New Approach for Differential Harvest Problem: The model checking way

Rim Saddem-Yagoubi, Olivier Naud, Karen Godary-Dejean, Didier Crestani

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-7 (2018) 57–63

New Approach for Differential Harvest Problem: The model checking way

Rim SADDEM YAGOUBI [*,**] Olivier NAUD [*]
Karen GODARY DEJEAN [**] Didier CRESTANI [**]

[*] *ITAP, Irstea, Montpellier SupAgro, Univ Montpellier, Montpellier, France*
(e-mail: {rim.saddem@irstea.fr},{olivier.naud@irstea.fr})
[**] *LIRMM, Univ Montpellier, CNRS, Montpellier, France,*
(e-mail: {godary,crestani}@lirmm.fr)

Abstract:
The development, in the last decades, of technologies for precision agriculture allows the acquisition of crop data with a high spatial resolution. This offers possibilities for innovative control and raises new logistics issues that may be solved using discrete event models. In vineyards, some technologies make it possible to define zones with different qualities of grapes and sort the grapes at harvest to make different vintages. In this context, the Differential Harvest Problem (DHP) consists in finding a trajectory of the harvesting machine in the field in order to obtain at least a given quantity of higher quality grapes and minimising working time. In available literature, the DHP has been solved using Constraint Programming. In this paper, we investigate if it is possible to solve the DHP using the Cost Optimal Reachability Analysis feature of a model-checking tool such as UppAal-CORA. A model named DHP_PTA has been designed based on the priced timed automata formalism and the UppAal-CORA tool. The method made it possible to obtain the optimal trajectory of a harvesting machine for a vine plot composed of up to 12 rows. The study is based on real vineyard data.

Keywords: Timed automata, Formal verification, Model checking, Vehicle routing, UppAal CORA, Optimisation, Precision agriculture

1. INTRODUCTION & RELATED WORK

Precision Agriculture (PAg) has been a research topic for many years. The purpose of PAg is to adapt agricultural actions to local situation in the field so that the right action is performed at the right place and at the right time (McBratney et al. (2005)). Technologies for PAg are now widespread in the farms and are being developed further. These technologies raise decision issues about logistics so that an adequate compromise can be found between the different criteria of the farmer and so that certain desired properties can be verified. Differential harvesting in vineyards is an example of such concern. Instead of harvesting everything in a plot for same wine quality, one would like to define zones with different qualities and sort the grapes. An example of such technology is described in (Briot et al. (2015c)). Sorting grapes may increase harvesting time and costs. The issue is then to find the best harvesting logistics, provided that grape quality criteria, expressed as constraints on harvesting system and harvesting output, are met.

Timed Automata (TA), as defined by Alur and Dill (1994), is a simple and expressive formalism to model the timed behaviour of real-time systems. It extends finite automata with a finite set of real-valued variables, called clocks. UppAal is a tool designed to validate systems that can be modeled as networks of TA. UppAal extends the TA formalism by adding integer variables, structured data types, user defined functions, and synchronisation channels (Bengtsson and Yi (2004)). TA and UppAal have already been used in the agricultural and ecosystem management domains (e.g in Largouët et al. (2012)).

UppAal-CORA (Behrmann et al. (2004)) is a variant of UppAal for Cost-Optimal Reachability Analysis of Priced Timed Automata. Cost-optimal reachability is the problem of finding the minimum cost to reach a certain target location in a given Priced Timed Automata (Bisgaard et al. (2016)). UppAal-CORA was developed in 2004 and Priced Timed Automata (PTA) were introduced independently by Behrmann et al. (2001a) and Alur et al. (2001) in 2001 (Fahrenberg et al. (2013)). PTA can be viewed as TA with one specific non-negative integer variable named cost. The cost is a monotonous function: it never decreases when time flows. UppAal-CORA has been used successfully for scheduling and routing problems in several case studies (Behrmann et al. (2004): Task Graph Scheduling, Aircraft Landing Problem, Vehicle Routing Problem with Time Windows, etc.). UppAal-CORA has also been used to solve a precision spraying problem for vineyards (Saddem et al. (2017)).

[⋆] This work has received the support of French National Research Agency under the grant number ANR-14-CE27-0004 attributed to AdAP2E project.

We investigate in this paper how UppAal-CORA can be applied to grape harvesting logistics, which present similarities with scheduling and routing problems. The reason to investigate this topic is that an UppAal-CORA based approach would have the advantage over alternatives for optimisation under constraints raised by PAg, such as Mixed Integer Programming methods or Constraints Programming, that system dynamics involved can be rather straightforwardly described in the PTA formalism.

Precisely, the problem studied in this paper is the Differential Harvest Problem (DHP). Let us consider a vine plot in which two qualities of grapes can be distinguished before harvest so that a map with zones having quality A and zones having quality B can be drawn. A vine plot is planted in rows, and each row has sections with quality A (higher quality) and sections with quality B. Let us consider a harvester having two hoppers which can operate in two modes: (selective) harvesting sequentially quality A and quality B grapes and directing them to hoppers "a" and "b" respectively and (non selective) filling "a" and "b" hoppers indistinctly with all grapes. In selective mode, as soon as one hopper is full, it must be emptied into a bin located at some place in the plot. Due to time needed to go to the bin and empty a hopper, the total working time to harvest the plot depends on the order and direction in which rows are harvested. The objective of DHP is to minimise the working time of the harvesting machine while ensuring that all grapes have been harvested and that a given minimum quantity of grapes of the higher quality has been sorted.

The DHP has been studied by Briot (Briot et al. (2015a), Briot et al. (2015b)), using Constraint Programming techniques (Dechter (2003)) and two alternative models. The first one is called Step model and is based on making 3 choices at each step of building the harvester route: which row next, in which direction, going afterwards to bin to dump harvest - or not. The second one is called the Precedence model. It was inspired by Kilby and Shaw (2006) in which a Vehicle Routing Problem (VRP) was solved using Constraint Programming. It uses a constraint called Circuit that prevents sub-tours in visits of locations (rows, bin). According to authors, solving with the Step model is time consuming and is limited by the high amount of memory needed while performances obtained with the Precedence model are better but require specific constraints. In this paper, we investigate if it is possible to solve the DHP using UppAal-CORA without dedicated heuristics to guide cost optimisation during reachability analysis.

The rest of the paper is organised as follows. In section 2, the Differential Harvest Problem is defined in details. In section 3, a model for expressing the DHP as a network of priced timed automata is described. It is based on the tool UppAal-CORA. The reachability property used to solve DHP with the model is provided. In Section 4, experimental results are described and discussed before concluding.

2. DETAILED DEFINITION OF DIFFERENTIAL HARVEST PROBLEM

In this section, we provide details about the DHP as it is defined in Briot et al. (2015a).
In a vine field, zones can be defined according to two grape quality grades A (higher quality) and B. A vine field is composed of several A and B zones, respectively containing A-grapes and B-grapes. Agronomists can define these zones using aerial photos. A Harvesting Machine (HM) ready for selective harvesting of two qualities has two hoppers that we name a-hopper and b-hopper. HM is able to direct the grapes to either hopper and can thus sort qualities according to A and B zones. The capacity of each hopper is limited to a maximum value $Cmax$. When HM is in selective mode, a-hopper contains only A-grapes and b-hopper contains B-grapes and may contain also A-grapes that will be lost for the high quality vintage.

The harvesting mechanism, named picking head, is composed of a set of shaking rods and conveyors that move fallen grapes to a mechanism on top of the machine that directs grapes to hoppers. Due to the longitudinal size of the picking head, there is a phenomenon called *latency* when the machine moves from a zone to another. We can distinguish two cases illustrated in figure 1.

The first case is that HM passes from a A-zone to a B-zone (see scenario 1). As soon as the machine enters the B-zone, grapes are directed to the b-hopper. The second case is the reverse zone change and is depicted as scenario 2. When the machines enters the A-zone, A-grapes and B-grapes can mingle in the picking head temporarily, and so they must be sorted to the b-hopper so that the a-hopper does not contain B-grapes. This causes the latency time on changing from a-hopper to b-hopper. Once this time has elapsed, there are only A-grapes in the picking head and they can be directed to the a-hopper. During operation, the value of latency time when moving from a B-zone to a A-zone is considered a fixed parameter. When moving from a A-zone to a B-zone, there is no latency.

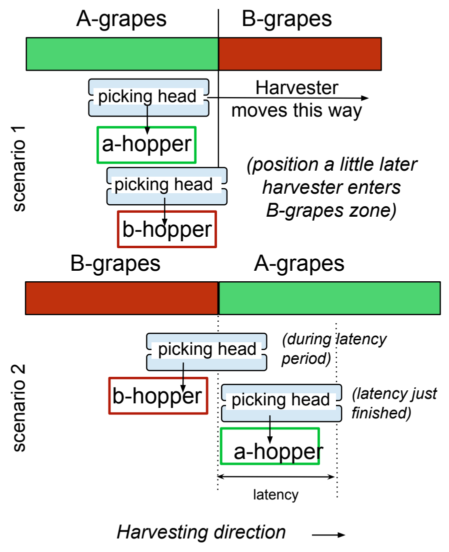


Fig. 1. Two cases: A-zone → B-zone or B-zone → A-zone

As a consequence of this latency time asymmetry, the quantities of A-grapes and B-grape that can be harvested in a row depend on the direction of HM in this row.

The agronomic objective of the DHP is to have at least a quantity $Rmin$ of A-grapes. If this threshold is reached, and once the contents of both hoppers have been dumped in the bin, HM switches to the non selective mode and put A-grapes and B-grapes in any of the hoppers. If $Rmin$ has been reached and hoppers have not yet been emptied, the machine cannot change from selective to non selective mode. It is supposed that HM starts operations in selective mode and switches to non selective only once and remains after in this latter mode.

A vine field is composed of n rows. Each row i is represented by two extremities N_i and S_i. All N_i are "on the same side" of the field, and the same applies to the S_i. Let us call *direct orientation*, the orientation from N_i to S_i and *inverse orientation*, the orientation from S_i to N_i. It is usual and efficient practice that if a row is harvested in the direct orientation, the row to be harvested next should be harvested in inverse orientation and vice versa. The model described hereafter takes advantage of this constraint, so that state space exploration can be reduced during problem solving, as was done in Briot et al. (2015b).

Let Q_A_D (resp. Q_B_D) be the array that specifies the quantity of A-grapes (resp. B-grapes) that will be collected in each row if it will be harvested in the Direct orientation. And Q_A_I, Q_B_I the equivalent in the Inverse orientation.
Finally, let DD be the cost matrix. DD represent distances (that may be represented as travel times) between all extremities and distances between each row extremity and the bin place.

The objective of the DHP is to find the order and orientation in which rows should be harvested so that at least $Rmin$ of A-grapes are harvested and so that the total travel distance of the harvesting machine is minimised.

3. MODELLING AND PROPERTIES TO CHECK

It can be acknowledged, as is done in Briot et al. (2015b), that the DHP has similarities with a VRP.

3.1 Definition of the Model

Let us recall a definition of vehicle routing problem that is taken from Kilby and Shaw (2006).

Definition 1. "A set of n customers is to be served by m vehicles. Each customer must be visited by exactly one vehicle. Customer i has demand r_i, and the sum of demands of customers assigned to vehicle k must be less than the vehicle capacity Q. All vehicles begin and end their route at a single depot. The objective is to minimise the sum of travel costs".

Each row to harvest in a DHP may be seen as a customer and the harvesting machine HM is the only vehicle that will serve all customers, going several times to the bin (in other words, the depot). This relates DHP to a multi-trip routing problem, the whole route being composed of several trips from the bin to a subset of the rows and back to the bin. It is supposed that there is only one bin place, with 2 containers, one for A-grapes only and one for B-grapes and extra A-grapes.

Solving the DHP is finding the sequence of harvested rows, with direction for each, and emptyings that optimises the travel (working) time of HM. The whole route of HM must start and finish at the bin. The question of the direction when harvesting a row distinguishes the DHP from a classical VRP. Indeed, in the cost matrix, because of the direction feature, a row is modelled by having two vertices per row instead of one in the graph of distances (travel and harvesting times).

In the following, the model representing the DHP as a network of PTA (called DHP_PTA) and a CORA (Cost Optimal Reachability Analysis) query are presented. The input data of DHP_PTA is a cost matrix described below.

Cost matrix: The graph provided in figure 2 illustrates how the matrix is built. Each row i in the field is represented by two extremities N_i and S_i. The cost matrix represents the required time to move from a row extremity to another or from a row extremity to the bin.

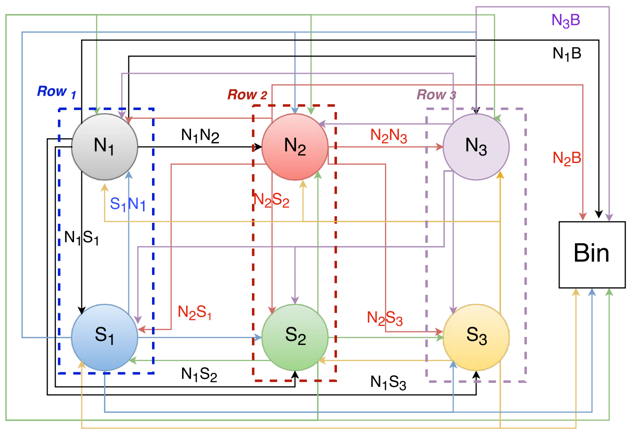


Fig. 2. DHP travel data: a case with 3 rows and a bin

In this figure, a colour is assigned to each row extremity and to the arcs from this extremity. For example, the black colour is assigned to N_1. From N_1, HM can visit N_2, N_3, S_1, S_2, S_3 and the bin (B). Because of the constraint that two rows that are harvested one after the other have to be harvested in opposite orientations, it follows that from N_1, HM does not need to visit S_2 and S_3. So N_1S_2 (travel time from N_1 to S_2) and N_1S_3 (travel time from N_1 to S_3) are not required (symbol $-$ in the matrix). In the matrix, $F.$ and B mean "From" and "Bin", respectively. Based on Figure 2, the cost matrix DD will be as follows:

$$DD = \begin{array}{c} \\ F.\ Bin \\ F.\ N_1 \\ F.\ S_1 \\ F.\ N_2 \\ F.\ S_2 \\ F.\ N_3 \\ F.\ S_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} Bin & N_1 & S_1 & N_2 & S_2 & N_3 & S_3 \end{array} \\ \begin{pmatrix} 0 & BN_1 & BS_1 & BN_2 & BS_2 & BN_3 & BS_3 \\ N_1B & 0 & N_1S_1 & N_1N_2 & - & N_1N_3 & - \\ S_1B & S_1N_1 & 0 & - & S_1S_2 & - & S_1S_3 \\ N_2B & N_2N_1 & - & 0 & N_2S_2 & N_2N_3 & - \\ S_2B & - & S_2S_1 & S_2N_2 & 0 & - & S_2S_3 \\ N_3B & N_3N_1 & - & N_3N_2 & - & 0 & N_3S_3 \\ S_3B & - & S_3S_1 & - & S_3S_2 & S_3N_3 & 0 \end{pmatrix} \end{array}$$

S_iS_i an N_iN_i are always equal to 0 because HM stays in the same place. XY is equal to YX with X and Y representing one of the row extremities and bin place.

Because this original matrix is symmetric, it can be reduced to:

$$dd = \begin{array}{c} \\ F.\ Bin \\ F.\ Row_1 \\ F.\ Row_2 \\ F.\ Row_3 \end{array} \begin{array}{c} \begin{array}{cccc} Bin & Row_1 & Row_2 & Row_3 \end{array} \\ \begin{pmatrix} BB & BN_1 & BN_2 & BN_3 \\ S_1B & N_1S_1 & N_1N_2 & N_1N_3 \\ S_2B & S_2S_1 & N_2S_2 & N_2N_3 \\ S_3B & S_3S_1 & S_3S_2 & N_3S_3 \end{pmatrix} \end{array} \quad (1)$$

In this reduced matrix dd, the upper triangular part represents travel times from bin to N_i extremities and travel times between N_i extremities. The diagonal column represents the harvesting time of each row. The lower triangular part represents travel times from S_i extremities to bin and travel times between S_i extremities.

The DHP_PTA model can now be described. It is a network of $n + 1$ automata: a harvesting machine automaton and n row automata. Each row automaton is instantiated with a row index number. The variables were defined in order to be consistent with Briot et al. (2015a).

Constants:

$CmaxHopper$: is the maximum capacity of each hopper.

$Rmin$: is the minimum quantity of A-grapes that must be harvested, provided as a percentage of the total quantity of A-grapes that is identified on the field map.

$Q\_A\_D[i]$ ($resp.\ Q\_B\_D[i]$): starting from $i = 0$, is the quantity of A-grapes (resp. B-grapes) that will be collected in the $(i+1)^{th}$ row if this row will be harvested in the direct orientation.

$Q\_B\_I[i](resp.Q\_A\_I[i])$: is the quantity of B-grapes (resp. A-grapes) that will be collected in the $(i+1)^{th}$ row if this row will be harvested in the inverse orientation.

dd: is the reduced cost matrix.

Variables:

$Mixed$: is a boolean variable that represents the harvest mode. If $Mixed = 0$, the harvest mode is differential ($Rmin$ quantity of A-grapes has not yet been reached and a-hopper can contain only A-grapes). Otherwise, A-grapes and B-grapes can be mixed in the two hoppers.

T: is a calculated integer variable that represents the time needed by HM to perform travel to next row and harvest it (see figure 3), or travel to bin and dump the harvest in the appropriate containers at the bin place (see figure 4).

$Location$: is an integer variable that represents the current location of HM. If $Location = 0$, HM is at the bin else if $Location = i$ with $i \in [1 : n]$ then HM is at i^{th} row.

$DumpNumber$: is an integer variable. It represents the number of dumps done since the beginning.

$Ori[i]$: is a boolean variable. If $Ori[i] = 1$, the $(i+1)^{th}$ row is harvested in the direct orientation else it is harvested in the inverse orientation. The status about whether a row has yet been harvested or not is given by state in the corresponding row automaton.

$Mix[i]$: is a boolean variable. If $Mix[i] = 0$, the $(i + 1)^{th}$ row is harvested in differential (selective) mode else it is harvested in non selective mode.

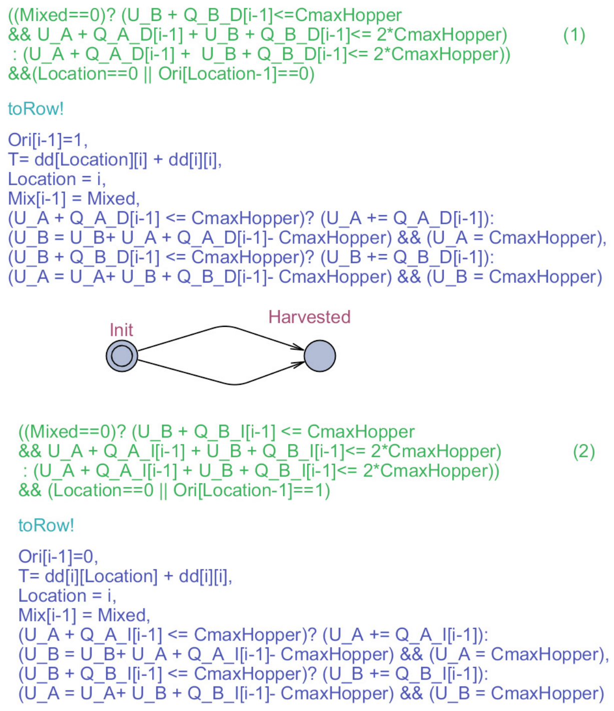
U_A ($resp.\ U\_B$): represents the current quantity of A-grapes (resp. B-grapes) in the a-hopper (resp. b-hopper).

Q_T_A ($resp.\ Q\_T\_B$): represents the quantity of A-grapes (resp. B-grapes) dumped until then at the bin place (in their respective containers).

c is a local clock for the vehicle model.

Row automata: There are two similar parts in the automaton depicted figure 3 that correspond to each potential orientation.

At $Init$ state, the row i is not harvested. It can be harvested according to direct orientation (top transition) or to inverse orientation (bottom transition). If the transition guard value is true ((1) or (2) in green in Figure 3), then a command event "toRow!" is sent to the vehicle automaton and the row passes to the $Harvested$ state.



```
((Mixed==0)? (U_B + Q_B_D[i-1]<=CmaxHopper
&& U_A + Q_A_D[i-1] + U_B + Q_B_D[i-1]<= 2*CmaxHopper)          (1)
 : (U_A + Q_A_D[i-1] +  U_B + Q_B_D[i-1]<= 2*CmaxHopper))
&&(Location==0 || Ori[Location-1]==0)

toRow!

Ori[i-1]=1,
T= dd[Location][i] + dd[i][i],
Location = i,
Mix[i-1] = Mixed,
(U_A + Q_A_D[i-1] <= CmaxHopper)? (U_A += Q_A_D[i-1]):
(U_B = U_B+ U_A + Q_A_D[i-1]- CmaxHopper) && (U_A = CmaxHopper),
(U_B + Q_B_D[i-1] <= CmaxHopper)? (U_B += Q_B_D[i-1]):
(U_A = U_A+ U_B + Q_B_D[i-1]- CmaxHopper) && (U_B = CmaxHopper)
```

```
((Mixed==0)? (U_B + Q_B_I[i-1] <= CmaxHopper
&& U_A + Q_A_I[i-1] + U_B + Q_B_I[i-1]<= 2*CmaxHopper)          (2)
 : (U_A + Q_A_I[i-1] + U_B + Q_B_I[i-1]<= 2*CmaxHopper))
&& (Location==0 || Ori[Location-1]==1)

toRow!

Ori[i-1]=0,
T= dd[i][Location] + dd[i][i],
Location = i,
Mix[i-1] = Mixed,
(U_A + Q_A_I[i-1] <= CmaxHopper)? (U_A += Q_A_I[i-1]):
(U_B = U_B+ U_A + Q_A_I[i-1]- CmaxHopper) && (U_A = CmaxHopper),
(U_B + Q_B_I[i-1] <= CmaxHopper)? (U_B += Q_B_I[i-1]):
(U_A = U_A+ U_B + Q_B_I[i-1]- CmaxHopper) && (U_B = CmaxHopper)
```

Fig. 3. Template row automaton for each row $i \in [1 : n]$

Let us examine the transition at the top, which has guard (1). The row i will be harvested in a direct orientation (Ori[i-1]=1). It follows that either the previous row has been harvested in the inverse orientation ($Ori[Location - 1] == 0$), or the location of HM before transition is the bin ($Location == 0$).

In addition, the row automaton also checks in the guard (1), and depending on the value of $Mixed$ variable, if enough space remains in the hopper to harvest the grapes of the row.

If $Mixed == 0$, the harvest mode is differential. The quantity $U\_B + Q\_B\_D[i - 1]$ (grapes in the b-hopper and B-grapes to be collected) must be limited to $CmaxHopper$. Furthermore, $U\_A + U\_B + Q\_A\_D[i - 1] + Q\_B\_D[i - 1]$ (grapes in the hoppers and A and B-grapes to be collected) must be limited to $2 \times CmaxHopper$ (note that A-grapes can be put in both a-hopper and b-hopper, regardless of harvest mode).

If $Mixed == 1$, the harvest mode is not differential. The row automaton checks that the quantity $U\_A + U\_B + Q\_A\_D[i-1] + Q\_B\_D[i-1]$ is limited to $2 \times CmaxHopper$.

If the value of the transition guard is true, the row automaton sends the control event toRow! and updates the variables $Ori[i-1]$, $Mix[i-1]$, $Location$, U_A, U_B and T.

$Ori[i-1]$ is updated to 1 if the i^{th} row is harvested in a direct orientation, else it is updated to 0. $Mix[i-1]$ is updated to the value of $Mixed$. $Location$ is updated to i.

The computing of U_A and U_B deserves some explanations. U_A computes the quantity of grapes in a-hopper at the end of row i. If $U\_A + Q\_A\_D[i-1]$ is lower than $CmaxHopper$, then U_A is updated to the value $U\_A + Q\_A\_D[i-1]$. Otherwise, U_A is updated to the value of $CmaxHopper$, the rest of A-grapes $(U\_A + Q\_A\_D[i-1] - CmaxHopper)$ is put in the b-hopper and U_B is updated accordingly.

U_B computes the quantity of grapes in the b-hopper at the end of row i. If $U\_B + Q\_B\_D[i-1]$ is lower than $CmaxHopper$, then U_B is updated to the value $U\_B + Q\_B\_D[i]$. Otherwise, U_B is updated to the value of $CmaxHopper$ and the rest of B-grapes $(U\_B + Q\_B\_D[i] - CmaxHopper)$ is put in a-hopper. So, U_A is updated with this rest of B-grapes. Please note that the latter will happen only in the mixed (non differential) harvest mode. Indeed, the case $U\_B + Q\_B\_D[i] > CmaxHopper$ can occur only in the mixed mode. On the guard of transistion, it is stated that if otherwise $(Mixed == 0)$, then $U\_B + Q\_B\_D[i] \leq CmaxHopper$.

T is updated to the sum of travel time and harvest time (service time in the VRP terminology). The service time is $dd[i][i]$. The travel time depends on the orientation with which the row will be harvested. If the row will be harvested in the direct orientation, then the travel time is equal to $dd[Location][i]$ else it is equal to $dd[i][Location]$ (see definition of cost matrix dd in equation 1).

The lower transition is constructed similarly and does not require further explanations.

Harvesting Machine automaton:

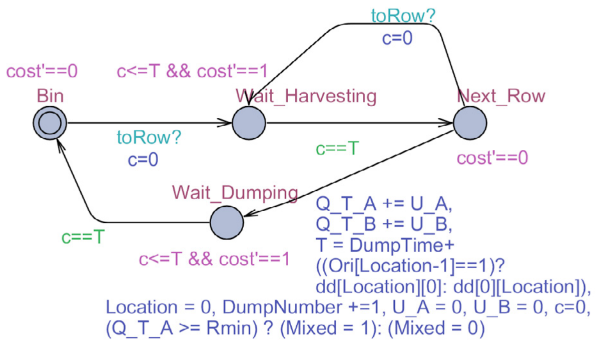


Fig. 4. The Harvesting Machine (HM) automaton

In figure 4, at init time, the starting location for the harvesting machine is the bin (state *"Bin"*). Once the HM automaton receives a synchronisation signal "toRow?" from a row automaton, the local clock c is reset and the next state is *"Wait_Harvesting"*.

At the state *"Wait_Harvesting"*, HM waits for T time units (time for travelling from location to row and time for harvesting the row). Once this time has elapsed, the HM automaton changes to state *"Next_Row"*.

At this point, HM is at the end of a row. Two cases are possible. The first case is that a synchronisation signal "toRow?" is received from a row automaton. In this case, the transition from the *"Next_Row"* to the state *"Wait_Harvesting"* will be executed. The second case is about going to the bin for dumping the hoppers. In this case, the transition from the state *"Next_Row"* to the state *"Wait_Dumping"* will be executed and the HM automaton will update the variables Q_T_A, Q_T_B, $Mixed$, $Location$, $DumpNumber$, U_A, U_B, c and T.

As for Q_T_A (resp. Q_T_B), the quantities of grapes in a-hopper U_A (resp. b-hopper U_B) are added. $Location$ is updated to 0 (the bin). $DumpNumber$ is incremented. U_A, U_B and c are reset. If $Rmin$ is reached ($Q\_T\_A >= Rmin$), $Mixed$ is set to 1, else it is set to 0 (continue differential harvest).

On this transition, T represents the travel time from the row to the bin and the dump time ($DumpTime$). The travel time depends on the orientation with which the previous row ($Location - 1$) was harvested. If it was harvested according to direct orientation ($Ori[Location - 1] == 1$) then the travel time is $dd[Location][0]$, otherwise it is $dd[0][Location]$.

At state *"Wait_Dumping"*, the HM automaton waits for T time units (time for travelling from location to bin and time for dumping hoppers). When this time has elapsed, the HM automaton passes to state *"Bin"*.

3.2 Properties

Many properties can be checked on this model to verify its consistency and its behaviour:

(1) All rows can be harvested. $E <> Row1.Harvested$ and $Row2.Harvested$ and ... $Rown.Harvested$
(2) Max bound of variables:
   - The number of dumps is always lower than number of rows. $A[\,]\ DumpNumber < nbRow + 1$
   - The quantity in hoppers is always lower than their maximum capacity of hoppers $A[\,]\ U\_A \leq CmaxHopper$ and $U\_B \leq CmaxHopper)$
(3) All rows can be harvested and at least Q_T_A is greater than $Rmin$ and HM starts and finishes at the bin.
   $E <> Row1.Harvested$ and $Row2.Harvested$ and ... $Rown.Harvested$ and $HM.Bin$ and $Q\_T\_A > Rmin$

The property that allows to solve the DHP is property 3.

4. RESULTS & DISCUSSION

In this section, we discuss some experimental results: we study the performance of DHP_PTA model for different instances from 7 to 12 rows extracted from a real vine field. The vineyard is located in southern France, in the vicinity of Gruissan.

The characteristics of the plot and the HM are as follows: the inter-row in the field is equal to 2.5 m, all rows are almost equal in length, which is roughly 189 m. HM speed during harvest in the rows is supposed to be 4 $km.h^{-1}$ and 9 $km.h^{-1}$ while moving otherwise in the field. Data were collected for $n_{max} = 12$ rows. The bin is supposed located near N_2. When instances have $n < n_{max}$ rows, the n first rows are considered.

In the following, we apply the query described as property 3 to DHP_PTA model with UppAal-CORA set to Random Best Depth First search. The C1 code means that $CmaxHopper$ is set to 1000 (unit: litre) and C2 means that $CmaxHopper$ is set to 2000. The code R1 (resp. R2) means that $Rmin$ is set to 50 % (resp. 70 %) of the total amount of A-grapes in the field.

In order to study empirically the complexity of solving DHP with DHP_PTA, the query was first applied without using the *remaining* feature of UppAal-CORA explained hereafter.

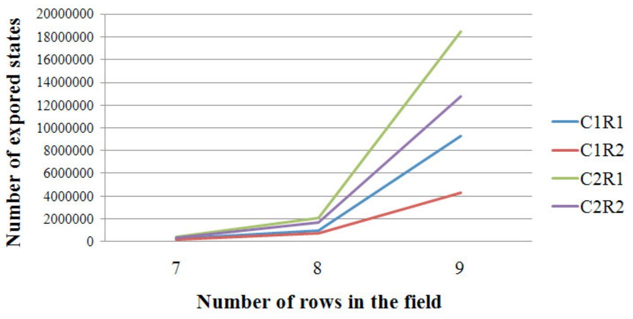


Fig. 5. Quantitative analysis of number of explored states

The high raise of practical complexity with the number of rows that can be observed in figure 5 was expected. The complexity of exploring which sequence over m visits is best, without the constraints of capacity, basically grows like $m!$. The DHP has some similarities with the VRP, which is known to be NP-hard. Besides, model checking is also well known to be exposed to combinatorial explosion. It can be noticed that curves C2R1 and C2R2 are above those with code C1. C2 denotes a large hopper capacity, which means that, more often than with C1, HM has the choice between going to the bin or harvesting another row. Because with this experiment, there is no filtering during optimisation according to a remaining cost lower bound, this increases the number of states to explore compared to C1. For same capacity, the curve with code R1 is higher than the one with code R2. This might result from the fact that once $Rmin$ A-grapes have been harvested, HM can switch to non differential harvest.

The performance of UppAal CORA can be improved by computing in each location an estimated lower bound of the remaining cost (*remaining* variable). This information is usually used in branch and bound algorithm to delete states and to search for promising ones (Behrmann et al. (2001b)). Furthermore, it can reduce the size of the explored states in a spectacular way.

For DHP_PTA model and at each location, the *remaining* variable was computed using the following formula:

$$remaining = ServiceTime + nEmptying * (DumpTime + 2 * minDBin) + (nbRowNotHarvested - nEmptying) * dmin$$

$ServiceTime$ is the service time needed to harvest the not yet harvested rows. $nEmptying$ is the lower bound for the number of hopper dumps still to be made. $minDBin$ is the smallest distance between the bin and a not harvested row. $nbRowNotHarvested$ is the number of not yet harvested rows. $dmin$ is the smallest distance between two not yet harvested rows.

The details about handling special cases are not given here.

Table 1 shows result for solving DHP, including the management of remaining cost, for a set of model instances (defined by nRCxRy, with n the number of rows in the field, Cx being the hopper capacity, and Ry being the Rmin value).

The following information is given for each instance: number of explored states ($States$), total cost which is the harvesting and travel time in seconds ($Cost$), execution time of the query in UppAal CORA in seconds (TE), and maximum memory size needed for the verification in KBytes ($Max\ RSS$, which means max Resident Set Size). Max RSS represents the pic in RAM memory usage and is an indicator of practical constraints to solve the problem on a computer with limited RAM. The meaning for C1, C2, R1, and R2 is as above.

Table 1. Experimental results using remaining

Instance	States	Cost	TE (s)	RSS (KB)
9RC1R1	974467	827	6.76	123252
9RC1R2	449599	827	3.48	60148
9RC2R1	2591458	809	17.45	305068
9RC2R2	1918631	809	13.30	220080
10RC1R1	181584	844	1.86	35916
10RC1R2	154027	844	1.63	30212
10RC2R1	800879	825	8.63	147708
10RC2R2	793346	825	8.43	144460
11RC1R1	29861147	990	203.31	3818508
11RC1R2	7659725	990	69.33	994700
11RC2R1	29557772	967	255.38	4105804
11RC2R2	30626152	967	266.12	4055288
12RC1R1	1598213	1007	19.18	297832
12RC1R2	1639723	1007	20.11	300416
12RC2R1	12823061	983	157.40	2429152
12RC2R2	12583816	983	156.28	2370280

Compared to the results without handling the remaining cost, the number of explored states is greatly reduced, with a factor of 5 for 7C2R2 and a factor of 117 for 8C1R1.

It can be observed that the number of explored states is higher for an odd number of rows. This results from limitations of currently implemented remaining cost estimator. Indeed, it is based on the lowest distance from row border to bin for dumping grapes. In the case of odd number of rows, one such dumping requires a much longer travel time, and the estimator is less efficient than for even number of rows cases. In Briot et al. (2015b), no results for instances with odd number of rows are provided.

Instances with high hopper capacity appear to be more difficult to solve than instances with lower hopper capacity. This behaviour is different of what was observed in Briot et al. (2015b) with the "Precedence" Model in a Constraint Programming approach. Indeed, the number of explored states and Time Execution increase considerably from C1 to C2 cases for the same values of n and $Rmin$. Yet, the performances of the UppAal-CORA based approach presented here compare favourably with Briot et al. (2015b). For the instance 12RC1R2, the UppAal-CORA query for DHP was solved in 20.11 s on an Intel(R) Xeon(R) CPU E5-2667 3.20 GHz processor. For same instance, the execution time reported in Briot et al. (2015b) was 494 s on an Intel(R) Xeon(R) CPU E5-2697 2.60 GHz processor. Memory requirements were not provided in Briot et al. (2015b).

Besides performance, the interest of our approach is that the same model allows for verification of several properties (see 3.2 section).

The UppAal-CORA model-checking tool currently presents some limitations, as the available binary has been built only for a 32 bit architecture thus limiting memory usage to 4GB of RAM. This is the reason why it was not possible to solve instances with more than 12 rows. Please note that the model may be optimised regarding memory by removing Mix and Q_T_B variables

5. CONCLUSION AND FUTURE WORK

In this paper, we investigated how to solve the Differential Harvest Problem (DHP) using the Cost Optimal Reachability Analysis feature implemented in the model-checking tool UppAal-CORA. We have designed the DHP_PTA model, based on the priced timed automata formalism and the UppAal-CORA tool. All model details and the property that allows to solve the DHP were explained.

The proposed approach offers good performances when compared to previous work using Constraint Programming techniques on same problem and offers faster execution time. The practical importance of using a tight enough lower bound of remaining cost in UppAal-CORA models was underlined by our experiments.

The model might be extended in order to cover multiple possible locations for the bins, which feature might be of practical interest. Other optimization criteria might be considered, like fuel consumption or length of time that harvested grapes stay in the bin before bin is entirely filled and transported to cave, in order to optimise quality.

The DHP_PTA model should be adaptable to other precision agriculture problems involving routing such as differential harvest for other crops or precision spreading. Similar models may also be of interest for optimising robotic operations in warehouses.

ACKNOWLEDGEMENTS

We thank the authors of Briot et al. (2015a) for sharing vineyard data.

REFERENCES

Alur, R. and Dill, D.L. (1994). A theory of timed automata. *Theoretical Computer Science*, 126, 183–235.

Alur, R., La Torre, S., and Pappas, G.J. (2001). Optimal paths in weighted timed automata. In *International Workshop on Hybrid Systems: Computation and Control*, 49–62. Springer.

Behrmann, G., Fehnker, A., Hune, T., Larsen, K., Pettersson, P., Romijn, J., and Vaandrager, F. (2001a). Minimum-cost reachability for priced timed automata. In *International Workshop on Hybrid Systems: Computation and Control*, volume 1, 147–161. Springer.

Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Petterson, P., and Romijn, J. (2001b). Guiding and cost-optimality in uppaal. In *AAAI-Spring Symposium on Model-based Validation of Intelligence*, 66–74.

Behrmann, G., Larsen, K.G., and Rasmussen, J.I. (2004). Priced timed automata: Algorithms and applications. In *FMCO*, volume 3657, 162–182. Springer.

Bengtsson, J. and Yi, W. (2004). *Timed Automata: Semantics, Algorithms and Tools*, 87–124. Springer, Berlin, Heidelberg.

Bisgaard, M., Gerhardt, D., Hermanns, H., Krčál, J., Nies, G., and Stenger, M. (2016). Battery-aware scheduling in low orbit: the gomx–3 case. In *FM 2016: Formal Methods: 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings 21*, 559–576. Springer.

Briot, N., Bessiere, C., Tisseyre, B., and Vismara, P. (2015a). Integration of operational constraints to optimize differential harvest in viticulture. In *Proc. 10th European Conference on Precision Agriculture (ECPA 2015)*, 487–494.

Briot, N., Bessiere, C., and Vismara, P. (2015b). A constraint-based approach to the differential harvest problem. In *Proc. 21st International Conference on Principles and Practice of Constraint Programming (CP 2015)*, volume 9255 of *Lecture Notes in Computer Science*, 541–556. Springer Berlin / Heidelberg.

Briot, N., Bessiere, C., and Vismara, P. (2015c). Programmation par contraintes pour la vendange sélective. In *Actes des Onzimes Journées Francophones de Programmation par Contraintes (JFPC 2015)*, 51–56.

Dechter, R. (2003). *Constraint processing*. Morgan Kaufmann.

Fahrenberg, U., Larsen, K.G., and Legay, A. (2013). Model-based verification, optimization, synthesis and performance evaluation of real-time systems. In *Unifying Theories of Programming and Formal Engineering Methods*, 67–108. Springer.

Kilby, P. and Shaw, P. (2006). Chapter 23 - vehicle routing. In F. Rossi, P. van Beek, and T. Walsh (eds.), *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*, 801 – 836. Elsevier.

Largouët, C., Cordier, M.O., Bozec, Y.M., Zhao, Y., and Fontenelle, G. (2012). Use of timed automata and model-checking to explore scenarios on ecosystem models. *Environmental Modelling & Software*, 30, 123–138.

McBratney, A., Whelan, B., Ancev, T., and Bouma, J. (2005). Future directions of precision agriculture. *Precision agriculture*, 6(1), 7–23.

Saddem, R., Naud, O., Cazenave, P., Godary-Dejean, K., and Crestani, D. (2017). Precision spraying: from map to sprayer control using model-checking. *Journal of Agricultural Informatics*, 8(3), 1–10.