A Continuous-Time *GDP* Framework for the Integrated Scheduling of Real-World Wine Production Systems

Germán M. Heim[1,2], Eduardo Vives Martínez[3], Erica Schulz[1,4], and Guillermo Durand[1,4,*]

[1] Universidad Nacional del Sur, Departamento de Ingeniería Química, Bahía Blanca, Buenos Aires, Argentina
[2] The Dow Chemical Company, Bahía Blanca, Buenos Aires, Argentina
[3] Bodega Vera de Estenas, Utiel, Valencia, España
[4] Planta Piloto de Ingeniería Química (PLAPIQUI CONICET-UNS), Bahía Blanca, Buenos Aires, Argentina
*gdurand@plapiqui.edu.ar

Abstract. The scheduling of multi-stage batch processes remains a challenging problem in process systems engineering. In the wine production industry, processing and storage decisions are tightly coupled, yet most existing formulations treat these aspects separately or rely on simplified representations of storage dynamics. This work presents a unified framework for the integrated scheduling of a real multi-product wine production facility in Spain (Vera de Estenas Winery), in close collaboration with the plant's engineering team.
The proposed model is formulated as a continuous-time, event-based problem using Generalized Disjunctive Programming (GDP). Key operational decisions, including task activation and unit assignment, are represented through disjunctions, enabling a compact and expressive formulation of complex logical relationships.
The objective function minimizes the makespan while satisfying product demand and preserving operational feasibility across shared resources. The study demonstrates how advanced optimization models can be effectively deployed in an industrial winemaking context to support informed, data-driven decision-making.

Keywords: scheduling, wine production, optimization

Una formulación de *GDP* en tiempo continuo para la programación integrada de sistemas reales de producción de vino.

Abstract. La programación de procesos batch multietapa sigue siendo un problema desafiante en la ingeniería de sistemas de procesos. En la industria de producción de vino, las decisiones de procesamiento y almacenamiento están estrechamente acopladas; sin embargo, la mayoría de las formulaciones existentes tratan estos aspectos por separado o se basan en representaciones simplificadas de la dinámica de almacenamiento. Este trabajo presenta un marco unificado para la programación integrada de una planta real de producción de vino multiproducto en España (Bodega Vera de Estenas), en estrecha colaboración con el equipo de ingeniería de la planta.
El modelo propuesto se formula como un problema en tiempo continuo, basado en eventos, utilizando Programación Disyuntiva Generalizada (GDP). Las decisiones operativas clave, incluyendo la activación de tareas y la asignación de unidades, se representan mediante disyunciones, lo que permite una formulación compacta y expresiva de relaciones lógicas complejas.
La función objetivo minimiza el tiempo total de producción, satisfaciendo al mismo tiempo la demanda de productos y preservando la factibilidad operativa en recursos compartidos. El estudio demuestra cómo modelos avanzados de optimización pueden implementarse eficazmente en un contexto industrial de elaboración de vino para apoyar la toma de decisiones informadas y basadas en datos.

Keywords: planeamiento a corto plazo, elaboración de vinos, optimización

1 Introduction

Batch process scheduling has been extensively studied through MILP formulations based on state-task network (STN) and resource-task network (RTN) representations (Kondili et al., 1993; Shah et al., 1993). While these frameworks handle multiple products and shared resources effectively, encoding logical decisions such as task activation and unit assignment via Big-M relaxations can become numerically demanding in tightly constrained instances (Grossmann, 2002; Méndez et al., 2006).

Wine production is a particularly challenging application: multiple interconnected stages are coupled through zero-wait transfers, limited intermediate storage, and tanks that remain dedicated once assigned. Existing works address isolated subproblems - grape harvesting (Bohle et al., 2010), fermentation (Cholette et al., 2019), or bottling (Lillo Otarola et al., 2025) - but full integration of processing and storage decisions within a unified framework is lacking.

Generalized Disjunctive Programming (GDP) addresses these limitations by representing logical relationships explicitly through disjunctions (Grossmann & Trespalacios, 2013; Raman & Grossmann, 1994). Hull-relaxation reformulations yield tight LP relaxations and strong computational behavior (Lee & Grossmann, 2000), yet GDP remains underexplored in integrated scheduling contexts.

This work proposes a GDP-based, continuous-time scheduling formulation for a real multi-product winery, implemented in Pyomo.GDP (Chen et al., 2022). The model integrates task activation, unit assignment, zero-wait and no-intermediate-storage constraints, and storage persistence within a single disjunctive framework. The main contributions are: (i) a comprehensive GDP scheduling model for multi-stage wine production; (ii) unified treatment of operational constraints including zero-wait transfers and tank persistence; and (iii) a computational study on a realistic industrial case study.

2 Problem Description

This work addresses the short-term scheduling of young wines production (i.e., wines that do not undergo barrel-aging) in a multi-product batch process representative of an operational winery. The operational scope spans from grape pressing through fermentation, stabilization, and final storage prior to dispatch, focusing on the most time- and resource-intensive stages of the campaign. Long-term aging decisions in barriques are intentionally excluded, as they follow different planning horizons and constraints.

The study is grounded in a real industrial setting: Bodega Vera de Estenas, located in Utiel (Valencia, Spain), a small-to-medium-sized winery with seasonal and highly constrained operations (see Figure 1). The modeled production network mirrors the actual plant layout and operating practices, capturing shared processing units, limited intermediate and final storage, and strict temporal relationships between consecutive tasks imposed by wine quality and stability requirements. The resulting process flowsheet, shown in Figure 2, includes only the stages represented in the model and reflects the structure encountered by production managers during the young-wine campaign. Within a finite planning horizon, the scheduling problem consists of deciding which tasks are executed, their start and finish times, the assignment of processing units, and the corresponding batch sizes. These decisions must satisfy market demand, unit capacity limits, storage availability, and all technological constraints inherent to wine production. The objective is to generate a feasible and efficient production schedule that reflects realistic winery operation while minimizing the makespan of the young-wine campaign.

The problem consists of determining, over a finite planning horizon, which tasks are executed, when they start and finish, which units are assigned, and what batch sizes are processed, while satisfying demand and all technological constraints. The objective is to obtain a feasible and efficient production schedule that minimizes makespan for the young-wine campaign.

3 Mathematical Formulation

3.1 Problem formulation

Building on Sec. 2, the case study is formulated as a continuous-time, event-based scheduling model over ordered event points. The formulation follows an

Fig. 1. Location of Bodega Vera de Estenas in Utiel, Valencia, Spain.

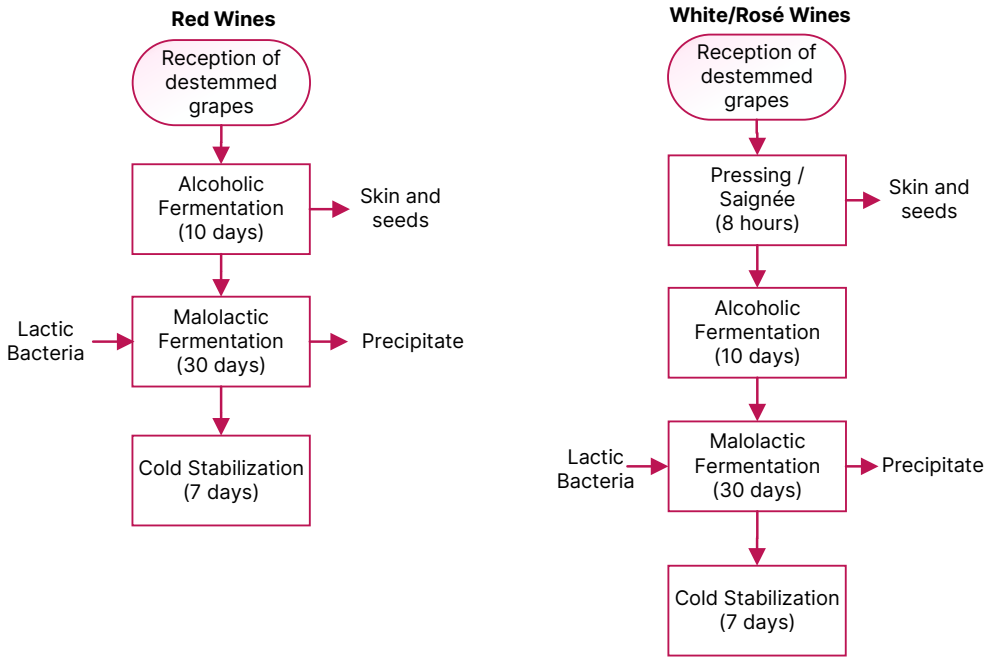

Fig. 2. Process flowsheet of the modeled stages in wine production for Red, White and Rosé young wines.

STN-type representation and simultaneously determines task activation, unit assignment, batch sizing, and timing decisions.

Given sets of tasks I, units J, states S, and event points N, together with processing-time parameters, capacity bounds, initial inventories, demand targets, and wine-specific rules (zero-wait, no-intermediate-storage, and Ecobulk constraints), the model computes a feasible schedule over horizon H that minimizes makespan. The complete model is written in GDP form and reformulated through the Hull transformation.

3.2 Sets, Indices, and Parameters

The model operates over four primary index sets. The set I contains all production tasks, partitioned into non-storage tasks I^{nst}, storage tasks I^{st}, and pre-storage tasks I^{pst}. The set J contains all processing and storage units, with $J^{\mathrm{st}} \subset J$ denoting storage units. The set S contains all material states, with subsets S^{P} (final products), S^{ZW} (zero-wait states), and S^{NIS} (no-intermediate-storage states). The set $N = \{1, 2, \ldots, |N|\}$ represents ordered event points imposing a temporal sequence. Task–unit compatibility is encoded by the set of feasible assignment pairs $\mathrm{IJ} \subseteq I \times J$, with projections $J_i = \{j : (i,j) \in \mathrm{IJ}\}$ and $I_j = \{i : (i,j) \in \mathrm{IJ}\}$ denoting the units compatible with task i and the tasks compatible with unit j, respectively. Key parameters include the fixed and variable processing time coefficients α_i and β_i for task i, the production and consumption stoichiometric coefficients $\rho_{i,s}^{\mathrm{prod}}$ and $\rho_{i,s}^{\mathrm{cons}}$ relating task i to state s, the minimum and maximum batch capacities $B_{i,j}^{\min}$ and $B_{i,j}^{\max}$ for task i in unit j, and the planning horizon H. The number of times a non-storage task may be activated is bounded by $iMax$. The bounds $jMin$, $jMax$, $jMin_{ST}$, and $jMax_{ST}$ bound the number of units simultaneously assigned to each active task.

3.3 Decision Variables

The model introduces both Boolean and continuous decision variables. Boolean variable $W_{i,n}$ indicates whether task i is active at event point n, and $y_{i,j,n}$ indicates whether task i uses unit j at event n. Continuous variable $b_{i,n} \geq 0$ denotes the total batch size of task i at event n, decomposed into unit-level contributions $b_{i,j,n}$. Inventory of state s at event n is tracked by $ST_{s,n} \geq 0$. The start and finish times of task i at event n are $Ts_{i,n}$ and $Tf_{i,n}$, respectively, with analogous unit-level timing variables $Tsj_{j,n}$ and $Tfj_{j,n}$. In the makespan formulation, the scalar variable $MS \geq 0$ represents the total schedule length to be minimized in Section 3.11.

3.4 GDP Formulation: Task Activation and Unit Assignment

The core scheduling logic uses Generalized Disjunctive Programming (GDP), encoding task-unit logic through disjunctions reformulated via hull transformation into a tractable MILP. For each task $i \in I$ and event point $n \in N$, an outer

disjunction captures task activation; when the task is active, a nested unit-selection disjunction for each compatible unit $j \in J_i$ enforces unit assignment, batch capacity bounds, and timing synchronization; when inactive, all associated variables are set to zero. Formally, the combined task-and-unit disjunction (Eq. 1) is:

$$\forall i \in I, n \in N : \begin{bmatrix} W_{i,n} \\ Tf_{i,n} = Ts_{i,n} + \alpha_i + \beta_i b_{i,n} \quad (i \in I^{\text{nst}}) \\ \forall j \in J_i : \begin{bmatrix} y_{i,j,n} \\ B_{i,j}^{\min} \leq b_{i,j,n} \leq B_{i,j}^{\max} \\ Tsj_{j,n} = Ts_{i,n} \\ Tfj_{j,n} = Tf_{i,n} \end{bmatrix} \vee \begin{bmatrix} \neg y_{i,j,n} \\ b_{i,j,n} = 0 \end{bmatrix} \end{bmatrix} \vee \begin{bmatrix} \neg W_{i,n} \\ b_{i,n} = 0 \\ Tf_{i,n} = Ts_{i,n} \\ \forall j \in J_i : \begin{bmatrix} \neg y_{i,j,n} \\ b_{i,j,n} = 0 \end{bmatrix} \end{bmatrix} \tag{1}$$

Storage tasks do not have a fixed-duration equation in the active disjunct. Their timing is defined by persistence constraints (Section 3.7) and a final-event condition: if a storage task is active at N, then its finish time equals MS. This bounds the storage horizon. A mutual-exclusion constraint enforces at most one task per unit and event:

$$\sum_{i \in I_j} y_{i,j,n} \leq 1, \ \forall j \in J, n \in N \tag{2}$$

3.5 Batch Aggregation and Unit Multiplicity

The total batch size of task i at event n equals the sum of unit-level batches:

$$b_{i,n} = \sum_{j \in J_i} b_{i,j,n}, \ \forall i \in I, n \in N \tag{3}$$

Within the active disjunct, unit-level batch sizes are already bounded by the physical capacity of the tank: when a unit is assigned the batch lies in $[B_{i,j}^{\min}, B_{i,j}^{\max}]$ and is forced to zero otherwise (Eq. 1). The minimum fill bound $B_{i,j}^{\min}$ is not merely an operative limit: leaving excessive headspace exposes the wine to oxygen, which alters its chemical composition, accelerates oxidation reactions, and degrades taste, aroma, and colour (Tarko et al., 2020). Enforcing a high minimum fill ratio is therefore a product-quality requirement embedded directly in the disjunctive assignment logic.

The minimum and maximum unit multiplicity per active task are enforced by the following algebraic constraint, which remains outside the disjunction as it aggregates across all units:

$$jMin_i W_{i,n} \ \leq \ \sum_{j \in J_i} y_{i,j,n} \ \leq \ jMax_i W_{i,n}, \quad \forall i \in I, \ n \in N \tag{4}$$

In the present case study, we set $jMax_i = 1$ for all $i \in I$, so that each active task is assigned to exactly one processing unit, in line with the winery's practice of not splitting a batch across tanks within a single task.

3.6 Material balances

Material inventories evolve across event points according to production and consumption events. At the initial event point ($n = 1$), the inventory of each state s is initialized from a known starting stock $ST_{s,0}$ and adjusted by consumption occurring at that event:

$$ST_{s,1} = ST_{s,0} + \sum_{i \in \mathrm{ICS}_s} \rho_{i,s}^{\mathrm{cons}} b_{i,1}, \ \forall s \in S \tag{5}$$

For all subsequent event points ($n > 1$), the inventory balance accounts for material produced at the preceding event and material consumed at the current event:

$$ST_{s,n} = ST_{s,n-1} + \sum_{i \in \mathrm{IPS}_s} \rho_{i,s}^{\mathrm{prod}} b_{i,n-1} + \sum_{i \in \mathrm{ICS}_s} \rho_{i,s}^{\mathrm{cons}} b_{i,n}, \ \forall s \in S, n > 1 \tag{6}$$

Here IPS_s and ICS_s denote the sets of tasks that produce and consume state s, respectively.

3.7 Temporal Sequencing and Precedence

Within-task sequencing requires that a task cannot begin at event $n + 1$ before it finishes at event n:

$$Ts_{i,n+1} \geq Tf_{i,n}, \ \forall i \in I, n \in \{1, \ldots, |N| - 1\} \tag{7}$$

An analogous sequence constraint applies at the unit level, preventing a unit from being reused before its previous engagement is completed:

$$Tsj_{j,n+1} \geq Tfj_{j,n}, \ \forall j \in J, n \in \{1, \ldots, |N| - 1\} \tag{8}$$

A companion constraint enforces non-negative unit occupancy at every event, so that a unit's finish time cannot precede its start time. This bounds the unit timing variables even at events where the unit is idle and therefore not synchronized to any task:

$$Tfj_{j,n} \geq Tsj_{j,n}, \ \forall j \in J, n \in N \tag{9}$$

Tasks linked by material flow are precedence-coupled. Let

$$\mathrm{Prec} = \{(i, i') : i \in \mathrm{ICS}_s, \ i' \in \mathrm{IPS}_s \text{ for some } s \in S, \ i \neq i'\}$$

collect the ordered pairs in which the consumer i draws a state produced by i'. A consumer i cannot begin at event $n+1$ until its producer i' has finished at event n, expressed as a conditional constraint on the activation variables:

$$Ts_{i,n+1} \geq Tf_{i',n} - H(2 - W_{i',n} - W_{i,n+1}),\ \forall (i,i') \in \text{Prec}, n \in \{1,\ldots,|N|-1\} \tag{10}$$

This conditional form binds only between a producer at event n and a consumer at event $n+1$. To ensure this is sufficient, each production line is treated as an ordered chain of tasks whose consecutive steps are locked to consecutive event points by a one-to-one activation coupling: a downstream task i is active at $n+1$ if and only if its immediate upstream task i' is active at n,

$$W_{i,n+1} = W_{i',n}, \quad \forall (i,i') \in C,\ n \in \{1,\ldots,|N|-1\}, \tag{11}$$

where $C \subseteq \text{Prec}$ collects the directly-coupled consecutive step pairs, excluding those already tied by an exact-timing rule (such as the zero-wait press-to-fermentation hand-off treated below). Consequently a producer at event n is always matched by its consumer at $n+1$. Material that must be buffered instead passes through an intermediate storage task that occupies the event in between, so every transfer remains between consecutive events.

3.8 Wine-Specific Operational Constraints

Several operational rules specific to the winemaking process require exact timing relationships between consecutive tasks. These are expressed as equality constraints on timing and are active only when both involved tasks are themselves active. Zero-wait (ZW) states arise when a material cannot be held in intermediate inventory after production - a common requirement in white wine processing where contact with air or equipment must be minimized. For states $s \in S^{\text{ZW}}$, the consuming task i must begin at exactly the moment production by i' ends:

$$W_{i',n} \wedge W_{i,n+1} \implies Ts_{i,n+1} = Tf_{i',n},\ \forall (i,i') : s \in S^{\text{ZW}} \tag{12}$$

The absolute-value bound is expanded into the equivalent pair of inequalities, with the horizon-based relaxation term deactivating the constraint when either task is inactive:

$$Ts_{i,n+1} - Tf_{i',n} \ \leq\ H\big(2 - W_{i',n} - W_{i,n+1}\big) \tag{13a}$$
$$Tf_{i',n} - Ts_{i,n+1} \ \leq\ H\big(2 - W_{i',n} - W_{i,n+1}\big) \tag{13b}$$

Rather than embedding these exact-timing relations in a disjunction and applying the hull reformulation used for the task-and-unit logic, they are stated directly in big-M form, with the planning horizon H serving as the big-M constant. This choice proved computationally more effective for the present problem.

First, H is a tight, physically meaningful bound on any timing gap, so the linear relaxation is already strong and the big-M slack introduces little looseness. Second, a hull reformulation would require disaggregating the continuous timing variables (Ts, Tf) into one copy per disjunct, inflating the model with auxiliary variables and balance constraints without a commensurate gain in relaxation tightness. The compact big-M form therefore yielded smaller models and shorter solution times.

No-intermediate-storage (NIS) constraints apply to states in which holding inventory between tasks is physically infeasible (e.g., due to a lack of buffer vessels). These are formulated identically to Eq. 13 and share the same linearized structure.

Ecobulk compatibility constraints model the requirement that certain products destined for flexible Ecobulk containers must be filled immediately upon production, imposing an NIS-like condition (Eq. 13). In practice, Ecobulk lots are prepared for near-immediate dispatch under committed logistics windows, so they cannot be treated as regular buffering inventory between processing stages. Moreover, allowing intermediate waiting would increase handling and temporary-holding needs, which is undesirable for quality preservation and lot traceability in this campaign. Additionally, if the parameter MustUseDep = 1, at least one storage (deposit) task must be activated for each Ecobulk-compatible product state:

$$\sum_{n} \sum_{i \in \mathrm{ICS}_s} W_{i,n} \geq 1, \ \forall s \in S^{\mathrm{FISEco}}, \ \text{if MustUseDep} = 1 \tag{14}$$

3.9 Storage Persistence

Storage tasks model tanks that, once filled, remain occupied for the remainder of the planning horizon. This behavior is captured by logical implication chains: if a storage task is active at event n, it must remain active at $n+1$ and, by transitivity, at all subsequent events. In the GDP representation, this is stated as:

$$W_{i,n} \implies W_{i,n+1}, \ \forall i \in I^{\mathrm{st}}, n \in \{1, \ldots, |N|-1\} \tag{15}$$

Similarly, unit-level persistence ensures that once a storage unit is assigned to a task, it is not released:

$$y_{i,j,n} \implies y_{i,j,n+1}, \ \forall i \in I^{\mathrm{st}}, (i,j) \in \mathrm{IJ}, n \in \{1, \ldots, |N|-1\} \tag{16}$$

For non-storage tasks, activation frequency is bounded globally:

$$\sum_{n \in N} W_{i,n} \leq iMax, \ \forall i \in I^{\mathrm{nst}} \tag{17}$$

3.10 Unit Capacity and Horizon Feasibility

Since no two tasks occupy a unit simultaneously, the cumulative processing time of all non-storage tasks assigned to unit j cannot exceed the schedule length:

$$\sum_{i \in I^{\text{nst}}} \sum_{n \in N} (\alpha_i y_{i,j,n} + \beta_i b_{i,j,n}) \leq MS, \ \forall j \in J \tag{18}$$

3.11 Makespan Formulation: Objective and Demand Constraints

In the makespan formulation, the scalar variable MS is bounded below by the finish time of all pre-storage tasks at the final event point:

$$Tf_{i,N} \leq MS, \ \forall i \in I^{\text{pst}} \tag{19}$$

The makespan is also bounded below by the time needed to traverse the longest production line. Let L be the set of product lines, each an ordered chain of tasks, the minimum completion time of line l is the sum of its fixed processing durations, giving the critical-path lower bound

$$MS \geq \max_{l \in L} \sum_{i \in l} \alpha_i \tag{20}$$

This valid inequality tightens the root relaxation of the makespan objective.

Final product demands D_s must be satisfied. The total quantity produced for state s is computed from the last-event inventory and any production occurring at event N:

$$ST_{s,N} + \sum_{i \in \text{IPS}_s} \rho_{i,s}^{\text{prod}} \, b_{i,N} = \text{FinalProd}_s, \quad \forall s \in S^{\text{P}} \tag{21a}$$

$$\text{FinalProd}_s \geq D_s, \quad \forall s \in S^{\text{P}} \tag{21b}$$

The objective minimizes the makespan:

$$\min \ Z = MS \tag{22}$$

3.12 Model Implementation and Solution Strategy

The formulation is implemented in Python using the Pyomo (`v6.10.0`) modeling environment (Bynum et al., 2021; Hart et al., 2011). GDP disjunctions are declared using `Pyomo.GDP` and automatically transformed to MILP via the Hull Reformulation using the `gdp.hull` transformation. The resulting MILP instances are solved using the commercial branch-and-cut solver `Gurobi 13.0` (Gurobi Optimization, LLC, 2026). All computational experiments are conducted on a Windows 10 machine with a Ryzen 5 5500 processor and 16 GB of RAM running at 3000 MHz. The number of event points $n_{\max}$ is treated as an input parameter

and determined by progressive enumeration: starting from a minimum feasible value, $n_{\max}$ is incremented until the objective function does not improve its value. This approach avoids over-specifying the event grid (Ierapetritou & Floudas, 1998).

4 Results

Table 1 summarises the instance characteristics and solver statistics. The model was solved to global optimality using Gurobi 13.0.

Table 1. Instance characteristics and computational results.

Instance characteristic	**Value**
Products (Red/White/Rosé)	11 (4/4/3)
Processing stages	5
Processing units	12
Event points ($n_{\max}$)	9
MILP size (post-hull)	
Constraints	81,721
Variables (binary)	36,943 (4,839)
Nonzeros	182,430
MILP size (post-presolve)	
Constraints	11,048
Variables (binary)	6,882 (577)
Solution	
Optimal makespan MS^*	2,024.75 h (84.4 days)
Initial optimality gap	67.8%
Final optimality gap	0.00%
B&B nodes explored	128,381
CPU time (s)	545.9

The number of event points was set to $n_{\max} = 9$, determined by progressive enumeration as described in Section 3.12. After the hull reformulation, Gurobi's presolve reduced the instance by 86% in constraints and 81% in variables. The solver explored 128,381 branch-and-bound nodes and proved global optimality in 545.9 seconds.

The optimal makespan is $MS^* = 2{,}024.75$ h (≈ 84.4 days), well within the planning horizon of $H = 3{,}600$ h (150 days). All product demands are satisfied. The resulting schedule and final production figures are shown in Figure 3. In the Gantt chart, the numbers in parentheses after each stage label identify the corresponding wine line (product index), which helps track each product path across shared units.

The dominant bottleneck is malolactic fermentation, which requires 720 h per batch, three times longer than alcoholic fermentation (240 h) and four times longer than cold stabilization (168 h). Nine of the eleven products require a

malolactic fermentation stage, keeping the six capable units occupied almost continuously from $t = 240$ h to $t = 1{,}848$ h (roughly 80% of the makespan). The critical path runs through three parallel malolactic fermentation to cold stabilization chains terminating simultaneously at $t = 2{,}016$ h.

The most heavily utilised units are the fermentation and malolactic tanks. The Stainless Steel tanks (Inox) run without interruption from the start of the schedule, accumulating 1,440 h of active time (71% utilization). Three subterranean tanks each sustain two back-to-back malolactic fermentation tasks covering 1,440 h respectively, also at 71% utilization. The cold-stabilisation tank handles six consecutive tasks from $t = 416$ h onward at 50% utilization.

By contrast, the press schedule is entirely driven by zero-wait constraints: it must run immediately before each downstream fermentation start and therefore contributes no bottleneck of its own. Similarly, seven of the long-term storage units remain unused, the solver activates long-term storage only for one wine, which requires it under the Ecobulk constraint. This highlights that Ecobulk handling is a targeted operational requirement in the optimal plan, not a system-wide storage bottleneck.

5 Conclusions

This work presented a GDP-based continuous-time scheduling formulation for a real multi-product winery, integrating task activation, unit assignment, and storage dynamics within a unified disjunctive framework. By replacing Big-M inequalities with explicit disjunctions and applying the hull reformulation, the model yields tighter LP relaxations and a more expressive representation of the underlying logical structure.

Applied to a real industrial case study, the model proved global optimality in under ten minutes, demonstrating that GDP, combined with the hull reformulation, can handle the combinatorial complexity of multi-product winery scheduling without sacrificing tractability. The result shows that all product demands can be met well within the available campaign window, leaving meaningful slack that plant managers can exploit for maintenance, unplanned batches, or demand variability.

Future work will extend the framework to include aging using an economic objective (profit maximization with outsourcing and lateness penalties), explore multi-period planning horizons to capture seasonal variability in grape supply, and address uncertainty in processing times and demand through stochastic or robust optimization approaches.

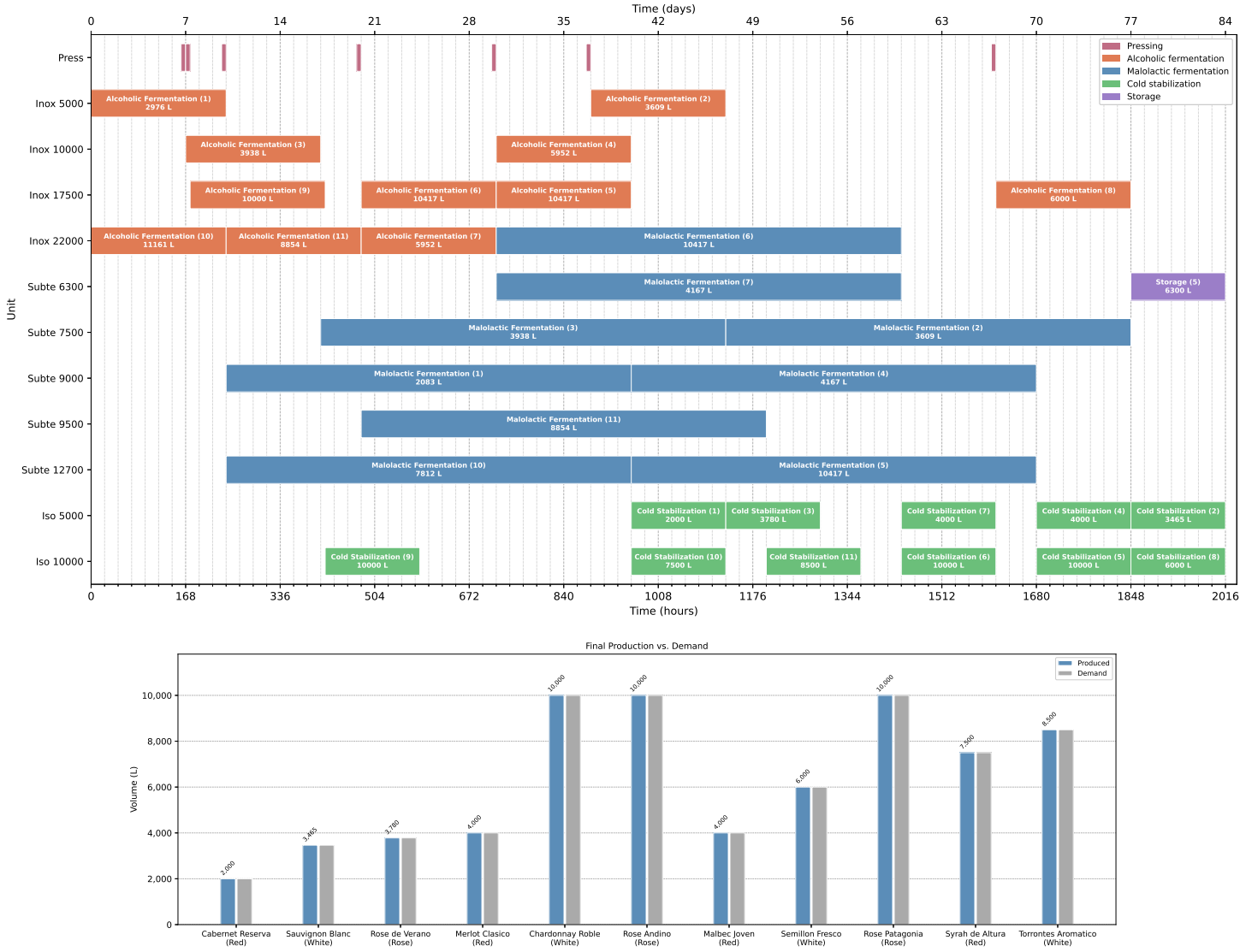

Fig. 3. Optimal schedule (top) and final production vs. demand (bottom) for the GDP makespan model. Makespan = 2,024.75 h.

Glossary

Symbol	Meaning
I	Set of tasks
I^{nst}	Set of non-storage tasks
I^{st}	Set of storage tasks
I^{pst}	Set of pre-storage tasks
J	Set of units
J^{st}	Set of storage units
S	Set of material states
S^{P}	Set of final product states
S^{ZW}	Set of zero-wait states
S^{NIS}	Set of no-intermediate-storage states
N	Ordered set of event points
L	Set of product lines (each an ordered chain of tasks)
IJ	Set of feasible task-unit assignment pairs
J_i	Units compatible with task i, i.e. $\{j : (i,j) \in \text{IJ}\}$
I_j	Tasks compatible with unit j, i.e. $\{i : (i,j) \in \text{IJ}\}$
IPS_s	Tasks producing state s
ICS_s	Tasks consuming state s
Prec	Precedence pairs (i,i'): consumer i draws a state produced by i'
C	Directly-coupled consecutive step pairs, $C \subseteq \text{Prec}$
α_i, β_i	Fixed and variable processing time coefficients for task i
$\rho_{i,s}^{\text{prod}}, \rho_{i,s}^{\text{cons}}$	Production (positive) and consumption (negative) coefficients of task i for state s
$B_{i,j}^{\min}, B_{i,j}^{\max}$	Minimum and maximum batch sizes of task i in unit j
H	Planning horizon
$iMax$	Maximum activations of a non-storage task
$jMin, jMax$	Min and max units assigned to an active non-storage task
$jMin_{ST}, jMax_{ST}$	Min and max units assigned to an active storage task
D_s	Demand for final product state s
$W_{i,n}$	Boolean activation of task i at event n
$y_{i,j,n}$	Boolean assignment of unit j to task i at event n
$b_{i,n}$	Total batch size of task i at event n
$b_{i,j,n}$	Batch size of task i in unit j at event n
$ST_{s,n}$	Inventory of state s at event n
$Ts_{i,n}, Tf_{i,n}$	Start and finish times of task i at event n
$Tsj_{j,n}, Tfj_{j,n}$	Start and finish times of unit j at event n
MS	Makespan
FinalProd_s	Total produced amount of final product state s

Acknowledgments. GMH acknowledges support from a scholarship from Universidad Nacional del Sur (85220250100037SU).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

Bohle, C., Maturana, S., & Vera, J. (2010). A robust optimization approach to wine grape harvesting scheduling. *European Journal of Operational Research*, *200*(1), 245–252. https://doi.org/https://doi.org/10.1016/j.ejor.2008.12.003

Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Siirola, J. D., Watson, J.-P., & Woodruff, D. L. (2021). *Pyomo–optimization modeling in python* (Third, Vol. 67). Springer Science & Business Media.

Chen, Q., Johnson, E. S., Bernal, D. E., Valentin, R., Kale, S., Bates, J., Siirola, J. D., & Grossmann, I. E. (2022). Pyomo.GDP: an ecosystem for logic based modeling and optimization development. *Optimization and Engineering*, *23*(1), 607–642. https://doi.org/10.1007/s11081-021-09601-7

Cholette, S., Maturana, S., & Mac Cawley, A. (2019). Scheduling fermentation tanks optimally. *Proceedings of the 13th Annual Conference of the American Association of Wine Economists (AAWE)*. https://www.wine-economics.org

Grossmann, I. E. (2002). Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, *3*(3), 227–252. https://doi.org/10.1023/A:1021039126272

Grossmann, I. E., & Trespalacios, F. (2013). Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. *AIChE Journal*, *59*(9), 3276–3295. https://doi.org/10.1002/aic.14088

Gurobi Optimization, LLC. (2026). Gurobi Optimizer Reference Manual. https://www.gurobi.com

Hart, W. E., Watson, J.-P., & Woodruff, D. L. (2011). Pyomo: Modeling and solving mathematical programs in python. *Mathematical Programming Computation*, *3*(3), 219–260.

Ierapetritou, M. G., & Floudas, C. A. (1998). Effective continuous-time formulation for short-term scheduling. 1. multipurpose batch processes. *Industrial & Engineering Chemistry Research*, *37*(11), 4341–4359. https://doi.org/10.1021/ie970927g

Kondili, E., Pantelides, C., & Sargent, R. (1993). A general algorithm for short-term scheduling of batch operations—I. MILP formulation [An International Journal of Computer Applications in Chemical Enginnering]. *Computers & Chemical Engineering*, *17*(2), 211–227. https://doi.org/10.1016/0098-1354(93)80015-F

Lee, S., & Grossmann, I. E. (2000). New algorithms for nonlinear generalized disjunctive programming. *Computers & Chemical Engineering*, *24*(9), 2125–2141. https://doi.org/10.1016/S0098-1354(00)00581-0

Lillo Otarola, L., de la Fuente-Mella, H., Peña Domarchi, A., Kundu, A., & Ceroni-Díaz, J. (2025). Optimal scheduling of the wine-bottling process: A multi-dependency model with hydraulic considerations. *Applied Sciences*, *15*(9), 4697. https://doi.org/10.3390/app15094697

Méndez, C. A., Cerdá, J., Grossmann, I. E., Harjunkoski, I., & Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering*, *30*(6), 913–946. https://doi.org/10.1016/j.compchemeng.2006.02.008

Raman, R., & Grossmann, I. (1994). Modelling and computational techniques for logic based integer programming [An International Journal of Computer Applications in Chemical Engineering]. *Computers & Chemical Engineering*, *18*(7), 563–578. https://doi.org/10.1016/0098-1354(93)E0010-7

Shah, N., Pantelides, C., & Sargent, R. (1993). A general algorithm for short-term scheduling of batch operations—II. Computational issues [An International Journal of Computer Applications in Chemical Engineering]. *Computers & Chemical Engineering*, *17*(2), 229–244. https://doi.org/10.1016/0098-1354(93)80016-G

Tarko, T., Duda-Chodak, A., Sroka, P., & Siuta, M. (2020). The impact of oxygen at various stages of vinification on the chemical composition and the antioxidant and sensory properties of white and red wines. *Int J Food Sci*, *2020*, 7902974.