A Constraint-based Approach to the Differential Harvest Problem

Nicolas Briot[1], Christian Bessiere[1], and Philippe Vismara[1,2]

[1] LIRMM, CNRS–University Montpellier, France
[2] MISTEA, Montpellier SupAgro, INRA, 2 place Viala, 34060 Montpellier, France
{briot,bessiere,vismara}@lirmm.fr

Abstract. In this paper, we study the problem of differential harvest in precision viticulture. Some recent prototypes of grape harvesting machines are supplied with two hoppers and are able to sort two types of grape quality. Given estimated qualities and quantities on the different areas of the vineyard, the problem is to optimize the routing of the grape harvester under several constraints. The main constraints are the amount of first quality grapes to harvest and the capacity of the hoppers. We model the differential harvest problem as a constraint optimization problem. We present preliminary results on real data. We also compare our constraint model to an integer linear programming approach and discuss expressiveness and efficiency.

1 Introduction

In precision viticulture, many studies have proposed to define field quality zones [14]. They demonstrated the technical and economic value of a differential harvesting of these different zones. This interest justifies the recent development of prototypes of conventional grape harvesting machines able to sort two types of harvest quality, such as the EnoControlTM system prototype (*newHolland Agriculture*, PA, USA). These grape harvesting machines have two tanks, called hoppers, able to differentiate two types of grape quality, named A and B, according to the harvested zone.

Optimizing harvest consists on minimizing the working time of grape harvester. This time corresponds to travel time and emptying time of the machine. Ideally, this goal requires that both hoppers of the machine are full at each emptying. In the case of selective harvesting, the simultaneous filling of the two hoppers is combinatorial and complex. Indeed, the hopper that contains A grapes can fill up at a speed different from the hopper that contains B grapes, depending on the harvested zone. Other issues have to be considered. Top quality grapes should not be altered (mixed with lower quality grapes) when the harvester moves from one quality zone to another. Turning radius of the machine must also be taken into account.

This problem, called *Differential Harvest Problem*, has not been studied in the literature except in a preliminary study we published in [4]. A comparison with routing problems can be established. For instance, Bochtis *et al.* [1, 2] show

various formulations for many related problems in agriculture, such as precision spraying, fertilizing in arable farming, etc. In [9], Kilby and Shaw give a Constraint Programming (CP) formulation for the *Vehicle Routing Problem* (VRP) and detail other exact methods and heuristics to solve the problem. *Balancing Bike Sharing System* (BBSS) is an example of real application solved with VRP formulation and CP approach. Di Gaspero *et al.* [7] show two novel CP models for BBSS and a Large Neighborhood Search approach is proposed.

In this paper, we explore how to solve the Differential Harvest Problem with a CP approach. We consider two models: a Step model we introduced in our previous work [4] and a new routing model that we call the Precedence model. We also compare these two models to an Integer Linear Programming approach.

The paper is organized as follows. Section 2 describes the problem of differential harvest in viticulture. Section 3 proposes the two CP models. We present preliminary results on real data in Section 4. We compare our CP model to an Integer Linear Programming approach and discuss expressiveness and efficiency in Section 5.

2 The Differential Harvest Problem

Precision viticulture is a new farming management approach that focuses on the intra-field variability in a vineyard. Thanks to high resolution satellite images and infield sensors, it is possible to predict the variability of quality and yield within a vine field. The Differential Harvest Problem (DHP) is defined on a vine field with two qualities of grapes. The first one will be denoted A grapes and corresponds to high quality grapes. These grapes will be used to produce high quality wine. The rest of the harvest will be put in the B-grapes class. Note that high quality grapes (A grapes) can be downgraded to the low quality B. But the converse is not true.

We consider a harvesting machine with two hoppers. We denote by $Cmax$ the maximum capacity of a hopper. With such a machine, two categories of grapes can be separated. Generally, one hopper (called A hopper) receives only A grapes whereas the other one (called $B\&A$ hopper) can contain either B grapes only or both A grapes and B grapes. It is important not to contaminate the grapes in A hopper with B grapes. When the hoppers are full, the machine can dump them into a bin located around the plot before continuing its work. Generally, the bin is immediately brought to the winery after each dump of the hoppers in order to preserve grapes quality. A second bin waits in stand-by to avoid delaying the harvester.

The harvesting machine takes some time between picking grapes and putting them into hoppers. This time ($\approx$ 10 seconds), required to empty the conveyor belt, is called *latency*. When the harvesting machine passes over two zones, the quality of grapes harvested is not guaranteed during this time because latency may lead to mixing both qualities of grapes. Hence, when the machine leaves a zone of A grapes to enter a zone of B grapes or leaves a zone of B grapes to enter a zone of A grapes, the grapes must be considered B grapes. For the same

row, the type of transitions may vary according to the direction that the row is harvested. Because of latency, the quantity of A grapes (resp. B grapes) that are collected within a given row can change with the direction. For instance, consider a row where the sequence of quality areas is A-B-A-B (see Figure 1). If the machine harvests the row from left to right, there is only one B-A transition. If the machine harvests the row in the opposite direction, two transitions B-A appear. Thus, quantities of A grapes and B grapes harvested will necessarily depend on the direction in which the rows are harvested.

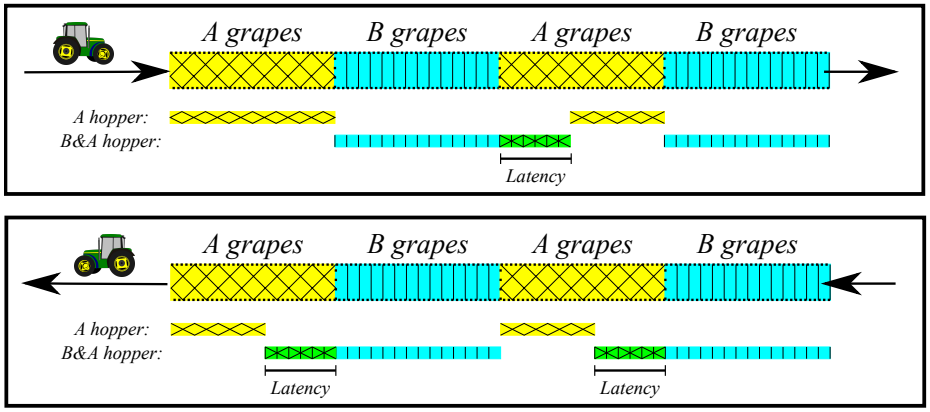


Fig. 1. Quantities of grapes depend on the direction of row harvest.

Consider *Rmin*, the desired minimum volume of A grapes harvested in the vine field. *Rmin* is a threshold. In practice, *Rmin* corresponds to shortfall of a volume of A grapes according to the objectives of the winery. If the vineyard contains more than *Rmin* A grapes, the excess amount can be downgraded to B-grapes quality. As long as *Rmin* has not been reached, A grapes are stored into the A hopper. Once *Rmin* has been reached and after the hoppers have been emptied, A grapes and B grapes can be mixed in both hoppers. Regarding *Rmin*, there are three possibilities to fill the two hoppers. When the harvesting machine must differentiate qualities in hoppers and the A hopper is not full, A grapes are put in the A hopper and B grapes in the $B\&A$ hopper (see Figure 2.a). When A hopper is full, the machine can put A grapes in the $B\&A$ hopper (see Figure 2.b). In such a case, these A grapes are downgraded. Once *Rmin* has been reached, the machine can mix grapes in both hoppers (see Figure 2.c).

The vine field, composed of n rows, is modelled by differentiating the two extremities of each row. A row $r \in \{0, \ldots, n-1\}$ is represented by extremities $2r$ and $2r+1$. For each row, we denote $Q^A_{2r\to 2r+1}$ and $Q^B_{2r\to 2r+1}$ (resp. $Q^A_{2r+1\to 2r}$ and $Q^B_{2r+1\to 2r}$) the quantities of A grapes and B grapes that will be collected

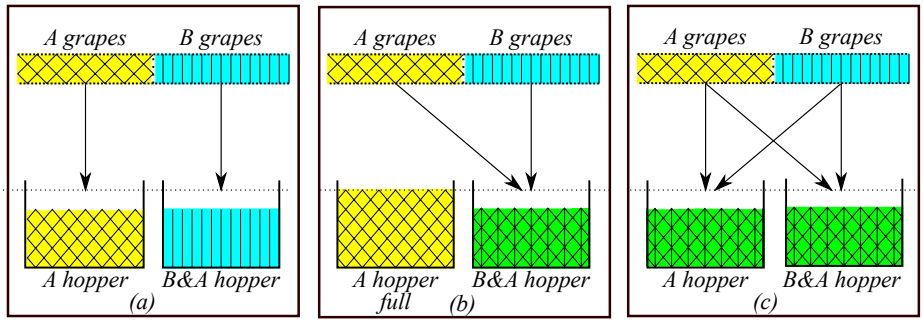


Fig. 2. Three possibilities to fill the hoppers. (a) A grapes and B grapes are separated. (b) When A hopper is full, A grapes and B grapes are mixed in the $B\&A$ hopper. (c) Once $Rmin$ has been reached, A grapes and B grapes are mixed in the two hoppers.

in row r with orientation $2r \rightarrow 2r+1$ (resp. $2r+1 \rightarrow 2r$). These quantities are computed according to the latency.

Another important information is the cost of the path between two extremities of different rows and between row extremities and the bin place. We denote $d(p,q) = d(q,p)\ \forall p,q \in \{0,1,\ldots,2n-1\} \cup \{2n\}$ the time required to go from an extremity p to an extremity q (where $2n$ denotes the bin place). This cost depends on the distances between extremities and the turning radius of the harvesting machine.

We are ready to define the problem.

Definition 1 (Differential Harvest Problem). *Given a vine field described by a cost path matrix between row extremities (or the bin place) and an estimation of the quantity of A grapes and B grapes on each row according to the direction, given a harvesting machine with a hopper capacity of Cmax and a latency delay, given a threshold of Rmin A grapes to harvest, the Differential Harvest Problem consists in finding a sequence of extremities (that is, an order and orientation of the rows) that minimizes the time required to harvest the vine field and ensures at least Rmin A grapes harvested.*

3 Constraint Programming for the DHP

The problem of differential harvest in precision viticulture is a recent problem. Its specification is not yet stable as it corresponds to a simplification of the real agronomic problem. In this context, other constraints will certainly appear when the first solutions will be applied. An example of probable extra constraint is shown in section 5.2. Constraint programming is well recognized for its flexibility and ease of maintenance. For these reasons, using constraint programming to model the DHP seems to be a good approach.

This section is devoted to present two CP models. First, the *Step model* consists in choosing which row to harvest step by step. Second, the *Precedence model* is devoted to find the best predecessor of each row.

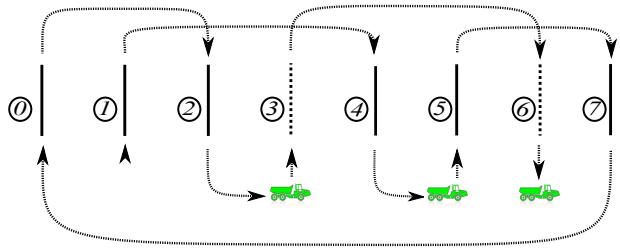
Given a set of variables and a set of possible values for each variable, given a set of constraints, and given an optimization criterion, constraint optimization consists in finding an instantiation of variables that satisfies all constraints of the problem and minimizes the criterion.

3.1 The Step model

In [4], we presented a preliminary CP model to solve the Differential Harvest problem. This model was dedicated to measure the gain of an optimized routing compared to the traditional approach. The traditional routing consists in systematically taking the second next row on the same side,[3] until the capacity of one hopper is reached. We showed in [4] that using a CP model to optimize the routing can reduce the harvest time of almost 40% compared to traditional harvest routing.

The *Step model* presented in [4] is illustrated in Figure 3. This model is based on three main kinds of variables that describe:

- the number of the row visited at each step of the route,
- the orientation of the visited row (harvest direction),
- the act of emptying the hoppers into the bin at the end of the visited row or not.



Step	0	1	2	3	4	5	6	7
Row	1	4	5	7	0	2	3	6
Orientation	0	1	0	1	0	1	0	1
Emptying into the bin?	0	1	0	0	0	1	0	1

Fig. 3. The Step model.

An *Alldifferent* constraint ensures that the machine passes exactly once in each row. Additional variables are used to represent the quantity of grapes har-

[3] Because of the turning radius of the machine, turning into the next row is generally longer than jumping to the second next row.

vested until each step. They are subject to a set of numeric constraints that control the capacities of the hoppers.

The first k steps of the route perform differential harvest. Before step k, B grapes cannot appear in A hoppers, i.e. harvest is separated. In step k, the machine goes to the bin and must have harvested at least a total of $Rmin$ A grapes. After step k, A grapes and B grapes are mixed in both hoppers.

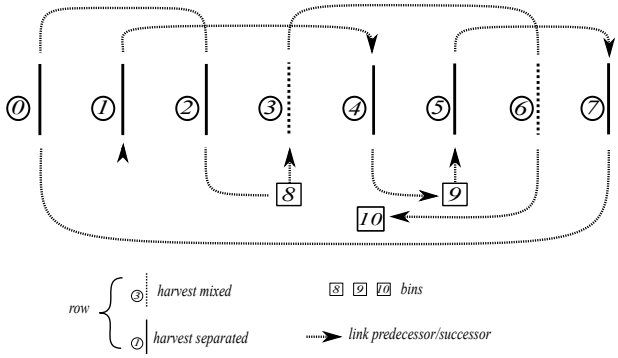
In the first version of this Step model we included k as a variable but experimental results were very bad. The combinatorial aspect of the problem decreases if we set k to a fixed value. Hence, each instance of the original problem is replaced by a set of sub-instances with different values of k. According to the capacity of the hoppers and the value of $Rmin$, few values of k must be considered. Since these instances are independent, they can be executed in parallel.

In section 4, we will see that this model is not effective. The next subsection gives another model for the DHP, based on models for routing problems.

3.2 The Precedence model

The *Precedence model* is based on variables that represent the previous row of a row. Our model shares similarities with the model presented in [9] for the Vehicle Routing Problem.

For a given instance of the Differential Harvest Problem, suppose that we have an upper bound λ on the number of times the hoppers have to be dumped into the bin. We note γ the number of dumps used in the differential part (beginning) of the harvest ($\gamma \leq \lambda \leq n$) where n is the number of rows in the vine field. We call $\mathcal{R} = \{0, \ldots, n-1\}$ the set of rows. $\mathcal{S} = \mathcal{R} \cup \{n, \ldots, n+\gamma-1, \ldots, n+\lambda-1\}$ denotes the set of *sites* (rows and hopper dumps into the bin).



Site	0	1	2	3	4	5	6	7	8	9	10
Predecessor	7	10	0	8	1	9	3	5	2	4	6
Orientation	0	0	1	0	1	0	1	1			
Mix	0	0	0	1	0	0	1	0	0	0	1

Fig. 4. The Precedence model.

Variables For each site $i \in \mathcal{S}$ we create:

- $\mathbf{P_i}, \mathbf{S_i}$: are integer variables that represent the direct predecessor (P_i) and direct successor (S_i) of site i. We have $D(P_i) = D(S_i) = \mathcal{S} \setminus \{i\}$;
- $\mathbf{Mix_i}$: is a Boolean variable that represents the harvest mode in site i. If $Mix_i = 0$ the A grapes are separated (differential harvest) otherwise they are mixed with B grapes. The domain of this variable is: $D(Mix_i) = \{0, 1\}$;
- $\mathbf{T_i}$: is an integer variable that represents the time to travel from P_i to site i, including the time to harvest row i. $D(T_i) = \mathbb{N}$;
- $\mathbf{U_i^A}$, $\mathbf{U_i^B}$ are integer variables that represent the quantity of A grapes and B grapes harvested up to site i since the last dump into the bin. The domains are: $D(U_i^A) = D(U_i^B) = \{0, \ldots, 2 \times Cmax\}$.

For each row $r \in \mathcal{R}$ we have:

- $\mathbf{Ori_r}$: is a Boolean variable that represents the orientation for row r (0 is the direction from odd to even extremities, as described in Section 2);
- $\mathbf{u_r^A}$ (resp. $\mathbf{u_r^B}$): represents the quantity of grapes of quality A (resp. B) harvested in row r according to the direction of harvest. $D(u_r^A) = D(u_r^B) = \mathbb{N}$;

Constraints Predecessor variables form a permutation of rows and are subject to the *alldifferent* constraint (1).

$$AllDifferent(P_0, \ldots, P_{n+\lambda-1}) \tag{1}$$

Constraint (2) is a *channelling* constraint between predecessor and successor variables.

$$P_{S_i} = i \qquad S_{P_i} = i \qquad \forall i \in \mathcal{S} \tag{2}$$

Constraints (3) and (4) force the harvest mode to be differential or mixed according to the index of the corresponding hopper dump. Constraint (5) is an *element* constraint that communicates the harvest mode between successors:

$$Mix_i = 0 \qquad \forall i \in \{n, \ldots, n + \gamma - 1\} \tag{3}$$

$$Mix_i = 1 \qquad \forall i \in \{n + \gamma, \ldots, n + \lambda - 1\} \tag{4}$$

$$Mix_r = Mix_{S_r} \qquad \forall r \in \mathcal{R} \tag{5}$$

The following constraints give the quantities of grapes according to the orientation of the row. It can be implement as a table constraint or as an *if ... then ... else* one. $\forall r \in \mathcal{R}$, we have :

$$u_r^A = Ori_r \times Q^A_{2r \to 2r+1} + (1 - Ori_r) \times Q^A_{2r+1 \to 2r} \tag{6}$$

$$u_r^B = Ori_r \times Q^B_{2r \to 2r+1} + (1 - Ori_r) \times Q^B_{2r+1 \to 2r} \tag{7}$$

Constraint (8) fixes quantities A and B for all sites representing dumps into the bin. Constraint (9) computes the quantities at the end of row i by adding the

accumulated amount from predecessor P_i and the quantity in row i given by the precedent constraints.

$$U_i^\alpha = 0 \qquad \forall \alpha \in \{A, B\} \;\; \forall i \in \mathcal{S} \setminus \mathcal{R} \tag{8}$$

$$U_i^\alpha = U_{P_i}^\alpha + u_i^\alpha \qquad \forall \alpha \in \{A, B\} \; \forall i \in \mathcal{R} \tag{9}$$

Harvested quantities are limited by the capacity of hoppers. Variable U_i^A always have an upper bound of $2Cmax$ because A grapes can be put in the two hoppers. When variable $Mix_i = 0$, harvest is not mixed and quantity of B grapes is bound by $Cmax$ (10). When variable $Mix_i = 1$, A grapes and B grapes are mixed in the two hoppers. Constraint (11) checks that the total amount of harvested grapes is smaller than the capacity of the two hoppers:

$$U_i^B \leq (1 + Mix_i) \times Cmax \qquad \forall i \in \mathcal{R} \tag{10}$$

$$U_i^A + U_i^B \leq 2 \times Cmax \qquad \forall i \in \mathcal{R} \tag{11}$$

Constraint (12) requires to harvest at least $Rmin$ A grapes. Only the A grapes stored in A hopper must be considered. This quantity corresponds to the part of U_i^A which is smaller than the capacity of the A hopper. It concerns the differential harvest mode only, i.e. dumps from n to $n + \gamma - 1$.

$$\sum_{i=n}^{n+\gamma-1} min(U_{p_i}^A, Cmax) \geq Rmin \tag{12}$$

It is possible to reformulate constraint (12) without the minimum operator. We add new variables called Ac_i $\forall i \in \{n, \ldots, n + \gamma - 1\}$ such that:

$$Ac_i \leq Cmax \qquad \text{that is: } D(Ac_i) = \{0, \ldots, Cmax\} \tag{12'a}$$

$$Ac_i \leq U_{P_i}^A \qquad \forall i \in \{n, \ldots, n + \gamma - 1\} \tag{12'b}$$

$$\sum_{i=n}^{n+\gamma-1} Ac_i \geq Rmin \tag{12'c}$$

Constraint (13) forces the exit from row P_i to be on the same side of the vine field as the entrance of row i. Hence, P_i and i have inverse orientations. This is the case in practice with traditional routing.

$$Ori_i = 1 - Ori_{P_i} \qquad \forall i \in \mathcal{R} \tag{13}$$

Next constraints ((14a) and (14b)) require a unique cycle (subtour elimination) on predecessor variables and successor variables. Figure 5 is an example of filtering for the circuit constraint described in [5, 12].

$$Circuit(1, (P_0, \ldots, P_{n+\lambda-1})) \tag{14a}$$

$$Circuit(1, (S_0, \ldots, S_{n+\lambda-1})) \tag{14b}$$

Constraints (15) force T_i to be equal to the time to travel from P_i to site i,

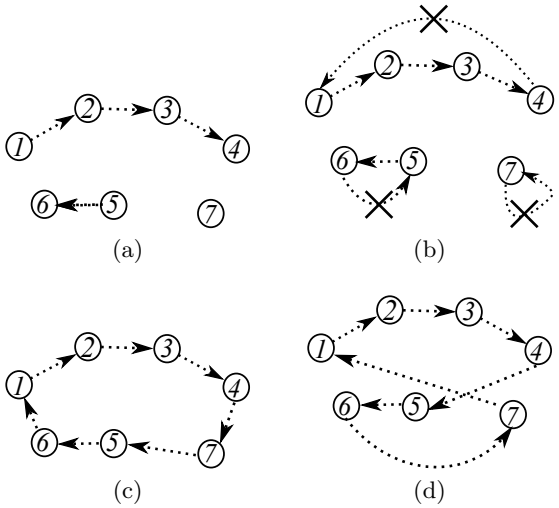


Fig. 5. An example of circuit constraint propagator on successor variables. Arrows represent the link (successor) between two variables (circles). (a) is a current instantiation. (b) filtering prohibits links between variables that represent the end and the start of a same path (i.e., the value that represents the start of a subpath is removed from the domain of variables that represent the end of this path). (c) and (d) represent two possible solutions in this example.

including the time to harvest row i if it is the case. Function d gives the time to travel between two extremities of rows (or an extremity and the bin) as defined in Section 2. Note that for two consecutive dumps into the bin ($i \notin \mathcal{R}$ and $P_i \notin \mathcal{R}$) the travel time is equal to 0 (15b).

$$\begin{aligned} &\text{if } P_r \in \mathcal{R} \text{ then } T_r = d(2P_r + Ori_{P_r}, 2r + 1 - Ori_r) \\ &\qquad \text{else } T_r = d(2n, 2r + 1 - Ori_r) && \forall r \in \mathcal{R} && (15a) \\ &\text{if } P_i \in \mathcal{R} \text{ then } T_i = d(2P_i + Ori_{P_i}, 2n) \text{ else } T_i = 0 && \forall i \in \mathcal{S} \setminus \mathcal{R} && (15b) \end{aligned}$$

Constraint (16) is the objective function that minimizes the travel time of the grape harvester:

$$Minimize \sum_{i=0}^{n+\lambda-1} T_i \tag{16}$$

Symmetry breaking The main variables of this model are predecessor variables. They take their values in a set of rows and dumps into the bin. For the set of dumps in the differential (or mixed) mode, there is a value symmetry caused by the possibility to interchange the dump indexes. To break this symmetry, we add an ordering constraint (17a) (resp. (17b)) on the variables that correspond

to the same harvest mode.

$$P_i < P_{i+1} \qquad \forall i \in \{n, \ldots, n+\gamma-2\} \tag{17a}$$

$$P_i < P_{i+1} \qquad \forall i \in \{n+\gamma, \ldots, n+\lambda-2\} \tag{17b}$$

4 Experimental Results

In this section, we present some experimental results. We have implemented the two models in Choco [13] using the variable ordering strategy DomOverWDeg [3] limited to the P_i variables. All experiments were executed on a Linux server with an Intel(R) Xeon(R) CPU E5-2697 2.60GHz processor. It has 14 cores that makes it possible to solve several instances in parallel but the solver was not configured to exploit the multicore architecture.

Our models were tested on a real data from an experimental vineyard of INRA Pech-Rouge (Gruissan) located in southern France (see Figure 6). In this experiment we want to compute the optimal solution. To test the models on instances of increasing size, we have extracted subsets of continuous rows from the 24 rows vine field.

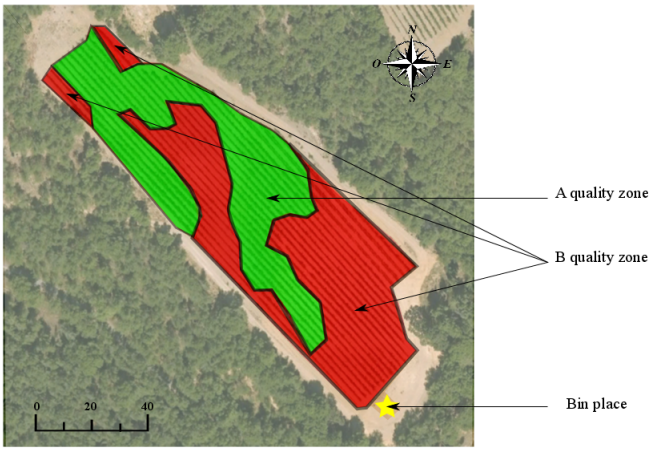


Fig. 6. A vine field with two qualities B grapes in red and A grapes in green.

Table 1 reports results with Step and Precedence models for 10, 12 and 14 rows. Each row in the table reports the average time on 12 instances of the given size generated from contiguous sequences of rows of the vine field. For each sequence, we experimented with two capacities of the hoppers (1000 and 2000) and two values for the desired threshold $Rmin$ (50% and 70% of the total amount of A grapes).

For the Step model, each instance of the Differential Harvest Problem is split into a set of more specific sub-instances of the constraint optimization problem. Each sub-instance is associated with a fixed value k of number of rows harvested on differential mode. For 10 (resp. 12) rows we varied k from 4 to 8 (resp. 6 to 10). Each set of sub-instances can be solved in parallel. So, Table 1 shows the maximum time needed to solve a sub-instance of a given set. The total time (for all sub-instances) corresponds to a single core architecture. Table 1 also gives the time for the sub-instance which has the optimal solution. With sequences of 14 rows, at least one sub-instance of each instance is not solved before the timeout (7200 sec).

Table 1. Comparison of the Step model and the Precedence model. All CPU times are averages on 12 instances and are given in seconds. Timeout at 7200 seconds.

# rows	Step model			Precedence model
	Parallel solving (maximum time)	Single processor (total time)	Time for the sub-instance with the optimal solution	Time
10	262	527	180	8
12	3414	8466	2749	118
14	timeout			8 instances solved

These results show that the Step model is clearly outperformed by the Precedence model. One disadvantage of the Step model is that it has to communicate the harvested quantities between steps. Hence, as long as the route is undefined, it is impossible to know if the capacity of hoppers is exceeded and if $Rmin$ is reached. This hinders propagation. There is another drawback of the Step model. It assumes a fixed value k for the step on which differential harvest stops (A grapes and B grapes are mixed from step k to step n). Finding an optimal solution requires to solve several instances of the problem with different values of k. This can be done in parallel as the problems are independent. But Table 1 shows that, even with an oracle predicting the value k of the sub-instance containing the optimal solution of the original instance, the Step model is outperformed by the Precedence model.

Table 2 shows results for instances of the Precedence model for sequences of 12, 14 and 16 rows. The first column shows the number of rows in the vine field. The second column gives the capacity of the hoppers ($Cmax$) and the third column gives the desired threshold $Rmin$. For each size of problem, we give the average time to solve it and the average number of nodes. Timeout is 7200 sec. For 16 rows with a $Cmax$ value of 1000 and for any greater number of rows, all instances exeeded the timeout.

These results indicate fairly high variability in CPU times. Instances with a small value of $Cmax$ seem to be more difficult to solve. When the threshold $Rmin$ is equal to 50% of A grapes, CPU time generally increases.

Table 2. Precedence model on 12, 14 and 16 rows.

# Rows	Cmax	Rmin	Time (s)	# Nodes
12	1000	50%	227	663267
		70%	170	667112
	2000	50%	35	153300
		70%	41	175017
14	1000	50%	2/3 instances solved	
		70%	0/3 instances solved	
	2000	50%	52	181262
		70%	21	72468
16	2000	50%	856	2835199
		70%	106	318246

Table 3 shows results for different configurations of the Precedence model. All CPU times and node numbers are average on all instances of the same number of rows. The first two columns correspond to the complete Precedence model. The next columns give the results for variants of this model: without symmetry breaking (constraints (17a) and (17b)); with constraint (12) instead of constraints (12'a), (12'b) and (12'c); and with the default Choco variable ordering (minDomain) instead of the DomOverWDeg strategy.

Despite a small number of bins, adding constraints to eliminate symmetric solutions helps a lot to reduce the search effort. It is also the case when constraint (12) with the minimum function is replaced by constraints (12'a), (12'b) and (12'c), or when the default Choco variable ordering is replaced by DomOverWDeg.

Table 3. Impact of different configurations for the Precedence model

	Precedence model		without symmetry breaking constraints		with min function in (12)		with minDomain variable ordering	
# Row	Time	# Nodes	Time	# Nodes	Time	# Nodes	Time	# Nodes
10	8	41811	17	95363	167	382345	101	1066550
12	118	441674	539	4017780	1042	4824446	776	3582405

5 Discussion

Our experimental results show that the Precedence model is orders of magnitude faster than the Step model. For small instances (up to 16 rows), the Precedence model solves the Differential Havest Problem in a reasonable amount of time. For larger instances, the timeout of 2 hours is generally reached. This amount

of time has to be compared to the time needed to collect data and to build the quality and yield map for the vine field. With additional improvements of the CP model, we can expect to solve larger real instances.

Our results confirm what other studies have reported on similar problems like the BBSS problem presented in [7]. In that paper, Di Gaspero *et al.* show that a Step formulation is outperformed by Routing (similar to Precedence) formulation for the optimal solution. In a second experiment, they show that large neighborhood search (LNS) is a good approach to decrease the time to solve, though the optimal solution can no longer be guaranteed. It could be interesting to explore that direction.

5.1 Comparison with an ILP formulation

Contrary to our expectations, it was not difficult to express our model of the Differential Harvesting Problem as a set of linear constraints on integer variables. Such a problem can be solved with Integer Linear Programming (ILP).

So we have designed an ILP model using the two-index vehicle flow formulation of the Capacited Vehicle Routing Problem (CVRP) introduced by Laporte, Nobert, and Desrochers [10]. There is one Boolean variable $x_{t:i\to j}$, for each pair i, j of rows extremities and bin on tour t. Each tour corresponds to a single mode of harvest (differential or not). $x_{t:i\to j}$ is equal to 1 if the harvesting machine goes from i to j on tour t and 0 otherwise. Constraints on the $x_{t:i\to j}$ variables ensure that each extremity is visited exactly once on all routes. Quantities of grapes harvested in each tour are expressed by a summation over the traversed rows and similarly for travel cost (objective function). As in CVRP formulation, there are constraints on capacity of hoppers according to the tour t. Variables and linear constraints (12'a), (12'b), and (12'c) on threshold $Rmin$ can directly be added to the ILP formulation.

Unfortunately, the cycle constraint cannot be added so easily. The basic ILP approach to forbid sub-tours (cycles that do not pass through the bin) is to post constraints of the form $\sum_{i,j\in S} x_{t:i\to j} < |S|$ for any subset S of extremities of rows. The number of such constraints is exponential. Another formulation introduced by Miller, Tucker, and Zemlin ($MTZ - formulation$ [11]) makes the number of constraints polynomial but its linear relaxation generally produces a significantly weaker lower bound compared to the basic model [6]. Hence, a typical approach consists in adding the sub-tour elimination constraints incrementally, as follows: Step 1: find an optimal solution (without sub-tour constraints at the beginning). Step 2: If the solution does not contain sub-tours, it is an optimal solution of the problem; Otherwise, new constraints are added to forbid all the sub-tours in the current solution. Then, proceed to Step 1. In the worst case, this algorithm has to solve an exponential number of instances with finally an exponential number of constraints on sub-tours. Thus, each intermediate ILP loop gives relaxed solutions that contain sub-tours between rows. It is only in the last ILP step that the reported (optimal) solution does not contain sub-tours and can be exploited. As a consequence, if we interrupt the ILP solver at a given

timeout limit, it has not yet produced any feasible solution, not even suboptimal ones. This is an advantage of the CP model over the ILP model. Any of its solutions, at any time in the solving process, can be exploited.

We have implemented the ILP model with the Cplex solver [8]. Table 4 shows a comparison between the Precedence model and the ILP formulation using a single core. Tests are performed with and without an initial upper bound. This bound is an estimation of the harvest time using a manual routing based on a repetitive pattern. This upper bound does not improve the ILP results so we give only one value for the ILP model. Preliminary experiments show that the ILP formulation clearly outperforms the CP model for small instances. But for hard instances ($n \geq 16$ and $Cmax = 1000$), ILP search fails to obtain a solution before the timeout of 7200 seconds. On hard instances the CP model can find suboptimal solutions that significantly improve the manual routing solution whilst ILP provides no solution at all.

Table 4. Comparison between CP model and ILP model for 12, 14, 16 and 24 rows, $Cmax = 1000$, $Rmin = 70\%$ and timeout of 7200s. In bold, optimal solutions.

# rows	upper-bound for	CP model		ILP model	
	harvesting time	harvesting time	CPU time	harvesting time	CPU time
12	$+\infty$	**960**	556s	**960**	3s
	2671	**960**	496s		
14	$+\infty$	*1120*	timeout	**1112**	113s
	2436	*1116*	timeout		
16	$+\infty$	*1260*	timeout	-	timeout
	2694	*1260*	timeout		
24	$+\infty$	*1805*	timeout	-	timeout
	2059	*1800*	timeout		

5.2 Complements to the model

Another advantage of the CP model is its ability to integrate new constraints. The formulation of the problem presented in this paper is a quite simplified version of the Differential Harvest Problem and it is dedicated to evolve. For instance, consider the case where the harvesting machine finishes a tour on an extremity of a row opposite to the bin. If the row is in the middle of the vineyard, the shortest path to the bin passes through a row. But it is not possible to go to the bin by passing through a non-harvested row. This can be expressed by the following rule: the path to the bin, for an extremity which is on the side opposite to the bin, must pass through a row that precedes the extremity in the global route. Such a constraint is very difficult to add to the ILP formulation but can be easily implemented in the CP approach. The propagate procedure of the circuit constraint already computes the set of rows that precede the last step of any tour in order to detect subtours. So it is easy to find in this set what

is the best row to go back to the bin. This can be done without changing the overall complexity of the propagate procedure.

6 Conclusion

In this paper, we have presented the Differential Harvest Problem in precision viticulture. We have proposed to use constraint programming to solve it. Two models have been presented, the Step model and the Precedence model. In the Step model, variables represent the row that is visited at a given time step and in which direction the row is traversed. In the Precedence model, variables connect a row to its predecessor and successor. The experiments we have performed to assess the behavior of these models show that the Precedence model is orders of magnitude faster than the Step model. We have also experimentally shown that an ILP formulation of the Differential Harvest Problem outperforms our CP approach on easy instances. However, such an ILP formulation requires an exponential space, and more importantly, fails to produce solutions on hard instances. All in all, our Precedence model seems to be a good approach. It allows to solve the problem on real data within reasonable time and it inherits the flexibility of CP models, that allows the addition of extra user-constraints in a simple way.

References

1. Dionysis Bochtis and Claus G. Sørensen. The vehicle routing problem in field logistics: part i. *Biosystems engineering*, 104:447–457, 2009.
2. Dionysis Bochtis and Claus G. Sørensen. The vehicle routing problem in field logistics: part ii. *Biosystems engineering*, 105:180–188, 2010.
3. Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, and Lakhdar Sais. Boosting systematic search by weighting constraints. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI'2004)*, pages 146–150, Valencia, Spain, 2004.
4. Nicolas Briot, Christian Bessiere, Bruno Tisseyre, and Philippe Vismara. Integration of operational constraints to optimize differential harvest in viticulture. In *Proc. 10th European Conference on Precision Agriculture*, July 2015, to appear.
5. Yves Caseau and François Laburthe. Solving small TSPs with constraints. In Lee Naish, editor, *Logic Programming, Proceedings of the Fourteenth International Conference on Logic Programming, Leuven, Belgium, July 8-11, 1997*, pages 316–330. MIT Press, 1997.
6. Martin Desrochers and Gilbert Laporte. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36, 1991.
7. Luca Di Gaspero, Andrea Rendl, and Tommaso Urli. Constraint-based approaches for balancing bike sharing systems. In *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, pages 758–773, 2013.
8. IBM ILOG. Cplex. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`, (accessed April, 2015).

9. Philip Kilby and Paul Shaw. Vehicle routing. In Francesca Rossi, Peter Van Beek, and Toby Walsh, editors, *Handbook of constraint programming*, chapter 23, pages 799, 834. Elsevier, 2006.
10. Gilbert Laporte, Yves Nobert, and Martin Desrochers. Optimal routing under capacity and distance restrictions. *Operations research*, 33(5):1050–1073, 1985.
11. Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.
12. Gilles Pesant, Michel Gendreau, Jean-Yves Potvin, and Jean-Marc Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32(1):12–29, 1998.
13. Charles Prud'homme, Jean-Guillaume Fages, and Xavier Lorca. *Choco3 Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S., 2014.
14. Bruno Tisseyre, Hernan Ojeda, and James Taylor. New technologies and methodologies for site-specific viticulture. *Journal International des Sciences de la Vigne et du Vin*, 41:63–76, 2007.