Exact method approaches for the differential harvest problem

Gabriel Volte, Eric Bourreau, Rodolphe Giroudeau, Olivier Naud

Exact method approaches for the differential harvest problem*

Gabriel Volte[1], Eric Bourreau[1], Rodolphe Giroudeau[1], and Olivier Naud[2]

[1] LIRMM, University of Montpellier, CNRS, Montpellier, France
{volte,bourreau,giroudeau}@lirmm.fr
[2] ITAP, Irstea, Montpellier SupAgro, University of Montpellier, Montpellier, France
olivier.naud@irstea.com

Abstract. The trend towards a precise, numerical, and data-intensive agriculture brings forward the need to design and combine optimization techniques to obtain decision support methodologies that are efficient, interactive, robust and adaptable. In this paper, we consider the Differential Harvest Problem (DHP) in precision viticulture. To tackle this problem, we dedicated a specific column generation approach with enumeration techniques and a constraint programming model. Therefore, a set of simulated instances (which differ in field shape, zone shape, and size) was created to perform a parametric study on our different approaches. The specific column generation approach presented in this paper is preliminary work in the development path of more sophisticated resolution methods such as robust optimization and column generation/constraint programming hybridization.

Keywords: Column generation · Enumeration technique · Constraints programming · Exact method · Precision agriculture.

1 Problem description

The Differential Harvest Problem, introduced by [5], consists of optimizing harvests of different grape qualities in vineyards so that we obtain a certain quantity, denoted by *Rmin*, of good quality grapes. In the problem, there are only two types of grape quality: A-grapes and B-grapes (let us assume that A-grapes are of better quality than B-grapes).

Thanks to agronomic information obtained a priori, it is possible to map (see Figure 1) a vineyard by distinguishing areas according to the quality of the grapes. Meanwhile, geolocated harvesting machines equipped with two hoppers (harvest tanks with a maximum load of $CapaMax$) are able to use such a map. When one of the two hoppers is full, both must be emptied into a bin located at the edge of the plot. These machines have two harvesting modes, that can be changed only when they are emptying the hoppers at the bin, the selective

* Supported by organization French National Research Agency under the Investments for the Future Program, referred as ANR-16-CONV-0004.

mode, and the non-selective mode. The selective mode corresponds to sorting good grapes quality in one hopper and other grapes in the second hopper. When the *Rmin* quantity of A-grapes is harvested in the selective mode the machine can change to the non-selective mode where the loading capacity is perfectly handled (cf. Figure 2).

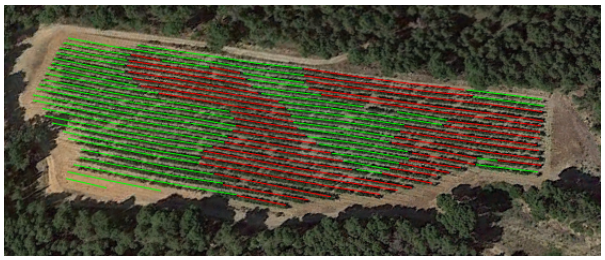

Fig. 1. Illustration of the agronomic map with different zones of grapes quality (real data from the Gruissan vineyard).

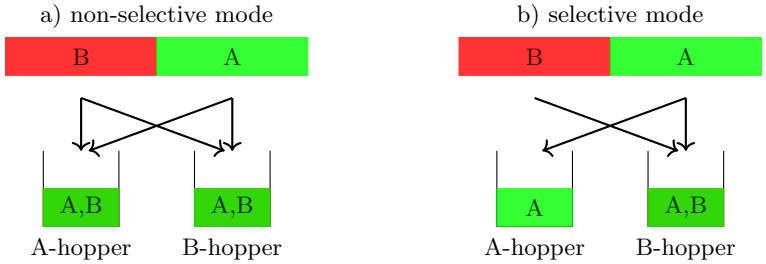

Fig. 2. Illustration of the different harvesting modes.

There is a technical issue with the harvesting machine which is related to the longitudinal size of its picking head of the harvester. This size is almost the length of wheelbase, approximately 5 meters. This problem occurs during zone changes and more particularly during the change from a zone with B-grapes to a zone with A-grapes, let us note this change BA (AB for the change from zone A to B).

When a BA transition occurs, three cases can be isolated (see Figure 3):

- case a), the machine head has just entered zone A. The harvesting hopper (which is hopper B) cannot be changed to hopper A because it is still harvesting B-grapes and therefore hopper A would be corrupted with B-grapes.
- In case b), the grape picker is overlapping both zones. The reasoning is the same as in case a) because we are still harvesting B-grapes.

– In case c), the harvester is completely in zone A, we no longer harvest B-grapes. The harvesting hopper can, therefore, be changed from hopper B to hopper A to harvest the A-grapes in hopper A.

When changing from AB, the harvesting hopper is changed from hopper A to hopper B as soon as the area transition has occurred, avoiding the corruption of A-grapes with B-grapes.

Any row r is composed with a succession of BA and AB transitions, this leads to these four following row representations (consecutive A or B can be merged):

– $r = (AB)^*$ and its symmetric $r = (BA)^*$. One can observe that in this row composition the difference between the number of BA transition when harvesting in the different directions is exactly one.
– $r = (AB)^*A$ and its symmetric $r = (BA)^*B$. In this configuration, the number of BA-transitions is the same in either direction.

Depending on the row configuration and the row direction the latency causes an asymmetry in the harvested quantities, therefore in the optimization part, we will need to determine the direction of each row to obtain the $Rmin$ quantity. For the sake of simplicity, the vineyard rows extremities are oriented upper to bottom, the upper extremity can be chosen arbitrarily.

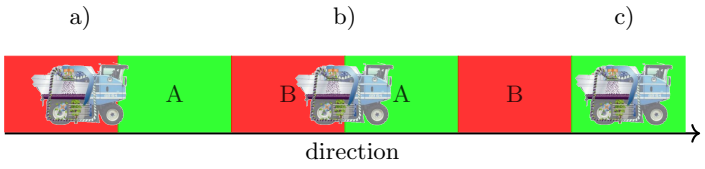

Fig. 3. Latency illustration.

The article is organized as follows: the first section 1 gives the context of this work and provides the problem definition. The second section 2 refers to the previous and related works on the problem. In section 3 we introduce a new graph model and decide to provide models for two exact methods; first a column generation approach (see subsection 3.1) then a constraint programming model (see subsection 3.2). In the section 4 we outline the results obtained with our different approaches.

2 Related Works

Precision agriculture is a principle of agricultural parcel management that aims to optimize yields and investments, by seeking to take better account of the variability of environments. In [18], the authors asserted that precision farming was

the future of crop nutrition offering benefits in crop quality, sustainability, food safety, etc. A few years later, [13] reviewed some precision agriculture advances and proposed new directions of research. [16] analyzed the adoption of precision agriculture technologies for several actors in the agricultural industry.

In the last decades, a lot of works was performed in digital and precision agriculture using operation research techniques involved in the resolution of vehicle routing problems, scheduling problems or stochastic problems.

The surveys [2] and [3] assert that most of the agricultural applications involve the motion of machines hence they first classify the agricultural field operations which can be modeled as a vehicle routing problem. They present an approach to represent the planning and scheduling of moving machines as a vehicle routing problem with time windows where the machine has to process deterministic, stochastic or dynamic requests.

VRP-specific optimization methods are addressed in [19] to solve their fleet routing problem, they have reduced the operating time by at most 17.3% using their approach rather than a human-made solution.

One can find some works on-field coverage problem [14,7], where the goal is to cover a field under technical constraints. The purpose of the work presented in [14] is to reduce the soil compaction. Smaller vehicles are used to cover the fields but those vehicles have smaller storage capacity. Thus the full field coverage in a single run is no longer possible and a path planning strategy is used to partially cover the field under *compacted area minimization constraints*. In comparison, the article [7] focuses both on finding minimal operating cost and on balancing the vehicle's workload.

The paper [17] explores the consequence of groundwater resource use under climate change scenarios. The problem is modeled as a dynamic stochastic problem with several temporal decision stages with multiple sources of risk that should impact farmer decisions.

Investment behavior under different policy and price scenarios was studied in [21]. They employ a dynamic multi-objective farm-household integer programming model on a northern Italy case study application highlighting the potentialities and the limits of the methodology applied.

A column generation approach is used to solve a crop rotation scheduling problem to produce a pre-determined demand for crops while respecting some ecological production constraints, this problem is called the sustainable vegetable crop demand-supply problem [10].

In few available literature, the DHP has been solved with methods based on Constraint Programming (CP), first a step model was proposed in [6] where they were able to optimally solve a 16 rows instance in 6 days, then a precedence model was introduced [5] which gives better results yet instances with 14 rows reach the 2 hours time limit, and Cost-Optimal Reachability Analysis [23] (CORA, which is a model-checking techniques branch) finding optimal solution on vineyards with up to 12 rows in few minutes.

3 Models

We propose a new graph model of the problem that we call the "flatten" representation of the vineyard (extremities are merged to consider only the rows) in opposition with the "physical" representation of the vineyard (rows are split into two extremities) used by [5] and [23]. Let n be the number of rows, $CapaMax$ be the hoppers capacity, $Rmin$ the minimum A-grapes quality needed to be harvested, b be the bin/depot, an undirected weighted complete graph $G = (V, E)$ with labeled edges, $V = \{0, \ldots, n\} \cup \{b\}$ and $E = \{(i, j, k) | i, j \in V^2,\ k \in \{0, 1\}\}$ (k refers to the row harvest direction see Figure 4). Taking the edge (i, j, k) means that the vehicle moves from row i to row j harvesting row i with direction k, if $k = 0$ the row is harvested from top to bottom, otherwise the row is harvested from bottom to top. Let say D is the weight of G computed from a distance matrix. It must be noted that because the vehicle has an important turning radius, the distance between rows are not exactly euclidean. qA/qB are the two harvesting functions: $(i, k) \mapsto \mathbb{N}, i \in V, k \in \{0, 1\}$.

The goal is to find a set of disjoint routes (except for the bin b), respecting the $CapaMax$ capacity, covering V and minimizing the harvesting time (here the distance) while harvesting at least $Rmin$ quantity of A-grapes.

This problem can be seen as a Heterogeneous Vehicle Routing Problem [20] with two resource constraints (the grapes quality, A and B) and two vehicle types (one selective and the other non-selective) with different capacities. If the vehicle is selective the harvested grapes are sorted according to their quality, thus the vehicle has two hoppers each with a capacity equal to $CapaMax$, otherwise both grapes quality are mixed in both hoppers, therefore, the vehicle hoppers can be merged into only one hopper with a capacity $2 * CapaMax$. Let us consider γ the number of selective vehicles and λ the total number of vehicles. The following bounds are easy to verify:

$$\left\lceil \frac{Rmin}{CapaMax} \right\rceil \leq \gamma \leq n \text{ and } \left\lceil \frac{Qtot}{2 * CapaMax} \right\rceil \leq \lambda \leq n$$

$Qtot$ refers to the total amount of grapes presents in the vineyard.

Previous works on the problem tackled it with constraint programming and model checking. We would like to investigate integer linear programming and more precisely column generation to observe the effectiveness of these techniques on the DHP. As early preliminary work, we tried to solve this problem with integer linear programming models and out-of-the-box solving methods. Because this was unsuccessful, we concluded that column generation techniques should be used here. The following subsection 3.1 explains the use of these techniques on the DHP. We identified that the principal weakness of the constraint programming approach used previously was to introduce a new global constraint that specifically fits the DHP. To avoid this, we decided to focus on designing constraint programming models and decision strategies that suit the most to our problem while using state-of-the-art generic constraints. We dedicate the subsection 3.2 to highlight our constraint programming approaches.

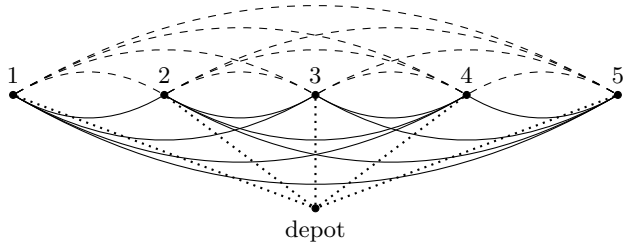

Fig. 4. The flatten graph representation of the vineyard, solid edges correspond to direction 0 and dashed edges refer to direction 1, dotted edges are direction free edges.

3.1 The column generation approach

We use Dantzig-Wolfe decomposition [9] on the DHP to obtain a set partitioning master problem with a side constraint which is the *Rmin* constraint and an elementary shortest path with resource constraints pricing problem.

Let us denote Ω the set of all feasible routes. The master problem selects routes in Ω to obtain at least the *Rmin* quantity. For the sake of memory usage, only solving the master problem, restricted to a subset R of Ω, is worth considering because the number of routes in Ω grows exponentially with the number of rows. Assume $R \subset \Omega$ is the set of feasible routes $R = (R^{\overline{s}} \cup R^s)$, with $R^{\overline{s}}$ be the set of non-selective routes and R^s be the set of selective routes. Selective routes must be generated independently of the non-selective routes, they use vehicle types with different capacities and resources consumption, therefore two pricing problems will be needed, one to generate selective routes and the other to generate non-selective routes.

Despite the classical branch-and-price scheme to VRP, we decided to opt for an enumeration technique [1]. There are several works on enumeration (see [22,15,1,8] for more details). The enumeration technique operates as follows. First, the relaxation of the restricted master problem is solved, like if we were solving the root node with branch-and-price, to obtain the linear relaxation optimal value z^*_{MP}. With the last solved master problem special dual costs are obtained, denoted by CR^*. Then any integer solution gives an upper bound z_{IP}. These bounds and the dual costs are provided to a specific pricing problem which will generate all columns with a reduced cost of $0 < z_{IP} - z^*_{MP} = \epsilon$. If we solve the new restricted master problem, this time without the linear relaxation, the solution obtained is optimal for the restricted master problem and thus for the master problem. The principal drawback of this technique is that too many columns may be generated if the linear relaxation of the restricted master problem is of poor quality, the gap with any integer solution would be too large.

Let us define the restricted master problem decision variables:

$$\forall r \in R^{\overline{s}},\ y^{\overline{s}}_r = \begin{cases} 1 \text{ if non-selective route } r \text{ is used} \\ 0 \text{ otherwise} \end{cases}$$

$$\forall r \in R^s,\ y_r^s = \begin{cases} 1 \text{ if selective route } r \text{ is used} \\ 0 \text{ otherwise} \end{cases}$$

Note $a_{ir} = 1$ if row i is harvested in the route r, 0 otherwise. Let q_r^t be the quantity of t-grapes collected in the route r and c_r the cost of the route r (total traveled distance computed from the weight matrix D). The route cost is the total traveled distance on the route. A non-selective route does not harvest A-grapes, all grapes are mixed in B-grapes.

Model M1: Restricted Master Problem

$$\text{Min} \sum_{r \in R} (y_r^{\overline{s}} + y_r^s) c_r \tag{1}$$

$$\sum_{r \in R} (y_r^{\overline{s}} + y_r^s) a_{ir} = 1 \qquad \forall i \in V \qquad [\pi_i] \tag{2}$$

$$\sum_{r \in R^s} q_r^A y_r^s \geq Rmin \qquad [\theta] \tag{3}$$

$$y_r^{\overline{s}} \in \{0,1\} \qquad \forall r \in R^{\overline{s}} \tag{4}$$

$$y_r^s \in \{0,1\} \qquad \forall r \in R^s \tag{5}$$

The aim is to minimize the cost of each route (1) while the entire field is harvested (2). The constraint (3) verifies that at least the $Rmin$ quantity of good quality grapes is harvested. And finally, the constraints (4) and (5) ensure the integrality of the y variables.

Pricing sub-Problem The pricing problem generates improving routes for the master problem based on the value of the constraints in the dual solution of the restricted master problem.

We solve the pricing problem, a shortest path problems with resource constraints, with dynamic programming using a labeling algorithm, introduced first by [12], enhanced by [11]. We adjusted this algorithm to our two pricing problems. Because extension and dominance rules are straightforward from the original, we decided not to reintroduce them in this paper. The only originality in our label algorithm is that two labels lists are used. One list, L^0, for labels with direction 0 and the other, L^1, for labels with direction 1. Therefore as the direction changes for every row in a route, the label obtained from the extension of the label L^0 in the labels list with direction 0 is added to the labels list with direction 1.

For each constraint (2), we obtain a dual cost π_i and each constraint (3) gives the θ dual cost. Using these dual costs we compute a reduced cost $\hat{c}_r$ for any

route r in the master problem. This reduced cost depends on whether the route is selective, thus let $\hat{c}_r^s = c_r - \sum_{i \in r} a_{ir}\pi_i + qA_r\theta$ be the reduced cost of the selective route r and $\hat{c}_r = c_r - \sum_{i \in r} a_{ir}\pi_i$ for the non-selective route. It also important when generating new selective routes to take the θ dual cost into account at each label extension to avoid interesting routes from being dominated:

$$\hat{c}_r^s = c_r - \sum_{i \in r} a_{ir}\pi_i + qA_r\theta = c_r - \sum_{i \in r} a_{ir}(\pi_i + qA_i\theta)$$

In the column generation part, routes with positive reduced cost can be discarded because they cannot belong to any optimal solution. Nevertheless, in the enumeration procedure, a new pricing problem is solved which brings out the routes with a reduced cost strictly less than ϵ.

3.2 Constraint Programming

We introduce in this section our constraint programming approach (CP). First, we describe a classical precedence model mainly used in VRP (partially used by [5], fully explained in [4]) enhanced by specific bin packing capacity constraints. Finally, a constructive search strategy is detailed.

Let V be the set of at most n vehicles($|V| = n$). For each vehicle, two dummy depots are created: an initial depot and a final depot. Note V^d (resp. V^f) the set of initial (resp. final) depots. The set of nodes N in the graph, such that $|N| = n + 2 * |V|$ (one node per row and two nodes per vehicle), is ordered as follows:

$$N = \{\overbrace{1, \ldots, n}^{rows}, \overbrace{n+1, \ldots, n+|V|}^{V^d}, \overbrace{n+|V|+1, \ldots, n+2*|V|}^{V^f}\}$$

Let us define the decision variables (see Figure 5 and Table 1, the distance matrix are displayed in the Figure 11 and Figure 12):

- $\forall i \in N, successor_i \in N$ (resp. $predecessor_i \in N$) variables giving the successor (resp. predecessor) of the row i in the route.
- $\forall i \in N, position_i \in N$, variables providing the position of the row i in the route, initial depots are in position 0.
- $\forall i \in N, assignment_i \in V$ variables indicating the vehicle assignment of the row i.
- $\forall i \in N, direction_i \in \{0, 1\}$, variables showing the direction of row i.
- $\forall v \in V, selective_v \in \{0, 1\}$, variables indicating whether a vehicle is selective.
- $\forall i \in N \forall u \in \{A, B\}, CapaSumq_i^u \in [0, CapaMax]$, variables measuring the cumulative sum of the harvested u-grapes quantity before harvesting row i.
- Obj is the objective variable which computes the total traveled distance.

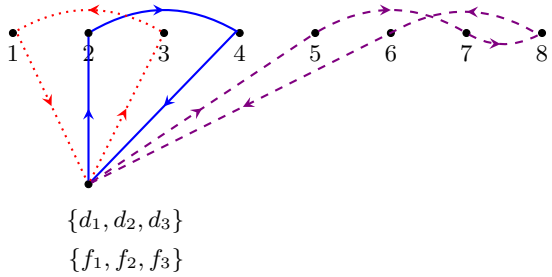

Fig. 5. Illustration of the optimal solution obtained with constraints programming for an instance with 8 rows. Plain arcs (resp. dotted and dashed arcs) representing vehicle one (resp. two and three), for the sake of clarity only the used vehicles are represented.

Node	1	2	3	4	5	6	7	8	d_1	d_2	d_3	f_1	f_2	f_3
qA	87	75	62	50	37	25	12	0	0	0	0	0	0	0
qB	13	25	38	50	63	75	88	100	0	0	0	0	0	0
successor	f_2	4	1	f_1	7	f_3	8	6	2	3	5	d_1	d_2	d_3
position	2	1	1	2	1	4	2	3	0	0	0	3	3	5
assignment	2	1	2	1	3	3	3	3	1	2	3	1	2	3
direction	0	1	1	0	1	0	0	1	0	0	0	1	1	1
selective	1	1	1	1	0	0	0	0	1	1	0	1	1	0
CapaSumqA	57	0	0	70	0	150	50	100	0	0	0	120	144	200
CapaSumqB	43	0	0	30	0	150	50	100	0	0	0	80	56	200

Table 1. Variables affectation for an eight rows instance, $CapaMax$ is set to 200 and $Rmin$ to 174, illustrated in Figure 5. The harvesting quantity qA/qB is displayed for the direction 1, but for the opposite direction, the harvesting quantity is computed by adding 5 (the latency) to qB and to subtract 5 to qA.

Model M2: Constraint Programming

$$\forall f_i \in V^f \quad successor_{f_i} = n + i \tag{6}$$

$$\forall d_i \in V^d \quad assignment_{d_i} = i \tag{7}$$

$$\forall f_i \in V^f \quad assignment_{f_i} = i \tag{8}$$

$$\forall i \in V^d \quad position_i = 0 \tag{9}$$

$$AllDifferent(successor_1, \ldots, successor_{|N|}) \tag{10}$$

$$\forall i \in N \quad direction_{successor_i} = 1 - direction_i \tag{11}$$

$$\forall i \in N \quad position_{successor_i} = 1 + position_i \tag{12}$$

$$\forall i \in N \quad assignment_{successor_i} = assignment_i \tag{13}$$

$$\sum_{i \in [1,n]} (qA_i * selective_{assignment_i}) \geq Rmin \tag{14}$$

$$\forall i \in N \quad CapaSumqA_{successor_i} = CapaSumqA_i + Q_i^A \tag{15}$$

$$\forall i \in N \quad CapaSumqB_{successor_i} = CapaSumqB_i + Q_i^B \tag{16}$$

$$\forall d_i \in V^d \quad s_{d_i} < s_{d_{i+1}} \tag{17}$$

$$\forall i \in N \quad predecessor_k = i \Leftrightarrow successor_i = k \tag{18}$$

$$\forall u \in \{A, B\} \quad diffN(assignment, CapaSumq^u, (1, \ldots, 1)^T, Q^u) \tag{19}$$

$$Obj = \sum_{i \in N} D_{i, successor_i, direction_i} \tag{20}$$

$$\min Obj \tag{21}$$

Constraints (6), (7), (8) and (9) assign the successor of a final depot to the corresponding initial depot, the same vehicle is affected to initial and final depot and initial depots are in position 0. The $AllDifferent$ constraint (10) ensures that routes are disjointed and that they cover all the rows, every successor is distinct implies that a row is taken exactly once (there are n values for n variables). The direction, position and assignment variables are updated depending on their successors (11),(12) and (13). Constraint (14) ensures that the $Rmin$ A-grapes quantity is collected. The cumulative amount of already harvested A-grapes (resp. B-grapes) is computed with (15) (resp. (16)), these constraints ensure that no vehicle exceeds its hoppers loading due to the maximum value of

the *CapaSum* domain. The constraints (20) and (21) minimize the total traveled distance.

We added redundant constraints to improve the performance of this model: first, symmetry breaking constraints (17) between vehicles are used then a channeling constraint (18) between *successor* and *predecessor* variables. The *diffN* constraints (19) are also redundant and handle the vehicle capacity taking account of the vehicle filling and the object positioning in the vehicle, this positioning depends on both the X and Y axis (see Figure 6). We create two objects per row: one indicates the A-grape quantity harvested in the rows and the other indicates the B-grapes quantity harvested. Objects are affected by vehicle assignment (X axis) and ordered depending on their position in the route with the $CapaSumq^u$ variables $\forall u \in \{A, B\}$ (Y axis) which link row with its direct successor. The height of a task, Q_i^u $\forall u \in \{A, B\}$, depends on the type of the node $i \in N$: $q^u_{i,selective_i,direction_i}$ for rows, 0 for initial depots and $CapaMax - CapaSum^u_{v_f}$ for final depots. The height of the final depots thereby defined is used to make the *diffN* constraint even more compact, all vehicles will have their maximum capacity reached due to the height of the final depots which fill the remaining space (see Figure 6).

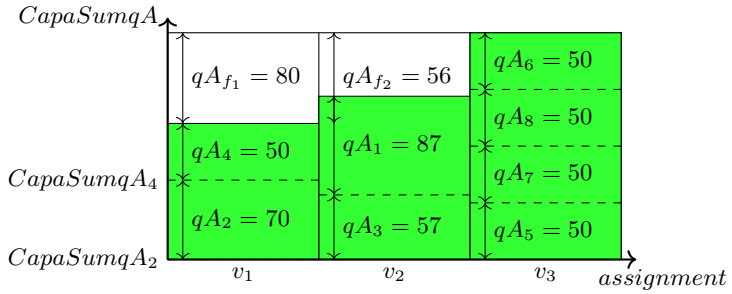

Fig. 6. Representation of the *diffN* constraint for the A-grapes for the instance shown in the Figure 5.

The Snake constructive search strategy One of the advantages of constraint programming is the possibility to choose the search strategy in the decision tree. The strategy is based on two axes: variable choice and value choice. The variable choice is a determining factor in a good strategy. Indeed, if the first variables used have a large domain or are weakly constrained then the decision tree is larger.

With this in mind, we wanted to create a strategy that mimics a greedy method, which we call: the Snake. The Snake strategy builds a route and then constructs entirely the next one and so on. The purpose is to find a good solution in a few decision nodes. Other strategies (as DomOverWDeg) generally build

route fragments, and, at the bottom of the decision tree, try to assemble these fragments to obtain a feasible route. This assembly is very combinatorial due to the many possible arrangements of the fragments, see Figure 7.

The Snake strategy works as follows: the first variable is the first available initial depot. The choice of the new variable is determined by the choice of the value of the previously instantiated variable. The first available value for the variable is selected, the values are increasingly sorted (we want to minimize the total distance). Once a final depot is instantiated, the next variable selected is the next initial depot. The variables that calculate the distance to the successor are used. Besides, these variables are sorted so that optimization begins with selective routes.

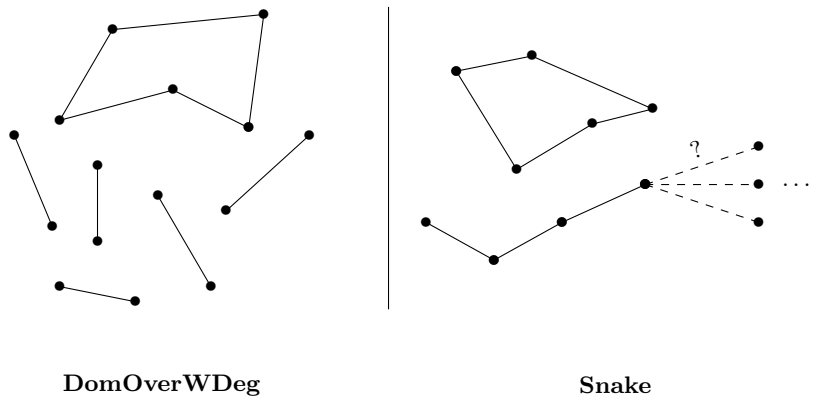

Fig. 7. The Snake strategy vs the DomOverWDeg strategy illustration.

4 Experimental results

In this section, we present the experimental results, first, we introduce the data sets used for the experimental study, then we outline the results obtained with our models. For the parametric study, we were focusing on exact methods and thus on finding optimal solutions.

4.1 Description of data sets

We have tested our models on different instances, some were artificially generated based on real data observed in a vineyard, at INRA Pech-Rouge (Gruissan), located in the southern France (cf. Figure 1).

As [5,23] we decompose this instance to create a set of smaller instances, denoted by G-instances. We create instances with 10 consecutive rows and up

to 24 rows (the original instances), with a hopper capacity of 900 and 1800 and the $Rmin$ value takes value in $\{0, 0.25, 0.5, 0.7, 0.9, 1\}$ of the total A-grapes in the vineyard.

Artificial instances were generated based on two vineyard shapes, the square instances (S-instances) and the triangle instances (T-instances) see Figure 10, that seems to be theoretically interesting. We assume that the triangle instances must be the worst-case instances because everything is symmetric. The square instances were determined to see the influence of the increase of the $Rmin$ value on the objective function.

The same construction as the original instance is used to construct artificial instances. The Table 2 summarizes the parameters variation of each instance type.

	#rows	$CapaMax$	$Rmin$	Total
G-instances	$\{10, 11, \ldots, 24\}$	$\{900, 1800\}$	$\{0, 0.25, 0.5, 0.7, 0.9, 1\}$	1140
T-instances	$\{10, 11, \ldots, 24\}$	$\{100, 200, \ldots, 500\}$	$\{0, 0.25, 0.5, 0.7, 0.9, 1\}$	540
S-instances	$\{10, 11, \ldots, 24\}$	$\{100, 200, \ldots, 500\}$	$\{0, 0.25, 0.5, 0.7, 0.9, 1\}$	540

Table 2. Parameters variation for each data set.

4.2 Results

In this section, we will detail the results obtained for our approaches. The results were performed on an Optiplex 5055 computer with an AMD RYZEN 7 Pro 1700 (8 cores/4 Mo/16 Threads/3 GHz) processor. For the column generation approach, we use CPLEX 12.8.0.0 and for the constraint programming, we use Choco 4.10.1.

Results for the column generation methods The time limit for all the instance was fixed to 10 minutes, except for the original instance with 24 rows where the time limit was 1 hour, due to the number of instances that has to be solved.

To the best of our knowledge, we are the first who were able to solve the real instance with 24 rows, see Figure 1, in 2210 seconds and almost all the computation time is used to generated routes and to find an integer solution, the enumeration technique took less than a few minutes and generates only 18 columns.

In the Table 5, a comparison is made between the results obtained with the variation of the $CapaMax$ parameter for the S and T instances and the same results for the G instances are shown in the Table 3. The results displayed are the average computation time and the average integrality gap of the $Rmin$ values for every row's number and $CapaMax$ value. When the $CapaMax$ parameter increases there is a huge increase in the computation time since the pricing

problem is more difficult to solve (the number of non-dominated routes generated is larger). For the G instances, the gap is really small (less than 0.002 %) however almost all the large instances reach the time limit.

	$CapaMax$			
	900		1800	
#rows	CPU(s)	gap(%)	CPU(s)	gap(%)
10	1.87	~ 0	10.71	~ 0
11	5.98	~ 0	38.43	0.016
12	**6.35**	~ 0	**81.78**	~ 0
13	**118.84**	~ 0	**344.90**	0.01
14	**6.46**	~ 0	**217.08**	0
15	85.04	~ 0	462.85	0.02
16	11.15	~ 0	429.40	~ 0
17	227.66	~ 0	538.73	0.02
18	19.27	~ 0	536.99	0.0003
19	313.19	~ 0	535.27	~ 0
20	**97.11**	~ 0	**534.45**	~ 0
21	331.28	~ 0	528.05	0.03
22	**165.95**	~ 0	**539.18**	0.04
23	545.24	~ 0	530.19	0.08

Table 3. Results for the G instances with the variation of the $CapaMax$ parameter.

The Figure 8 and Figure 9 indicate the results obtained for the G instances according to the *Rmin* parameter variation and each point represents the average value of the $CapaMax$ values. One can see that the objective value and the computation time appear not to increase from 0% to 90% *Rmin* but when demanding to harvest all the A-grapes the objective value grows due to the latency, all the rows have to be harvested in the direction that maximizes the A-grapes harvested, yet the computation time is low even for the biggest instances.

Results for the constraint programming approaches Precedent works using constraint programming to solve the DHP were able to solve optimally a 16 rows instances in 6 days, then optimally solve 10 and 12 rows instances with a 2 hours time limit. We propose to compare our approach to its ability to obtain a good feasible solution in a short time.

With this in mind, we only show the results for a few instances (one with 10 rows, 13 rows, and 16 rows), with a 10 minutes time limit. The results for the constraint programming approach are summarized in the Table 4, we are looking for statistics for the first, the best solution (whether the optimal solution is found according to the column generation solution) and for optimality proof. For the 10 rows instance, with the *DomOverWDeg* strategy a poor quality first solution has been found. Moreover, the last solution is not improving so much the first solution (4379 to 4358). Meanwhile, the Snake strategy gives a good first solution in a few decision nodes (27 nodes). This solution is improved to optimality in under 2 minutes. The *diffN* constraint adds some reasoning to the model because

the nodes number decreases but the total time to compute an optimal solution is higher.

For the instance with 13 rows, the analysis is even worse for the *DomOverWDeg* strategy cannot find any solution within the 10 minutes time limit even with the *diffN* constraint. Nevertheless with the *Snake* strategy a solution is found in less than 1 second but at the end of the time limit, no optimal solution has been proved. For the 16 rows instance, the Snake strategy also finds a solution in less than 1 second but is not able to improve it to optimality in less than 10 minutes.

These results are interesting for a hybridization approach because the feasible solution found by the constraint programming in less than a few minutes can be used in the enumeration part for the column generation approach.

		10 rows				13 rows				16 rows			
		V1	V2	V3	V4	V1	V2	V3	V4	V1	V2	V3	V4
DOWDeg		✓	✓			✓	✓		✓	✓			
diffN			✓		✓		✓		✓		✓		✓
Snake				✓	✓			✓	✓		✓	✓	
First Sol	obj	4379	4370	4222	4222	∅	∅	5241	5241	∅	∅	6603	6603
	Nodes	1669	1306	27	27	\	\	31	31	\	\	39	39
	CPU(s)	3	2	0	0	\	\	0	0	\	\	0	0
Best Sol	obj	∅	∅	4066	4066	∅	∅	**5234**	**5234**	∅	∅	∅	∅
	Nodes	\	\	91k	81k	\	\	466k	452k	\	\	\	\
	CPU(s)	\	\	26	36	\	\	159	231	\	\	\	\
	#sol	\	\	10	10	\	\	6	6	\	\	\	\
Proof	obj	4358	4 206	**4066**	**4066**	∅	∅	**5234**	**5234**	∅	∅	6447	6447
	Nodes	547k	525k	398k	339k	492k	433k	1767k	1171k	399k	308k	2033k	1281k
	CPU(s)	600	600	115	148	600	600	600	600	600	600	600	600

Table 4. Constraint programming results for the instance with 10, 13 and 16 rows with a 600 seconds time limit.

5 Conclusion

In this paper, we have proposed both a column generation, with enumeration technique, approach based on a new graph representation of the DHP and a constraint programming model using global constraints. We have performed a parametric study on various instance sets, some based on real data and other purposely generated. The main difficulty is that computing the improving routes in the pricing problem may be awful with the growth of the $CapaMax$ parameter. The small integrality gap obtained gives us good hope to solve larger instances with a bigger time limit, and validate the efficiency of the column generation approach to tackle the DHP, the real instance with 24 rows was optimally solved. The constraint programming results are promising for a hybridization approach, computing all the ϵ-improving routes with the constraint programming integer solution (found in a few minutes) for the enumeration part.

Acknowledgements

This work was supported by the French National Research Agency under the Investments for the Future Program, referred as ANR-16-CONV-0004

Appendix

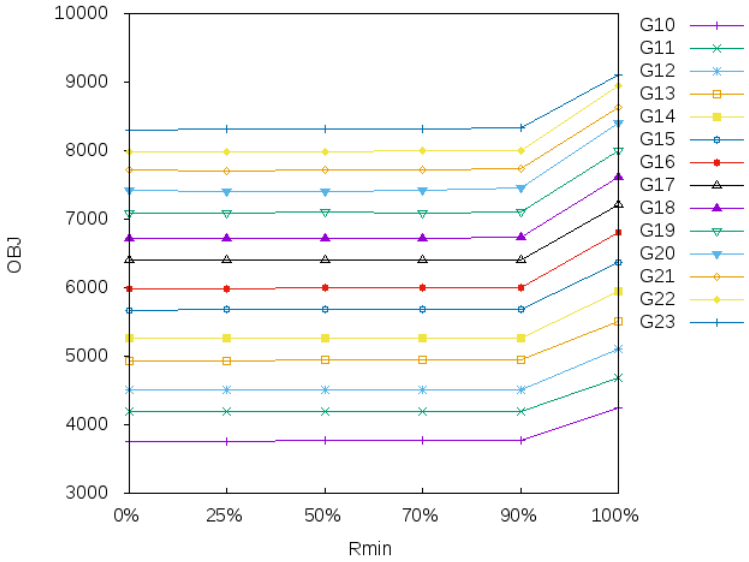

Fig. 8. The evolution of the objective function value when the *Rmin* parameters increase for the G instances.

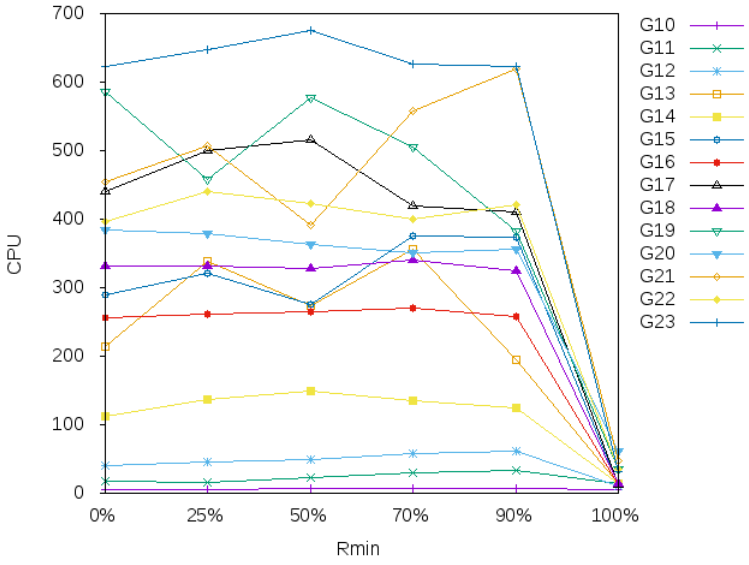

Fig. 9. The evolution of the computation time when the *Rmin* parameters increase for the G instances.

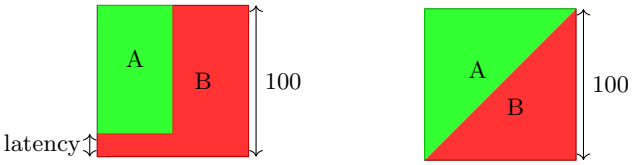

Fig. 10. Vineyard shape representation of the square instances (left side) and the triangle instances (right side).

		T		S				T		S	
#rows	$CapaMax$	CPU(s)	gap(%)	CPU(s)	gap(%)	#rows	$CapaMax$	CPU(s)	gap(%)	CPU(s)	gap(%)
	100	0.29	4.25	0.2	3.37		100	0.82	0.13	47.66	3.70
	200	0.49	0.11	0.94	0.41		200	16.52	4.55	216.75	5.25
10	300	2.36	0.04	2.64	0.64	17	300	529.01	5.12	603.39	5.72
	400	6.52	0.41	9.05	0.87		400	521.21	1.77	525.95	2.48
	500	9.40	1.10	16.40	1.67		500	542.55	1.80	624.74	4.15
	100	1.32	4.95	0.15	2.26		100	0.71	0.99	2.78	3.94
	200	14.96	7.03	0.91	8.16		200	6.55	0.22	7.11	0.54
11	300	48.99	7.28	79.15	8.45	18	300	441.58	0.52	512.01	1.04
	400	87.25	7.30	52.64	8.24		400	518.07	1.59	549.86	0.27
	500	59.52	7.53	98.07	8.35		500	559.98	2.15	620.04	3.87
	100	0.23	1.22	0.24	**2.56**		100	0.55	1.67	0.08	0.64
	200	0.90	0.58	0.93	**0.50**		200	310.40	3.79	311.51	4.63
12	300	8.96	0.38	10.50	0.27	19	300	521.54	4.20	522.08	4.93
	400	15.42	0.68	20.34	0.58		400	517.21	4.35	626.03	7.15
	500	42.56	0.50	26.88	1.12		500	512.78	2.23	613.55	3.85
	100	1.35	3.46	2.34	**3.32**		100	0.047	0	106.80	1.91
	200	**20.37**	6.07	159.77	**7.017**		200	10.36	0.26	16.08	0.31
13	300	**318.57**	5.95	419.87	7.16	20	300	500.71	0.53	513.86	0.73
	400	523.88	6.56	518.98	7.51		400	504.41	0.76	620.26	2.10
	500	455.87	6.83	267.43	7.39		500	512.33	2.53	607.29	3.03
	100	0.52	1.93	0.33	2.73		100	0.14	1.67	0.08	0.53
	200	**2.87**	0.82	3.09	0.48		200	415.4	3.71	515.31	4.24
14	300	**21.14**	1.07	25.82	0.75	21	300	520.53	3.83	623.61	4.34
	400	125.38	0.44	92.45	0.60		400	506.63	3.95	608.38	4.78
	500	249.35	0.65	245.66	0.39		500	519.92	2.30	607.89	3.07
	100	**1.97**	2.19	**0.591**	2.04		100	1.08	2.04	0.18	1.09
	200	**307.76**	4.83	**454.34**	5.88		200	48.26	0.87	38.33	0.58
15	300	428.17	5.32	524.62	6.29	22	300	515.78	2.96	522.21	1.34
	400	526.90	5.54	545.35	6.36		400	519.62	0.70	626.18	2.26
	500	515.87	3.29	530.25	6.18		500	549.81	2.48	640.07	2.57
	100	0.36	1.10	0.21	0.36		100	0.22	0.98	0.10	1.34
	200	3.65	1.34	7.00	0.77		200	418.62	3.47	418.97	3.78
16	300	156.90	0.24	136.39	1.07	23	300	515.7	2.51	542.45	4.02
	400	526.57	0.44	513.76	0.26		400	524.80	2.57	621.25	4.65
	500	517.73	0.99	550.97	2.74		500	521.29	2.32	621.82	2.63

Table 5. Results for T and S instance type with the variation of the $CapaMax$ parameter.

$$\begin{pmatrix} 0 & 5 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\ 5 & 0 & 5 & 4 & 6 & 8 & 10 & 12 & 14 \\ 4 & 5 & 0 & 5 & 4 & 6 & 8 & 10 & 12 \\ 6 & 4 & 5 & 0 & 5 & 4 & 6 & 8 & 10 \\ 8 & 6 & 4 & 5 & 0 & 5 & 4 & 6 & 8 \\ 10 & 8 & 6 & 4 & 5 & 0 & 5 & 4 & 6 \\ 12 & 10 & 8 & 6 & 4 & 5 & 0 & 5 & 4 \\ 14 & 12 & 10 & 8 & 6 & 4 & 5 & 0 & 5 \\ 16 & 14 & 12 & 10 & 8 & 6 & 4 & 5 & 0 \end{pmatrix}$$

Fig. 11. Distance matrix between rows and the depot with direction 0, for the example instance Figure 5.

$$\begin{pmatrix} 0 & 105 & 104 & 106 & 108 & 110 & 112 & 114 & 116 \\ 105 & 0 & 5 & 4 & 6 & 8 & 10 & 12 & 14 \\ 104 & 5 & 0 & 5 & 4 & 6 & 8 & 10 & 12 \\ 106 & 4 & 5 & 0 & 5 & 4 & 6 & 8 & 10 \\ 108 & 6 & 4 & 5 & 0 & 5 & 4 & 6 & 8 \\ 110 & 8 & 6 & 4 & 5 & 0 & 5 & 4 & 6 \\ 112 & 10 & 8 & 6 & 4 & 5 & 0 & 5 & 4 \\ 114 & 12 & 10 & 8 & 6 & 4 & 5 & 0 & 5 \\ 116 & 14 & 12 & 10 & 8 & 6 & 4 & 5 & 0 \end{pmatrix}$$

Fig. 12. Distance matrix between rows and the depot with direction 1, for the example instance Figure 5.

References

1. R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. Mathematical Programming, 115(2):351–385, 2008.
2. D.D. Bochtis and C. G Sørensen. The vehicle routing problem in field logistics part i. Biosystems engineering, 104(4):447–457, 2009.
3. D.D. Bochtis and C.G. Sørensen. The vehicle routing problem in field logistics: Part ii. Biosystems engineering, 105(2):180–188, 2010.
4. E. Bourreau, M. Gondran, P. Lacomme, and M. Vinot. De la Programmation Linéaire à la Programmation Par Contraintes. 2019.
5. N. Briot, C. Bessiere, and P. Vismara. A constraint-based approach to the differential harvest problem. In Gilles Pesant, editor, Principles and Practice of Constraint Programming, pages 541–556, Cham, 2015. Springer International Publishing.
6. Nicolas Briot, Christian Bessiere, Bruno Tisseyre, and Philippe Vismara. Integration of operational constraints to optimize differential harvest in viticulture. In Precision agriculture'15, pages 111–129. Wageningen Academic Publishers, 2015.
7. M. Burger, M. Huiskamp, and T. Keviczky. Complete field coverage as a multi-vehicle routing problem. IFAC Proceedings Volumes, 46(18):97–102, 2013.
8. C. Contardo and R. Martinelli. A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. Discrete Optimization, 12:129–146, 2014.
9. George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. Operations research, 8(1):101–111, 1960.
10. L. M. R dos Santos, A. M Costa, M. N Arenales, and R. H. S Santos. Sustainable vegetable crop supply problem. European Journal of Operational Research, 204(3):639–647, 2010.
11. D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks, 44(3):216–229, 2004.
12. Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Column generation, pages 33–65. Springer, 2005.
13. A. McBratney, B. Whelan, T. Ancev, and J. Bouma. Future directions of precision agriculture. Precision agriculture, 6(1):7–23, 2005.
14. M. Plessen. Partial field coverage based on two path planning patterns. Biosystems engineering, 171:16–29, 2018.
15. Julie Poullet and Axel Parmentier. Ground staff shift planning under delay uncertainty at air france. arXiv preprint arXiv:1811.00171, 2018.
16. M. Reichardt and Carste. Jürgens. Adoption and future perspective of precision farming in germany: results of several surveys among different agricultural target groups. Precision Agriculture, 10(1):73–94, 2009.
17. M. Robert, J.-E. Bergez, and A. Thomas. A stochastic dynamic programming approach to analyze adaptation to climate change–application to groundwater irrigation in india. European Journal of Operational Research, 265(3):1033–1045, 2018.
18. P. C. Robert. Precision agriculture: a challenge for crop nutrition management. In Progress in Plant Nutrition: Plenary Lectures of the XIV International Plant Nutrition Colloquium, pages 143–149. Springer, 2002.
19. Hasan Seyyedhasani and Joseph S Dvorak. Reducing field work time using fleet routing optimization. Biosystems engineering, 169:1–10, 2018.

20. Paolo Toth and Daniele Vigo. The vehicle routing problem. SIAM, 2002.
21. D. Viaggi, M. Raggi, and S. y Paloma. An integer programming dynamic farm-household model to evaluate the impact of agricultural policy reforms on farm investment behaviour. European Journal of Operational Research, 207(2):1130–1139, 2010.
22. Laurence A Wolsey and George L Nemhauser. Integer and combinatorial optimization. John Wiley & Sons, 2014.
23. Rim Saddem Yagoubi, Olivier Naud, Karen Godary Dejean, and Didier Crestani. New approach for differential harvest problem: The model checking way. IFAC-PapersOnLine, 51(7):57–63, 2018.