name	V	$n_1$	$n_2$	m	$z^*$	$\operatorname{Cost}$	Load	#r	#o	% ut	ac
A-n32-k5a	32	15	16	4	1.8264	705	386	4	13	96.5	7.0
A-n32-k5b	32	23	8	4	1.8677	734	393	4	6	98.3	7.3
A-n33-k5a	33	16	16	4	1.3984	523	374	4	11	93.5	6.8
A-n33-k5b	33	24	8	5	1.3960	557	399	5	3	79.8	5.4
A-n33-k6a	33	16	16	4	1.1654	444	381	4	5	95.3	5.3
A-n33-k6b	33	24	8	6	1.2990	630	485	6	4	80.8	4.7
A-n34-k5a	34	16	17	4	1.5106	500	331	4	8	82.8	6.0
A-n34-k5b	34	24	9	5	1.5606	682	437	5	5	87.4	5.8
A-n36-k5a	36	17	18	4	1.7247	664	385	4	12	96.3	7.3
A-n36-k5b	36	26	9	5	1.7532	689	393	4	5	98.3	7.8
A-n37-k5a	37	18	18	3	1.7508	520	297	3	7	99.0	8.3
A-n37-k5b	37	27	9	4	1.6292	637	391	4	6	97.8	8.3
A-n37-k6a	37	18	18	4	1.6051	634	395	4	6	98.8	6.0
A-n37-k6b	37	27	9	5	1.6862	833	494	5	4	98.8	6.2
A-n38-k5a	38	18	19	4	1.3813	547	396	4	10	99.0	7.0
A-n38-k5b	38	27	10	5	1.4846	677	456	5	7	91.2	6.8
A-n39-k5a	39	19	19	4	1.4987	559	373	4	10	93.3	7.3
A-n39-k5b	39	28	10	5	1.5894	658	414	5	4	82.8	6.4
A-n39-k6a	39	19	19	5	1.4158	664	469	5	12	93.8	6.2
A-n39-k6b	39	28	10	6	1.4969	723	483	6	4	80.5	5.3
A-n44-k6a	44	21	22	4	1.6286	627	385	4	9	96.3	7.5
A-n44-k6b	44	32	11	6	1.6313	907	556	6	10	92.7	7.0
A-n45-k6a	45	22	22	4	1.6500	627	380	4	6	95.0	7.0
A-n45-k6b	45	33	11	6	1.5650	867	554	6	$\overline{7}$	92.3	6.7
A-n45-k7a	45	22	22	4	1.7051	665	390	4	$\overline{7}$	97.5	7.3
A-n45-k7b	45	33	11	6	1.7082	960	562	6	6	93.7	6.5
A-n46-k7a	46	22	23	4	1.5291	604	395	4	7	98.8	7.3
A-n46-k7b	46	33	12	6	1.4663	827	564	6	8	94.0	6.8
A-n48-k7a	48	23	24	5	1.6511	814	493	5	12	98.6	7.0
A-n48-k7b	48	35	12	6	1.6712	986	590	6	9	98.3	7.3
A-n53-k7a	53	26	26	5	1.4429	720	499	5	10	99.8	7.2
A-n53-k7b	53	39	13	6	1.4661	821	560	6	5	93.3	7.3
A-n54-k7a	54	26	27	5	1.6430	810	493	5	12	98.6	7.6
A-n54-k7b	54	39	14	6	1.6965	1,006	593	6	7	98.8	7.7
A-n55-k9a	55	27	27	6	1.2271	724	590	6	12	98.3	6.5
A-n55-k9b	55	40	14	8	1.2105	897	741	8	6	92.6	5.8
A-n60-k9a	60	29	30	5	1.4888	734	493	5	7	98.6	7.2
A-n60-k9b	60	44	15	7	1.5768	1,099	697	7	8	99.6	7.4
A-n61-k9a	61	30	30	6	1.1142	634	569	6	10	94.8	6.7
A-n61-k9b	61	45	15	8	1.1207	882	787	8	7	98.4	6.5
A-n62-k8a	62	30	31	4	1.8693	744	398	4	6	99.5	9.0
A-n62-k8b	62	45	16	7	1.7125	1,120	654	7	8	93.4	7.6
A-n63-k10a	63	31	31	6	1.3548	798	589	6	10	98.2	6.8
A-n63-k10b	63	46	16	8	1.4383	1,142	794	8	8	99.3	6.8
A-n63-k9a	63	31	31	6	1.7487	1,044	597	6	10	99.5	6.8
A-n63-k9b	63	46	16	8	1.7898	1,405	785	8	8	98.1	6.8
A-n64-k9a	64	31	32	6	1.5874	935	589	6	9	98.2	6.7
A-n64-k9b	64	47	16	7	1.6440	1,136	691	7	5	98.7	7.4
A-n65-k9a	65	32	32	6	1.3232	786	594	6	10	99.0	7.0
A-n65-k9b	65 66	48	16	8	1.3197	1,036	785	8	10	98.1	7.1
A-n69-k9a	69	54	34	5	1.4061	696	495	5	10	99.0	8.8
A-n69-k9b	69	51	17	7	1.3968	968	693	7	7	99.0	8.3
A-n80-k10a	80	39	40	7	1.8371	1,286	700	7	14	100.0	7.6
A-n80-k10b	80	59	20	9	1.8401	1,623	882	9	11	98.0	7.8

Table 7: Details of the best solutions found for class  $[\min |c/l]$ 

Table 8: Details of the best s	solutions found for	class $[\max  p/t]$
--------------------------------	---------------------	---------------------

A-n32-k5a       32       15       16       3       0.3835       306       798       3       5       9.78       6.77         A-n32-k5b       33       23       8       4       0.3809       406       1.066       4       6       9.80.       7.3         A-n33-k5b       33       16       16       3       0.7856       483       720       3       15       9.84.       9.7         A-n33-k6a       33       16       16       3       0.7856       488       583       3       10       82.3       8.7         A-n34-k5b       34       16       17       3       0.5759       368       639       3       9       91.4       8.3         A-n36-k5a       36       17       18       4       0.5614       421       1.0411       4       6       90.90       8.0         A-n37-k5a       37       18       18       3       0.4067       305       750       3       5       98.4       7.7         A-n37-k5a       37       18       8       0.5014       510       1.23       4       19.8       7.7         A-n37-k5a       37       18	Name	V	$n_1$	$n_2$	m	$z^*$	Profit	Time	#r	#o	% ut	ac
A-n32-k5b3223840.38094061.0664698.09.7.3A-n33-k5b33161630.601443372331398.09.7.3A-n33-k6b33161630.785645858331082.38.7A-n33-k6b3324840.6521506776458.1.57.3A-n34-k5a34161730.57593686393991.48.3A-n34-k5b3424940.56174377784583.17.3A-n36-k5a36171880.4067305750359.847.7A-n37-k5a37181830.4067305750359.847.7A-n37-k6b3727950.43253979184790.48.5A-n37-k6a37181840.501450737788886.27.5A-n38-k5a38181940.50703977834880.27.5A-n39-k6b39191940.507039778349.808.0A-n39-k6b3919100.51704037593.687.68A-n39-k6b391910	A-n32-k5a	32	15	16	3	0.3835	306	798	3	5	97.8	6.7
A-n33-k5a       33       16       16       3       0.6014       433       720       3       13       98.0       9.7         A-n33-k5b       33       24       8       4       0.6521       506       766       4       5       81.5       7.3         A-n33-k6b       33       24       8       4       0.6521       506       776       4       5       81.5       7.3         A-n34-k5a       34       16       17       3       0.5759       368       639       3       9       91.4       8.3         A-n36-k5b       36       26       9       4       0.5617       437       778       4       5       83.1       7.3         A-n37-k5b       37       18       18       3       0.4067       305       750       3       5       98.4       7.7         A-n37-k5a       37       18       18       4       0.5074       307       78       4       8       97.5       8.8       8       7.3       A-n38-k5a       9       19       4       0.5310       403       759       3       4       95.8       8.7       A-n38-k6a       8       8.7	A-n32-k5b	32	23	8	4	0.3809	406	1.066	4	6	98.0	7.3
A-n33-k5b3324840.5989433723359.8.49.7A-n33-k6a33161630.785648858331082.38.7A-n34-k5b34161730.57593686393999.1.48.3A-n34-k5b36171840.56174377784583.17.3A-n36-k5b3621940.40444211.041469.08.0A-n37-k5b37181830.4067305750359.8.47.7A-n37-k5b3727950.43253979184790.48.5A-n37-k6b3727960.43715591.279579.8.86.8A-n37-k6b38271040.511143785541286.27.5A-n38-k5a38181940.5107307783495.87.7A-n39-k6a39191940.5507307783495.87.7A-n39-k6b39281050.4774791009449.58.0A-n48-k6b43211250.46774791009498.610.5A-n48-k6b432115 <td< td=""><td>A-n33-k5a</td><td>33</td><td>16</td><td>16</td><td>3</td><td>0.6014</td><td>433</td><td>720</td><td>3</td><td>13</td><td>98.0</td><td>9.7</td></td<>	A-n33-k5a	33	16	16	3	0.6014	433	720	3	13	98.0	9.7
A-n33-k6a33161630.785645858331082.38.7A-n33-k6b3324840.65215067764581.57.3A-n34-k5b34161730.57593686393991.48.3A-n36-k5a36171840.38344031.05141399.97.5A-n36-k5a361718180.40444211.0414699.08.0A-n37-k5b37181830.4067305750359.847.7A-n37-k6b3717181840.50445161.02341198.77.88A-n37-k6b3717960.43715591.279579.86.8A-n38-k5b38181940.51114378554128.07.5A-n38-k5b38271040.5210397783487.58.8A-n39-k6b39191940.53104037593495.88.7A-n44-k6b44321150.47714791.0094495.58.0A-n39-k6b39281050.47674.149.98.68.7A-n45-k6a45	A-n33-k5b	33	24	8	4	0.5989	433	723	3	5	98.4	9.7
A-n33-k6b3324840.65215067764581.57.3A-n34-k5a34161730.575936863939991.48.3A-n34-k5b36171840.38344031.05141399.97.5A-n36-k5b3626940.40444211.0414699.08.0A-n37-k5b371818180.40673057703598.47.73A-n37-k6b3717950.43253979184790.48.5A-n37-k6b3727960.43715591.2795798.86.8A-n38-k5b38271040.5070397783487.58.8A-n39-k5a39191940.50703977834494.98.0A-n39-k6b39281050.47714791.0094495.58.0A-n44-k6a442122250.566751490741480.79.0A-n44-k6a442122250.566751490741480.79.0A-n44-k6a45222250.566751490741480.79.0<	A-n33-k6a	33	16	16	3	0.7856	458	583	3	10	82.3	8.7
A-n34-k5a34161730.57593686393999.48.33A-n34-k5b36171840.38344031,05141399.97.5A-n36-k5a361718180.40444211,0414699.08.0A-n37-k5a371818180.40673057503598.47.7A-n37-k6b3727950.43253979184790.48.5A-n37-k6b3727960.43715591,279798.86.8A-n37-k6b3727960.43715591,279798.86.8A-n38-k5a3819940.55114378754878.66.8A-n39-k6b39281050.47514499454494.98.0A-n44-k6b44321150.47474791,0094495.58.8A-n44-k6b44321150.44705401,2085789.27.8A-n44-k6b44321150.44705401,2085789.27.8A-n44-k6b44321150.566751490741480.790A-n44-k7a46 <td>A-n33-k6b</td> <td>33</td> <td>24</td> <td>8</td> <td>4</td> <td>0.6521</td> <td>506</td> <td>776</td> <td>4</td> <td>5</td> <td>81.5</td> <td>7.3</td>	A-n33-k6b	33	24	8	4	0.6521	506	776	4	5	81.5	7.3
A-n34-k5b3424940.56174377784583.17.3A-n36-k5b36171840.38344031.05141399.97.5A-n36-k5b3626940.40444211.0414699.08.0A-n37-k5a37181830.40673057503598.47.7A-n37-k5b3727950.43253979184790.48.5A-n37-k5b3718181940.511143785541286.27.5A-n38-k5b38181940.51114378554128.68A-n39-k5b39191940.5310403759349.88.6A-n39-k5b39281050.47474791.009449.58.0A-n44-k6b44212240.40844371.07041098.77.8A-n44-k6b442122250.566751490741480.79.0A-n45-k7b45331170.36665691.52778.927.8A-n45-k7b45331170.36665691.52759.005.4 <td< td=""><td>A-n34-k5a</td><td>34</td><td>16</td><td>17</td><td>3</td><td>0.5759</td><td>368</td><td>639</td><td>3</td><td>9</td><td>91.4</td><td>8.3</td></td<>	A-n34-k5a	34	16	17	3	0.5759	368	639	3	9	91.4	8.3
A-n36-k5a36171840.38344031,05141399.97.5A-n36-k5b3626940.40444211,0414699.08.0A-n37-k5b371818180.40673057503598.47.7A-n37-k6b371818180.50445161,02341198.77.3A-n37-k6b3727960.43715591.2795798.86.8A-n38-k5a38181940.511143785541286.27.5A-n38-k5b39191940.5070397783487.66.8A-n39-k6b39281050.47474791,0094495.58.0A-n49-k6b44212240.40844371,07041098.77.8A-n44-k6b44211150.44705401,208578.07.6A-n45-k6b45222250.56675149074480.79.0A-n45-k7b45331170.36665691,5527596.05.4A-n45-k7b45331170.566751490741398.8A-n45-k7b45	A-n34-k5b	34	24	9	4	0.5617	437	778	4	5	83.1	7.3
A-n36-k5b3626940.40444211.0414699.08.0A-n37-k5a3718181830.40673057503598.47.7A-n37-k6a37181840.50445161.02341198.77.3A-n37-k6b3727960.43715591.2795798.86.8A-n38-k5a38181940.511143785541286.27.5A-n38-k5b38271040.49224769674897.58.8A-n39-k5a39191940.53104037593495.87.7A-n39-k6a39181050.47714791.0094495.58.0A-n44-k6a42212240.40844371.07041098.77.8A-n44-k6b4321150.44705401.2085789.27.8A-n44-k6b4321150.52615551.124998.610.5A-n44-k6b4321150.52615557556.05.4A-n44-k7a46222340.555453195641398.4A-n46-k7a4623125 </td <td>A-n36-k5a</td> <td>36</td> <td>17</td> <td>18</td> <td>4</td> <td>0.3834</td> <td>403</td> <td>1.051</td> <td>4</td> <td>13</td> <td>99.9</td> <td>7.5</td>	A-n36-k5a	36	17	18	4	0.3834	403	1.051	4	13	99.9	7.5
A-n37-k5a37181830.4067305750359.8.47.7A-n37-k6b3727950.43253979184790.48.5A-n37-k6b371818140.50445161,02341198.77.3A-n37-k6b3727960.437155512,79579.886.8A-n38-k5b38181940.511143785541286.27.5A-n38-k5b39191940.50703977834878.66.8A-n39-k5a39191940.53104037593495.88.0A-n39-k6a39191940.53104037593495.88.7A-n39-k6a39281050.47474791,0094495.58.0A-n44-k6a44212240.044705401,20857<89.2	A-n36-k5b	36	26	9	4	0.4044	421	1,041	4	6	99.0	8.0
A-n37-k5b372795 $0.4325$ 3979184790.48.5A-n37-k6a3718184 $0.5044$ 516 $1.023$ 41198.77.3A-n37-k6b3818194 $0.5111$ 43785541286.27.5A-n38-k5b3827104 $0.4922$ 476967489.758.8A-n39-k5b3928105 $0.4751$ 4499454494.98.0A-n39-k6b3928105 $0.4747$ 479 $1.009$ 4495.58.0A-n44-k6b4421224 $0.4084$ 437 $1.070$ 41098.77.8A-n45-k6b4533115 $0.4747$ 479 $1.009$ 449.807.8A-n44-k6b44211224 $0.4044$ 437 $1.070$ 41480.79.0A-n45-k6b4533115 $0.5261$ $558$ $1.112$ 4998.610.5A-n45-k7a4522225 $0.5667$ $514$ 90741480.79.0A-n46-k7a4623245 $0.5672$ $574$ $1.012$ 5783.88.0A-n46-k7a4623245 $0.5672$ $574$ $1.012$ 5 <td< td=""><td>A-n37-k5a</td><td>37</td><td>18</td><td>18</td><td>3</td><td>0.4067</td><td>305</td><td>750</td><td>3</td><td>5</td><td>98.4</td><td>7.7</td></td<>	A-n37-k5a	37	18	18	3	0.4067	305	750	3	5	98.4	7.7
A-n37-k6a3718184 $0.5044$ 516 $1,023$ 411 $98.7$ $7.3$ A-n37-k6b372796 $0.4371$ $559$ $1,279$ 57 $98.8$ $6.8$ A-n38-k5b3818194 $0.5070$ $397$ $783$ 48 $75.5$ $8.8$ A-n39-k5b3928105 $0.4751$ $449$ $945$ 44 $94.9$ $8.0$ A-n39-k6a3919194 $0.5070$ $397$ $783$ 44 $94.5.8$ $7.7$ A-n39-k6a3912105 $0.4747$ $479$ $1,009$ 4 $4$ $95.5$ $8.0$ A-n44-k6a4212224 $0.4084$ $437$ $1,070$ 410 $98.7$ $7.8$ A-n45-k6a45222225 $0.5667$ $514$ $907$ 414 $80.7$ $9.0$ A-n45-k7a4522225 $0.5667$ $514$ $907$ 414 $80.7$ $9.0$ A-n46-k7a4623117 $0.3666$ $569$ $1,552$ 75 $96.0$ $5.4$ A-n46-k7a4623245 $0.5672$ $574$ $1,012$ 5 $7.3$ A-n46-k7a4823245 $0.5672$ $574$ $1,012$ 5 $7.98.8$ $8.0$ A-n46-k7a4823245 $0.5672$	A-n37-k5b	37	27	9	5	0.4325	397	918	4	7	90.4	8.5
A-n37-k6b372796 $0.4371$ 559 $1.279$ 5798.86.8A-n38-k5b3818194 $0.51111$ 43785541286.27.5A-n38-k5b3919194 $0.5070$ 3977834897.58.8A-n39-k5a3919194 $0.5070$ 3977834494.98.0A-n39-k6a3919194 $0.5310$ 4037593495.87.7A-n39-k6b3928105 $0.4747$ 479 $1.009$ 4495.58.0A-n44-k6a44212226 $0.4707$ 41098.77.8A-n44-k6b44212226 $0.4470$ 540 $1.208$ 5789.77.8A-n45-k6a4522225 $0.5667$ 51490741480.79.0A-n45-k7a4522225 $0.4470$ 462 $1.141$ 51098.86.4A-n45-k7a4523117 $0.3666$ 530 $1.213$ 51295.44A-n46-k7b4633125 $0.5672$ 574 $1.012$ 5783.38.0A-n48-k7a4823245 $0.5672$ 574 $1.025$ 41398.9 <td>A-n37-k6a</td> <td>37</td> <td>18</td> <td>18</td> <td>4</td> <td>0.5044</td> <td>516</td> <td>1.023</td> <td>4</td> <td>11</td> <td>98.7</td> <td>7.3</td>	A-n37-k6a	37	18	18	4	0.5044	516	1.023	4	11	98.7	7.3
A-n38-k5a3818194 $0.5111$ 43785541286.27.5A-n38-k5b3827104 $0.4922$ 4769674897.58.8A-n39-k5a3919194 $0.5070$ 3977834494.98.0A-n39-k6a3919194 $0.5310$ 40375934495.58.0A-n39-k6b3928105 $0.4747$ 479 $1.009$ 4495.58.0A-n44-k6a4421224 $0.4084$ 437 $1.070$ 41098.77.8A-n45-k6a4522225 $0.5667$ 51490741480.79.0A-n45-k6b4533115 $0.5261$ 5851.1124998.610.5A-n45-k7a45222225 $0.5667$ 5741.012578.3.38.0A-n45-k7a45222223 $4$ $0.5554$ 53195641398.48.8A-n46-k7a4633125 $0.5672$ 574 $1.012$ 578.3.38.0A-n48-k7a4832245 $0.5672$ 574 $1.012$ 578.27.3A-n48-k7b4835126 $0.3955$ 598 $1.512$ 6	A-n37-k6b	37	27	9	6	0.4371	559	1,279	5	7	98.8	6.8
A-n38-k5b3827104 $0.4922$ 4769674897.58.8A-n39-k5b3928105 $0.4751$ 4499454494.98.0A-n39-k6b3919194 $0.5310$ 4037593495.87.7A-n39-k6b3928105 $0.4747$ 479 $1.009$ 4495.58.0A-n44-k6a4421224 $0.4084$ 437 $1.070$ 41098.77.8A-n44-k6b4432115 $0.4470$ 540 $1.208$ 5789.27.8A-n45-k6a4522225 $0.5667$ 51490741480.79.0A-n45-k7a4533115 $0.5261$ 585 $1.112$ 4998.610.5A-n46-k7a4622234 $0.5554$ 53195641398.48.8A-n46-k7a4623245 $0.5672$ 574 $1.012$ 578.38.0A-n46-k7a4823245 $0.5405$ 558 $1.512$ 699.27.3A-n53-k7a5326264 $0.5405$ 554 $1.025$ 41398.99.8A-n54-k7b5439147 $0.3949$ 605 $1.532$ 77	A-n38-k5a	38	18	19	4	0.5111	437	855	4	12	86.2	7.5
A-n39-k5a39191940.50703977834878.66.8A-n39-k6a39191940.53104037593494.98.0A-n39-k6a39281050.47474791,0094495.87.7A-n39-k6b39281050.47474791,0094495.88.0A-n44-k6b44212240.40844371,07041098.77.8A-n44-k6b442122250.566751490741480.79.0A-n45-k7a45222250.566751490741480.79.0A-n45-k7a45222250.566751490741480.79.0A-n45-k7a45222250.40494621,14151098.86.4A-n46-k7a46222340.555453195641398.48.8A-n46-k7b46331250.56725741,0125783.38.0A-n48-k7a48232450.43695301,21351295.57.0A-n48-k7b48351260.39555981,5126999.27.3	A-n38-k5b	38	27	10	4	0.4922	476	967	4	8	97.5	8.8
A-n39-k5b3928105 $0.4751$ 4499454494.98.0A-n39-k6a3919194 $0.5310$ 4037593495.87.7A-n39-k6b3928105 $0.4747$ 479 $1.009$ 4495.58.0A-n44-k6a4421224 $0.4084$ 437 $1.070$ 41098.77.8A-n44-k6b4432115 $0.4747$ 540 $1.208$ 5789.27.8A-n44-k6b4432115 $0.4767$ 540 $1.208$ 5789.77.8A-n45-k7a4522225 $0.5667$ 514 $907$ 41480.79.0A-n45-k7a4522225 $0.4049$ 462 $1.141$ 51098.86.4A-n45-k7a4522234 $0.5554$ 53195641398.48.8A-n46-k7b4633125 $0.5672$ 574 $1.012$ 57783.38.0A-n48-k7a4823245 $0.4369$ 530 $1.213$ 51295.57.0A-n48-k7b4835126 $0.3955$ 598 $1.512$ 6999.27.3A-n53-k7b5339137 $0.4699$ 594 $1.264$ 5<	A-n39-k5a	39	19	19	4	0.5070	397	783	4	8	78.6	6.8
A-n39-k6a39191940.53104037593495.87.7A-n39-k6b39281050.47474791,0094495.58.0A-n44-k6a442112240.40844371,07041098.77.8A-n44-k6b44321150.47475401,2085789.27.8A-n45-k6a45222250.566751490741480.79.0A-n45-k7a45222250.40494621,14151098.86.4A-n45-k7a4522222340.555453195641398.48.8A-n46-k7a46222340.555453195641398.48.8A-n46-k7a46232450.43695301,21351295.57.0A-n48-k7a48351260.39555981,512699.27.3A-n53-k7a53262640.54055541,02541398.99.8A-n54-k7b43391470.39496051,532778.726.6A-n54-k7b54391470.39496051,532778.28.7<	A-n39-k5b	39	28	10	5	0.4751	449	945	4	4	94.9	8.0
A-n39-k6b3928105 $0.4747$ 479 $1,009$ 4495.58.0A-n44-k6a4421224 $0.4084$ 437 $1,070$ 41098.77.8A-n44-k6b4432115 $0.4470$ 540 $1,208$ 5789.27.8A-n45-k6a4522225 $0.5667$ 51490741480.79.0A-n45-k7a4522225 $0.4049$ 462 $1,111$ 51098.86.4A-n45-k7a4522234 $0.5554$ 53195641398.48.8A-n46-k7a4622234 $0.5554$ 53195641398.48.8A-n46-k7b4633125 $0.5672$ 574 $1,012$ 5783.38.0A-n48-k7a4823245 $0.5672$ 574 $1,012$ 5783.38.0A-n48-k7b4835126 $0.3955$ 598 $1,512$ 6999.27.3A-n53-k7b5339137 $0.4699$ 594 $1,264$ 5555.08.8A-n54-k7b5439147 $0.3949$ 605 $1,532$ 7787.26.6A-n55-k9a5527275 $0.7779$ 746959416 </td <td>A-n39-k6a</td> <td>39</td> <td>19</td> <td>19</td> <td>4</td> <td>0.5310</td> <td>403</td> <td>759</td> <td>3</td> <td>4</td> <td>95.8</td> <td>7.7</td>	A-n39-k6a	39	19	19	4	0.5310	403	759	3	4	95.8	7.7
A-n44-k6a44212240.40844371,07041098.77.8A-n44-k6b44321150.44705401,2085789.27.8A-n45-k6a45222250.566751490741480.79.0A-n45-k7a45222250.566751490741480.79.0A-n45-k7a45222250.40494621,14151098.86.10.5A-n45-k7a45222250.40494621,14151098.86.4A-n45-k7a46222340.555453195641398.48.8A-n46-k7a46222340.555453195641398.48.8A-n46-k7b46331250.56725741,0125783.38.0A-n48-k7a48232450.43695301,21351295.57.0A-n48-k7a48331370.46995941,6245595.08.8A-n53-k7b53391370.46995941,6245595.78.2A-n54-k7b4991470.39496051,532778.26.6	A-n39-k6b	39	28	10	5	0.4747	479	1.009	4	4	95.5	8.0
A-n44-k6b4432115 $0.4470$ 540 $1,208$ 57 $89.2$ 7.8A-n45-k6a4522225 $0.5667$ $514$ $907$ 414 $80.7$ $9.0$ A-n45-k7a4522225 $0.4049$ $462$ $1,111$ 5 $10$ $98.8$ $6.4$ A-n45-k7a4522225 $0.4049$ $462$ $1,141$ 5 $10$ $98.8$ $6.4$ A-n45-k7a4622234 $0.5554$ $531$ $956$ 4 $13$ $98.4$ $8.8$ A-n46-k7a4633125 $0.5672$ $574$ $1,012$ 57 $83.3$ $8.0$ A-n48-k7a4835126 $0.3955$ $598$ $1,512$ 6 $9$ $99.2$ $7.3$ A-n54-k7a4835126 $0.3955$ $598$ $1,512$ 6 $9$ $99.2$ $7.3$ A-n53-k7a5326264 $0.5405$ $554$ $1,025$ 4 $13$ $98.9$ $9.8$ A-n54-k7a5426275 $0.4814$ $594$ $1,234$ 5 $15$ $98.7$ $8.2$ A-n54-k7b5439147 $0.3949$ $605$ $1,532$ 77 $87.2$ $6.6$ A-n55-k9a557077 $87.2$ $6.6$ $1,116$ 5 $13$ $85.8$ $8.4$ A-n60-k9a $60$ <td>A-n44-k6a</td> <td>44</td> <td>21</td> <td>22</td> <td>4</td> <td>0.4084</td> <td>437</td> <td>1.070</td> <td>4</td> <td>10</td> <td>98.7</td> <td>7.8</td>	A-n44-k6a	44	21	22	4	0.4084	437	1.070	4	10	98.7	7.8
A-n45-k6a45222250.566751490741480.79.0A-n45-k6b45331150.52615851,1124998.610.5A-n45-k7a45222250.40494621,14151098.86.4A-n45-k7b45331170.36665691,5527596.05.4A-n46-k7a46222340.555453195641398.48.8A-n46-k7a48232450.56725741,0125783.38.0A-n48-k7a48232450.56725741,0125783.38.0A-n48-k7a48232450.54055541,02541398.99.8A-n54-k7a53262640.54055541,02541398.78.2A-n54-k7b53391370.46995941,2645595.08.8A-n54-k7b54391470.39496051,5327787.26.6A-n55-k9a55272750.777974695941694.010.8A-n60-k9a60293050.58786561,11651385.88.7 <t< td=""><td>A-n44-k6b</td><td>44</td><td>32</td><td>11</td><td>5</td><td>0.4470</td><td>540</td><td>1.208</td><td>5</td><td>7</td><td>89.2</td><td>7.8</td></t<>	A-n44-k6b	44	32	11	5	0.4470	540	1.208	5	7	89.2	7.8
A-n45-k6b45331150.52615851,1124998.610.5A-n45-k7a45222250.40494621,14151098.86.4A-n45-k7b45331170.36665691,5527596.05.4A-n46-k7a46222340.555453195641398.48.8A-n46-k7b46331250.56725741,0125783.38.0A-n48-k7a48232450.43695301,21351295.57.0A-n48-k7b48351260.39555981,5126999.27.3A-n53-k7a53262640.54055541,02541398.99.8A-n53-k7b53391370.46995941,2645595.08.8A-n54-k7b54391470.39496051,5327787.26.6A-n55-k9a55272750.777974695941694.010.8A-n60-k9a60293050.58786561,11651385.88.4A-n61-k9b61451560.52817601,4396892.28.7<	A-n45-k6a	45	22	22	5	0.5667	514	907	4	14	80.7	9.0
A-n45-k7a45222250.40494621,14151098.86.4A-n45-k7b45331170.36665691,5527596.05.4A-n46-k7a46222340.555453195641398.48.8A-n46-k7b46331250.56725741,0125783.38.0A-n48-k7a48232450.43695301,21351295.57.0A-n48-k7b48351260.39555981,5126999.27.3A-n53-k7a53262640.54055541,02541398.99.8A-n54-k7a54262750.46995941,2645595.08.8A-n54-k7a54262750.777974695941694.010.8A-n55-k9a55272750.777974695941694.010.8A-n60-k9a60293050.58786561,11651385.88.4A-n61-k9a61303050.797671789941697.311.5A-n61-k9a61303050.797671789941697.311.5 <t< td=""><td>A-n45-k6b</td><td>45</td><td>33</td><td>11</td><td>5</td><td>0.5261</td><td>585</td><td>1.112</td><td>4</td><td>9</td><td>98.6</td><td>10.5</td></t<>	A-n45-k6b	45	33	11	5	0.5261	585	1.112	4	9	98.6	10.5
A-n45-k7b45331170.36665691,5127596.05.4A-n46-k7a46222340.555453195641398.48.8A-n46-k7b46331250.56725741,0125783.38.0A-n48-k7a48232450.43695301,21351295.57.0A-n48-k7b48351260.39555981,5126999.27.3A-n53-k7a53262640.54055541,02541398.99.8A-n54-k7a54262750.48145941,23451598.78.2A-n54-k7b54391470.39496051,5327787.26.6A-n55-k9a55272750.777974695941694.010.8A-n60-k9a60293050.58786561,11651385.88.4A-n61-k9a61303050.797671789941697.31.15A-n61-k9b61451560.77548251,0645992.110.8A-n62-k8a62451670.40857191,76061192.59.3	A-n45-k7a	45	22	22	5	0.4049	462	1.141	5	10	98.8	6.4
A-n46-k7a46222340.555453195641398.48.8A-n46-k7b46331250.56725741,0125783.38.0A-n48-k7a48232450.43695301,21351295.57.0A-n48-k7b48351260.39555981,5126999.27.3A-n53-k7a53262640.54055541,02541398.99.8A-n53-k7b53391370.46995941,2645595.08.8A-n54-k7a54262750.48145941,23451598.78.2A-n54-k7b54391470.39496051,5327787.26.6A-n55-k9a55272750.777974695941694.010.8A-n60-k9a60293050.58786561,11651385.88.4A-n61-k9a61303050.797671789941697.311.5A-n61-k9a61451560.77548251,0645992.110.8A-n62-k8a62303150.69938001,14451689.79.4	A-n45-k7b	45	33	11	7	0.3666	569	1.552	7	5	96.0	5.4
A-n46-k7b46331250.56725741,0125783.38.0A-n48-k7a48232450.43695301,21351295.57.0A-n48-k7b48351260.39555981,5126999.27.3A-n53-k7a53262640.54055541,02541398.99.8A-n53-k7b53391370.46995941,2645595.08.8A-n54-k7a54262750.48145941,23451598.78.2A-n54-k7b54391470.39496051,5327787.26.6A-n55-k9a55272750.777974695941694.010.8A-n60-k9a60293050.58786561,11651385.88.4A-n60-k9a60293050.58786561,11651385.88.4A-n61-k9a61303050.797671789941697.311.5A-n61-k9a61451560.77548251,0645992.110.8A-n63-k10a63313150.69938001,14451689.79.4<	A-n46-k7a	46	22	23	4	0.5554	531	956	4	13	98.4	8.8
A nilo nilo no10101010101010101010A-n48-k7a4823245 $0.3469$ 530 $1,213$ 5 $12$ $95.5$ $7.0$ A-n48-k7b4835 $12$ 6 $0.3955$ $598$ $1,512$ 6 $9$ $99.2$ $7.3$ A-n53-k7a53 $26$ $26$ $4$ $0.5405$ $554$ $1,025$ $4$ $13$ $98.9$ $9.8$ A-n53-k7b53 $39$ $13$ $7$ $0.4699$ $594$ $1,264$ $5$ $5$ $95.0$ $8.8$ A-n54-k7a $54$ $26$ $27$ $5$ $0.4814$ $594$ $1,234$ $5$ $15$ $98.7$ $8.2$ A-n54-k7b $54$ $39$ $14$ $7$ $0.3949$ $605$ $1,532$ $7$ $7$ $87.2$ $6.6$ A-n55-k9a $55$ $27$ $27$ $5$ $0.7779$ $746$ $959$ $4$ $16$ $94.0$ $10.8$ A-n60-k9a $60$ $29$ $30$ $5$ $0.5878$ $656$ $1,116$ $5$ $13$ $85.8$ $8.4$ A-n61-k9a $61$ $30$ $30$ $5$ $0.7976$ $717$ $899$ $4$ $16$ $97.3$ $11.5$ A-n61-k9a $61$ $45$ $15$ $6$ $0.7754$ $825$ $1,064$ $5$ $9$ $92.1$ $10.8$ A-n62-k8a $62$ $30$ $31$ $5$ $0.6993$ $800$ $1,144$ $5$ <t< td=""><td>A-n46-k7b</td><td>46</td><td>33</td><td>12</td><td>5</td><td>0.5672</td><td>574</td><td>1 012</td><td>5</td><td>7</td><td>83.3</td><td>8.0</td></t<>	A-n46-k7b	46	33	12	5	0.5672	574	1 012	5	7	83.3	8.0
A-n48-k7b48511660.39555981,5126999.27.3A-n53-k7a53262640.54055541,02541398.99.8A-n53-k7b53391370.46995941,2645595.08.8A-n54-k7a54262750.48145941,23451598.78.2A-n54-k7b54391470.39496051,5327787.26.6A-n55-k9a55272750.777974695941694.010.8A-n60-k9a60293050.58786561,11651385.88.4A-n61-k9a61303050.797671789941697.311.5A-n61-k9a61303050.797671789941696.59.2A-n62-k8a62303150.43356191,42851696.59.2A-n63-k10a63313150.60938001,14451689.79.4A-n63-k9a63313170.32035831,8207799.25.4A-n63-k9a63313170.32035831,8207799.25.4	A-n48-k7a	48	23	24	5	0.4369	530	1.213	5	12	95.5	7.0
A-nt3-k7a5326264 $0.5405$ 554 $1,022$ 41398.99.8A-nt53-k7a5339137 $0.4699$ $594$ $1,264$ 55 $95.0$ 8.8A-n54-k7a5426275 $0.4814$ $594$ $1,234$ 515 $98.7$ 8.2A-n54-k7b5439147 $0.3949$ $605$ $1,532$ 77 $87.2$ $6.6$ A-n55-k9a5527275 $0.7779$ $746$ $959$ 4 $16$ $94.0$ $10.8$ A-n60-k9a6029305 $0.5878$ $656$ $1,116$ 5 $13$ $85.8$ $8.4$ A-n60-k9b6044156 $0.5281$ $760$ $1,439$ 6 $8$ $92.2$ $8.7$ A-n61-k9a6130305 $0.7976$ $717$ $899$ 4 $16$ $97.3$ $11.5$ A-n61-k9b6145156 $0.7754$ $825$ $1,064$ 59 $92.1$ $10.8$ A-n62-k8a6230315 $0.4335$ $619$ $1,428$ 5 $16$ $96.5$ $9.2$ A-n63-k10a6331315 $0.6093$ $800$ $1,144$ 5 $16$ $99.46$ $9.2$ A-n63-k9a6331317 $0.3203$ $583$ $1,820$ 77 $99.2$ $5.4$ A-n63-k9a63<	A-n48-k7b	48	35	12	6	0.3955	598	1,210 1.512	6	9	99.2	7.3
A-n63-k7b5391070.46695941,2645595.08.8A-n54-k7a54262750.48145941,23451598.78.2A-n54-k7b54391470.39496051,5327787.26.6A-n55-k9a55272750.777974695941694.010.8A-n65-k9a60293050.58786561,11651385.88.4A-n60-k9a60293050.58786561,11651385.88.4A-n61-k9a61303050.797671789941697.311.5A-n61-k9a61303050.797671789941697.311.5A-n62-k8a62303150.43356191,42851696.59.2A-n62-k8a62303150.43356191,42851696.59.2A-n63-k10a63313150.69938001,1445689.79.4A-n63-k9a63313170.32035831,820779.925.4A-n63-k9a63313270.43337701,77771998.87.1 <tr< td=""><td>A-n53-k7a</td><td>53</td><td>26</td><td>26</td><td>4</td><td>0.5405</td><td>554</td><td>1.025</td><td>4</td><td>13</td><td>98.9</td><td>9.8</td></tr<>	A-n53-k7a	53	26	26	4	0.5405	554	1.025	4	13	98.9	9.8
A-n54-k7a5426275 $0.4814$ 594 $1,234$ 51598.78.2A-n54-k7b5439147 $0.3949$ 605 $1,532$ 77787.26.6A-n55-k9a5527275 $0.7779$ 74695941694.010.8A-n55-k9b5540145 $0.8000$ 7609504893.112.0A-n60-k9a6029305 $0.5878$ 656 $1,116$ 51385.88.4A-n60-k9b6044156 $0.5281$ 760 $1,439$ 6892.28.7A-n61-k9a6130305 $0.7976$ 71789941697.311.5A-n61-k9a6130305 $0.7754$ 825 $1,064$ 5992.110.8A-n62-k8a6230315 $0.4335$ 619 $1,428$ 51666.59.2A-n62-k8b6245167 $0.4085$ 719 $1,760$ 61192.59.3A-n63-k10a6331315 $0.6993$ 800 $1,144$ 5699.2A-n63-k9a6331317 $0.3203$ 583 $1,820$ 779.9.25.4A-n63-k9a6346169 $0.3597$ 838 $2,330$	A-n53-k7b	53	39	13	7	0.4699	594	1.264	5	5	95.0	8.8
A-n63-k7b54391470.39496051,532778 7.26.6A-n55-k9a55272750.777974695941694.010.8A-n55-k9b55401450.80007609504893.112.0A-n60-k9a60293050.58786561,11651385.88.4A-n60-k9b60441560.52817601,4396892.28.7A-n61-k9b61451560.77548251,0645992.110.8A-n62-k8a62303150.43356191,42851696.59.2A-n62-k8b62451670.40857191,76061192.59.3A-n63-k10a63313150.69938001,14451689.79.4A-n63-k9b63461670.61058871,4536994.69.2A-n64-k9a64313270.43337701,77771998.86.2A-n64-k9b64471660.45488211,80561297.49.8A-n64-k9a64313270.68797871,14452097.410.4 <td>A-n54-k7a</td> <td>54</td> <td>26</td> <td>27</td> <td>5</td> <td>0 4814</td> <td>594</td> <td>1 234</td> <td>5</td> <td>15</td> <td>98.7</td> <td>8.2</td>	A-n54-k7a	54	26	27	5	0 4814	594	1 234	5	15	98.7	8.2
A-n55-k9a5527275 $0.7779$ 74695941694.010.8A-n55-k9b5527275 $0.7779$ 74695941694.010.8A-n55-k9b5540145 $0.8000$ 7609504893.112.0A-n60-k9a6029305 $0.5878$ 6561,11651385.88.4A-n60-k9b6044156 $0.5281$ 7601,4396892.28.7A-n61-k9a6130305 $0.7976$ 71789941697.311.5A-n61-k9b6145156 $0.7754$ 8251,0645992.110.8A-n62-k8a6230315 $0.4335$ 6191,42851696.59.2A-n63-k10a6331315 $0.6993$ 8001,14451689.79.4A-n63-k9a6331317 $0.3203$ 5831,8207799.25.4A-n63-k9a6341327 $0.4333$ 7701,77771998.86.2A-n64-k9a6431327 $0.4333$ 7701,77771998.87.1A-n64-k9b6447166 $0.4548$ 8211,80561297	A-n54-k7b	54	39	14	7	0.3949	605	1,201 1,532	7	7	87.2	6.6
A-n65-k9b55401450.8007609504893.112.0A-n60-k9a60293050.58786561,11651385.88.4A-n60-k9b60441560.52817601,4396892.28.7A-n61-k9a61303050.797671789941697.311.5A-n61-k9b61451560.77548251,0645992.110.8A-n62-k8a62303150.43356191,42851696.59.2A-n63-k10a63313150.69938001,14451689.79.4A-n63-k10b63461670.61058871,4536994.69.2A-n63-k9a63313170.32035831,8207799.25.4A-n63-k9b63461690.35978382,33091098.86.2A-n64-k9a64313270.43337701,77771998.87.1A-n64-k9b64471660.45488211,80561297.49.8A-n65-k9a65323270.68797871,14452097.410.4	A-n55-k9a	55	27	27	5	0.7779	746	959	4	16	94.0	10.8
A-n60-k9a6029305 $0.5878$ $656$ $1,116$ 5 $13$ $85.8$ $8.4$ A-n60-k9b6029305 $0.5878$ $656$ $1,116$ 5 $13$ $85.8$ $8.4$ A-n60-k9b6144156 $0.5281$ $760$ $1,439$ 6 $8$ $92.2$ $8.7$ A-n61-k9a6130305 $0.7976$ $717$ $899$ 4 $16$ $97.3$ $11.5$ A-n61-k9b6145156 $0.7754$ $825$ $1,064$ 59 $92.1$ $10.8$ A-n62-k8a6230315 $0.4335$ $619$ $1,428$ 5 $16$ $96.5$ $9.2$ A-n63-k10a6331315 $0.6993$ $800$ $1,144$ 5 $16$ $89.7$ $9.4$ A-n63-k10a6331317 $0.3203$ $583$ $1,820$ 77 $99.2$ $5.4$ A-n63-k9a6331317 $0.3203$ $583$ $1,820$ 77 $99.2$ $5.4$ A-n63-k9b6346169 $0.3597$ $838$ $2,330$ 9 $10$ $98.8$ $6.2$ A-n64-k9a6431327 $0.4333$ $770$ $1,777$ 7 $19$ $98.8$ $7.1$ A-n65-k9a6532327 $0.6879$ $787$ $1,144$ $5$ $20$ $97.4$ $10.4$ A-n65-	A-n55-k9b	55	40	14	5	0.8000	760	950	4	8	93.1	12.0
A-n60-k9b6044156 $0.5281$ 760 $1,439$ 68 $92.2$ $8.7$ A-n61-k9a6130305 $0.7976$ 717 $899$ 416 $97.3$ $11.5$ A-n61-k9b6145156 $0.7754$ $825$ $1,064$ 59 $92.1$ $10.8$ A-n62-k8a6230315 $0.4335$ $619$ $1,428$ 5 $16$ $96.5$ $9.2$ A-n62-k8b6245167 $0.4085$ $719$ $1,760$ 6 $11$ $92.5$ $9.3$ A-n63-k10a6331315 $0.6993$ $800$ $1,144$ 5 $16$ $89.7$ $9.4$ A-n63-k10b6346167 $0.6105$ $887$ $1,453$ 69 $94.6$ $9.2$ A-n63-k9b6346167 $0.6105$ $887$ $1,453$ 69 $94.6$ $9.2$ A-n63-k9b6346169 $0.3597$ $838$ $2,330$ 9 $10$ $98.8$ $6.2$ A-n64-k9a6431327 $0.4333$ $770$ $1,777$ $7$ $19$ $98.8$ $7.1$ A-n65-k9a6532327 $0.6879$ $787$ $1,144$ $5$ $20$ $97.4$ $10.4$ A-n65-k9a6548168 $0.6189$ $851$ $1,375$ $6$ $13$ $97.1$ $10.2$ A-n69-k9a	A-n60-k9a	60	29	30	5	0.5878	656	1 116	5	13	85.8	8.4
A-n61-k9a $61$ $30$ $30$ $5$ $0.7976$ $717$ $899$ $4$ $16$ $97.3$ $11.5$ A-n61-k9b $61$ $45$ $15$ $6$ $0.7754$ $825$ $1,064$ $5$ $9$ $92.1$ $10.8$ A-n62-k8a $62$ $30$ $31$ $5$ $0.4335$ $619$ $1,428$ $5$ $16$ $96.5$ $9.2$ A-n62-k8b $62$ $45$ $16$ $7$ $0.4085$ $719$ $1,760$ $6$ $11$ $92.5$ $9.3$ A-n63-k10a $63$ $31$ $31$ $5$ $0.6993$ $800$ $1,144$ $5$ $16$ $89.7$ $9.4$ A-n63-k10b $63$ $46$ $16$ $7$ $0.6105$ $887$ $1,453$ $6$ $9$ $94.6$ $9.2$ A-n63-k9a $63$ $31$ $31$ $7$ $0.3203$ $583$ $1,820$ $7$ $7$ $99.2$ $5.4$ A-n63-k9b $63$ $46$ $16$ $9$ $0.3597$ $838$ $2,330$ $9$ $10$ $98.8$ $6.2$ A-n64-k9a $64$ $31$ $32$ $7$ $0.4333$ $770$ $1,777$ $7$ $19$ $98.8$ $7.1$ A-n65-k9a $65$ $32$ $32$ $7$ $0.6879$ $787$ $1,144$ $5$ $20$ $97.4$ $10.4$ A-n65-k9a $65$ $32$ $32$ $7$ $0.6879$ $787$ $1,144$ $5$ $20$ $97.4$ $10.4$ A-n69-k9a $69$ $34$ $34$ $4$	A-n60-k9b	60	44	15	6	0.5281	760	1,110 1 439	6	8	92.2	87
A-n61-k9b6160606061626210.6A-n62-k8a62303150.43356191,42851696.59.2A-n63-k10a63313150.69938001,14451689.79.4A-n63-k10b63461670.61058871,4536994.69.2A-n63-k9a63313170.32035831,8207799.25.4A-n63-k9b63461690.35978382,33091098.86.2A-n64-k9a64313270.43337701,77771998.87.1A-n64-k9b64471660.45488211,80561297.49.8A-n65-k9a65323270.68797871,14452097.410.4A-n65-k9a69343440.66416781,02141699.712.5A-n69-k9a69511760.59397561,2735899.111.8 <td< td=""><td>A-n61-k9a</td><td>61</td><td>30</td><td>30</td><td>5</td><td>0.7976</td><td>717</td><td>899</td><td>4</td><td>16</td><td>97.3</td><td>11.5</td></td<>	A-n61-k9a	61	30	30	5	0.7976	717	899	4	16	97.3	11.5
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	A-n61-k9b	61	45	15	6	0.7754	825	1.064	5	9	92.1	10.8
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	A-n62-k8a	62	30	31	5	0 4335	619	1 428	5	16	96.5	9.2
A-n63-k10a63313150.69938001,14451689.79.4A-n63-k10b63461670.61058871,453699.469.2A-n63-k9a63313170.32035831,8207799.25.4A-n63-k9b63461690.35978382,33091098.86.2A-n64-k9a64313270.43337701,77771998.87.1A-n64-k9b64471660.45488211,80561297.49.8A-n65-k9a65323270.68797871,14452097.410.4A-n65-k9b65481680.61898511,37561397.110.2A-n69-k9a69343440.66416781,02141699.712.5A-n69-k9b69511760.59397561,2735899.111.8A-n69-k10a80394060.39117611,94661699.89.2A-n69-k10b8059270384490.323197110.89.0	A-n62-k8h	62	45	16	7	0.4085	719	1,420 1 760	6	11	92.5	9.3
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	A_n63_k10a	63	31	31	5	0.4000	800	1,700 1 144	5	16	89.7	9.0
A-n63-k9a6331317 $0.3203$ 583 $1,850$ 7799.25.4A-n63-k9b6346169 $0.3597$ 838 $2,330$ 91098.86.2A-n64-k9a6431327 $0.4333$ 770 $1,777$ 71998.87.1A-n64-k9b6447166 $0.4548$ 821 $1,805$ 61297.49.8A-n65-k9a6532327 $0.6879$ 787 $1,144$ 52097.410.4A-n65-k9b6548168 $0.6189$ 851 $1,375$ 61397.110.2A-n69-k9a6934344 $0.6641$ 678 $1,021$ 41699.712.5A-n69-k9b6951176 $0.3911$ 761 $1,946$ 61699.89.2A-n80-k10b8039406 $0.3911$ 761 $1,946$ 61699.89.2	A_n63_k10b	63	46	16	7	0.6105	887	1 / 53	6	0	94.6	0.1
A-n63-k9b6346169 $0.3597$ 838 $2,330$ 91098.86.2A-n64-k9a6431327 $0.4333$ 770 $1,777$ 71998.87.1A-n64-k9b6447166 $0.4548$ 821 $1,805$ 61297.49.8A-n65-k9a6532327 $0.6879$ 787 $1,144$ 52097.410.4A-n65-k9b6548168 $0.6189$ 851 $1,375$ 61397.110.2A-n69-k9a6934344 $0.6641$ 678 $1,021$ 41699.712.5A-n69-k9b6951176 $0.5939$ 756 $1,273$ 5899.111.8A-n69-k9b6951176 $0.3911$ 761 $1,946$ 61699.89.2A-n80-k10b8059207 $0.3804$ 90323197119810.0	A-n63-k9a	63	31	31	7	0.0100	583	1 820	7	7	00.2	5.4
A-n64-k9a $64$ $31$ $32$ $7$ $0.4333$ $770$ $1,777$ $7$ $19$ $98.8$ $7.1$ A-n64-k9b $64$ $47$ $16$ $6$ $0.4548$ $821$ $1,805$ $6$ $12$ $97.4$ $9.8$ A-n65-k9a $65$ $32$ $32$ $7$ $0.6879$ $787$ $1,144$ $5$ $20$ $97.4$ $10.4$ A-n65-k9b $65$ $48$ $16$ $8$ $0.6189$ $851$ $1,375$ $6$ $13$ $97.1$ $10.2$ A-n69-k9a $69$ $34$ $34$ $4$ $0.6641$ $678$ $1,021$ $4$ $16$ $99.7$ $12.5$ A-n69-k9b $69$ $51$ $17$ $6$ $0.5939$ $756$ $1,273$ $5$ $8$ $99.1$ $11.8$ A-n80-k10a $80$ $39$ $40$ $6$ $0.3911$ $761$ $1,946$ $6$ $16$ $99.8$ $9.2$ A-n80-k10b $80$ $59$ $20$ $7$ $0.3804$ $903$ $2.319$ $7$ $11$ $98$ $10.0$	A-n63-k9h	63	46	16	à	0.3203 0.3507	838	2 330	à	10	08.8	6.2
A-n64-k9b $64$ $47$ $16$ $6$ $0.4535$ $110$ $1,111$ $1$ $15$ $50.6$ $111$ A-n64-k9b $64$ $47$ $16$ $6$ $0.4548$ $821$ $1,805$ $6$ $12$ $97.4$ $9.8$ A-n65-k9a $65$ $32$ $32$ $7$ $0.6879$ $787$ $1,144$ $5$ $20$ $97.4$ $10.4$ A-n65-k9b $65$ $48$ $16$ $8$ $0.6189$ $851$ $1,375$ $6$ $13$ $97.1$ $10.2$ A-n69-k9a $69$ $34$ $34$ $4$ $0.6641$ $678$ $1,021$ $4$ $16$ $99.7$ $12.5$ A-n69-k9b $69$ $51$ $17$ $6$ $0.5939$ $756$ $1,273$ $5$ $8$ $99.1$ $11.8$ A-n80-k10a $80$ $39$ $40$ $6$ $0.3911$ $761$ $1,946$ $6$ $16$ $99.8$ $9.2$ A-n80-k10b $80$ $59$ $20$ $7$ $0.3894$ $903$ $2.319$ $7$ $11$ $88$ $10.0$	A-n64-k99	64	31	32	7	0.0001	770	1,330	7	10	08.8	71
A-n65-k9a6532327 $0.6879$ 787 $1,144$ 52097.4 $10.4$ A-n65-k9b6532327 $0.6879$ 787 $1,144$ 52097.4 $10.4$ A-n65-k9b6548168 $0.6189$ 851 $1,375$ 61397.1 $10.2$ A-n69-k9a6934344 $0.6641$ 678 $1,021$ 41699.7 $12.5$ A-n69-k9b6951176 $0.5939$ 756 $1,273$ 5899.1 $11.8$ A-n80-k10a8039406 $0.3911$ 761 $1,946$ 61699.8 $9.2$ A-n80-k10b8059207 $0.3894$ $903$ $2.319$ 7 $11.989$ $10.0$	A-n64-k9a	64	47	16	6	0.4535	821	1 805	6	12	90.0 07 /	0.8
A-n65-k9b       65 $32$ $32$ $1$ $0.6373$ $131$ $1,144$ $5$ $20$ $97.4$ $10.4$ A-n65-k9b       65 $48$ $16$ $8$ $0.6189$ $851$ $1,375$ $6$ $13$ $97.1$ $10.2$ A-n69-k9a $69$ $34$ $34$ $4$ $0.6641$ $678$ $1,021$ $4$ $16$ $99.7$ $12.5$ A-n69-k9b $69$ $51$ $17$ $6$ $0.5939$ $756$ $1,273$ $5$ $8$ $99.1$ $11.8$ A-n80-k10a $80$ $39$ $40$ $6$ $0.3911$ $761$ $1,946$ $6$ $16$ $99.8$ $9.2$ A-n80-k10b $80$ $59$ $20$ $7$ $0$ $3804$ $903$ $2$ $310$ $7$ $11$ $0.8$ $10.0$	A_n65_k00	65	30	30	7	0.4040	787	1 144	5	20	07 /	10.4
A-n69-k9a       69       34       4 $0.6641$ 678 $1,073$ 6       16       99.1 $10.2$ A-n69-k9a       69       34       34       4 $0.6641$ 678 $1,021$ 4       16       99.7 $12.5$ A-n69-k9b       69       51       17       6 $0.5939$ 756 $1,273$ 5       8       99.1       11.8         A-n80-k10a       80       39       40       6 $0.3911$ 761 $1,946$ 6       16       99.8       9.2         A-n80-k10b       80       59       20       7 $0.3804$ 903 $2.319$ 7       11 $0.8$ 10.0	A_n65_k0h	65	<u>1</u> 8	16	2	0.6180	851	1 275	Б	12	97.4	10.4
A-n69-k9b       69       51       17       6 $0.5939$ 756 $1,273$ 5       8       99.1       12.5         A-n69-k9b       69       51       17       6 $0.5939$ 756 $1,273$ 5       8       99.1       11.8         A-n80-k10a       80       39       40       6 $0.3911$ 761 $1,946$ 6       16       99.8       9.2         A-n80-k10b       80       59       20       7 $0.3804$ 903       2       310       7       11       88       100	A_n60_b0	60	34	34	1	0.66/1	678	1 021	4	16	90.7	10.2 12.5
A-n80-k10a       80 $39$ $40$ $6$ $0.3937$ $700$ $1,245$ $5$ $6$ $99.1$ $11.6$ A-n80-k10a       80 $39$ $40$ $6$ $0.3911$ $761$ $1,946$ $6$ $16$ $99.8$ $9.2$ A-n80-k10b $80$ $59$ $7$ $0$ $3804$ $903$ $2$ $310$ $7$ $11$ $0.8$ $10.0$	A-n60 k0k	60	51	17	4 6	0.0041	756	1.021	-1 5	10	99.1 00.1	11 Q
A-n80-k10b 80 59 20 7 0 3894 003 2310 7 11 08 0 10 $99.8$ 9.2	Δ_n80 k100	80	30	11 10	6	0.0909	750	1.273	0 6	0 16	99.1 00 S	11.0
	A-n80-k10h	80	59	20	7	0.3894	903	2.319	7	11	98.9	10.0