Technical Documentation

A. Definition MLCLSP-L-B with Probabilistic Demand

The MIP formulation of the MLCLSP-L-B with probabilistic demand is represented by a two-stage stochastic problem formulation. Many studies in the literature like Hu and Hu (2016) and Azizi et al. (2020) use that topology to model a two-stage stochastic programming formulation for lot size optimization. Scenarios are inserted into production, inventory, backorder, setup, and linked lot size decision variables as new indexes, whereby the total setup decision variable stays untouched. The total setup decision variable ensures that the model's outcome generates a unique setup plan across different scenarios. The model determines lot sizes so that setup, inventory, and backorder costs are kept at a minimum while demand has to be satisfied across several demand scenarios.

$x^{su}_{s,p,t}$	Equals 1, if $p \in \mathscr{P}$ is prepared for setup in $t \in \mathscr{T}$ and scenario $s \in \mathscr{S}$, otherwise 0
$x^{l}_{s,p,t}$	Equals 1, if the production of $p \in \mathscr{P}$ is continued from t to $t+1$ on period domain $\mathscr{T}_0$ and scenario $s \in \mathscr{S}$, otherwise 0
$x^{tsu}_{p,t}$	First-stage total setup decision that equals $x^{su}_{s,p,t} + x^{l}_{s,p,t-1}$ for all $s \in \mathscr{S}$, $p \in \mathscr{P}$, and $t \in \mathscr{T}$
$x^{p}_{s,p,t}$	Production quantity of product $p \in \mathscr{P}$ in period $t \in \mathscr{T}$ and scenario $s \in \mathscr{S}$
$x^{inv}_{s,p,t}$	Inventory quantity of a product $p \in \mathscr{P}$ in period $t \in \mathscr{T}_0$ and scenario $s \in \mathscr{S}$
$x^{bo}_{s,p,t}$	Backorder quantity of a product $p \in \mathscr{P}$ in period $t \in \mathscr{T}_0$ and scenario $s \in \mathscr{S}$

Table 1: Decision variables of the MLCLSP-L-B with probabilistic demand

Let $S, M, P, T \in \mathbb{N}$ be the number of scenarios, machines, materials, and planning periods, respectively. Each material is allocated to exactly one machine, but one machine can produce several materials. Thus, a machine index is not required for production-related decision variables and model parameters. Each finished good is requested by scneario- and period-specific deterministic demand. Probabilistic demand is expressed by the demand scenarios with equal probability. Moreover, a material can be stocked or backordered. For both cases, holding and backorder costs must be considered per unit at the end of each period. Each machine has a period-specific capacity. The production of materials requires variable production and fixed setup times. A setup operation is associated with product-specific setup costs. This paper assumes lead times to equal 0 periods, sequence-independent setups, and linked lot sizes' validity. If a material is produced, then several ingredients are issued. The set of issued ingredients can intersect with other sets of issued materials. Table 1 and Table 2 summarize model decision variables and parameters of the MLCLSP-L-B with probabilistic

$\mathscr{S}$	Set of scenarios $\{0,\dots,S\}$
$\mathscr{M}$	Set of machines $\{1,\dots,M\}$
$\mathscr{P}$	Set of products $\{1,\dots,P\}$
$\mathscr{T}$	Set of periods $\{1,\dots,T\}$
$\mathscr{T}_0$	Set of periods including initial period $\{0,\dots,T\}$
$\mathscr{P}_p^{suc}$	Set of successors of a product $p \in \mathscr{P}$
$\mathscr{P}_m$	Set of products that can be produced on machine $m \in \mathscr{M}$
$\mathscr{P}^{Int}$	Set of intermediate products $\{p \in \mathscr{P} \mid \mathscr{P}_p^{suc} \neq \emptyset\}$
$b_{m,t}$	Capacity of machine $m \in \mathscr{M}$ in period $t \in \mathscr{T}$
$d_{s,p,t}$	Demand of product $p \in \mathscr{P}$ in scenario $s \in \mathscr{S}$ and period $t \in \mathscr{T}$
c_p^{su}	Setup cost for a product $p \in \mathscr{P}$
c_p^{inv}	Inventory holding cost for a product $p \in \mathscr{P}$
c_p^{bo}	Backorder cost for a product $p \in \mathscr{P}$
t_p^{su}	Setup time for a product $p \in \mathscr{P}$
t_p^{p}	Production time for a unit of product $p \in \mathscr{P}$
t_p^{as}	Advanced shift for a product $p \in \mathscr{P}$
$r_{p,q}$	Number of units of product $p \in \mathscr{P}$ required to produce one unit of successor product $q \in \mathscr{P}_p^{suc}$
$\bar{x}_{s,p,0}^{inv}$	Initial inventory quantity for $p \in \mathscr{P}$ in scenario $s \in \mathscr{S}$
$\bar{x}_{s,p,T}^{inv}$	Final inventory quantity for $p \in \mathscr{P}$ in scenario $s \in \mathscr{S}$
$\bar{x}_{s,p,0}^{bo}$	Initial backorder quantity for $p \in \mathscr{P}$ in scenario $s \in \mathscr{S}$
$\bar{x}_{s,p,0}^{l}$	Initial setup state for $p \in \mathscr{P}$, such that $\sum_{p \in \mathscr{P}_m} \bar{x}_p^l \leq 1$ for all $s \in \mathscr{S}$ and $m \in \mathscr{M}$
$M_{s,p,t}$	Large number, e.g. $M_{s,p,t} = \min\{\sum_{\tau \in \mathscr{T}, \tau \geq t} d_{s,p,\tau}, b_{m,t} / t_p^p\}$ for $s \in \mathscr{S}$, $p \in \mathscr{P}_m$ for a fixed $m \in \mathscr{M}$, and $t \in \mathscr{T}$ whereby $d_{s,p,\tau}$ represents primary demand in case of finished goods and is replaced by secondary demand for intermediates

Table 2: Model sets and parameters of the MLCLSP-L-B with probabilistic demand

demand represented by scenarios. The correspondent MIP is based on the work of Quadt and Kuhn (2008), Helber and Sahling (2010), and Simonis and Nickel (2023b). Among this research, the MLCLSP-L-B with probabilistic demand is formulated as follows:

$$\min Z = \min \left\{ \frac{1}{S} \sum_{s \in \mathscr{S}} \left(\sum_{p \in \mathscr{P}} \sum_{t \in \mathscr{T}} c_p^{su} x_{s,p,t}^{su} + \sum_{p \in \mathscr{P}} \sum_{t \in \mathscr{T}} c_p^{inv} x_{s,p,t}^{inv} + c_p^{bo} x_{s,p,t}^{bo} \right) \right\}, \quad (1)$$

$$x^{inv}_{s,p,t-1} + x^{bo}_{s,p,t} + x^{p}_{s,p,t-t^{as}_{p}} = x^{inv}_{s,p,t} + x^{bo}_{s,p,t-1} + d_{s,p,t} + \sum_{p' \in \mathscr{P}^{suc}_{p}} r_{p,p'} x^{p}_{s,p',t}, \tag{2}$$

$$\sum_{p' \in \mathscr{P}_m} t^{su}_{p'} x^{su}_{s,p',t} + t^{p}_{p'} x^{p}_{s,p',t} \leq b_{m,t}, \tag{3}$$

$$x^{p}_{s,q,t} \leq M_{s,q,t} \left(x^{su}_{s,q,t} + x^{l}_{s,q,t-1} \right), \tag{4}$$

$$x^{tsu}_{p,t} = x^{su}_{s,p,t} + x^{l}_{s,p,t-1}, \tag{5}$$

$$\sum_{p' \in \mathscr{P}_m} x^{l}_{s,p',t} \leq 1, \tag{6}$$

$$x^{l}_{s,q,t} - x^{su}_{s,q,t} - x^{l}_{s,q,t-1} \leq 0, \tag{7}$$

$$x^{l}_{s,q,t} + x^{l}_{s,q,t-1} - x^{su}_{s,q,t} + x^{su}_{s,r,t} \leq 2, \tag{8}$$

$$x^{bo}_{s,p',t} = 0, \tag{9}$$

$$x^{inv}_{s,p,0} = \bar{x}^{inv}_{s,p,0}, x^{inv}_{s,p,T} = \bar{x}^{inv}_{s,p,T}, x^{l}_{s,q,0} = \bar{x}^{l}_{s,q,0}, x^{bo}_{s,p,0} = \bar{x}^{bo}_{s,p,0}, \tag{10}$$

$$x^{tsu}_{p,t} \in \{0,1\}, x^{su}_{s,q,t} \in \{0,1\}, x^{l}_{s,q,t} \in \{0,1\}, x^{bo}_{s,p,t} \geq 0, x^{p}_{s,q,t} \geq 0, x^{inv}_{s,p,t} \geq 0,$$

$$\forall s \in \mathscr{S}, m \in \mathscr{M}, p \in \mathscr{P}, q, r \in \mathscr{P}_m, q \neq r, p' \in \mathscr{P}^{Int}, t \in \mathscr{T}.$$

(1) aims to minimize average setup, inventory, and backorder costs across all scenarios for each machine and material over the planning horizon. The material balance equation is covered by (2), capacity constraints are included by (3), and (4) binds a positive production quantity to a setup in the same or a linked lot size in the previous period. (5) sets the final setup decision $x^{tsu}_{s,p,t}$ equal a setup operation $x^{su}_{p,t}$ plus a linked lot sizes from previous period $x^{l}_{s,p,t-1}$ for all scenarios. Thus $x^{su}_{p,t}$ is the only decision variable set at time zero (scenario independent) while production, inventory, backorder, setup and linked lot size decision variables are decided within the scenario afterwards. (6) satisfies that at most one linked lot size per period on a specific machine can be operated, (7) guarantees that a linked lot size is only allowed when a setup in the same period or a linked lot size in the previous period take place, and (8) synchronizes production runs that continue over more than two periods on a machine $m \in \mathscr{M}$ in scenario $s \in \mathscr{S}$. If $x^{l}_{s,p,t} = x^{l}_{s,p,t-1} = 1$, then either $x^{su}_{s,q,t-1} = 0$ for all $p, q \in \mathscr{P}_m$, $q \neq p$ or for some $q \neq p$, $x^{su}_{s,q,t-1} = 1$ and $x^{su}_{s,p,t-1} = 1$. That is, either product p is produced exclusively in period $t-1$ or product p is produced at the beginning of $t-1$, some other products were produced next, and the facility was reset to produce product p at the end of period $t-1$. (9) restricts the backorders to final products. This prohibits final products from being processed further if intermediate product shortages occur. Moreover, (10) sets the initial inventory and setup state and the initial and final backorder quantities, respectively.

B. Modeling Background

This section presents the modeling background of the RDS applied on the MLCLSP-L-B with probabilistic demand. It focuses on the data representation for the MLCLSP-L-B with probabilistic demand provided by Simonis and Nickel (2023a). A detailed description of the mathematical formulation of the MLCLSP-L-B with probabilistic demand is provided by Appendix A. This section explains first the role of time-dependent value substitutions. Second, it summarizes the representation of probabilistic demand. Third, the section outlines the representation of multi-level production structures.

B.1. Time-Dependent Values Substituted by Validity Horizons

MachineId	ValidityDateFrom	ValidityDateTo	TotalCapacity
Machine1	2021-01-01	2021-12-31	5200

(a) Validity horizon representation

MachineId	Date	Capacity
Machine1	2021-01-04	100
Machine1	2021-01-11	100
...	...	...
Machine1	2021-12-20	100
Machine1	2021-12-27	100

(b) Multi period representation

Figure 1: Two different representations of time-dependent values for capacity data

Data in manufacturing processes (manufacturing costs, processing times, or production recipes) are updated yearly. Thus, planning parameters stay constant over a significant part of the planning horizon. The spreadsheets profit from that behavior by substituting time dependencies with validity horizons instead of periods as suggested by Dutta and Fourer (2008). A validity horizon has a start and an end date and represents a particular value for the entire horizon. Figure 1 illustrates this representation: A machine *Machine1* might have a capacity of 100 for each week in 2021. In this example, a validity horizon represents a total capacity of 5200 in 2021 by one observation. At the same time, a multi-period representation consists of 52 observations with a capacity value of 100 per observation. Thus, validity horizons keep the data compact and easily maintainable in the spreadsheets. The price of this simplicity is that these time-dependent values mapped on validity horizons must be assigned to the correct planning period in the optimization model (part of the model formulation). Furthermore, the validity horizons are prohibited from overlapping for primary keys to avoid unique constraint issues (part of the business rules in the data storage).

B.2. Probabilistic demand stored in simulation scenarios

SimulationInstanceId	MaterialId	DeliveryDate	Quantity
Sim1	Material1	2021-01-04	100
Sim2	Material1	2021-01-04	110
Sim1	Material1	2021-01-05	100
Sim2	Material1	2021-01-05	90
...	...	...	...

Figure 2: Storage of simulation scenarios for demand in spreadsheet *Demand*

Pharmaceutical tablets demand is uncertain. Almost all approaches from the literature simulate sufficiently many scenarios and search for efficient lot sizes across all scenarios. This data becomes large even on medium-sized problem instances since each demand scenario might increase the dataset by multiple. Demand scenarios are represented by a scenario instance identified by a unique ID in the developed spreadsheets. These instances are listed in spreadsheet *SimulationInstance* and referenced in spreadsheet *Demand*. Figure 2 illustrates the demand scenario storage in entity *Demand*. Simulation instances *Sim2* represent a demand shift of 10 units of material *Material1* from 2021-01-05 to 2021-01-04, while simulation instance *Sim1* assumes a constant demand of 100 across both months. The template supports demand scenarios stored in daily date buckets. Scenario representation becomes intuitive and easy for the end-users to fill out. However, the price for that is that daily demand has to be mapped and aggregated to model planning periods (part of model formulation).

B.3. Multi-level BOM representation

BOMHeaderId	MachineId	MaterialId	ValidityDateFrom	ValidityDateTo	ProductionTime	...
Header1	Machine1	Material1	2021-01-01	2021-12-31	0.05	...
...						

(a) Spreadsheet extract of *BOMHeader*

BOMHeaderId	BOMItemId	MaterialId	Ratio	...
Header1	Item1	Material2	1.0	
Header1	Item2	Material2	2.0	
...				

(b) Spreadsheet extract of *BOMItem*

Figure 3: Representations of multi-level production structures

Multi-level production structures are complex and usually stored in a multi-level BOM in planning systems. They contain production-relevant information

and definitions of production relationships stored in the BOM header and BOM item entity, respectively. This representation provides a unified structure to describe different multi-level production processes. The developed spreadsheets follow this approach. They strictly separate production activities from product dependencies by the entities *BOMHeader* and *BOMItem*. Figure 3 shows that representation. A material *Material1* can be produced in the entire year 2021 on machine *Machine1*. For one production unit of *Material1*, the machine requires 0.05 days. If *Material1* is issued by *Header1*, then two ingredients *Material2* and *Material3* are issued by the production with 1.0 and 2.0 units, respectively.

C. Modeling Techniques

The MLCLSP-L-B is a complex MIP that relies on different data sources stored in a structured format, such as demand, capacity, material cost, production cost, setup matrix, and the BOM. A list of all developed entities and virtual tables is provided in repository documentation of Simonis (2024). Appendix D documents the mapping rules of the optimization model indices, sets, and coefficients to the RDS components This section organizes the applied modeling techniques as follows: First, it introduces the raw data template for the MLCLSP-L-B with probabilistic demand and how to build an Entity Relationship (ER) model. Second, it presents how to develop the Extended ER (EER) model to cover all required data preparation tasks.

C.1. Entity Relationship Model

Simonis and Nickel (2023a) provided a data template developed in MS Excel spreadsheets to store the metadata information of the MLCLSP-L-B with probabilistic demand in 10 spreadsheets. They are named *Material*, *Capacity*, *ProblemInstance*, *SimulationInstance*, *Demand*, *MaterialCost*, *SetupMatrix*, *BOMHeader*, *BOMItem*, and *InitialLotSizingValues*. Complexity and hierarchy drivers of the data are the amount of data (machines, materials, and periods), demand scenarios, and multi-level production structures. Cunha et al. (2009) studied the relationship between non-hierarchical spreadsheets and RDS. A rule framework was created that defines the design of an ER model by the data representation of the spreadsheets. The authors created an ER model entity for each spreadsheet with primary and foreign keys. For each spreadsheet defined in Simonis and Nickel (2023a), a persistent table is created consistent with the spreadsheet's name, column names, column data types, and data relationships. An ER diagram of persistent tables in the ER model is shown in Figure 4. In terms of simplicity, not all relationships between production and simulation instances are listed. The entities *ProblemInstance* and *SimulationInstance* are the baseline of the relational data model. Each defined problem instance has multiple simulation

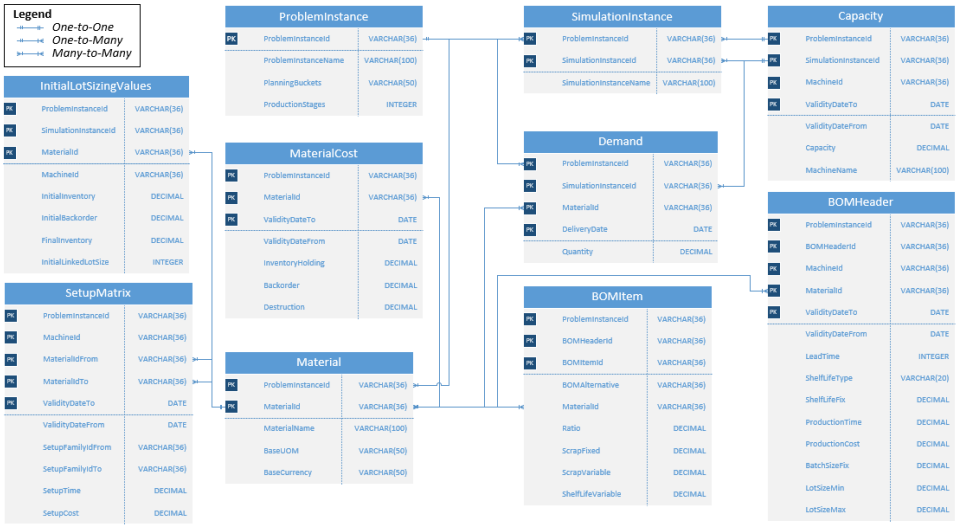

Figure 4: ER diagram of persistent tables

instances assigned. For each problem instance, materials (*Material*), material-related costs (*MaterialCost*), machines and product-to-machine assignments (*BOMHeader*), setup structure (*SetupMatrix*), and product structure (*BOMItem*) are set. Simulated scenarios are supported for demand (*Demand*), machine capacities (*Capacity*), and initial planning values (*InitialLotSizingValues*).

C.2. Extended Entity Relationship Model

Based on the ER model from the previous section, the persistent entities of the ER model store only metadata information of the MLCLSP-L-B. Thus, data transformations must be executed to meet the requirements of MLCLSP-L-B instantiation while ensuring that data can be processed at scale. Data transformations are developed with virtual tables defined by Structured Query Language (SQL) queries. An overview of the EER model is shown in Figure 5. The EER model consists of the 10 persistent tables (blue shaded rectangles) outlined in the previous section and, in addition, 16 virtual tables (green shaded rectangles). The circles represent join operations whereby the letter "I", "L", and "R" represent inner, left outer, and right outer joins, respectively. Grouping operations are denoted by the "GROUP BY" label. In the following, the most essential virtual table modeling techniques are described: First, two essential virtual tables that transform validity horizons into a multi-period representation are described in detail. Second, the mapping and aggregation of demand scenarios into a multi-period representation are outlined. Third, the material cost mapping based on

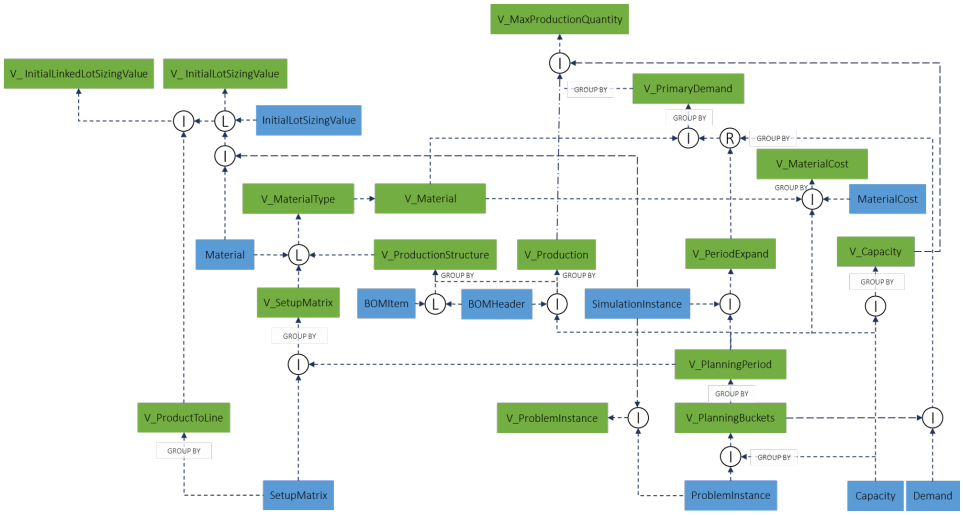

Figure 5: EER model with persistent and virtual tables

validity horizons is explained.

Map Validity Horizons to Planning Periods. The following describes the transformation from validity horizons to multi-period representations in detail. These queries are a central transformation step on which almost all other virtual table definitions depend. The MLCLSP-L-B represents time by periods within the planning horizon. The planning horizon is determined by a start and end date, and each period has a particular duration (planning bucket). The query *V_PlanningBuckets* answers the question, "Which planning buckets, start and end date of the planning horizon are available for problem instance?". Figure 6 defines the virtual table. The planning horizon is determined by the minimum

```
CREATE VIEW "V_PlanningBuckets" AS
SELECT t_pi."ProblemInstanceId", t_pi."PlanningBuckets", j_c."PlanningEndDate", j_c."PlanningStartDate",
   CASE
      WHEN UPPER(t_pi."PlanningBuckets") LIKE '%DAY' THEN 'DAY'
      WHEN UPPER(t_pi."PlanningBuckets") LIKE '%WEEK' THEN 'WEEK'
      WHEN UPPER(t_pi."PlanningBuckets") LIKE '%MONTH' THEN 'MONTH'
      WHEN UPPER(t_pi."PlanningBuckets") LIKE '%QUARTER' THEN 'QUARTER'
      WHEN UPPER(t_pi."PlanningBuckets") LIKE '%YEAR' THEN 'YEAR'
      ELSE 'DAY' END AS "PlanningBucketType"
FROM "ProblemInstance" AS t_pi
INNER JOIN (
   SELECT t_c."ProblemInstanceId", MAX(t_c."ValidityDateTo") AS "PlanningEndDate",
   MIN(t_c."ValidityDateFrom") AS "PlanningStartDate"
   FROM "Capacity" AS t_c
   GROUP BY t_c."ProblemInstanceId"
) AS j_c ON t_pi."ProblemInstanceId" = j_c."ProblemInstanceId";
```

Figure 6: Syntax definition of query *V_PLanningBuckets*

and maximum validity dates of the capacity for each problem instance from table *Capacity* (line 12-15). This information is merged with table *ProblemInstance* based on the identifier *ProblemInstanceId* (line 10, 11, and 16). Based on defined planning buckets, the type is derived by case-when statement. The query supports the types *'DAY'*, *'WEEK'*, *'MONTH'*, *'QUARTER'*, and *'YEAR'* as duration of a period (line 3-9). Finally, the problem instance (*ProblemInstanceId*), planning buckets (*PlanningBuckets*), start and end horizon dates (*PlanningStartDate* and *PlanningEndDate*), and the planning bucket type (*PlanningBucketType*) are selected from the query.

The query *V_PlanningPeriod* answers the question, "Which model period is mapped on which planning date for a problem instance within the planning horizon?". Figure 7a shows the syntax of the query. An illustrative example of the data transformations is summarized in Figure 7b. The nested query starts

```
1  CREATE VIEW "V_PlanningPeriod" AS
2  SELECT s_v_pb."ProblemInstanceId", s_v_pb."PlanningStartDate", s_v_pb."PlanningEndDate",
3     s_v_pb."PlanningBuckets", s_v_pb."PlanningDate",
4     DATE(s_v_pb."PlanningDate" + s_v_pb."PlanningBuckets"::INTERVAL - '1 DAY'::INTERVAL) AS "PlanningDateIntervalEnd",
5     ROW_NUMBER() OVER(PARTITION BY s_v_pb."ProblemInstanceId" ORDER BY s_v_pb."PlanningDate") AS "PlanningPeriod"
6  FROM
7  (
8     SELECT v_pb."ProblemInstanceId", v_pb."PlanningStartDate", v_pb."PlanningEndDate",
9     v_pb."PlanningBuckets", DATE(DATE_TRUNC(
10       v_pb."PlanningBucketType",
11       DATE(GENERATE_SERIES(
12          v_pb."PlanningStartDate",
13          v_pb."PlanningEndDate",
14          v_pb."PlanningBuckets"::INTERVAL)))) AS "PlanningDate"
15    FROM "V_PlanningBuckets"  AS v_pb
16    GROUP BY v_pb."ProblemInstanceId", v_pb."PlanningStartDate", v_pb."PlanningEndDate",
17       v_pb."PlanningBuckets", v_pb."PlanningBucketType"
18 ) AS s_v_pb;
```

(a) Syntax definition

Problem InstanceId	PlanningStartDate	PlanningEndDate	PlanningBuckets	PlanningDate	PlanningDate IntervalEnd	PlanningPeriod
Problem1	2021-01-01 MIN(Capacity. ValidityDateFrom)	2021-06-30 MAX(Capacity. ValidityDateTo)	2 MONTH (V_PlanningBuckets. PlanningBuckets)	2021-01-01	2021-02-28	1
				2021-03-01	2021-04-30	2
				2021-05-01	2021-06-30	3
				DATE_TRUNC(-) GENERATE_SERIES(-)	PlanningDate + PlanningBuckets::Interval – 1 Day	ROW_NUMBER(-)

(b) Illustration data transformations

Figure 7: Details for query *V_PlanningPeriod*

with "Get for each problem instance the planning buckets and types, and start and end date of the planning horizon." from virtual view *V_PLanningBuckets* (line 15). This view provides, for each problem instance, the planning start and end date and the date buckets. Next, this information is used by a sub-query that answers the question "Which planning dates are valid for associated planning buckets and a problem instance within the planning horizon?" (line 8-17).

It uses the *GENERATE_SERIES* function (line 11). The function has the inputs *start* (start timestamp), *stop* (stop timestamp), and *step* (interval buckets of the time-series generation). It returns a set of timestamps from *start* to *stop* with frequency *step*. The output of the function *GENERATE_SERIES* is formatted by the *DATE_TRUNC* function (line 10). The function takes the inputs *field* (valid value for truncate) and *source* (set of timestamps). It returns a truncated set of timestamps (e.g., the start of day, week, month, quarter, or year). Finally, the virtual table *V_PlanningPeriod* selects the problem instance (*ProblemInstanceId*), planning buckets (*PlanningBuckets*), start and end horizon dates (*PlanningStart-Date* and *PlanningEndDate*), and determines the planning end of each interval bucket (*PlanningDateIntervalEnd*) and planning period (*PlanningPeriod*) by the row number partitioned by the problem instances and ordered by the planning date (line 1-7).

Map and Aggregate Demand Scenarios to Planning Periods. The query *V_Demand* answers the question, "How much demand of a material is expected per planning period for a problem and simulation instance?". Figure 8a shows the syntax of the query. Furthermore, code transformations are illustrated in Figure 8b and Figure 8c. The nested query starts with "How much demand of a material is expected in a particular delivery date truncated on date buckets" (line 11-18). The *DATE_TRUNC* is used to format the column *DeliveryDate* in entity *Demand* against the *PlanningBucketType* provided by the view *V_PlanningBuckets*. After this operation, the demand quantity is summed over the problem and simulation instance ID, material ID, and the truncated delivery dates. Next, the demand per planning bucket is right-outer joined with the *V_PeriodExpand* (equals the view *V_PlanningPeriod*, but contains additionally the simulation instance and material id for all planning periods) view on the keys problem and simulation instance, planning date, and material id (line 20-22) whereby the *DeliveryDate* is joined with an *BETWEEN* condition on the interval *PlanningDate* and *PlanningDateIntervalEnd*. Suppose no demand quantity was defined for a particular planning period. In that case, the right outer join creates an observation that associates a delivery date and demand quantity of NULL with such a planning period. Thus, the resulting sub-query contains a demand quantity for each planning period, material, problem, and simulation instance. Base units and currencies are mapped from entity *Material* by an inner join on problem instance id and material (line 23-24). Next, this information is used by a sub-query that answers the question, "Which demand quantity per material is available for the planning periods across all problem and simulation instances?" (line 6-9). It transforms NULL values for the demand quantity to 0 and renames the mapped planning date from the multi-period grid to the delivery date. Finally, the origin question

```
1  CREATE VIEW "V_PrimaryDemand" AS
2     SELECT s_t_d."ProblemInstanceId", s_t_d."SimulationInstanceId", s_t_d."MaterialId",
3     s_t_d."DeliveryDate", SUM(s_t_d."Quantity") AS "Quantity", s_t_d."PlanningPeriod",
4     s_t_d."BaseUOM", s_t_d."BaseCurrency"
5     FROM (
6        SELECT j_pe."ProblemInstanceId", j_pe."SimulationInstanceId", j_pe."MaterialId",
7           j_pe."PlanningDate" AS "DeliveryDate", j_pe."PlanningPeriod",
8           j_v_m."BaseUOM", j_v_m."BaseCurrency",
9           CASE WHEN j_t_d."Quantity" IS NULL THEN 0 ELSE j_t_d."Quantity" END AS "Quantity"
10       FROM (
11          SELECT t_d."ProblemInstanceId", t_d."SimulationInstanceId", t_d."MaterialId",
12             DATE(DATE_TRUNC(j_v_pp."PlanningBucketType", t_d."DeliveryDate")) AS "DeliveryDate",
13             SUM(t_d."Quantity") AS "Quantity"
14          FROM
15             "Demand" AS t_d
16          INNER JOIN "V_PlanningBuckets" AS j_v_pp ON t_d."ProblemInstanceId" = j_v_pp."ProblemInstanceId"
17          GROUP BY t_d."ProblemInstanceId", t_d."SimulationInstanceId", t_d."MaterialId",
18             DATE_TRUNC(j_v_pp."PlanningBucketType", t_d."DeliveryDate")
19       ) AS j_t_d
20       RIGHT JOIN "V_PeriodExpand" AS j_pe ON j_t_d."ProblemInstanceId" = j_pe."ProblemInstanceId" AND
21          (j_t_d."DeliveryDate" BETWEEN j_pe."PlanningDate" AND j_pe."PlanningDateIntervalEnd") AND
22          j_t_d."MaterialId" = j_pe."MaterialId" AND j_t_d."SimulationInstanceId" = j_pe."SimulationInstanceId"
23       INNER JOIN "V_Material" AS j_v_m
24          ON j_pe."ProblemInstanceId" = j_v_m."ProblemInstanceId" AND j_pe."MaterialId" = j_v_m."MaterialId"
25    ) AS s_t_d
26 GROUP BY s_t_d."ProblemInstanceId", s_t_d."SimulationInstanceId", s_t_d."MaterialId", s_t_d."DeliveryDate",
27    s_t_d."PlanningPeriod", s_t_d."BaseUOM", s_t_d."BaseCurrency";
```

(a) Syntax definition

Problem InstanceId	Simulation InstanceId	MaterialId	DeliveryDate	Quantity	PlanningBucketType	PlanningBuckets
Problem1	Simulatio1	Material1	2021-01-01 2021-01-07	100	MONTH (V_PlanningBuckets. PlanningBucketType)	2 MONTH (V_PlanningBuckets. PlanningBuckets)
			2021-01-01 2021-01-15	100		
			2021-02-01 2021-02-05	100		
			2021-06-01 2021-06-15	400		

DATE_TRUNC(·) SUM(·)

(b) Illustration data transformations (line 11-18)

Problem InstanceId	Simulation InstanceId	MaterialId	DeliveryDate	Quantity	PlanningDate	PlanningDateIntervalEnd	PlanningPeriod
Problem1	Simulation1	Material1	2021-01-01	200	2021-01-01	2021-02-28	1
			2021-02-01	100	2021-01-01	2021-02-28	
			NULL	0 NULL	2021-03-01	2021-04-30	2
			2021-06-01	400	2021-05-01	2021-06-30	3

CASE [...] WHEN [...] SUM(·)

V_PeriodExpand
RIGHT OUTER DeliveryDate BETWEEN PlanningDate AND PlanningDateIntervalEnd

(c) Illustration data transformations (line 6-9, 20-22)

Figure 8: Details for query *V_PrimaryDemand*

of *V_Demand* is answered by summing the demand quantity overall problem and simulation instances, materials, delivery dates, planning periods, base units, and currencies.

```
1  CREATE VIEW "V_MaterialCost" AS
2  SELECT t_mc."ProblemInstanceId", t_mc."MaterialId", j_v_pp."PlanningDate", j_v_pp."PlanningPeriod"
3     AVG(t_mc."InventoryHolding") AS "InventoryHolding",
4     AVG(t_mc."Backorder") AS "Backorder"
5  FROM "MaterialCost" AS t_mc
6  INNER JOIN "V_PlanningPeriod" AS j_v_pp ON t_mc."ProblemInstanceId" = j_v_pp."ProblemInstanceId" AND
7     (j_v_pp."PlanningDate" BETWEEN t_mc."ValidityDateFrom" AND t_mc."ValidityDateTo")
8  INNER JOIN "V_Material" AS j_v_m ON t_mc."ProblemInstanceId" = j_v_m."ProblemInstanceId" AND
9     t_mc."MaterialId" = j_v_m."MaterialId"
10 GROUP BY t_mc."ProblemInstanceId", t_mc."MaterialId", j_v_pp."PlanningDate", j_v_pp."PlanningPeriod";
```

(a) Syntax definition

Problem InstanceId	MaterialId	ValidityDate From	ValidityDate To	InventoryHolding	Backorder	PlanningDate	PlanningPeriod
Problem1	Material1	2021-01-01	2021-02-28	0.06	0.12	2021-01-01	1
		2021-03-01	2021-06-30	0.075	0.15	2021-03-01	2
						2021-05-01	3

AVG(·) AVG(·) V_PlanningPeriod INNER PlanningDate BETWEEN ValidityDateFrom AND ValidityDateTo

(b) Illustration data transformations

Figure 9: Details for query *V_MaterialCost*

Map Material Costs Based on Validity Horizons. Almost all MIP parameters are mapped from the data templates by validity horizons. As discussed in the previous sections, this keeps the raw data easily maintainable. The downside of validity horizons is that they must be mapped to the planning periods. This section explains this mapping based on the view *V_MaterialCost*. However, analog transformations are implemented in the views *V_Capacity*, *V_Production*, and *V_ProductStructures*. The query *V_MaterialCost* answers the question, "Which inventory and backorder costs are associated with each material per planning period and problem instance?". Figure 9a shows the syntax of the query. Furthermore, code transformations are illustrated in Figure 9b. The entity *MaterialCost* is merged with the view *V_PlanningPeriod* (line 6-7) on the problem instance id and the planning date that is between the validity horizon determined by columns *ValidityDateFrom* and *ValidityDateTo*. Next, the results are merged with the view *V_Material* (line 8-9) on the problem instance and material id. Finally, the data is grouped by the problem instance, material ID, planning date, and planning period to average inventory-holding and backorder costs (line 2-4).

D. Documentation of Optimization Model Instantiation

The mapping of model coefficients with their value representation from the EER model instantiates the MLCLSP-L-B with probabilistic demand. Table 3 shows a summary of mapping rules. The group index sets, capacity, demand, costs, production, and initial values are used to classify all coefficients. Then,

Class	Coefficient	EER model component	Column mapping rule	
			Value	Index
Index sets	$\mathscr{S}$	V_ProblemInstance	SimulationInstanceId	-
	$\mathscr{M}$	V_Capacity	MachineId	-
	$\mathscr{P}$	V_Material	MaterialId	-
	$\mathscr{T}$	V_PeriodExpand	PlanningPeriod	-
	$\mathscr{T}_0$	V_PeriodExpand	PlanningPeriod	-
	$\mathscr{P}_p^{suc}$	V_ProductionStructures	GoodsReceived	GoodsIssued
	$\mathscr{P}_m$	V_ProductToLine	MaterialId	MachineId
Capacity	$b_{s,m,t}$	V_Capacity	CapacityPerPeriod	SimulationInstanceId, MachineId, PlanningPeriod
	$t_{p,t}^{p}$	V_Production	ProductionTimePerBaseUOM	MaterialId, PlanningPeriod
	$t_{p,t}^{su}$	V_SetupMatrix	SetupTime	MaterialId, PlanningPeriod
Demand	$d_{s,p,t}$	V_PrimaryDemand	Quantity	SimulationInstanceId, MaterialId, PlanningPeriod
Costs	$c_{p,t}^{inv}$	V_MaterialCost	InventoryHolding	MaterialId, PlanningPeriod
	$c_{p,t}^{bo}$	V_MaterialCost	Backorder	MaterialId, PlanningPeriod
	$c_{p,t}^{su}$	V_SetupMatrix	SetupCost	MaterialId, PlanningPeriod
Production	$r_{p,q}$	V_ProductionStructures	Ratio	GoodsReceived, GoodsIssued
	t_p^{as}	V_Production	LeadTime	MaterialId
	$M_{s,m,p,t}$	V_MaxProductionQuantity	BigM	SimulationInstanceId, MachineId, MaterialId, PlanningPeriod
Initial values	$\tilde{x}_{s,p,0}^{inv}$	V_InitialLotSizingValues	InitialInventory	SimulationInstanceId, MaterialId
	$\tilde{x}_{s,p,0}^{bo}$	V_InitialLotSizingValues	InitialBackorder	SimulationInstanceId, MaterialId
	$\tilde{x}_{s,p,T}^{inv}$	V_InitialLotSizingValues	FinalInventory	SimulationInstanceId, MaterialId
	$\tilde{x}_{s,p,0}^{l}$	V_InitialLinkedLotSizingValues	InitialLinkedLotSize	SimulationInstanceId, MachineId, MaterialId

Table 3: Mapping rules for MLCLSP-L-B instantiation for a given problem instance

the EER model component that contains the mapping values is listed. Next, the mapping value is derived from the referenced technical column name. A referenced column reads the index value if the model coefficient has an index. Thus, the instantiation is executed in two steps: First, index sets are created. Required indices for scenarios, machines, products, periods, product successors, and machine allocations are read from *V_ProblemInstance*, *V_Capacity*, *V_Material*, *V_PeriodExpand*, *V_ProductionStructures*, and *V_ProductToLine*. Second, model coefficients are defined. Capacity-related model coefficients are read from views *V_Capacity*, *V_Production*, and *V_SetupMatrix*. Demand data

is collected by virtual tables *V_PrimaryDemand*. Cost-related coefficients are mapped by *V_MaterialCost* and *V_SetupMatrix*. Production-related model coefficients are set by the virtual tables *V_Production*, *V_ProductionStructure*, and *V_MaxProductionQuantity*. Finally, initial values are read from the virtual tables *V_InitialLotSizingValues* and *V_InitialLinkedLotSizingValues*. Mapping rules instantiate the MLCLSP-L-B with probabilistic demand. Afterward, it can be used by a solver API to run optimization procedures on the instantiated model.

References

Azizi, V., Hu, G., Mokari, M., 2020. A two-stage stochastic programming model for multi-period reverse logistics network design with lot-sizing. Computers & Industrial Engineering 143, 106397.

Cunha, J., Saraiva, J., Visser, J., 2009. From spreadsheets to relational databases and back, in: Proceedings of the 2009 ACM SIGPLAN workshop on Partial evaluation and program manipulation, pp. 179–188.

Dutta, G., Fourer, R., 2008. Database structure for a class of multi-period mathematical programming models. Decision Support Systems 45, 870–883.

Helber, S., Sahling, F., 2010. A fix-and-optimize approach for the multi-level capacitated lot sizing problem. International Journal of Production Economics 123, 247–256.

Hu, Z., Hu, G., 2016. A two-stage stochastic programming model for lot-sizing and scheduling under uncertainty. International Journal of Production Economics 180, 198–207.

Quadt, D., Kuhn, H., 2008. Capacitated lot-sizing with extensions: a review. 4OR 6, 61–83.

Simonis, M., 2024. PostgreSQL: Relational database structures application on capacitated lot-sizing for pharmaceutical tablets manufacturing processess. `https://github.com/MichaelSimois/mlclsplb_eerm/blob/software_impacts`. doi:`https://github.com/MichaelSimois/mlclsplb_eerm`. v1.1.

Simonis, M., Nickel, S., 2023a. Generalized data model for real-world capacitated lot-sizing problems with linked lot sizes and backorders. Data in Brief 49, 109440.

Simonis, M., Nickel, S., 2023b. A simulation–optimization approach for a cyclic production scheme in a tablets packaging process. Computers & Industrial Engineering 181, 109304.