



Decision support for grape harvesting at a South African winery

A van der Merwe* FE van Dyk† JH van Vuuren‡

Received: 26 July 2011; Revised: 4 October 2011; Accepted: 19 October 2011

Abstract

Recent technological advances have had a major impact on the management of traditional wineries, giving rise to the prospect of computerised decision support with respect to a range of complex harvesting and wine making decisions which have to be taken routinely. In this paper, two nested scheduling problems are considered. The first, referred to as the *active cellar scheduling problem*, is concerned with making good scheduling decisions within a winery (*i.e.* optimal assignments of grape intake batches to different processor sets inside the active part of the cellar). The *harvest scheduling problem*, on the other hand, refers to the larger, over-arching problem of selecting the best possible dates on which to harvest the respective vineyard blocks in order to preserve grape quality. A nested tabu search approach is presented to solve these two scheduling problems simultaneously. This solution approach has been implemented as a computerised decision support tool, called VINDSS, and the practical workability of this tool is demonstrated by means of a special case study at a winery in the South African Western Cape.

Key words: Grape harvesting, winery, scheduling, decision support, metaheuristics.

1 Introduction

Automated assistance in the form of decision support systems has become a very popular research topic in the wine industry [3, 5, 7, 8]. In fact, the wealth of interesting and challenging operational research and logistical problems occurring at wineries have led to the founding of the *Wine Supply Chain Council (WSCC)* [1]. This international wine industry research network attempts to collaborate on issues in global wine supply chains. WSCC member, Simon Dunstall has been working with Australian Orlando Wyndham Group since 2003 in order to achieve the shared supply network goal of maximizing the value that is realised from material and intellectual assets in the supply network [3]. Furthermore, Chilean member Sergio Maturana has been involved in a project where

*Department of Logistics, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa.

†CSIR Built Environment, PO Box 320, Stellenbosch, 7599, South Africa.

‡Corresponding author: (**Fellow of the Operations Research Society of South Africa**), Department of Logistics, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa, email: vuuren@sun.ac.za

certain crucial factors of the wine making process, such as the reception and pressing capacities of a cellar, were studied by simulating the reception of grapes at the cellar [2]. Maturana, together with WSCC member Alejandro MacCawley, has also been involved in the design of a practical tool for the scheduling of wine grape harvesting operations which adopts an optimization approach, taking into account both operational costs and grape quality [4].

2 Problem description

At South African producer cellars, grapes are typically received from a number of different suppliers. The vineyards of these suppliers are usually divided into *vineyard blocks*, grouped according to their cultivar and quality. During the harvesting period, roughly spanning the period January to April, grape samples are required from the suppliers in order to perform sugar, pH and acidity analyses. Based on the results of these analyses, the viticulturist and winemakers, along with the cellar manager, in most cases sort manually through large volumes of data in order to agree on a suitable harvesting block selection strategy for each day. During the height of the harvesting season this volume of data can often be overwhelming, hence giving rise to the need for some kind of automated decision support on a daily basis, suggesting a shortlist of vineyard blocks to be harvested from which the viticulturist, winemakers and cellar manager may finalise a harvesting block selection strategy for each day.

The main criterion for the selection of vineyard blocks is that the blocks selected should contain grapes that are fully ripened. Furthermore, there should be sufficient capacity at the cellar to receive and process these grapes. Two nested scheduling problems therefore arise naturally. The first is the over-arching problem of scheduling vineyard blocks for harvesting on a daily basis over a set period of time and is referred to as the *harvest scheduling problem*. The second, smaller problem, which we call the *active cellar scheduling problem*, involves the scheduling of grape loads being processed on the different types of machinery inside the cellar during a particular day, given a vineyard block harvesting schedule, and may be used as a means of evaluating the feasibility of the larger harvesting schedule (since it should be possible to accommodate all grapes harvested during any particular day on that same day at the cellar).

3 Outline of modelling approach

In this paper we report on the design and implementation of a nested tabu search approach towards solving the two scheduling problems described above. Solving the over-arching harvest scheduling problem approximately by means of Glover's tabu search methodology [6] requires knowledge of the ongoing processes and tank availability inside the active cellar in order to test for feasibility of a given harvest schedule. This process of testing for feasibility is achieved by performing an inner active cellar scheduling tabu search for the relevant harvesting schedule during each iteration of the outer harvest scheduling tabu search. An iteration of the over-arching harvest scheduling tabu search may be divided into three phases, as shown in Figure 1.

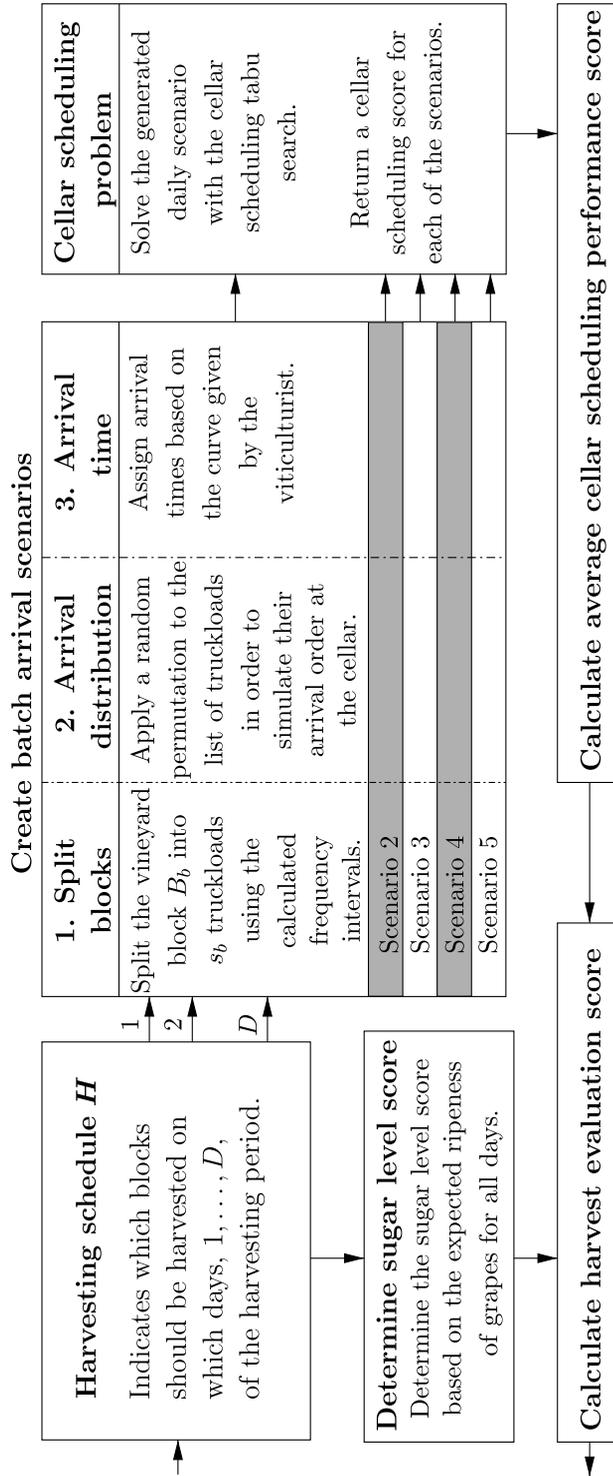


Figure 1: The three phases of an iteration of the over-arching harvest scheduling problem tabu search: (1) The generation of a daily vineyard block harvest selection strategy, (2) the stochastic generation of a number of daily grape truckload or batch arrival scenarios, and (3) solving the active cellar scheduling problem approximately via the inner tabu search.

During the first phase, a new daily vineyard block harvest selection strategy is generated from the entire list $\mathcal{B} = \{B_1, \dots, B_N\}$ of vineyard blocks to be harvested during the scheduling window. This is done by performing an outer harvesting schedule tabu search move on the daily vineyard block harvest selection evaluated during the previous tabu search iteration. The harvest schedule is represented by means of a list, \mathbf{H} , of vineyard block lists to be harvested each day. In particular, the d -th entry of \mathbf{H} , denoted by $\mathbf{H}(d)$, is itself an *unordered* list of vineyard blocks to be harvested (in some order) during day d of the harvesting season.

During the second phase, a number of daily grape truckload or arrival batch scenarios are generated stochastically as input to the active cellar scheduling problem. For each of these scenarios, the active cellar scheduling problem is solved approximately during the third phase by means of an inner tabu search in order to determine an active cellar performance score for each of the scenarios. These performance scores are then used to evaluate the quality of the vineyard block harvest selection.

4 The inner active cellar tabu search

Before the inner active cellar scheduling tabu search may be described, a minimal amount of terminology and background information is required.

4.1 Cellar jobs and their characteristics

The j -th arrival batch of grapes within the active cellar is referred to as a *job*, denoted by J_j . Job J_j is further divided into k_j tasks, $T_{j,1}, \dots, T_{j,k_j}$, according to the different phases of grape processing and specific machine requirements within the cellar.

There are five types of tasks (shown below in bold face) which may be performed on any one of a specified subset of processors (shown in italics):

1. **Crushing and destemming** of grapes takes place at *tipping bins* immediately upon arrival in the cellar.
2. **Separation** of juice from the skins of white grapes may then be performed on *separators* or *presses*.
3. Thereafter, **pressing** of the skins of white grapes is performed on *presses*. The pressing of red grapes only occurs after fermentation.
4. **Fermentation** of red grapes takes place in *fermentation tanks*.
5. *Fermentation tanks* have to be **drained** before red grapes are sent to the presses.

Jobs involving red grapes are henceforth referred to as TYPE I jobs, while TYPE II jobs refer to jobs involving white grapes. This distinction is necessary, because processors typically have to be cleaned before processing white grapes directly after having processed red grapes. Since red grape fermentation typically requires several days, while the process of active cellar scheduling typically has a time horizon of one day, a further job type distinction is required: TYPE III jobs refer to red grape juice and skins that require pressing after primary fermentation.

4.2 Generating an initial solution

Generating a sensible initial solution based on problem-specific characteristics may have a significant influence on the success of a tabu search. Therefore, a considerable amount of effort went into the generation of such an initial solution for the inner tabu search.

In particular, jobs were considered in their order of estimated arrival times at the cellar. Although Type III jobs do not actually *arrive* at the cellar, they are nevertheless assigned appropriate starting times since it may take a number of hours to empty a fermentation tank and to transfer the grapes to a press from a fermentation tank. Jobs are assigned to the tipping bins (points of entry into the active cellar) according to the number of Type I or Type II jobs received. The list of jobs, ordered according to their estimated arrival times, is traversed and jobs are assigned to available tipping bins in that order. After this initial assignment, the ordered list is again traversed. The next set of assignments to processors in the cellar is applied, depending on the job types and their allowable subsequent processors. The list is repeatedly traversed in this way until all jobs have been assigned to all the required processors.

4.3 Evaluating the quality of a candidate solution

An active cellar scheduling candidate solution should ideally be evaluated on two levels. The first concerns the processing time required to apply the particular schedule and whether or not the schedule will fit into one cellar working day after minimizing the makespan of the schedule. This is achieved in approximate fashion by applying a so-called *cellar packing algorithm*. The algorithm does not take into account the assignments to the red fermentation tanks, since such assignments have no influence on the daily active cellar schedule makespan, as explained above. A further evaluation of the scheduling candidate solution is therefore necessary, specifically focusing on the efficiency and effectiveness of assignments to the red fermentation tanks.

4.3.1 The cellar packing algorithm

The goal of the cellar packing algorithm is to evaluate the feasibility of a candidate solution to the cellar scheduling problem by determining bounds on its job completion times. Of particular importance is the daily *completion time*, which refers to the completion time of the last task on any processor other than a red fermentation tank. For the active cellar, all jobs that do not end at the fermentation tanks, are assumed to end at the presses. Therefore, the completion time is, in fact, the termination time of the last task performed on a press for the day under consideration.

The cellar packing algorithm is applied in three steps:

1. All tasks are packed in the correct order next to their assigned processors when adopting a Gantt-chart view. The only constraint is that the processing times and setup times should be respected.
2. The starting and ending times at the tipping bins are updated according to their relevant arrival times.

3. The starting time of any task, T_{jk} , is taken to be at least as large as the ending time of its predecessor task, $T_{j(k-1)}$, also ensuring that a task being performed on parallel machines, has the same starting and ending times on all the relevant machines. This step is referred to as a *forward shift*.

A Gantt-chart representation of the candidate solution to the active scheduling problem obtained by applying the cellar packing algorithm to a small hypothetical cellar consisting of three tipping bins P_1, P_2, P_3 , three separators P_4, P_5, P_6 , three presses P_7, P_8, P_9 and three red fermentation tanks P_{10}, P_{11}, P_{12} , is shown in Figure 2 for the seven jobs described in Table 1. An upper bound on the completion time for these seven jobs is 14.2 hours. The striped blocks in the figure refer to times assigned unnecessarily, including instances where a task has to be shifted forward in order for the following task of the same job to follow on it directly. For example, task $T_{4,1}$ ends at 2.25 hours. However the following task, $T_{4,2}$, only starts at 3.7 hours. Therefore, the ending time of task $T_{4,1}$ should, in fact, be 3.7 and the starting time 3.45 hours. If the schedule were to be applied in practice, these shortcomings could have been rectified by applying a *backwards shift* in the obvious manner, but since only the completion time of all jobs is of importance here (in view of the inner tabu search being applied as a feasibility test for candidate solutions to the outer harvest scheduling problem), the backwards shift is not actually applied by the cellar packing algorithm.

| Job | Cultivar | Quality class | Total mass | Task | Allowed P_i | Processing time |
|-----|--------------------|---------------|------------|------|--------------------------|-----------------|
| 1 | Cabernet Sauvignon | 1 | 40 | 1 | P_1, P_2, P_3 | 0.2 |
| | | | | 2 | P_{10}, P_{11}, P_{12} | 144 |
| 2 | Cabernet Sauvignon | 1 | 25 | 1 | P_1, P_2, P_3 | 0.25 |
| | | | | 2 | P_{10}, P_{11}, P_{12} | 144 |
| 3 | Pinotage | 2 | 15 | 1 | P_1, P_2, P_3 | 0.2 |
| | | | | 2 | P_{10}, P_{11}, P_{12} | 120 |
| 4 | Chardonnay | 1 | 40 | 1 | P_1, P_2, P_3 | 0.25 |
| | | | | 2 | P_4, P_5, P_6 | 3.0 |
| | | | | 3 | P_7, P_8, P_9 | 2.5 |
| 5 | Chardonnay | 1 | 30 | 1 | P_1, P_2, P_3 | 0.2 |
| | | | | 2 | P_4, P_5, P_6 | 3.0 |
| | | | | 3 | P_7, P_8, P_9 | 2.5 |
| 6 | Sauvignon Blanc | 2 | 20 | 1 | P_1, P_2, P_3 | 0.2 |
| | | | | 2 | P_4, P_5, P_6 | 1.75 |
| | | | | | P_7, P_8, P_9 | 0* |
| 7 | Merlot | 3 | 50 | 1 | P_{10}, P_{11}, P_{12} | 5 |
| | | | | 2 | P_7, P_8, P_9 | 3 |

Table 1: Allowable processors and corresponding processing times (measured in hours) for a set of seven jobs in a hypothetical cellar. Job masses are specified in tonnes (entire vineyard blocks). *Chardonnay grapes require some time to lie with their skins before separation and therefore the Chardonnay grapes can only be separated in a separator, whereas Sauvignon Blanc grapes may be separated in a press or separator.

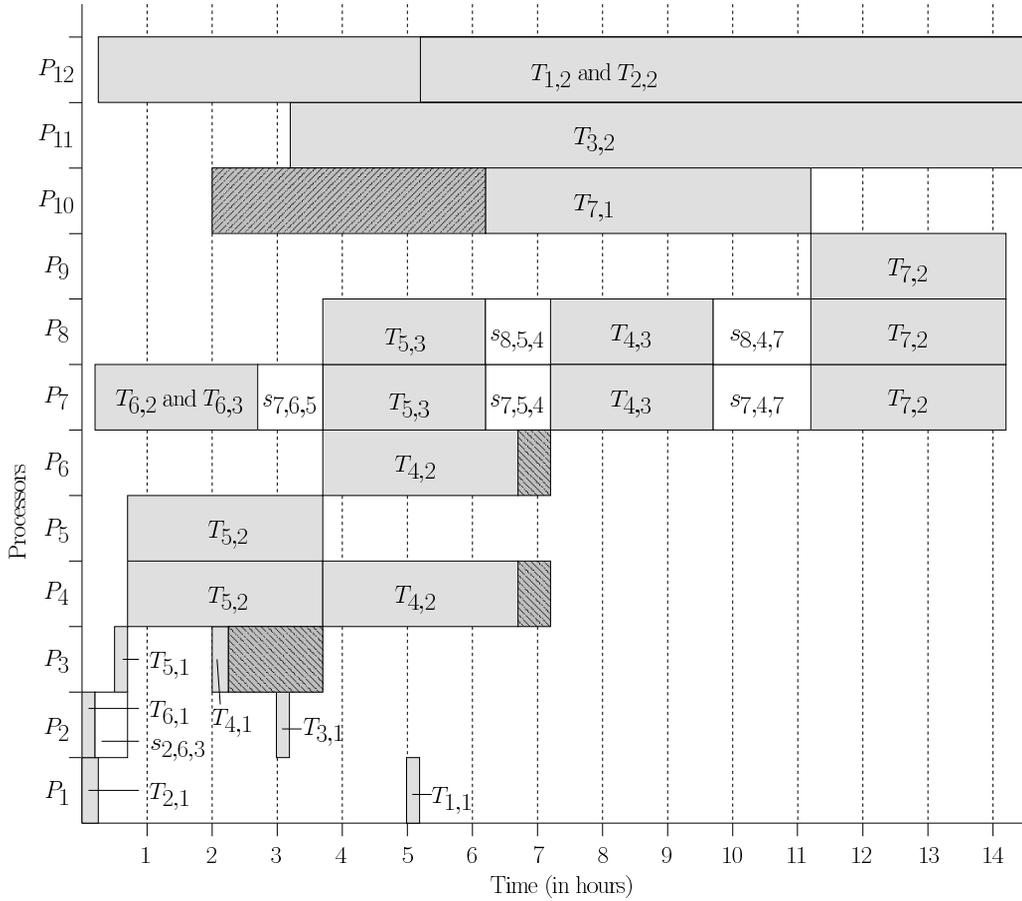


Figure 2: Gantt-chart representation of the candidate solution to the active scheduling problem obtained by applying the cellular packing algorithm to the initial solution for a small hypothetical cellar and for the seven jobs in Table 1, resulting in an upper bound of 14.2 hours on the completion time. The setup time required on processor P_i between jobs J_{j_1} and J_{j_2} is denoted by s_{i,j_1,j_2} .

4.3.2 Red fermentation tank effectiveness-of-use evaluation

When assigning jobs to the red fermentation tanks, the order in which these jobs are assigned is not of concern. The grapes of jobs assigned to the same tank are mixed and do not follow one another; however, only grapes of the same cultivar and quality are allowed to be mixed and this constraint is included when moves are generated for the inner tabu search.

Two criteria are of importance when measuring the quality of a candidate solution to the active cellar scheduling problem. The first is to minimise the total mass of unassigned jobs, thereby ensuring that a superior solution is one in which as many as possible of the jobs that arrive at the cellar are processed on the desired day. This is the most important of the two evaluation criteria. The other criterion involves minimising wasted space, *i.e.* the sum of the volumes that are still available in each of the assigned tanks. From these two criteria a red fermentation tank effectiveness-of-use score is calculated.

The final evaluation score of a candidate solution to the active cellar scheduling problem

is expressed as a weighted sum of the red fermentation tank effectiveness-of-use score and the upper bound on the makespan determined by the cellar packing algorithm.

4.4 Inner tabu search move generation and selection

Each set of processors in the active cellar exhibits different characteristics and requirements in terms of scheduling. These processor sets therefore require different rules guiding the generation of a list of inner tabu search candidate moves. Some also require different criteria for the selection of the best candidate move. Five different move types (shown in bold face) are considered for each of the processor sets (shown in italics) in our implementation of the inner tabu search:

1. **Move Type A** is applied as diversification move to the assignment and order of jobs processed in the *tipping bins*. Each job of Type I or II is assigned to a tipping bin exactly once, regardless of its mass.
2. **Move Type B** is applied to the assignment and order of jobs processed on the *separators*.
3. **Move Type C** is applied to the assignment, number of assignments and order of jobs processed on the *presses*.
4. **Move Type D** is concerned with the assignment of a task of a Type II job which may be assigned to *either a separator or a press* (achieved by means of a boolean variable indicator).
5. **Move Type E** is employed to assign Type I jobs to the *red fermentation tanks*. Jobs assigned to the same tank are mixed and do not follow one another as with any of the previously mentioned move types. Therefore, the relevant processor availability and starting volumes should be kept in mind.

All the move types described above, save for a move of Type D where binary switches are considered, employ ejection chain switches applied to the order and assignment of tasks currently assigned to the processor sets [6]. The most significant difference between the ejection chains for the various processor sets lies in the method of testing for move feasibility, since different requirements have to be met for a solution to be feasible with respect to each job type and processor set combination.

4.5 The inner tabu search implementation

A pseudo-code listing for a tabu search customised for the active cellar scheduling problem is outlined in Algorithm 1. Once either of the stopping criteria in lines 23 or 24 is satisfied, the move types that may be used in the tabu search are updated accordingly.

A variable α is used to store the shortest makespan achieved during the tabu search, while ϵ denotes the best red fermentation tank effectiveness-of-use evaluation score. The search may terminate once both the makespan goal α_{\max} and the red fermentation tank assignment effectiveness-of-use goal ϵ_{\max} have been achieved. If neither of these stopping criteria is satisfied within *counterMaximum* iterations, the tabu search is nevertheless terminated and the current best scores, α and ϵ , are considered approximately optimal objective function values for the input arrival batch scenario.

Algorithm 1: Inner active cellular scheduling problem tabu search.

Input: An arrival batch scenario for which to solve the active cellular scheduling problem.
Output: A feasible active cellular schedule for processing the various job tasks involved in the arrival batch.

- 1 Generate an initial solution;
- 2 Evaluate the initial solution and set aspiration criteria, a_α and a_ϵ , to the evaluation scores, α and ϵ , respectively;
- 3 **counter** \leftarrow 0;
- 4 **unchanged** \leftarrow 0;
- 5 **solutionFound** \leftarrow false;
- 6 **firstType** \leftarrow B;
- 7 **lastType** \leftarrow E;
- 8 **previousType** \leftarrow -1;
- 9 **while** **solutionFound** = false **and** **counter** \leq **counterMaximum** **do**
- 10 Set **moveType** to a randomly generated move type between **firstType** and **lastType**, not equal to **previousType**;
- 11 **if** $a_\alpha = \alpha$ **then**
- 12 **unchanged** \leftarrow **unchanged** + 1;
- 13 **if** **unchanged** \geq 10 **then**
- 14 **moveType** \leftarrow A;
- 15 **unchanged** \leftarrow 0;
- 16 **end**
- 17 **else**
- 18 $a_\alpha \leftarrow \alpha$;
- 19 **unchanged** \leftarrow 0;
- 20 **end**
- 21 **end**
- 22 **previousType** \leftarrow **moveType**;
- 23 Apply the selected move type, **moveType**;
- 24 **if** $\epsilon \leq \epsilon_{max}$ **then**
- 25 **lastType** \leftarrow D;
- 26 **end**
- 27 **if** (**firstType** = E) **and** (**lastType** = D) **then**
- 28 **solutionFound** \leftarrow true;
- 29 **end**
- 30 **counter** \leftarrow **counter** + 1;
- 31 **end**

Whenever the number of iterations during which the value of α remains unchanged reaches 10, a move of Type A replaces the randomly generated move in order to introduce some measure of diversity into the search process.

In order to prevent the search from repeatedly considering the same solution sets, tabu lists are maintained for each move type. Rather than adding the inverse of a move (which is a complex move in this case) to such a list, the partial solution attributes (*i.e.* the solution matrix part referring to the specific processor set) obtained by applying the move is added to the list. These previously visited partial solutions are avoided for a pre-determined number of iterations, unless such a move results in an improvement of the aspiration criterion (which is, of course, determined by considering the full solution matrix over all processors).

5 The generation of arrival batch scenarios

Each arrival batch scenario generated for the purposes of solving the active scheduling problem (approximately) relies on three attributes:

1. *The number of batches (truckloads) into which a vineyard block is subdivided when its grapes are delivered to the cellar.* Previous harvesting data are considered to form a general distribution for these arrival batches in terms of the frequencies with which a specific vineyard block size is subdivided into the different batch sizes considered.
2. *The order in which these batches arrive at the cellar.* A random permutation is applied to the list of arrival batches over all vineyard blocks selected for harvesting on a particular day.
3. *The distribution of batch arrival times.* A distribution of the arrival times of loads of grapes at the cellar over time should be discussed with the viticulturist of the winery, since this may vary between different South African producer cellars. This distribution is employed to assign an estimated arrival time to each of the jobs in the scenario.

For the sake of simplicity, the arrivals of grapes from one vineyard block are assumed to occur in equally sized batches. Although this is not necessarily the case in practice, this simplifying assumption is not expected to impact significantly on the accuracy of the model. Processors inside the cellar typically operate according to specified cycles (*i.e.* the various wine making processes last for specified time durations) and the arrival batch sizes have no influence on these cycle lengths; they only have to adhere to the capacity restrictions of the various processors.

6 The outer harvest scheduling tabu search

The most significant attribute of a good harvesting schedule is that as much grapes as possible should be received at the cellar when they are close to their optimal ripeness levels. It is assumed that vineyard block sample data are maintained by the cellar's viticulturist. From these data, the most recent sugar levels of grapes in each block may be used to approximate the timing of optimal sugar levels during the current harvesting season based on previous years' harvesting data.

6.1 Generating an initial harvesting schedule

A list containing all vineyard blocks with current sugar levels that lie within an acceptable sugar level interval is generated. These vineyard blocks are then ordered in decreasing order of ripeness levels per cultivar as determined by the viticulturist, based on the sample sugar levels of the blocks.

In order to generate an initial harvesting schedule, the total mass of the expected yield of all the vineyard blocks in this list is considered. This mass is compared to the average yield tonnage per harvesting day of previous years, denoted by \bar{x} , in order to estimate the number of blocks to be scheduled for harvesting per day. For each of the first $D - 1$ days,

the total mass of the expected yield for the scheduled blocks should be at least as large as \bar{x} , with all remaining blocks scheduled for harvesting on day D , for some value D . The shorter the rolling time horizon, D , over which the harvest scheduling problem has to be solved, the more accurate the schedule is expected to be (in terms of harvesting grapes as close as possible to their optimal ripeness levels); D is typically taken as 5 days in order to generate a weekly schedule.

6.2 Evaluating the quality of a candidate solution

There are three main factors to be taken into account when evaluating the quality of a candidate harvesting schedule:

1. The (expected) ripeness of the grapes from the selected vineyard blocks on the days they are scheduled for harvesting.
2. The required daily processing time inside the active cellar associated with the harvesting schedule.
3. The lack of space or wasted space as a result of job assignment to the red fermentation tanks caused by the harvesting schedule.

It should be possible for vineyard blocks shortlisted as available for scheduling not to be selected for harvesting during the scheduling time period under consideration. However, any vineyard block not selected for harvesting during a particular harvesting day, should be eligible for harvesting during a following harvesting day in the scheduling period. Therefore, any vineyard block not selected for harvesting still appears in our harvest scheduling matrix, \mathbf{H} , but a binary variable is employed to distinguish between blocks selected for harvesting and blocks that are eligible, but which are not selected.

The first step in forming a harvesting schedule starts with the generation of the daily arrival batch scenarios for various harvesting days within the schedule. If at least one vineyard block is scheduled for harvesting during a particular day, the active cellar scheduling problem instance associated with such a scenario is solved (approximately) for that day via the inner tabu search. If the best completion time found, α , does not fit within the required business hours of the cellar, the active cellar scheduling score is severely penalized by setting α equal to $10\,000\alpha$. The reason for such a drastic penalization is that such a completion time may, in fact, cause the harvesting schedule to be infeasible. However, rather than considering the harvesting schedule infeasible and excluding it from further consideration, a very poor harvest evaluation score is assigned to it, causing it to be a highly undesirable harvesting schedule.

We split the calculation of the sugar level score into two parts, the first part concerning blocks that are available for harvesting, but are not selected for harvesting, and the second part concerning blocks that are scheduled to be harvested. If any vineyard block with grapes of a very high quality (*i.e.* very close to optimal ripeness) is not selected for harvesting, the sugar level score is penalized severely.

6.3 Outer tabu search move generation and selection

There are not as many different characteristics and requirements present within the harvest scheduling problem as there are within the cellar scheduling problem. Hence the only types of moves considered within the outer harvesting schedule tabu search are swaps. Two entries in the harvesting schedule are selected (following a set of selection rules) and their positions are exchanged, thereby applying a swap with respect to the positions of two vineyard blocks within the schedule \mathbf{H} .

The next block to which to apply a swap, is chosen to be a vineyard block with the largest contribution to the sugar level score. In order to avoid the same block being selected repeatedly, a tabu list is maintained in order to limit the frequency with which each block may be selected.

All feasible moves (swaps) are sorted in order of non-increasing harvest evaluation score. Starting at the first move in this list, each move is considered in turn until a suitable move has been found. Such a move is found either if the move is not in any tabu list or if the aspiration criterion is met. Upon applying a move, two further tabu lists are updated. The first list prevents the two vineyard blocks exchanging positions in the harvesting schedule as a result of the swap, from being exchanged again too soon thereafter. The second tabu list limits a vineyard block from moving back to its previous harvesting day in the schedule too soon after being assigned a new harvesting day.

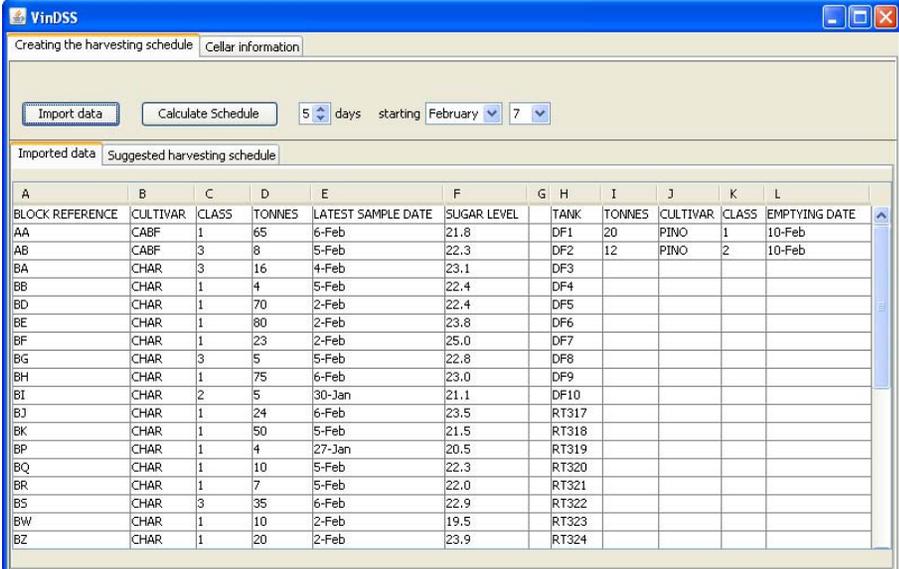
6.4 The outer tabu search implementation

A pseudo-code listing of the harvest scheduling tabu search is given in Algorithm 2. First, an initial harvesting schedule, \mathbf{H} , is generated. The harvest evaluation score ϑ of this harvesting schedule is then calculated and recorded as the best score encountered thus far during the search. A list of candidate swap moves is then generated, evaluated and the best move is applied. This process is repeated until a satisfactory harvesting schedule is found.

As explained in §4, the inner tabu search is called in order to test each of the generated batch arrival scenarios for feasibility as part of the harvest schedule evaluation.

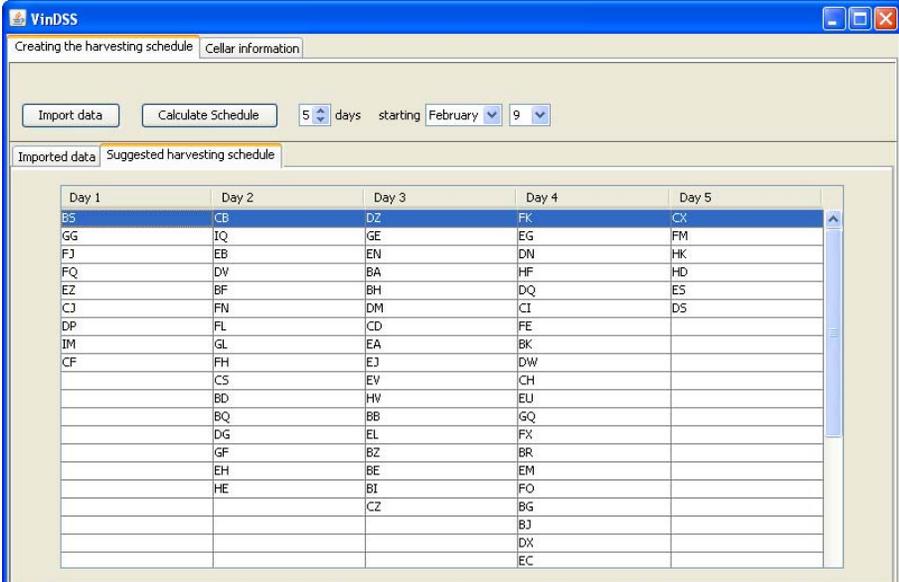
7 A decision support system, VinDSS

A simple computerised decision support system, called VINDSS, was developed in Java, employing the nested tabu search approach described above in order to solve the active cellar and harvest scheduling problems. VINDSS imports data from Excel for ease of use, and computes a harvesting schedule for the number of days specified from a fixed starting date. An example of such a set of imported data is shown in Figure 3(a). The harvesting schedule computed for the five harvesting days selected in Figure 3(a), is shown in Figure 3(b).



| A | B | C | D | E | F | G | H | I | J | K | L |
|-----------------|----------|-------|--------|--------------------|-------------|-------|--------|----------|-------|---------------|---|
| BLOCK REFERENCE | CULTIVAR | CLASS | TONNES | LATEST SAMPLE DATE | SUGAR LEVEL | TANK | TONNES | CULTIVAR | CLASS | EMPTYING DATE | |
| AA | CABF | 1 | 65 | 6-Feb | 21.8 | DF1 | 20 | PINO | 1 | 10-Feb | |
| AB | CABF | 3 | 8 | 5-Feb | 22.3 | DF2 | 12 | PINO | 2 | 10-Feb | |
| BA | CHAR | 3 | 16 | 4-Feb | 23.1 | DF3 | | | | | |
| BB | CHAR | 1 | 4 | 5-Feb | 22.4 | DF4 | | | | | |
| BD | CHAR | 1 | 70 | 2-Feb | 22.4 | DF5 | | | | | |
| BE | CHAR | 1 | 80 | 2-Feb | 23.8 | DF6 | | | | | |
| BF | CHAR | 1 | 23 | 2-Feb | 25.0 | DF7 | | | | | |
| BG | CHAR | 3 | 5 | 5-Feb | 22.8 | DF8 | | | | | |
| BH | CHAR | 1 | 75 | 6-Feb | 23.0 | DF9 | | | | | |
| BI | CHAR | 2 | 5 | 30-Jan | 21.1 | DF10 | | | | | |
| BJ | CHAR | 1 | 24 | 6-Feb | 23.5 | RT317 | | | | | |
| BK | CHAR | 1 | 50 | 5-Feb | 21.5 | RT318 | | | | | |
| BP | CHAR | 1 | 4 | 27-Jan | 20.5 | RT319 | | | | | |
| BQ | CHAR | 1 | 10 | 5-Feb | 22.3 | RT320 | | | | | |
| BR | CHAR | 1 | 7 | 5-Feb | 22.0 | RT321 | | | | | |
| BS | CHAR | 3 | 35 | 6-Feb | 22.9 | RT322 | | | | | |
| BW | CHAR | 1 | 10 | 2-Feb | 19.5 | RT323 | | | | | |
| BZ | CHAR | 1 | 20 | 2-Feb | 23.9 | RT324 | | | | | |

(a) Input data imported from Microsoft Excel.



| Day 1 | Day 2 | Day 3 | Day 4 | Day 5 |
|-------|-------|-------|-------|-------|
| BS | CB | DZ | FK | CX |
| GG | IQ | GE | EG | FM |
| FJ | EB | EN | DN | HK |
| FQ | DV | BA | HF | HD |
| EZ | BF | BH | DQ | ES |
| CJ | FN | DM | CI | DS |
| DP | FL | CD | FE | |
| IM | GL | EA | BK | |
| CF | FH | EJ | DW | |
| | CS | EV | CH | |
| | BD | HV | EU | |
| | BQ | BB | GQ | |
| | DG | EL | FX | |
| | GF | BZ | BR | |
| | EH | BE | EM | |
| | HE | BI | FO | |
| | | CZ | BG | |
| | | | BJ | |
| | | | DX | |
| | | | EC | |

(b) The resulting harvesting schedule, consisting of two-letter vineyard block names.

Figure 3: User interface screen shots of the decision support system, VINDSS.

Algorithm 2: Outer harvest scheduling tabu search.

Input: A list of vineyard blocks to be harvested together with their sugar levels, a limit, `counterMax`, on the number of tabu search iterations, and a harvesting schedule aspiration criterion, ϑ_{\max} .

Output: A harvesting schedule, H , aspiring in quality to ϑ_{\max} .

- 1 Generate an initial solution;
- 2 Determine the evaluation score, ϑ , of the initial solution;
- 3 `found` \leftarrow false;
- 4 `counter` \leftarrow 0;
- 5 **while** `found = false and counter` \leq `counterMax` **do**
- 6 Generate a list of swap moves, as explained in §6.3;
- 7 Generate batch arrival scenarios for each of the D days for each of the resulting schedules, as explained in §5;
- 8 Evaluate each resulting schedule;
- 9 Apply the best move, update all tabu lists and the evaluation score, ϑ ;
- 10 **if** $\vartheta \leq \vartheta_{\max}$ **then**
- 11 | `found` \leftarrow true;
- 12 **end**
- 13 **end**

8 Wamakersvallei winery: A case study

Wamakersvallei Winery, a South African producer cellar, agreed to participate in the evaluation of the feasibility and desirability of utilising the solution approach described in this paper with respect to harvesting decisions at the cellar. At Wamakersvallei, grapes are received from more than eighty different suppliers. During the harvesting season grape samples are received from the suppliers on a two-weekly basis per vineyard block. Sugar, pH and other acidity analyses are performed on these samples. The viticulturist, winemakers and cellar manager manually sort through the resulting analysis data by hand in order to agree on a suitable harvesting block selection strategy for each day during the harvesting season. Farmers are notified via telephone the day before they have to deliver the grapes of a particular vineyard block to the winery.

Once harvested, the scheduled loads of grapes arrive at the winery where they are weighed and their quality and origin are documented. Each load of grapes is then assigned to a tipping bin which serves as entry point to the active cellar. The wine making process already starts inside the tipping bins and, depending on the grape cultivar and type of wine desired, the various loads of grapes are then assigned to different sets of available processors. All of these processors are connected by means of permanent stainless steel pipes and/or temporary pipes that are easily moved between the processors. The interconnectivity of the cellar processors is shown in Figure 4. In this figure, the open circles are a means of representing the joining or splitting of routes that grape batches may follow in the cellar, and do not refer to any physical aspect of the cellar. Arrows represent the possible flow routes of grape batches (one arrow may, in fact, refer to more than one pipe). Different wine types require different processes performed on a specific grape type. Each of these processes satisfies unique requirements and each set of processors has a unique set of characteristics.

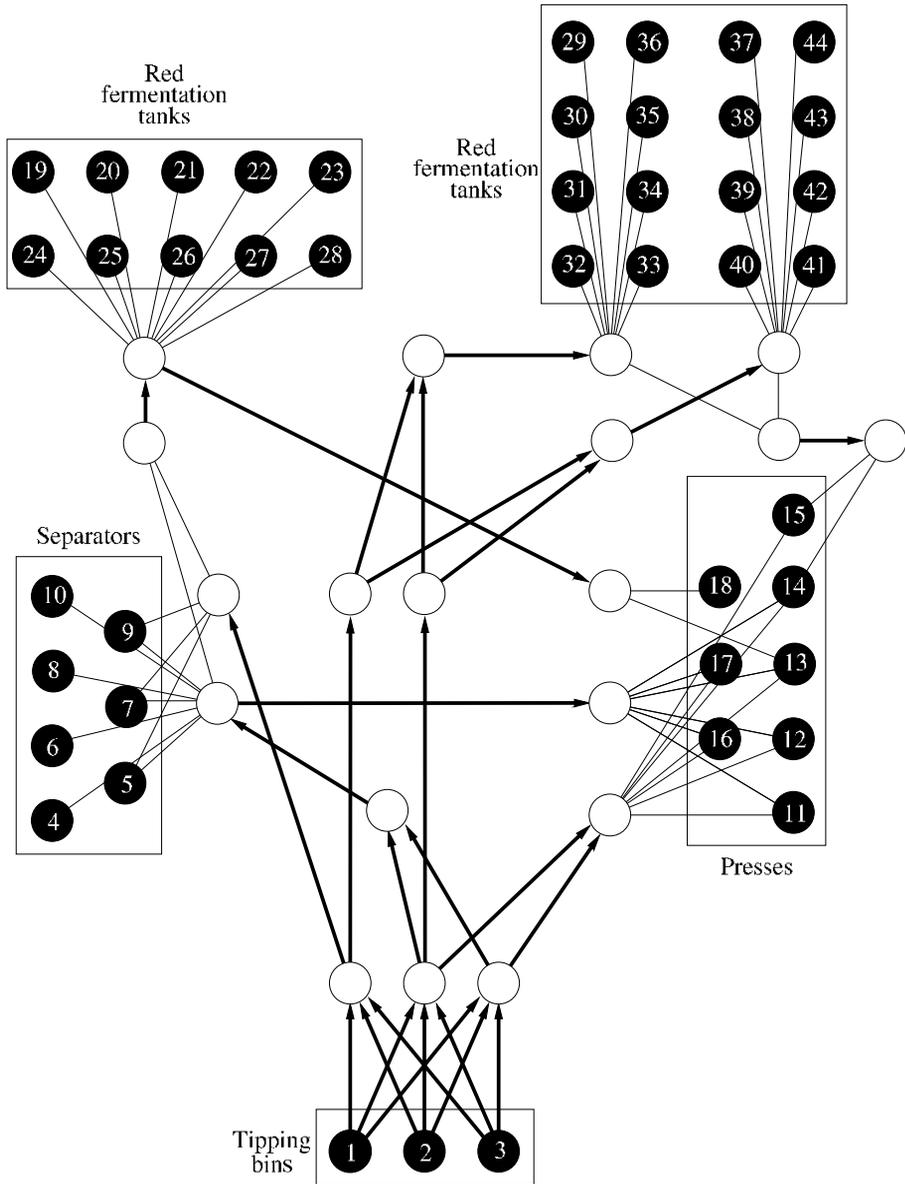


Figure 4: Interconnectivity of the various processors at Wamakersvallei Winery.

| Day | Actual 2009 | |
|----------------|--------------------|---------|
| | \mathbf{H}_{100} | harvest |
| 1 | 351 | 393 |
| 2 | 371 | 497 |
| 3 | 427 | 561 |
| 4 | 329 | 390 |
| 5 | 117 | 23 |
| Total | 1 595 | 1 865 |
| \overline{W} | 319 | 373 |
| α | 118.6 | 208.6 |

Table 2: An analysis of the mass of grapes scheduled for intake during a five-day period for the harvesting schedule \mathbf{H}_{100} and the actual 2009 harvesting schedule implemented at Wamakersvallei Winery. Here \overline{W} denotes the average mass of grapes received per day and α denotes the standard deviation of these daily masses.

Wamakersvallei Winery made their historical 2009 harvesting data available to us. After a painstaking process of piecing together a minimal amount of information from these data in order to be able to apply our tabu search approach towards finding a good hindsight harvest scheduling strategy for a five-day week during the 2009 Wamakersvallei harvesting season via VINDSS, the system required approximately two hours to execute 100 moves of the outer tabu search in order to suggest a harvesting schedule which we call \mathbf{H}_{100} .

A number of problems arose when attempting to evaluate the quality of \mathbf{H}_{100} . Firstly, most of the data received from Wamakersvallei Winery were not in electronic format, resulting in a tedious process of determining when each vineyard block was actually harvested. Secondly, different referencing names were used by the viticulturist in different sets of data when referring to the same vineyard blocks. Thirdly, the list of bi-weekly vineyard block sugar levels was incomplete and also contained a few nonsensical entries. Some vineyard blocks arrived at the cellar for which no sugar level entries were recorded. Upon investigation it was found that this might be due to the personal relationship between the Wamakersvallei team and some of the suppliers. The process of cleaning the data was therefore nontrivial and was conducted in collaboration with the viticulturist at Wamakersvallei Winery. In cases where sugar level data were missing, similar data from previous years were used instead.

The problems described above made it impossible to compare the quality of \mathbf{H}_{100} directly to that of the actual 2009 Wamakersvallei harvesting schedule in terms of the timing of vineyard block harvesting (or the closeness of grapes to optimal ripeness levels). However, the viticulturist found \mathbf{H}_{100} to be a plausible schedule, based on his harvesting experience. Moreover, it was possible to compare the hypothetical daily masses of grape intakes at the winery had \mathbf{H}_{100} been adopted rather than the actual 2009 harvesting schedule. This comparison is presented in Table 2.

Two aspects of the values in Table 2 stand out:

- A very small volume of grapes was received at the winery during the last day of

harvesting. On closer inspection, it was found that the 23 tonnes of grapes harvested on day 5 were all from vineyard blocks where harvesting had started the previous day, but was not completed. There exists a variety of reasons for starting the process of harvesting a specific vineyard block and not completing the process on the same day. However, the most common reason is that there is no space or time to receive grapes at the cellar.

- An enormous volume of grapes was received at the winery on day 3 of the actual 2009 Wamakersvallei harvesting schedule, which is not at all typical. During the five day harvesting period considered, a total of over 400 tonnes of grapes were received at the cellar without any trace of grape samples being received beforehand. This definitely influences the outcome of the schedule, since these grapes were most likely not scheduled for intake at the winery, but instead received from a supplier who had a large volume of excess grapes; this often happens when a supplier delivers to more than one winery, in which case the staff at Wamakersvallei seem inclined to help out by taking in the additional grapes.

It is clear from Table 2 that VINDSS is capable of producing a more stable harvesting schedule in terms of consistent grape intake volumes (and hence activity within the cellar) than what actually occurred during the 2009 season at Wamakersvallei Winery. Furthermore, given the *ad hoc* manner in which harvesting decisions seem to have been made during 2009, it may even be that the VINDSS schedule H_{100} also compares favourably to the actual 2009 harvesting schedule in terms of the timing of vineyard block harvesting (or the closeness of grapes to optimal ripeness levels), but in order to be able to verify such an expectation, better data keeping practices at the winery are required.

9 Conclusion

In this paper we developed a nested tabu search approach towards solving the complex harvest scheduling problem experienced at producer cellars in South Africa. We also implemented this solution methodology in a computerised decision support system, called VINDSS, which may be useful to cellar managers. However, the effectiveness of this decision support tool is heavily dependent on large volumes of high-quality historical and sample data, a lesson learnt during the case study reported in §8 above.

An aspect of the decision support system which may yet be further enhanced is the design of a plug-in capability of performing accurate forecasts with respect to sugar and ripeness levels of grapes, perhaps taking weather forecasts into consideration.

References

- [1] WINE SUPPLY CHAIN COUNCIL, 2009, *From grapes to glass*, [Online], [cited February 20th, 2010], Available from <http://www.tli.gatech.edu/wsc/>
- [2] AUGER A, FERRER JC, MATURANA S & VERA J, 2008, *Simulation of the grape reception at a winery*, [Online], [cited November 5th 2008], Available from <http://www.gepuc.cl/publicaciones.html>
- [3] DUNSTALL S & JOHNSTONE R, 2005, *Applying innovative decision support technologies to achieve harmony and adaptability in an Australian wine supply network*, Paper presented at the Smart 2005 Conference, Australia.

- [4] FERRER JC, MAC CAWLEY A, MATURANA S, TOLOZA S & VERA J, 2008, *An optimization approach for scheduling wine grape harvest operations*, International Journal of Production Economics, **112**, pp. 985–999.
- [5] GERTIOSO C, 1988, *A decision support system for wine-making cooperatives*, European Journal of Operational Research, **33**, pp. 273–278.
- [6] GLOVER F & LAGUNA M, 1993, *Tabu search*, pp. 70–150 in REEVES CR (ED), *Modern heuristic techniques for combinatorial problems*, Blackwell Scientific Publications, Oxford.
- [7] MUSEE N, LORENZEN L & ALDRICH C, 2005, *Decision support for waste minimization in wine making processes*, Environmental Progress, **25**(1), pp. 56–63.
- [8] VAN DER MERWE A, 2009, *A decision support system for scheduling the harvesting and wine making processes at a winery*, MSc Thesis, University of Stellenbosch, Stellenbosch.