European Journal of Operational Research 245 (2015) 581–590

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

A branch-price-and-cut method for the vegetable crop rotation scheduling problem with minimal plot sizes

Lana M. R. Santos [a], Pedro Munari [b,*], Alysson M. Costa [c], Ricardo H. S. Santos [d]

[a] Departamento de Matemática, Universidade Federal de Viçosa, Viçosa, Brazil
[b] Departamento de Engenharia de Produção, Universidade Federal de São Carlos, São Carlos, Brazil
[c] School of Mathematics and Statistics, University of Melbourne, Melbourne, Australia
[d] Departamento de Fitotecnia, Universidade Federal de Viçosa, Viçosa, Brazil

ARTICLE INFO

Article history:
Received 31 March 2014
Accepted 23 March 2015
Available online 28 March 2015

Keywords:
OR in agriculture
Crop rotation scheduling
Branch-price-and-cut
Strong branching
Subadditive valid inequalities

ABSTRACT

Crop rotation plays an important role in agricultural production models with sustainability considerations. Commonly associated strategies include the alternation of botanical families in the plots, the use of fallow periods and the inclusion of green manure crops. In this article, we address the problem of scheduling vegetable production in this context. Vegetables crop farmers usually manage a large number of crop species with different planting periods and growing times. These crops present multiple and varied harvesting periods and productivities. The combination of such characteristics makes the generation of good vegetable crop rotation schedules a hard combinatorial task. We approach this problem while considering two additional important practical aspects: standard plot sizes (multiples of a base area) and total area minimisation. We propose an integer programming formulation for this problem and develop a branch-price-and-cut algorithm that includes several performance-enhancing characteristics, such as the inclusion of a family of subadditive valid inequalities, two primal heuristics and a strong branching rule. Extensive computational experiments over a set of instances based on real-life data validate the efficiency and robustness of the proposed method.

1. Introduction

Sustainable agricultural production and distribution models have gained attention in the literature in recent years (Alfandari, Lemalade, Nagih, & Plateau, 2011; Bachinger & Zander, 2007; Detlefsen & Jensen, 2007; Haneveld & Stegeman, 2005). In particular, we have been interested in the use of crop rotation as a pest control and conservation of productive resources strategy in the farming of vegetables. The scheduling of vegetable crop rotation, especially when the goal is to supply a given demand, presents several complicated characteristics associated with the fact that crops with very different planting, growing and (multiple and coupled) harvesting times must be addressed simultaneously.

In the past decade, a few mathematical formulations have been proposed in the literature with the purpose of considering ecologically based constraints. Santos (2009) developed an integer programming model by considering the following three ecological based criteria: (i) crops of the same botanical family cannot be grown one after another in the same land plot, (ii) a green manure crop must be periodically grown in each plot and (iii) there are requirements in terms of fallowness that must be respected. These criteria aim at reducing the population growth of pests and weeds; recovering of the physical, chemical and biological soil properties; increasing soil fertility and incorporating nitrogen into the system through biological fixation (Altieri, 1995; Dias, Dukes, & Antunes, 2014; Wezel et al., 2013). They have been used in a number of follow-up articles addressing variants of the model proposed in Santos (2009), including, for instance, concerns regarding the supply of demands (Santos, Costa, Arenales, & Santos, 2010), the management of perishable stocks (Costa, Santos, Alem, & Santos, 2014) and the extension of these ecologically based constraints to avoid growing crops from the same botanical family simultaneously in neighbouring plots (Santos, Arenales, Costa, & Santos, 2010; Santos, Michelon, Arenales, & Santos, 2011). These problems have been solved with variants of a column generation algorithm (Lübbecke & Desrosiers, 2005; Vanderbeck & Wolsey, 2010) in which the generated columns carry the crop planting schedule information.

The articles mentioned above deal with the sizing of the plots, but following two different approaches. Either they assume that the plot size is an exogenous parameter (Santos, Arenales et al., 2010; Santos et al., 2011) or they let the optimisation model to freely decide on plot

* Corresponding author. Tel.: +55 16 33519525.
E-mail addresses: lanamara@ufv.br (L. M. R. Santos), munari@dep.ufscar.br (P. Munari), alysson.costa@unimelb.edu.au (A. M. Costa), rsantos@ufv.br (R. H. S. Santos).

http://dx.doi.org/10.1016/j.ejor.2015.03.035
0377-2217/

sizes by using endogenous variables (Costa et al., 2013; Santos, Costa et al., 2010). While both approaches might have their applicabilities, they also have limitations. Fixing candidate plot sizes a priori reduces the solution space and might, therefore, lead to sub-optimal configurations. On the other hand, imposing no sizing constraints usually lead to a large number of plots, with very different and frequently small areas. This is contrary to what is expected in practice, where plot size standardisation is a welcome characteristic because it eliminates the need of dealing with very small plots (with sizes smaller than the base area) and also facilitates farming management. Indeed, many caring techniques, such as irrigation, are planned for or more suitably used in plots with standard sizes.

In this paper, we focus on a crop rotation scheduling problem that deals with the sizing of the plots endogenously, while imposing that they are multiples of a base area. We also consider the ecological based criteria mentioned earlier and an explicit area minimisation objective. The idea of area minimisation has been already addressed in Alfandari et al. (2011) and has clear advantages, such as the reduction of infrastructure and operational costs and the facilitation of the maintenance of conservation spaces with wider environments, thus reducing the pressure on areas that are more susceptible to degradation. To tackle this problem, we propose an integer programming model that minimises the planting area while guaranteeing that the obtained plot sizes are multiples of a base value and supply a given demand. Since the proposed model corresponds to an extensive formulation, we develop a branch-price-and-cut (BPC) algorithm.

BPC algorithms have shown to be a powerful tool for solving hard combinatorial problems (Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance, 1998; Vanderbeck & Wolsey, 2010). In the context of crop scheduling, we are aware of two other applications of the method, which deal with crops with different growth cycles (Santos, Arenales et al., 2010) and equal growth cycles (Alfandari, Plateau, & Schepler, 2015). In Santos, Arenales et al. (2010), the authors propose a BPC for a vegetable crop rotation (using criteria (i)–(iii)) with additional adjacency constraints imposing that crops of the same botanical family cannot be grown simultaneously in neighbouring areas. Plots locations and sizes were determined a priori, allowing the model to be reformulated as a pure 0–1 master problem. In Alfandari et al. (2015), the authors use a BPC algorithm to solve a crop rotation problem that select plots from a candidate list in order to minimise the combined area of chosen plots while meeting a given demand. They deal with crops that have a single growing cycle (and therefore, one harvest period) and only two possible planting periods per year, following the same ideas proposed in Alfandari et al. (2011). These crop characteristics allow the feasible crop rotations to be expressed by means of a clever transition graph, so the subproblems can be solved by using a shortest path algorithm. In both Santos, Arenales et al. (2010) and Alfandari et al. (2015), the models are defined using a set of potential plot sizes chosen a priori. This allows the use of binary variables associated with the selection of scheduling sequences for each plot and the efficient use of traditional branching rules.

The BPC algorithm proposed in this paper is structurally different from those proposed in Santos, Arenales et al. (2010) and Alfandari et al. (2015). Indeed, since the model includes decisions on both plot sizes and schedules simultaneously, the decision variables must be defined as general integers, i.e., they are allowed to take any positive integer value. To be effective in such context, the BPC has to rely on branching rules based on sets of variables, aided with a strong branching strategy. In addition, we aim at tackling practical situations that involve up to 20 crops with diverse characteristics and have to follow a fine granularity of the time discretisation (each year represented as 48 time periods). These features lead to large instances which require a number of further performance-enhancing characteristics, such as the use of a set of newly developed family of valid inequalities, which improve the quality of the bounds obtained by the column generation subroutine, and two primal heuristics.

To verify the performance of the proposed BPC algorithm in practice, we have run an extensive set of computational experiments using instances based on real-life data. For all instances, the proposed method was able to prove 1 percent-optimality within very reasonable computational times. The results also indicate the positive effect of the proposed performance-enhancing features.

The remainder of this paper is organised to present this whole set of ideas coherently. Section 2 defines the problem and presents the developed mixed-integer model. The solution methodology is then described in Section 3, in which we detail, in order, the column generation procedure, the branching strategy, the proposed valid inequalities, and the primal heuristic procedures that compose our branch-price-and-cut method. Section 4 presents the results of the computational experiments while Section 5 ends this paper with conclusions and suggested routes for further investigation.

2. Model

Let N be the number of different available crops and M be the number of time periods (days, weeks, months, etc.) in which the growing of these crops is going to be scheduled to satisfy a given known demand. The crop rotation scheduling problem (CRSP) consists of determining schedules for the crop rotation activity such that the production of each crop at each period is sufficient to satisfy its corresponding demand, while respecting a set of ecological criteria and biological methods. We consider that the main objective is to minimise the size of the planting area. In addition, we assume that the planting area is divided into plots, which have a predefined minimum size denoted by $a_{\min}$ and that the size of any plot can only be a multiple of this value. Consider the following additional parameters:

C	a set of crops that are planted to satisfy the demand;
G	a set of crops that are available for green manuring;
N	total number of crops, i.e., cardinality of $C \cup G$;
$N+1$	index of a fictitious crop, representing the presence of fallow;
NF	number of botanical families;
$F(p)$	set of crops from the botanical family p, $p = 1, \ldots,$ NF;
t_i	production time for crop i, including soil preparation and harvesting;
I_i	set of time periods in which crop i can be planted;
t_f	minimum time interval to wait between planting two crops from the same botanical family.

We can write linear constraints forcing a single plot schedule to respect the ecologically based criteria mentioned previously. Indeed, using binary variables:

$$x_{ij} = \begin{cases} 1, & \text{if crop } i \text{ is planted in time period } j, \\ 0, & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, N+1$ and $j = 1, \ldots, M$, these constraints read (Santos, Costa et al., 2010):

$$\sum_{i=1}^{N+1} \sum_{r=0}^{t_i-1} x_{i,j-r} \leq 1, \quad j = 1, \ldots, M, \tag{1}$$

$$\sum_{i \in F(p)} \sum_{r=0}^{t_i-1+t_f} x_{i,j-r} \leq 1, \quad p = 1, \ldots, \text{NF}, \quad j = 1, \ldots, M, \tag{2}$$

$$\sum_{i \in G} \sum_{j \in I_i} x_{ij} = 1, \tag{3}$$

$$\sum_{j=1}^{M} x_{N+1,j} = 1, \tag{4}$$

$$x_{ij} \in \{0,1\}, \quad i = 1,\ldots,N+1, \quad j \in I_i \tag{5}$$

where $j - r < 0$ is replaced by $j - r + M$ in constraints (1) and (2). Constraints (1) ensure that at most one crop is planted at each time period (including fallow), whereas constraints (2) forbid planting crops from the same botanical family in a time interval smaller than t_f. Constraint (3) imposes that a crop must be planted for green manuring and constraint (4) guarantees the presence of fallow. Both constraints (3) and (4) can be easily modified to incorporate the need of more fallows or green manure crops during the rotation period (Santos, 2009).

The right-hand side in constraint (4) enforces the requirement of a single period of fallow. This is not a limitation of the model, but a choice that follows our experience with vegetable farmers in Brazil. The common practice among these farmers is to make intense use of the land. In addition, we base this requirement on the recent literature that addresses the effect of a single short fallow on crop yield (Babu, Rao, & Veeraraghavaiah, 2014; Dias et al., 2014; Gabriel & Quemada, 2011; Hunt & Kirkegaard, 2012) and on the fact that we also consider green manuring as a way of recovering soil chemical and physical properties and interrupting the population growth of pests and weeds (Allen, Pikul, Waddell, & Cochran, 2011; Kröbel et al., 2014; Manyanga, Mafongoya, & Tauro, 2014; Miller et al., 2011).

Let X be the set of all feasible schedules that satisfy constraints (1) to (5). Additionally, let S be the set of indices associated with the schedules in X, such that each schedule in X can be denoted by x^s for a unique $s \in S$. Given a schedule x^s, we are interested in determining the production that results from the use of such schedule in a given plot. To obtain this information, the following parameters are needed:

p_{ir}	total production of crop i on its rth harvesting;
o_i	time interval between the period crop i is planted and the period of its first harvesting.

The production of crop i at time period j in an area of size equal to the base area, when using schedule x^s, is then given by

$$a_{ij}^s = \sum_{r=1}^{t_i - o_i - 1} a_{\min} p_{ir} x_{i,j-o_i-r}^s, \tag{6}$$

where $a_{\min}$ stands for the base plot area.

Relationship (6) converts schedule-based information to crop production values. This is crucial for obtaining the following model, which aims at finding schedules with a combined production that meets the required demand while minimising the total planting area:

$$\min \sum_{s \in S} u^s \tag{7}$$

$$\text{s.t.} \sum_{s \in S} a_{ij}^s u^s \geq d_{ij}, \quad i \in C, \quad j = 1,\ldots,M, \tag{8}$$

$$\sum_{s \in S} u^s \leq L, \tag{9}$$

$$u^s \in \mathbb{Z}_+, \quad s \in S, \tag{10}$$

where u is the vector of decision variables such that each component u^s is the number of base-area plots of size $a_{\min}$ that is assigned to schedule $s \in S$. Parameter d_{ij} is the demand of crop i in time period j, for $i \in C$ and $j = 1, \ldots, M$, and therefore constraints (8) ensure that the required demand is met. Constraint (9) imposes that the number of base-area plots does not exceed the upper limit L. This constraint can be tightened by changing L in (9) to any known upper bound on the solution of the problem. As we have observed in preliminary computational experiments, this strategy was helpful to speed up the proposed solution method. We assume that planting area is homogeneous, as in practice the plots used to grow the vegetable crops have typically similar characteristics, as a result of years of similar cropping management. The model can be easily extended to cope with heterogeneous areas (see, for instance Santos, Arenales et al., 2010; Santos, Costa et al., 2010).

The usual large number of possible schedules and the integrality of variables u^s makes the resolution of such a model a difficult challenge. The following section describes the method that has been proposed for this purpose.

3. Methodology

In this section, we describe the main components of the branch-price-and-cut method that we implemented to solve problem (7)–(10). First, we describe the column generation procedure used to solve the linear relaxations at each branching node. Then, we present a strong branching strategy and propose a class of valid inequalities used to tighten these linear relaxations. Finally, we present two heuristics used to further improve the computational performance of the method.

3.1. Column generation technique

As mentioned above, the number of schedules in S is usually very large even for problems with a small number of crops and periods, making the explicit resolution of model (7)–(10) impractical. A common workaround in such cases is the use of the column generation technique which is based on the idea of artificially reducing the number of variables and dynamically generating promising columns during the solution method (Lübbecke & Desrosiers, 2005; Vanderbeck & Wolsey, 2010).

In the case of this paper, this aim can be achieved by initially solving a so-called restricted master problem (RMP), in which the integrality of variables u^s is ignored and which contains just a (small) subset $\bar{S} \subset S$ of the variables:

$$\min \sum_{s \in \bar{S}} u^s \tag{11}$$

$$\text{s.t.} \sum_{s \in \bar{S}} a_{ij}^s u^s \geq d_{ij}, \quad i \in C, \quad j = 1,\ldots,M, \tag{12}$$

$$\sum_{s \in \bar{S}} u^s \leq L, \tag{13}$$

$$u^s \geq 0, \quad s \in \bar{S}. \tag{14}$$

After solving the RMP, the method searches for promising columns, i.e., columns with negative reduced cost. Let $(\pi_{11}^*, \pi_{12}^*, \ldots, \pi_{|C|M}^*)$ and σ^* be the dual variables associated with constraints (12) and (13), respectively, in the optimal solution of the RMP. The reduced cost of a column u^s is given by:

$$\tilde{c}^s = 1 - \sum_{i \in C} \sum_{j=1}^{M} a_{ij}^s \pi_{ij}^* - \sigma^*.$$

Therefore, the aim is to find a schedule $x_{ij} \in \{0,1\}, i = 1, \ldots, N+1$, $j = 1, \ldots, M$, respecting constraints (1)–(5) with associated column u^s minimising the above expression. Relations (6) enable such pricing subproblem to be written as follows:

$$\max \quad \sum_{i \in C} \sum_{j=1}^{M} \sum_{r=1}^{t_i - o_i - 1} p_{ir} x_{i,j-o_i-r} \pi_{ij}^* \quad \text{s.t. } x \in X.$$

Let $x^\star$ represent the optimal solution of this subproblem. If

$$1 - a_{\min} \sum_{i \in C} \sum_{j=1}^{M} \sum_{r=1}^{t_i - o_i - 1} p_{ir} x_{i,j-o_i-r}^\star \pi_{ij}^* - \sigma^* < 0,$$

then the column obtained from $x^\star$ is attractive. This column can be written as $(a^*_{11}, a^*_{12}, \ldots, a^*_{|C|M}, 1)^T$ and be inserted in the RMP. As before, parameters a^*_{ij} can be obtained using the relationship given in (6).

After a new column is inserted, the RMP is solved again, and the method iterates. The optimal solution of the linear relaxation of the master problem is obtained when no column with a negative reduced cost can be found. Such an optimal solution corresponds to a lower bound for the optimal solution of the integer programming problem (7)–(10). An integer solution, in turn, can be obtained by branching on fractional u^s variables. This is discussed in the following section.

3.2. Branching

In a branch-price-and-cut algorithm, defining an efficient branching strategy is typically more complicated than in the classical branch-and-bound method. Indeed, the standard variable dichotomy rule may compromise the structure of the subproblem as well as lead to an unbalanced tree (Vanderbeck, 2000b; Vanderbeck & Wolsey, 2010). Hence, more elaborate strategies are often required to achieve good computational performance.

We propose branching constraints of the type (Vance, 1998; Vanderbeck, 1999; 2000a):

$$\sum_{s\in Q} u^s \le \overline{\alpha} - 1, \qquad \sum_{s\in Q} u^s \ge \overline{\alpha}, \tag{15}$$

which simultaneously use several RMP variables at a time. Set Q, which contains the indices of the variables in the branching constraints, and the RHS value $\overline{\alpha}$ are obtained as follows. For each $i \in C$ and $j = 1, \ldots, M$, we define the set

$$S_{ij} = \{s \in S : a^s_{ij} > 0\}, \tag{16}$$

which is given by the indices of master variables u^s with positive coefficient a^s_{ij} in the associated constraint (12). In other words, S_{ij} contains the indices of the columns (schedules) that contribute to the production of crop i in period j. If the following summation

$$\alpha_{ij} = \sum_{s\in S_{ij}} \overline{u}^s \tag{17}$$

is fractional, then S_{ij} is a candidate branching set. Q is chosen as the candidate set corresponding to pair (i, j) leading to the minimum value

$$\theta_{ij} = \sum_{s\in S_{ij}} a^s_{ij}\overline{u}^s - d_{ij}, \tag{18}$$

over all $i \in C$ and $j = 1, \ldots, M$, where θ_{ij} is the extra production of crop i in period j in the current solution $\overline{u}$. The value $\overline{\alpha}_{ij} = \lceil \alpha_{ij} \rceil$ is used to set $\overline{\alpha}$ in (15). By using the minimum θ_{ij} to define set Q, we aim at selecting a branching set composed by master variables with nonzero coefficients in an active constraint of the RMP, according to the current optimal solution. Therefore, this strategy increases the chances of effectively perturbing the RMP of the child nodes.

Alternative rules can be used to define the candidate branching sets. For instance, let $a^\star_{ij}$ be the largest coefficient in the RMP row associated with pair (i, j). The candidate branching set may be defined as follows:

$$S_{ij} = \{s \in S : a^s_{ij} \ge a^\star_{ij}\}. \tag{19}$$

Notice that in this case, only a subset of the master variables with a nonzero coefficient in row ij will be selected, namely those with the largest coefficient. In Section 4, we compare the impact of these two choices within the branch-price-and-cut method.

3.2.1. Strong branching and implementation details

An efficient branching strategy should have a relatively small computational cost and deliver the most significant increase in the lower bounds of each child node. The branching rule proposed above uses the minimum θ_{ij} to decide which candidate set S_{ij} should be chosen for branching. Although this increases the chances of making an effective choice, additional analysis may be helpful to improve the quality of the branching decision. Hence, we propose a strong branching strategy (Achterberg, Koch, & Martin, 2005; Linderoth & Savelsbergh, 1999) in which we estimate the candidate branching sets that are likely to most improve the lower bounds of the generated child nodes.

For each $i \in C$ and $j = 1, \ldots, M$, let z^L_{ij} and z^R_{ij} be the optimal values of the two linear programming problems obtained from the current RMP after adding the constraints $\sum_{s\in S_{ij}} u^s \le \overline{\alpha}_{ij} - 1$ and $\sum_{s\in S_{ij}} u^s \ge \overline{\alpha}_{ij}$, respectively. No columns are generated when solving these two problems to quickly obtain z^L_{ij} and z^R_{ij}. As a result, these values only provide estimates for the lower bounds of the left and right child nodes in cases where S_{ij} is chosen as the branching set. The pair (i, j) yielding the largest estimates z^L_{ij} and z^R_{ij} is the one selected for branching in the branch-price-and-cut algorithm.

To reduce the computational cost of this search, we sort the pairs (i, j) in nondescending order of the associated values θ_{ij} and restrict our search to those pairs in the first positions. More specifically, we initially define set Q as the candidate set S_{ij} corresponding to the smallest θ_{ij}. Then, by analysing each pair (i, j) in the pre-defined order, we redefine Q whenever better values for z^L_{ij} and z^R_{ij} are found. We stop this search when Q is redefined for the second time. Typically, only few candidates are investigated using this strategy before a good quality choice is obtained, as the pairs with small θ_{ij} are expected to generate better branching constraints.

Some remarks are in order. In cases where the problem used to obtain z^L_{ij} or z^R_{ij} is infeasible, the associated estimate value assumes LB+2 (where LB is the current lower bound of the current node), somehow prioritising candidates that lead to pruning. Because the LHS in the constraints (15) is a summation over variables, this value might be an integer even if the individual variables are not. There is, therefore, a risk of finding no valid branching constraint. In these cases, one must use the traditional dichotomy rule, i.e., given a fractional component $\overline{u}^s$ of the optimal solution, we create two child nodes by imposing one of the following variable bounds to each subproblem

$$u^s \le \lceil \overline{u}^s \rceil - 1, \quad u^s \ge \lceil \overline{u}^s \rceil. \tag{20}$$

In our tests, this branching rule was rarely required.

3.2.2. Changes in the subproblem

Branching with column generation is not a trivial task. In addition to the typical difficulty in finding an effective branching strategy, we must also consider the changes they imply at the subproblem level. Branching rules of type (20) generally imply quite complex changes, as they must avoid a subset of schedules to become the optimal solutions of the subproblem. The difficulty of solving the subproblem increases for each new variable bound added to the RMP. Branching constraints of type (15) require milder modifications to the subproblem. In fact, assume that in a given node of the branch-and-bound tree we have K branching constraints in the RMP, which are determined by K branching sets $Q_1, Q_2, \ldots, Q_K$. In association with the branching constraints, we have their dual variables $(\delta_1, \delta_2, \ldots, \delta_K)$ that must be considered at the subproblem level. Hence, for each $k = 1, \ldots, K$, we add a binary variable w_k to the subproblem, which assumes the value 1 only if schedule x leads to a positive production of crop i at time period j, such that pair (i, j) was used to define branching set Q_k. This means that the index of the column resulting from this schedule must be added to the kth branching set. To ensure this relationship holds, we add the following constraint to the subproblem, for each $k = 1, \ldots, K$:

$$\sum_{r=1}^{t_{i_k}-o_{i_k}-1} x_{i_k j_k - o_{i_k} - r} = w_k,$$

where i_k and j_k are the crop and period associated with the branching set Q_k.

When the left-hand-side of the above equation equals 1, this value signals that there is production of crop i_k at time period j_k and thus we must have $w_k = 1$. Notice that the result of this summation can be at most 1, as the production originates from a single planting (see model (1)–(5)). Finally, it remains to be mentioned that the binary variables must also be added to the objective function of the subproblem. Their costs are given by the dual variables associated with the branching constraints in the RMP. Hence, we add the following summation to the objective function of the subproblem

$$-\delta_1 w_1 - \delta_2 w_2 - \ldots - \delta_K w_K.$$

The changes in the objective function must also be applied to the computation of the reduced costs. It is worth mentioning that all these required changes do not affect significantly the difficulty of solving the subproblem in practice, as we will see in the computational results presented in Section 4.2.

3.3. Valid inequalities

To improve the quality of the lower bounds provided by the master problem, we propose a family of valid inequalities. These inequalities, called the special rounding inequalities, correspond to general-purpose cuts such as the Chvátal–Gomory inequalities (Nemhauser & Wolsey, 1988). They are inspired by the dual feasible functions proposed in Fekete and Schepers (2001) for the bin-packing problem.

3.3.1. Special rounding inequalities

The special rounding inequalities are derived from constraints (8) of the RMP. They have the following form:

$$\sum_{s\in\bar{S}} F_{ij}^k(a_{ij}^s)u^s \geq F_{ij}^k(d_{ij}) \tag{21}$$

for each $i \in C$ and $j = 1, \ldots, M$, where $d_{ij} > 0$ and $a_{ij}^s \leq d_{ij}$, for each $s \in \bar{S}$. The family of functions $F_{ij}^k : [0, d_{ij}] \to [0, k+1]$ is defined as

$$F_{ij}^k(x) = \begin{cases} \frac{(k+1)x}{d_{ij}}, & \text{if } \frac{kx}{d_{ij}} \in \mathbb{Z}, \\ \lceil \frac{kx}{d_{ij}} \rceil, & \text{otherwise,} \end{cases} \tag{22}$$

for $k = 1, \ldots, d_{ij}$ and $0 \leq x \leq d_{ij}$. Notice that $F_{ij}^k(d_{ij}) = k+1$, for any $i \in C$, $j = 1, \ldots, M$ and $k = 1, \ldots, d_{ij}$. Based on Proposition 4.1 in Nemhauser and Wolsey (1988, p. 231), to verify that (21) is a valid inequality for $P = \mathbb{Z}^n \cap \{u \in \mathbb{R}_+^{|\bar{S}|} : \sum_{s\in\bar{S}} a_{ij}^s u^s \geq d_{ij}\}$, it is sufficient to show that each function F_{ij}^k is subadditive, non-decreasing and satisfies $F_{ij}^k(0) = 0$. From definition (22), it is clear that $F_{ij}^k(0) = 0$ and F_{ij}^k is non-decreasing. In Proposition 3.1, we show that this function is subadditive as well.

Proposition 3.1. *The function F^k: $[0, d] \to [0, k+1]$ defined as*

$$F^k(x) = \begin{cases} \frac{(k+1)x}{d}, & \text{if } \frac{kx}{d} \in \mathbb{Z}, \\ \lceil \frac{kx}{d} \rceil, & \text{otherwise,} \end{cases} \tag{23}$$

is subadditive, for any $d \in \mathbb{Z}_+$ and $k = 1, \ldots, d$ such that $d > 0$.

Proof. To be subadditive, the function must satisfy $F^k(x) + F^k(y) \geq F^k(x+y)$, for any $x, y, x+y \in \mathbb{R} \cap [0, d]$. Hence, given $x, y \in [0, d]$ such that $x + y \in [0, d]$, we have one out of the following possible cases:

1. $\frac{kx}{d}, \frac{ky}{d} \in \mathbb{Z}$. Hence, $F^k(x) + F^k(y) = \frac{(k+1)(x+y)}{d} = F^k(x+y)$;
2. $\frac{kx}{d} \in \mathbb{Z}$, $\frac{ky}{d} \notin \mathbb{Z}$. Consequently, $\frac{k(x+y)}{d} \notin \mathbb{Z}$. Hence, $F^k(x) + F^k(y) = \frac{(k+1)x}{d} + \lceil \frac{ky}{d} \rceil \geq \frac{kx}{d} + \lceil \frac{ky}{d} \rceil = \lceil \frac{kx}{d} + \frac{ky}{d} \rceil = F^k(x+y)$;
3. $\frac{kx}{d}, \frac{ky}{d} \notin \mathbb{Z}$, but $\frac{k(x+y)}{d} \in \mathbb{Z}$. First, as $\frac{kx}{d}$ and $\frac{ky}{d}$ are fractional, then $\lceil \frac{kx}{d} \rceil + \lceil \frac{ky}{d} \rceil$ must be strictly larger than $\frac{kx}{d} + \frac{ky}{d}$. Because $\frac{kx}{d} + \frac{ky}{d}$ is an integer, it follows that $\lceil \frac{kx}{d} \rceil + \lceil \frac{ky}{d} \rceil = \frac{kx}{d} + \frac{ky}{d} + 1$. Additionally, $x + y \in [0, d] \Rightarrow \frac{x+y}{d} \leq 1$. Hence, $F^k(x) + F^k(y) = \frac{kx}{d} + \frac{ky}{d} + 1 \geq \frac{(k+1)(x+y)}{d} = F^k(x+y)$;
4. $\frac{kx}{d}, \frac{ky}{d}, \frac{k(x+y)}{d} \notin \mathbb{Z}$. Thus, $F^k(x) + F^k(y) = \lceil \frac{kx}{d} \rceil + \lceil \frac{ky}{d} \rceil \geq \lceil \frac{kx}{d} + \frac{ky}{d} \rceil = \lceil \frac{k(x+y)}{d} \rceil = F^k(x+y)$.

Since $F^k(x) + F^k(y) \geq F^k(x+y)$ in all cases, the proof is complete. □

Inequalities (21) have two advantageous properties. First, they are equivalent to or dominate a class of rank-1 Chvátal–Gomory inequalities. Second, we need to generate only those inequalities with k odd, as they dominate the inequalities obtained with other values of k. Proposition 3.2 state and prove these properties.

Proposition 3.2. *Special rounding inequalities (21) are equivalent to or dominate any rank-1 Chvátal–Gomory inequalities of the form*

$$\sum_{s\in\bar{S}} \lceil \gamma a_{ij}^s \rceil u^s \geq \lceil \gamma d_{ij} \rceil \tag{24}$$

with $\gamma \in (\frac{1}{d_{ij}}, 1)$, for $i \in C$, $j = 1, \ldots M$ and $d_{ij} > 0$. Moreover, the inequalities defined with $F_{ij}^k(x)$ are equivalent to or dominate those defined with $F_{ij}^{2k}(x)$, for any $k \in \{1, \ldots, d_{ij}\}$ and $0 \leq x \leq d_{ij}$.

Proof. To prove that inequalities (21) are equivalent to or dominate inequalities (24), we have to show that $\lceil \gamma a_{ij}^s \rceil \geq F_{ij}^k(a_{ij}^s)$ whenever $\lceil \gamma d_{ij} \rceil = F_{ij}^k(d_{ij})$, with $\gamma \in (\frac{1}{d_{ij}}, 1)$. Given γ such that $\lceil \gamma d_{ij} \rceil = F_{ij}^k(d_{ij}) = k+1$, it follows that $\gamma d_{ij} > k$ and hence $\gamma x > \frac{kx}{d_{ij}}$, for any $x \in (0, d_{ij})$. If $\frac{kx}{d_{ij}} \in \mathbb{Z}_+$ then $\lceil \gamma x \rceil \geq \frac{kx}{d_{ij}} + 1 \geq \frac{kx}{d_{ij}} + \frac{x}{d_{ij}} = \frac{(k+1)x}{d_{ij}} = F_{ij}^k(x)$. Otherwise, $\frac{kx}{d_{ij}} \notin \mathbb{Z}_+$ implies in $\lceil \gamma x \rceil \geq \lceil \frac{kx}{d_{ij}} \rceil = F_{ij}^k(x)$, which completes the first part of the proof.

We prove now the dominance of special rounding inequalities defined with $F_{ij}^k(x)$ over those defined with $F_{ij}^{2k}(x)$. From (21), it follows that

$$\sum_{s\in\bar{S}} 2F_{ij}^k(a_{ij}^s)u^s \geq 2k+2,$$

as $F_{ij}^k(d_{ij}) = k+1$. On the other hand, we have also from (21) that

$$\sum_{s\in\bar{S}} F_{ij}^{2k}(a_{ij}^s)u^s \geq 2k+1. \tag{25}$$

For any feasible solution of the master problem we have that $\sum_{s\in\bar{S}} u^s \geq 1$. Using this fact and (25), we can write

$$\sum_{s\in\bar{S}} (F_{ij}^{2k}(a_{ij}^s) + 1)u^s \geq 2k+2.$$

Following the same reasoning of the first part of the proof, it suffices to show now that $(F_{ij}^{2k}(a_{ij}^s) + 1) \geq 2F_{ij}^k(a_{ij}^s)$. Indeed, we have one out of the following three cases:

1. $\frac{2kx}{d_{ij}} \notin \mathbb{Z}$. In such case, $\frac{kx}{d_{ij}} \notin \mathbb{Z}$. Hence, $2F_{ij}^k(x) = 2\lceil \frac{kx}{d_{ij}} \rceil = \lceil \frac{kx}{d_{ij}} \rceil + \lceil \frac{kx}{d_{ij}} \rceil \leq \lceil \frac{2kx}{d_{ij}} \rceil + 1 = F_{ij}^{2k}(x) + 1$.
2. $\frac{kx}{d_{ij}} \in \mathbb{Z}$, so that $\frac{2kx}{d_{ij}} \in \mathbb{Z}$. Hence, $2F_{ij}^k(x) = 2\frac{(k+1)x}{d_{ij}}$ and $F_{ij}^{2k}(x) + 1 = \frac{(2k+1)x}{d_{ij}} + 1 = \frac{2kx}{d_{ij}} + \frac{x}{d_{ij}} + 1 \geq \frac{2kx}{d_{ij}} + \frac{2x}{d_{ij}}$, as $\frac{x}{d_{ij}} \leq 1$. As a result, $F_{ij}^{2k}(x) + 1 \geq \frac{2kx}{d_{ij}} + \frac{2x}{d_{ij}} \geq 2\frac{(k+1)x}{d_{ij}} = 2F_{ij}^k(x)$.
3. $\frac{2kx}{d_{ij}} \in \mathbb{Z}$ and $\frac{kx}{d_{ij}} \notin \mathbb{Z}$. Hence, $2\lceil \frac{kx}{d_{ij}} \rceil - 2\frac{kx}{d_{ij}} < 2$ which implies that $2\lceil \frac{kx}{d_{ij}} \rceil - 2\frac{kx}{d_{ij}} = 1$. It follows that $2\lceil \frac{kx}{d_{ij}} \rceil = 1 + 2\frac{kx}{d_{ij}} = 1 + \lceil 2\frac{kx}{d_{ij}} \rceil$.

Since $F_{ij}^{2k}(x) = \frac{(2k+1)x}{d_{ij}}$ and $2F_{ij}^k(x) = 2\lceil \frac{kx}{d_{ij}} \rceil$, we obtain from the above implications that $2F_{ij}^k(x) = F_{ij}^{2k}(x) + 1$, which completes the proof. □

The separation of special rounding inequalities can be achieved with a simple full enumeration algorithm, which is called each time the column generation algorithm converges. The violated inequalities of type (21) are added to the RMP and the process is restarted until no violated inequality is found.

3.3.2. Changes in the subproblem

Similar to the discussion presented in Section 3.2, adding the special rounding inequalities to the RMP also requires modifying the subproblem to take into account the corresponding dual variables. However because each valid inequality is based on the coefficients of a single master constraint of type (12), the original dual variable can be used and we do not need to create any additional variable in the subproblem. More specifically, let $i_p \in C$, $j_p \in \{1, \ldots, M\}$ and $k_p \in \{1, \ldots, d_{i_p j_p}\}$ be the parameters that were used to generate the pth valid inequality in the RMP. Let σ_p be the dual variable for constraint (12) associated with such parameters. The subproblem is then modified by adding the term $F_{i_p j_p}^{k_p}(a_{\min} p_{i_p r})\sigma_p$ to the cost of each variable $x_{i_p, j_p - o_{i_p} - r}$, for $r = 1, \ldots, t_{i_p} - o_{i_p} - 1$. Needless to say, the changes in the objective function must also be applied to the computation of the reduced costs. As it was the case when adding branching constraints, the changes required by the valid inequalities did not significantly increase the difficulty of solving the subproblem.

3.4. Primal heuristics

Primal heuristics are typically useful to accelerate the convergence of a branch-and-price algorithm. Hence, we propose two simple heuristic procedures to improve the performance of the branch-price-and-cut method, namely the rounding heuristic (RH) and the constructive heuristic (CH).

3.4.1. Rounding heuristic (RH)

The RH is called just before the branching procedure, after the column generation has finished. The purpose of this heuristic is to obtain a feasible integer solution based on the columns currently available at the last solved RMP. The heuristic works by simply imposing integrality to the master variables and then calling a generic MIP solver for a limited amount of time. At the end of this procedure, if a feasible solution is found, this solution corresponds to a feasible solution of the original problem (7)–(10). MIP-based heuristics have been widely used in the column generation literature (Joncour, Michel, Sadykov, Sverdlov, & Vanderbeck, 2010). Although very simple, they have shown to be powerful in practice, in special for the case we address here, as we will verify later in Section 4.2.4.

3.4.2. Constructive heuristic (CH)

The CH is a classical residual heuristic that is called in the middle of the column generation procedure. The main purpose of this heuristic is to find an integer solution that is feasible for the original MP (7)–(10). In addition, it helps in the diversification of the types of columns obtained in the column generation procedure. At each iteration of this heuristic, a new column is generated by calling a modified version of the subproblem. The schedules corresponding to the generated columns are combined to provide a tentative solution, which may lead to a feasible solution when the heuristic procedure terminates.

We may call the CH during any iteration of the column generation procedure after solving the subproblem and obtaining an attractive new column. To describe the heuristic procedure, let $\tilde{d}$ be the residual demand vector, which is initially defined as the original demand d. Additionally, let $(1, a'_{11}, a'_{12}, \ldots, a'_{|C|M}, 1)^T$ represent the last generated column. The procedure begins by adding this column to the current RMP, setting $\tilde{d}$ as the right-hand side and then solving this problem. Using the entries of the column and $\tilde{d}$, we compute the value

$$u' = \min\left\{\left\lceil \frac{\tilde{d}_{ij}}{a'_{ij}} \right\rceil : \tilde{d}_{ij} > 0 \text{ and } a'_{ij} > 0; i \in C; j = 1, \ldots, M\right\},$$

which corresponds to the minimum relative area required to satisfy at least one of the demand values $\tilde{d}_{ij}$ by the corresponding production a'_{ij}. The resulting u' is used to fix the current column in the tentative solution, i.e., the schedule used to generate the column is assigned to a plot of relative size u', as part of a potential feasible solution of the original problem. After computing u', we update the residual demand as follows:

$$\tilde{d}_{ij} = \max\{\tilde{d}_{ij} - u' a'_{ij}, 0\}, \quad \text{for } i \in C,\ j = 1, \ldots, M.$$

In the case $\tilde{d}_{ij} = 0$ for all $i \in C, j = 1, \ldots, M$, the tentative solution is complete and feasible for the original MP (7)–(10), and thus the procedure terminates. Otherwise, a new column should be generated through another call to the subproblem. In our implementation, we use a modified version of the subproblem, in which the objective function is adjusted according to the residual demand $\tilde{d}$. For each variable x_{ij} in the subproblem, with $i \in C$ and $j = 1, \ldots, M$, we assign the residual cost

$$c_{ij}^{\text{res}} = \sum_{r=1}^{t_i - o_i - 1} a_{\min} p_{ir} \tilde{d}_{i, j + o_i + r}.$$

The rationale behind this idea is to force the subproblem to generate schedules for the production of crops in time periods for which there is still a residual demand. After solving this subproblem, a new column is generated and the process is repeated. Calling the CH at each iteration of the column generation algorithm may be computationally expensive. Hence, we call the CH heuristic periodically (e.g., at every five outer iterations of column generation and at every 10 nodes of the branch-and-price tree). In Section 4.2.4, we present computational experiments to show the behaviour of this heuristic when it is embedded in the branch-price-and-cut algorithm.

4. Computational experiments

In this section the performance of the BPC method is assessed through computational experiments. To test the proposed method we developed a C++ implementation on top of the CPLEX 12.4 Concert Library, which is used to solve each RMP and subproblem.

4.1. Data generation

We generated a set of CRSP instances based on real-life data obtained from a producer in Barbacena, in the countryside of Brazil (21 degrees 3 arcminutes 30 arcseconds S and 43 degrees 46 arcminutes 25 arcseconds W, 1165 metres above sea level). The original dataset included 26 crops with their different planting and growing times and productivities (Santos, Costa et al., 2010). This set was used to randomly generate classes of instances with different planning horizons, namely $M = 48$ and 96 periods, corresponding to 1 and 2 years, respectively. At each period, a percentage (d_f) of the possible harvesting periods of each crop (according to periods of crop planting) was assumed to have positive demands, with d_f = 20–50 percent or 60–90 percent. These assumptions yielded four classes, which we designate P48-20/50, P48-60/90, P96-20/50 and P96-60/90. Each of these classes contains 15 instances with different numbers of crops (five instances for each $N = 10, 15$, and 20). In all experiments, $a_{\min}$ has been fixed at 1 square metre and the initial L has been set to 50, 000, a sufficiently large value so that no instance would become infeasible.

4.2. Results

We are interested in the overall performance of the proposed BPC method, as well as in the effect of the various features of the

algorithm discussed in the previous sections. The complete version of this method has been implemented with the following features:

- The strong branching strategy presented in Section 3.2 and using branching sets (16) which includes all variables with positive coefficients in the constraint;
- The special rounding valid inequality (SR) proposed in Section 3.3. The separation procedure is called at every node, when the column generation procedure finishes;
- The rounding heuristic (RH) described in Section 3.4.1. This heuristic is called once at every 50 processed nodes (the first time being in the root node) and is allowed to run for at most 100 seconds in each call;
- The constructive heuristic (CH) described in Section 3.4.2. This heuristic is called once at every five outer iterations of column generation, but only at every 10 processed nodes (the first time being in the root node);
- Best-bound search node selection strategy.

We have chosen this configuration based on the performances observed in preliminary computational experiments with different versions of the BPC method. Some of these experiments are later summarised in this section. Before that, we present and analyse the overall results of the complete version. To obtain these results, we have limited the maximum running time to 1 hour.

4.2.1. Overall performance

Table 1 shows the results of solving the CRSP instances using the complete version of the proposed BPC method. For each instance, the first two columns of Table 1 give the class and the name of the instance, respectively. The name is given in the format *N*–*k*, where *N* is the number of crops and *k* is the instance identifier. The third column (UB) gives the value of the best integer solution found by the method. The fourth column (Gap) shows the relative gap (in percent) between the UB value and the best lower bound (LB) found by the method. This gap is computed as $100(\text{UB} - \text{LB})/(10^{-10} + \text{UB})$. The relative gap at the *root node* is given on the fifth column (Gap root) and is computed using the UB and LB available at the end of the root node, just before branching and after solving the rounding heuristic. Columns 6 and 7 give the total number of nodes (Nodes) and cuts (Cuts) generated in the branching tree, respectively. Columns 8–11 give the total CPU time (Total), the CPU time spent in the root node only (Root), the total CPU time spent solving all the pricing subproblems and the total CPU time spent with the rounding heuristic (Rounding), all in seconds. The time spent in the root node includes the time required by the first run of the rounding heuristic. The instances that reached the maximum running time are marked with the symbol '*' in the total CPU time column. Finally, the last two columns of Table 1 show the number of processed nodes and the elapsed time needed to find the best integer solution corresponding to the UB value.

The results in Table 1 show that the proposed BPC algorithm was able to solve all instances to 1 percent-optimality (i.e., with a relative gap less than or equal to 1 percent). Moreover, most of the instances could be solved to optimality. The class P96-60/90 was the most challenging for the method, as many instances could not be solved to optimality with the imposed maximum running time. On the other hand, class P96-20/50 was the easiest, as the method could solve to optimality all of the instances in this class. The total CPU time and the number of nodes were typically small for the instances solved to optimality. In most of the instances, the best integer solution was found at the root node (node 0) by using the rounding heuristic. Nevertheless, branching was useful in several instances to reduce the relative gap and to prove optimality.

Having discussed the overall performance of the proposed BPC method, we are interested now in analysing the effect of each component of this algorithm. Hence, in the following subsections, we present the results of computational experiments with different variants of the method. For the sake of clarity, we analyse the performance of these variants in terms of the percentage of proven optimal solutions obtained in 1 hour of running time. To evaluate each of these individual algorithm aspects, we present the results obtained when a single feature is turned off at a time. In all of the tables below, the first column gives the name of the instance class; columns 2–4 present the results for the complete (best) version of the algorithm; and the remaining columns present the results for other versions of the algorithm. For each version, the tables show the percentage of instances that were solved to optimality (Opt), to 0.1 percent-optimality ($\leq$0.1), and to 1 percent-optimality ($\leq$1).

4.2.2. Branching rules

Table 2 presents the percentages of optimal, 0.1 percent-optimal and 1 percent-optimal solutions for different variants of the branching strategy proposed in Section 3.2. We have included the results of the following strategies:

- Rule 1: The standard strategy used in the complete version of the algorithm;
- Rule 2: Same configuration as in Rule 1, except for not using strong branching. The candidate branching sets are analysed in the non-descending order of values θ_{ij} for each row ij, as described in Section 3.2;
- Rule 3: Same configuration as in Rule 1, but using (19) as branching set;
- Rule 4: Same configuration as in Rule 1, but using strong branching with the variable dichotomy rule described in (20).

The results indicate that Rule 1 (which includes all proposed features) is indeed the most efficient algorithmic version, as optimality was proven for a larger percentage of instances. By comparing Rules 1 and 2, we were able to observe the importance of using the strong branching technique. The choice of an appropriate branching set is also important, as fewer instances were solved to optimality using Rule 3. As expected, using the variable dichotomy rule (Rule 4) resulted in a drastic reduction of the method's performance, even though the strong branching technique was still used. Computational experiments using Rules 3 and 4 without using strong branching yielded significantly worse results. A number of other variants of the addressed rules were also tested during the computational experiments, but none of them presented interesting results.

4.2.3. Valid inequalities

In this section, we compare different versions of the BPC method regarding the use of valid inequalities. The complete version of the method uses the special rounding (SR) inequality, as proposed in Section 3.3. The first variant that we consider does not use any valid inequality, while the second one uses only the rank-1 Chvátal–Gomory (ChG) inequality. As mentioned earlier, the SR inequality dominates the ChG inequality. The results presented in Table 3 indicate that only the proposed SR inequalities were beneficial to the convergence of the algorithm. On the other hand, the bound improvement obtained by the use of the ChG inequalities did not compensate for the time spent generating them, as the overall performance of the corresponding variant was worse than the performance of the variant that did not rely on any valid inequality.

4.2.4. Heuristic procedures

In Section 3.4, we described two primal heuristics to be used within the BPC method, namely the rounding heuristic (RH) and the constructive heuristic (CH). Table 4 presents the results obtained with different configurations of the method regarding the use of these heuristics: when CH and RH are both used (CH: yes, RH: yes), when only one is used (CH: no, RH: yes/CH: yes, RH: no), and when both are ignored (CH: no, RH: no). The results indicate that RH is crucial for

Table 1
Results of solving the CRSP instances using the complete version of the proposed branch-price-and-cut algorithm.

Instance		UB	Gap (percent)	Gap root (percent)	Nodes	Cuts	CPU time (seconds)				UB	
Class	Name						Total	Root	Subproblem	Rounding	Node	Time (seconds)
P48-20/50	10-1	1090	0	0	1	6	0.46	0.46	0.34	0.10	0	0.46
	10-2	518	0.193	0.386	2735	262	*	129.16	693.38	2406.54	0	129.16
	10-3	1349	0	0	1	90	25.28	25.28	24.69	0.13	0	25.28
	10-4	805	0	0.124	5	55	5.77	5.31	4.34	1.10	0	5.31
	10-5	839	0	0	1	5	0.33	0.33	0.20	0.06	0	0.33
	15-1	1900	0	0.053	101	235	246.61	185.04	209.51	0.57	100	246.61
	15-2	837	0.119	0.239	1201	351	*	209.83	694.35	2004.26	0	209.83
	15-3	1966	0	0.102	11	107	134.74	131.00	129.77	0.22	0	131.00
	15-4	1720	0	0.058	13	176	254.86	249.89	147.66	100.26	0	249.89
	15-5	1496	0.067	0.267	4959	120	*	21.29	1641.63	411.65	100	147.61
	20-1	1977	0	0.051	3	55	204.02	202.42	195.79	0.08	0	202.42
	20-2	1032	0	0.194	239	266	772.58	517.74	595.66	14.97	0	517.74
	20-3	1051	0.285	0.381	701	488	*	738.30	1044.80	1448.76	0	738.30
	20-4	1920	0	0.104	33	96	94.58	80.07	86.71	0.21	0	80.07
	20-5	1624	0	0.062	487	282	1010.35	524.72	707.31	101.38	0	524.72
P48-60/90	10-1	280	0	1.071	1805	612	1658.11	158.24	704.42	347.61	0	158.24
	10-2	390	0	0	1	0	9.75	9.75	9.63	0.01	0	9.75
	10-3	754	0	0.265	365	293	397.20	195.41	169.06	156.11	0	195.41
	10-4	283	0	0.353	11	4	6.95	5.51	6.60	0.07	0.00	5.51
	10-5	2503	0	0	1	10	0.41	0.41	0.28	0	0	0.41
	15-1	5464	0	0	1	75	34.35	34.35	33.06	0.11	0	34.35
	15-2	3737	0	0.080	347	392	1696.67	515.21	499.86	602.41	200	1383.61
	15-3	3019	0	0.132	91	309	836.63	637.64	536.08	200.48	0	637.64
	15-4	1605	0.062	0.187	2001	369	*	225.02	1060.61	1690.68	0	225.02
	15-5	2741	0	0.109	65	177	170.52	145.32	161.73	0.25	0	145.32
	20-1	2377	0	0.084	763	881	2366.11	618.93	1298.25	6.83	0	618.93
	20-2	1773	0.056	0.113	571	1093	*	1175.99	1659.94	426.60	0	1175.99
	20-3	2284	0	0.088	37	361	835.14	785.16	689.97	100.40	0	785.16
	20-4	3079	0	0.032	31	581	596.01	550.70	553.91	0.98	0	550.70
	20-5	2033	0.049	0.148	1181	585	*	660.47	1558.58	1106.76	0	660.47
P96-20/50	10-1	1491	0	0.134	7	45	26.96	23.31	25.22	0.20	0	23.31
	10-2	568	0	0.176	3	73	50.44	49.04	49.12	0.21	0	49.04
	10-3	792	0	0	1	13	10.48	10.48	9.82	0.17	0	10.48
	10-4	2169	0	0	1	25	0.78	0.78	0.66	0.01	0	0.78
	10-5	543	0	0	1	80	14.30	14.30	13.23	0.11	0	14.30
	15-1	1090	0	0.183	451	230	1279.45	150.07	290.74	105.30	450	1279.45
	15-2	1477	0	0.135	17	115	277.81	245.32	262.89	2.44	0	245.32
	15-3	1392	0	0	1	14	115.53	115.53	110.62	0.15	0	115.53
	15-4	1019	0	0	1	156	107.12	107.12	101.55	0.70	0	107.12
	15-5	1285	0	0	1	40	100.31	100.31	97.30	0.15	0	100.31
	20-1	979	0	0	1	167	357.79	357.79	310.84	18.34	0	357.79
	20-2	803	0	0.124	51	394	1453.30	1020.29	1062.72	121.67	50	1453.30
	20-3	685	0	0.292	285	541	2178.49	442.20	808.55	301.12	0	442.20
	20-4	779	0	0.384	101	468	1979.87	581.28	582.68	213.79	100	1979.87
	20-5	1033	0	0	1	346	577.04	577.04	535.36	4.04	0	577.04
P96-60/90	10-1	38258	0	0.003	7	0	24.28	6.75	6.43	0.04	6	24.28
	10-2	43529	0	0.002	25	118	89.16	17.98	22.74	0.04	0	17.98
	10-3	8963	0	0.011	133	170	840.38	131.38	113.15	200.24	0	131.38
	10-4	16753	0	0.006	69	246	200.69	11.16	48.26	0.36	0	11.16
	10-5	34209	0	0.006	151	346	650.69	29.68	105.65	0.51	150	650.69
	15-1	9633	0.073	0.073	15	291	*	2751.71	2744.67	100.47	0	2751.71
	15-2	6888	0.174	0.174	35	540	*	2060.10	2039.75	100.93	0	2060.10
	15-3	8272	0.169	0.181	18	889	*	1138.94	1794.76	100.51	0	1138.94
	15-4	9216	0.065	0.076	34	500	*	641.65	1412.30	100.49	0	641.65
	15-5	16552	0	0.030	195	268	1539.49	130.53	418.49	100.27	0	130.53
	20-1	9053	0.044	0.066	59	562	*	661.94	892.25	201.21	0	661.94
	20-2	6873	0.291	0.291	4	843	*	1568.64	2767.16	100.64	0	1568.64
	20-3	9729	0.051	0.062	22	595	*	817.21	1300.17	100.60	0	817.21
	20-4	9229	0.108	0.130	41	694	*	295.27	2524.47	100.60	0	295.27
	20-5	8980	0.145	0.156	6	709	*	2016.85	2711.29	100.80	0	2016.85

* The maximum running time was reached (3600 seconds).

improving the convergence of the method. Indeed, the performance of the method was considerably affected when this heuristic was disregarded. On the other hand, all instances could be solved within a 1 percent optimality gap when this heuristic was used. Regarding CH, the percentages in Table 4 lead to a mixed conclusion. By comparing the case in which both heuristics are ignored against the case using CH alone, the constructive heuristic has a negative effect on convergence. Nevertheless, when RH is present, the inclusion of CH improves performance. One possible explanation for this result is that CH generates extra columns that have little value by themselves but can be usefully incorporated in the rounding procedure of RH.

4.2.5. *Effect of increasing demand*

In this section, we analyse how the performance of the complete version of the algorithm varies when the demands are increased. Table 5 shows the results for the following different demand

Table 2
Results of using different branching rules.

	Rule 1			Rule 2			Rule 3			Rule 4		
Class	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1
P48-20/50	73.3	80.0	100.0	40.0	73.3	100.0	66.7	80.0	100.0	26.7	46.7	100.0
P48-60/90	80.0	100.0	100.0	33.3	60.0	93.3	60.0	100.0	100.0	20.0	46.7	93.3
P96-20/50	100.0	100.0	100.0	66.7	80.0	100.0	86.7	86.7	100.0	53.3	60.0	100.0
P96-60/90	40.0	66.7	100.0	6.7	73.3	100.0	26.7	73.3	100.0	6.7	66.7	100.0

Table 3
Results of the branch-price-and-cut method using the special rounding inequality (SR), the rank-1 Chvátal–Gomory inequality (ChG), and without using any inequality.

	SR inequalities			No valid inequalities			ChG inequalities		
Class	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1
P48-20/50	73.3	80.0	100.0	40.0	66.7	100.0	40.0	66.7	100.0
P48-60/90	80.0	100.0	100.0	60.0	93.3	100.0	53.3	86.7	100.0
P96-20/50	100.0	100.0	100.0	80.0	80.0	100.0	80.0	80.0	100.0
P96-60/90	40.0	66.7	100.0	20.0	66.7	100.0	13.3	20.0	46.7

Table 4
Results of different versions of the branch-price-and-cut method regarding the use of the heuristics.

	CH: yes, RH: yes			CH: no, RH: yes			CH: yes, RH: no			CH: no, RH: no		
Class	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1
P48-20/50	73.3	80.0	100.0	73.3	80.0	100.0	40.0	40.0	53.3	33.3	33.3	60.0
P48-60/90	80.0	100.0	100.0	66.7	93.3	100.0	40.0	40.0	40.0	40.0	40.0	40.0
P96-20/50	100.0	100.0	100.0	73.3	80.0	100.0	26.7	26.7	26.7	33.3	33.3	33.3
P96-60/90	40.0	66.7	100.0	40.0	66.7	100.0	13.3	20.0	46.7	40.0	26.7	46.7

Table 5
Results obtained with the original demands and with the demands multiplied by two and by four.

	Original demand			Demand * 2			Demand * 4		
Class	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1	Opt	≤ 0.1	≤ 1
P48-20/50	73.3	80.0	100.0	80.0	100.0	100.0	73.3	100.0	100.0
P48-60/90	80.0	100.0	100.0	73.3	100.0	100.0	53.3	93.3	100.0
P96-20/50	100.0	100.0	100.0	86.7	100.0	100.0	80.0	100.0	100.0
P96-60/90	40.0	66.7	100.0	26.7	93.3	100.0	26.7	100.0	100.0

scenarios: using the original demands d_{ij} of the instances (Original demand), using d_{ij} multiplied by two (Demand * 2), and using d_{ij} multiplied by four (Demand * 4). No clear conclusions can be drawn from these results because in some cases the performance was slightly improved whereas in others, the performance was slightly degraded. More importantly, however, these results appear to suggest that the method is robust to demand variations, as it was still able to solve all instances within 1 percent of optimality (considering the same 1 hour running time limit).

Table 5 can also be seen as an analysis of different values for the $a_{\min}$ parameter, as increasing all the demand is equivalent to reducing the size of the minimum accepted plot. These results indicate that the method seems to be robust to variations on this parameter.

5. Conclusions

Here, we have addressed a sustainable vegetable crop rotation scheduling problem with the goal of minimising the planting area while supplying the given demands and respecting standardised plot sizes. The production of vegetables usually occurs in coupled sequential harvesting periods, which differentiates this situation from other multi-period scheduling problems, such as the cutting stock problem. The existence of multiple harvests, in which a single planting might supply the demands for various future periods, is associated with the need to use integer variables to model the requirement of standard plot sizes and the fact that real-life instances usually address a large number of periods, making this a very difficult combinatorial problem. To solve this problem, we have proposed a branch-price-and-cut method that incorporates enhanced features, such as a strong branching strategy using branching sets, a family of newly developed subadditive valid inequalities, and primal heuristics procedures.

To verify the performance of the proposed method, we randomly generated four classes of instances based on real-life data. The results of the computational experiments on these instances suggest that the resulting method is efficient and robust to demand variation. Most of the instances could be solved to optimality. The relative gap of the 16 instances for which optimality was not achieved was less than 1 percent. In addition, we analysed the impact of the main features of the method on its overall performance. Our analyses indicate that the column generation combined within a MIP-based rounding heuristic at the root node could be used to provide high quality solutions and are probably sufficient in practice. Nevertheless, branching could improve both lower and upper bounds for a number of instances. The strong branching technique and the proposed valid inequalities contributed significantly to the improvement of the performance of the branch-price-and-cut method. Furthermore, primal heuristics were shown to be crucial to the accelerated convergence of the method.

Future research in this area should incorporate additional aspects that might make the obtained solutions even more suitable in practice, including a multi-objective analysis of factors as total area, number of plots and other ecologically based criteria.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable contributions to this paper. This research was supported by Fundação Arthur Bernardes (FUNARBE), Brazil.

References

Achterberg, T., Koch, T., & Martin, A. (2005). Branching rules revisited. *Operations Research Letters, 33*(1), 42–54.

Alfandari, L., Lemalade, J., Nagih, A., & Plateau, G. (2011). A MIP flow model for crop-rotation planning in a context of forest sustainable development. *Annals of Operations Research, 190*(1), 149–164.

Alfandari, L., Plateau, A., & Schepler, X. (2015). A branch-and-price-and-cut approach for sustainable crop rotation planning. *European Journal of Operational Research, 241*(3), 872–879.

Allen, B. L., Pikul, J. L., Waddell, J. T., & Cochran, V. L. (2011). Long-term lentil green-manure replacement for fallow in the semiarid northern great plains. *Agronomy Journal, 103*(4), 1292–1298.

Altieri, M. A. (1995). *Agroecology: The science of sustainable agriculture* (2nd ed.). Westview Press.

Babu, P., Rao, C. P., & Veeraraghavaiah, R. (2014). Growth, yield and economics of rice fallow castor under different levels of *n* and *p* applied to preceding kharif rice and different fertilizer schedules given to succeeding castor crop. *Indian Journal of Agricultural Research, 48*(3), 217–221.

Bachinger, J., & Zander, P. (2007). ROTOR, a tool for generating and evaluating crop rotations for organic farming systems. *European Journal of Agronomy, 26*(2), 130–143.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., & Vance, P. H. (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research, 46*(3), 316–329.

Costa, A. M., Santos, L. M. R., Alem, D. J., & Santos, R. H. S. (2014). Sustainable vegetable crop supply problem with perishable stocks. *Annals of Operations Research, 219*(1), 265–283.

Detlefsen, N. K., & Jensen, A. L. (2007). Modelling optimal crop sequences using network flows. *Agricultural Systems, 94*(2), 566–572.

Dias, T., Dukes, A., & Antunes, P. M. (2014). Accounting for soil biotic effects on soil health and crop productivity in the design of crop rotations. *Journal of the Science of Food and Agriculture*.

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

Gabriel, J., & Quemada, M. (2011). Replacing bare fallow with cover crops in a maize cropping system: yield, *n* uptake and fertiliser fate. *European Journal of Agronomy, 34*(3), 133–143.

Haneveld, W. K., & Stegeman, A. W. (2005). Crop succession requirements in agricultural production planning. *European Journal of Operational Research, 166*(2), 406–429.

Hunt, J., & Kirkegaard, J. (2012). Re-evaluating the contribution of summer fallow rain to wheat yield in southern Australia. *Crop and Pasture Science, 62*(11), 915–929.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., & Vanderbeck, F. (2010). Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics, 36*, 695–702.

Kröbel, R., Lemke, R., Campbell, C. A., Zentner, R., McConkey, B., Steppuhn, H., et al. (2014). Water use efficiency of spring wheat in the semi-arid Canadian prairies: effect of legume green manure, type of spring wheat, and cropping frequency. *Canadian Journal of Soil Science, 94*(2), 223–235.

Linderoth, J. T., & Savelsbergh, M. W. (1999). A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing, 11*(2), 173–187.

Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research, 53*(6), 1007–1023.

Manyanga, M. A., Mafongoya, P. L., & Tauro, T. P. (2014). Soil macrofauna order diversity and abundance under improved fallows and organic matter transfer system in Zimbabwe. *African Journal of Ecology*.

Miller, P., Lighthiser, E., Jones, C., Holmes, J., Rick, T., & Wraith, J. (2011). Pea green manure management affects organic winter wheat yield and quality in semiarid Montana. *Canadian Journal of Plant Science, 91*(3), 497–508.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and combinatorial optimization*. Wiley-Interscience.

Santos, L. M. R. (2009). *Programação de rotação de culturas – modelos e métodos de solução*. Universidade de São Paulo (Ph.D. thesis).

Santos, L. M. R., Arenales, M. N., Costa, A. M., & Santos, R. H. S. (2010). A linear optimization approach for increasing sustainability in vegetable crop production. In H. A. Prado, A. J. B. Luiz, & H. Chaib Filho (Eds.), *Computational methods applied to agricultural research: Advances and applications* (pp. 234–265). Hershey: IGI Global.

Santos, L. M. R., Costa, A. M., Arenales, M. N., & Santos, R. H. S. (2010). Sustainable vegetable crop supply problem. *European Journal of Operational Research, 204*(3), 639–647.

Santos, L. M. R., Michelon, P., Arenales, M. N., & Santos, R. H. S. (2011). Crop rotation scheduling with adjacency constraints. *Annals of Operations Research, 190*(1), 165–180.

Vance, P. H. (1998). Branch-and-price algorithms for the one-dimensional cutting stock problem. *Computational Optimization and Applications, 9*(3), 211–228.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming, 86*(3), 565–594.

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

Vanderbeck, F. (2000b). On Dantzig–Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research, 48*(1), 111–128.

Vanderbeck, F., & Wolsey, L. A. (2010). Reformulation and decomposition of integer programs. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, & L. A. Wolsey (Eds.), *50 years of integer programming 1958–2008* (pp. 431–502). Berlin, Heidelberg: Springer.

Wezel, A., Casagrande, M., Celette, F., Vian, J.-F., Ferrer, A., & Peigné, J. (2013). Agroecological practices for sustainable agriculture: a review. *Agronomy for Sustainable Development*, 1–20.