University of Southampton

Optimisation of Box Allocation in Meal Kit Delivery

Student: Thi Nguyen

Project sponsor: Mr Loïc Genest

Academic supervisor: Dr Alain Zemkoho

Date: 1st Oct 2024

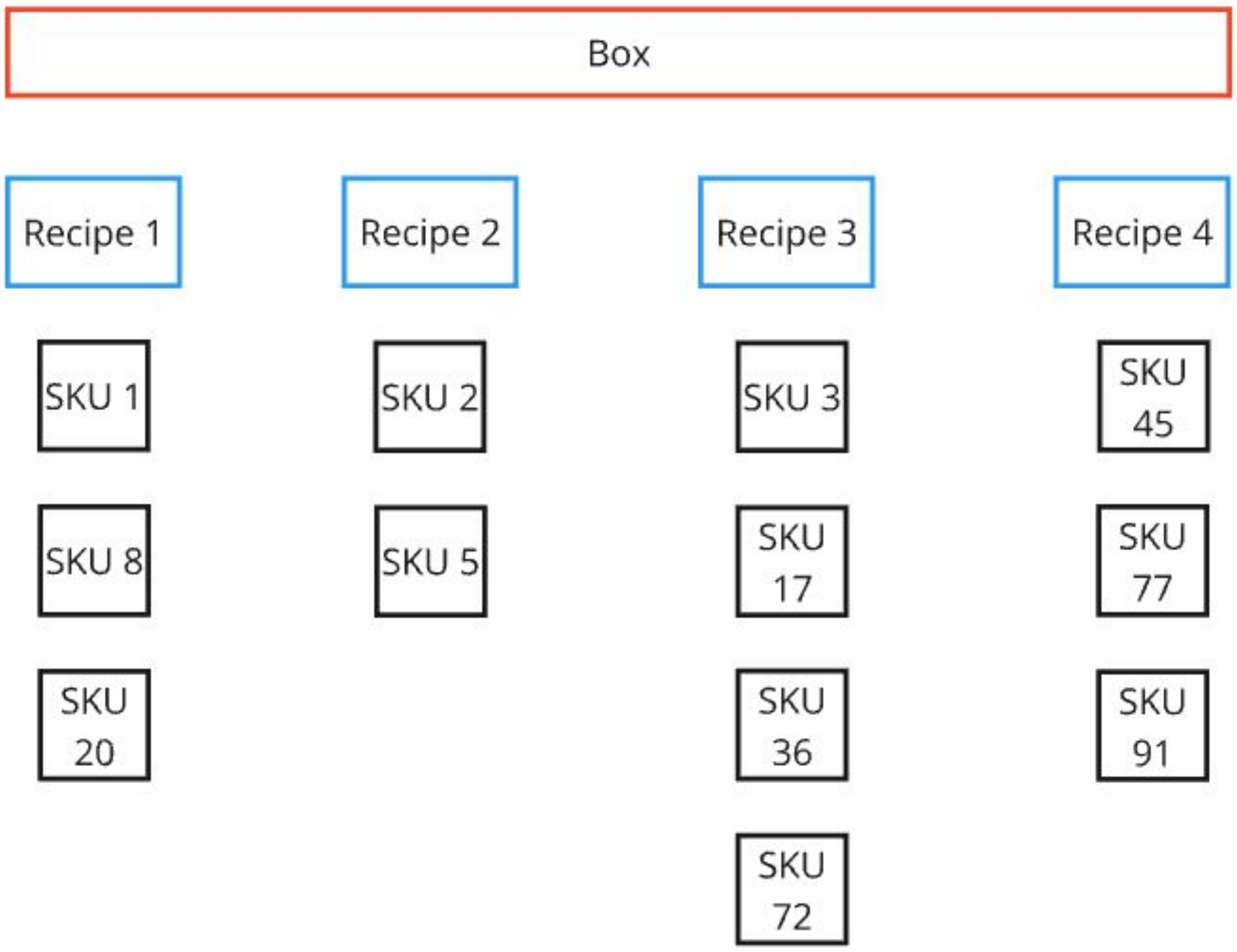
Contents

University of Southampton

Contents

Box structure

University of Southampton



Source: Loic Genest (Gousto)

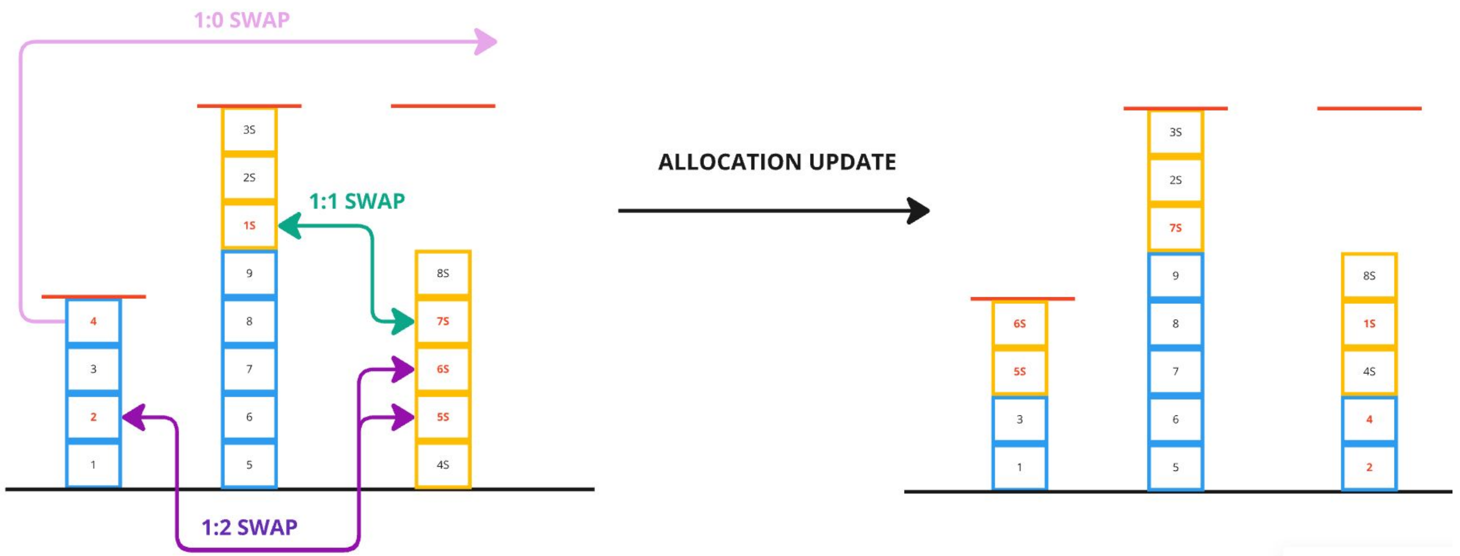
Source: Woop (2024)

Box allocation problem

1. Twice a day:

Source: Loic Genest (Gousto)

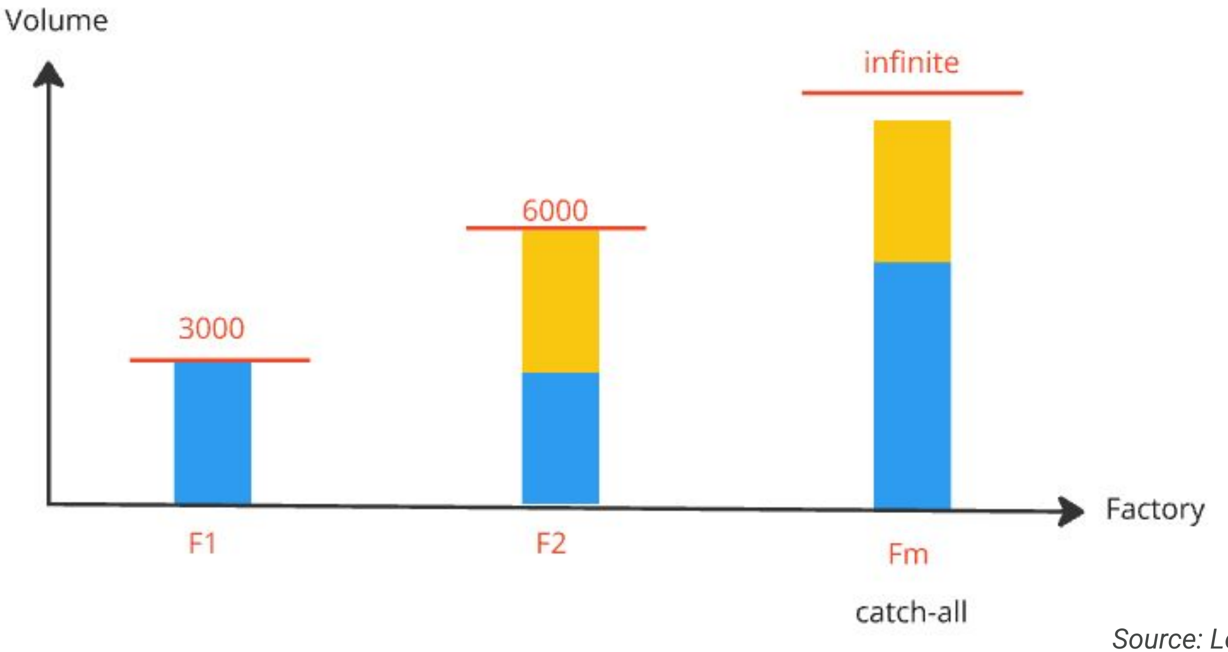
2. Optimize allocation decision by swapping orders between factories:



Source: Loic Genest (Gousto)

Constraints

1. Factory full capacity

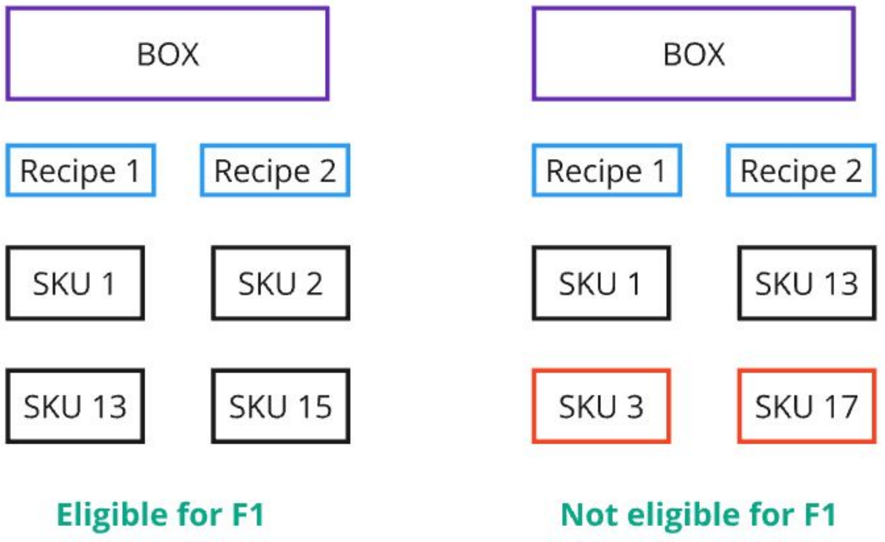


Source: Loic Genest (Gousto)

2. Recipe and SKU eligibility

Factory 1 has below ingredients:

- SKU 1
- SKU 2
- SKU 13
- SKU 15



Source: Loic Genest (Gousto)

Mathematical model

UNIVERSITY OF Southampton

Objective function

$$WMAPE_{site} : \frac{\sum_{i=1}^{N} \sum_{j=1}^{S} |A_{i,j} - F_{i,j}|}{\sum A_i}$$

$$\text{Minimize } f(x) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j,t} - a_{i,j,t-1}|}{\sum_{i=1}^{n} q_{i,t}} \tag{3.1}$$

subject to

$$\sum_{o=1}^{k} x_{o,j,t} = C_{j,t}, \quad \forall j \neq m, \forall t \tag{3.2}$$

$$x_{o,j,t} \leq \min_{i \in o} E_{i,j,t}, \quad \forall o, \forall j, \forall t \tag{3.3}$$

$$\sum_{j=1}^{m} x_{o,j,t} = 1, \quad \forall o, \forall t \tag{3.4}$$

$$x_{o,j,t} \in \{0, 1\}, \quad \forall o, \forall j, \forall t \tag{3.5}$$

No site involved *(Lower bound of $WMAPE_{site}$)*

$$WMAPE_{global} = \frac{\sum_{i=1}^{N} |A_i - F_i|}{\sum A_i}$$

Source: Loic Genest (Gousto)

The following notations are used in the formulation of BAP:

- t: Allocation day
- $N = \{1, 2, \ldots, n\}$: Set of recipes, indexed by i
- $S = \{1, 2, \ldots, m\}$: Set of factories, indexed by j
- $O = \{1, 2, \ldots, k\}$: Set of orders, indexed by o
- $a_{i,j,t}$: Number of times recipe i is allocated to factory j at day t
- $a_{i,j,t-1}$: Number of times recipe i is allocated to factory j at day $t-1$
- $q_{i,t}$: The quantity of recipe i at day t
- $q_{i,t-1}$: The quantity of recipe i at day $t-1$
- $x_{o,j,t}$: Binary decision variable indicating the allocation of order o to factory j at day t
- $C_{j,t}$: Capacity of factory j at day t
- $E_{i,j,t}$: Binary parameter indicating the eligibility of recipe i at factory j at day t

WMAPE calculation and its importance

University of Southampton

Recipe	Factory	Day -15	Day -14	Absolute site-recipe difference
1	F1	16	16	0
1	F2	0	0	0
1	F3	5	2	3
2	F1	15	15	0
2	F2	4	5	1
3	F1	3	3	0
3	F2	3	3	0
3	F3	2	3	1
4	F1	2	2	0
4	F2	2	2	0
4	F3	3	2	1
5	F1	0	0	0
5	F2	2	2	0
5	F3	3	3	0
6	F1	0	0	0
6	F2	30	27	3
6	F3	8	7	1
7	F1	0	0	0
7	F2	23	31	8
7	F3	5	5	0
8	F1	0	0	0
8	F2	23	30	7
8	F3	4	9	5
9	F1	0	0	0
9	F2	28	32	4
9	F3	11	5	6
10	F1	0	0	0
10	F2	0	0	0
10	F3	23	24	1
Sum			**232**	**41**
WMAPE site			**41 / 232 = 0.177**	

Recipe	Day -15	Day -14	Absolute recipe difference
1	21	18	3
2	19	20	1
3	8	9	1
4	7	6	1
5	9	9	0
6	38	34	4
7	28	36	8
8	27	39	12
9	39	37	2
10	23	24	1
Sum		**232**	**33**
WMAPE global		**33 / 232 = 0.142**	

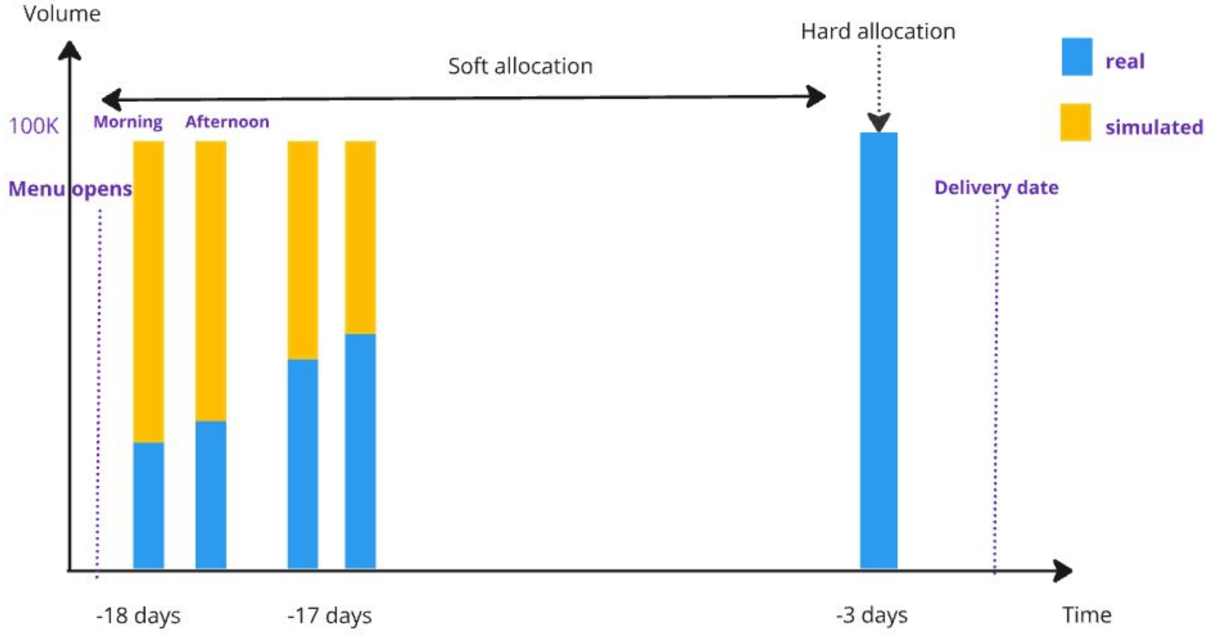
Minimizing WMAPE site is crucial for Gousto:

1. **Reduce waste:** consistent allocations, accurate forecasting, less over ordering, reduced spoilage.

2. **Ensure availability**: stable ingredient stocks, fewer stockouts, reliable order fulfillment.

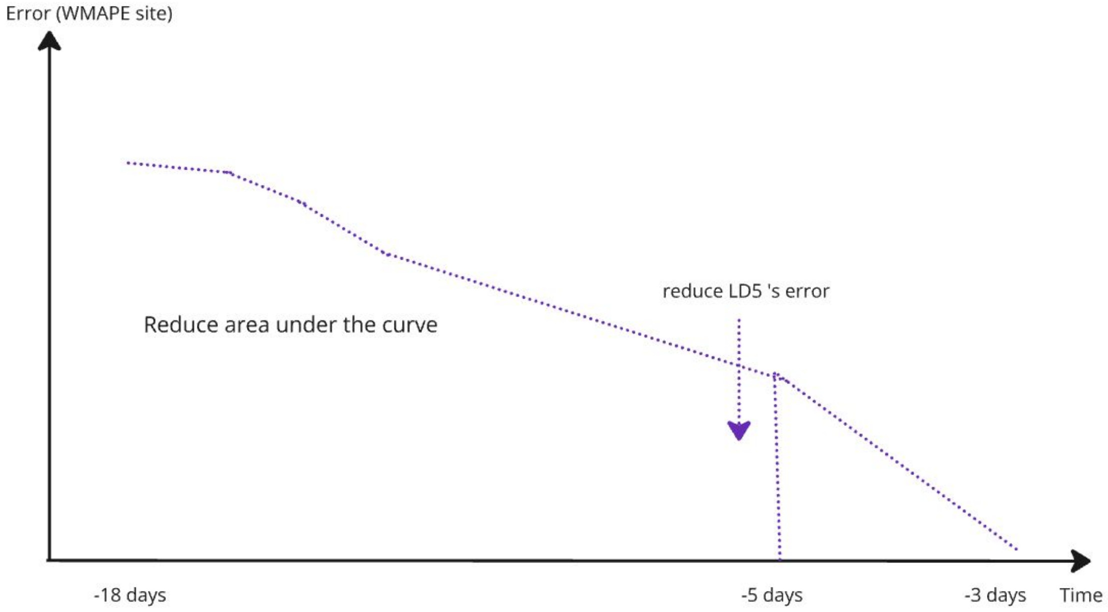
Proxy optimization

1. Temporal aspect:



Source: Loic Genest (Gousto)

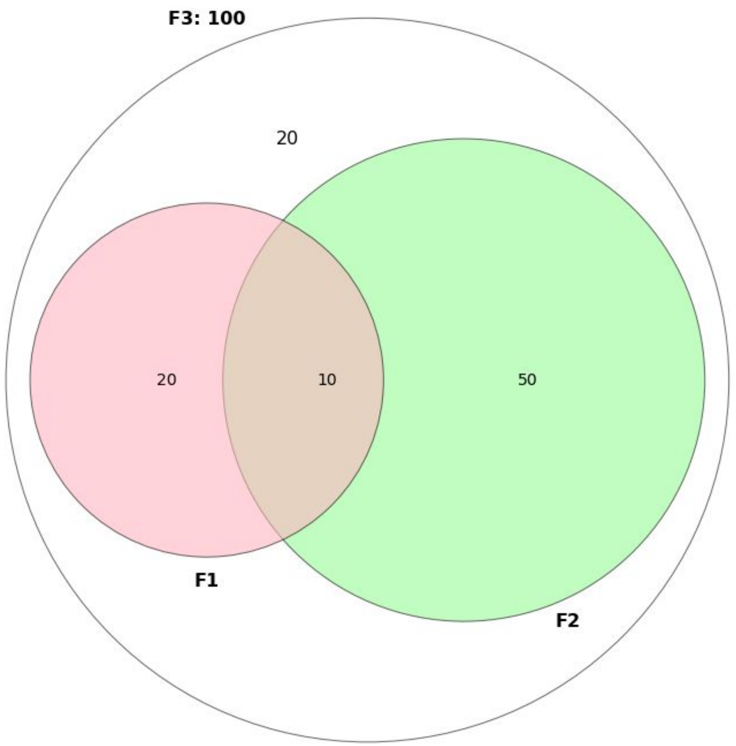
2. Ultimate goal:

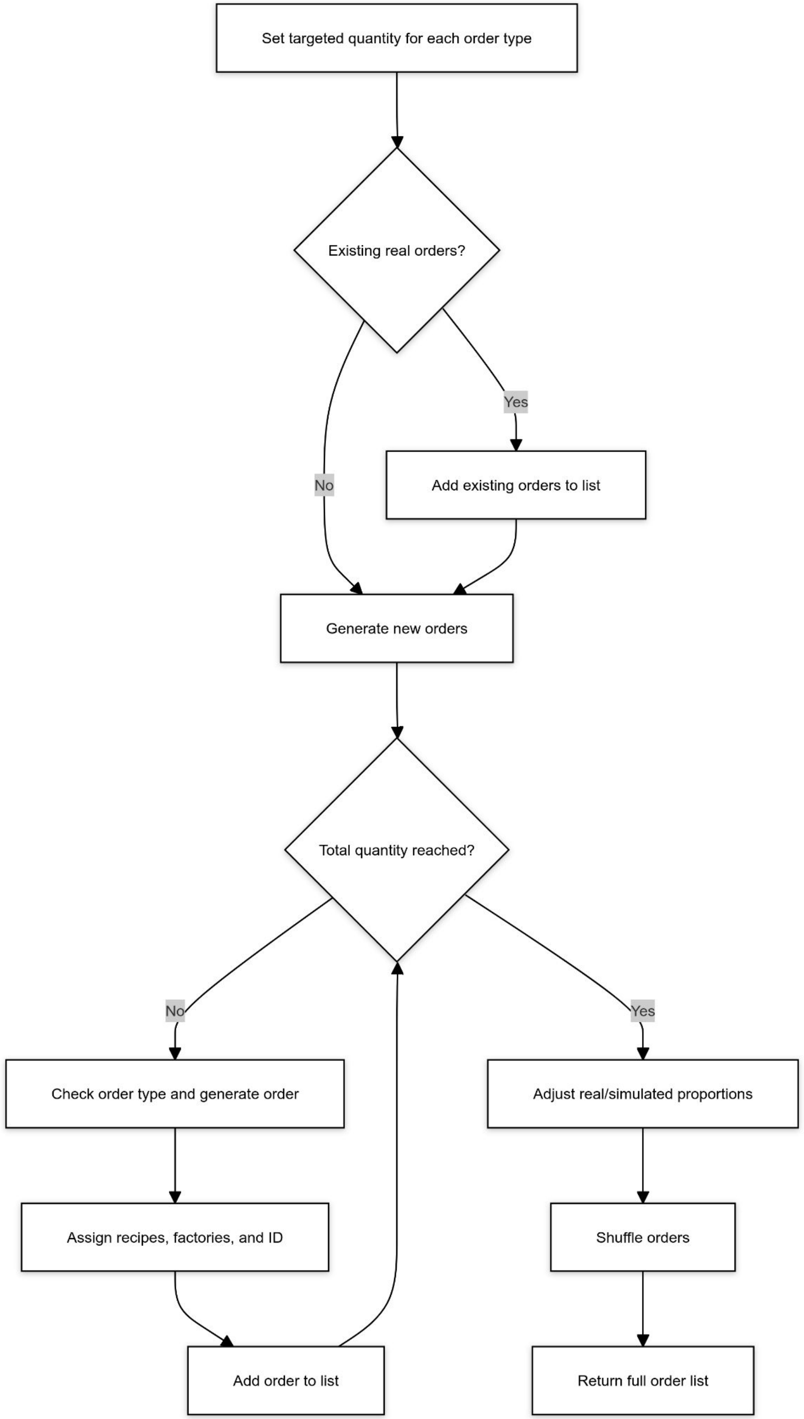


Source: Loic Genest (Gousto)

Testing data

- Generate simulated data
- Flexibility for testing
- Order distribution among factories: *30% F1, 60% F2*





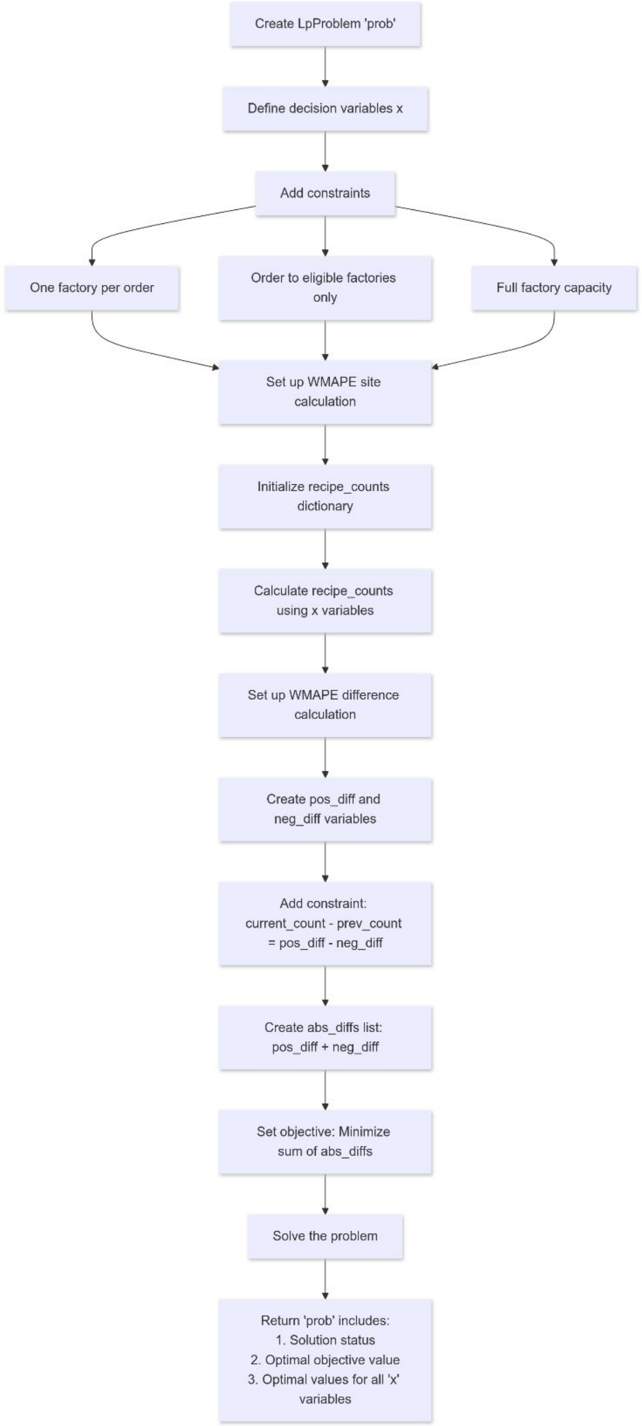
University of Southampton

Contents

Exact method

- CBC *(COIN-OR Branch and Cut)* solver: B&B with primal heuristics (Feasibility Pump, Coefficient Diving).

- Directly allocate current day's orders based on the previous day's allocation to minimize WMAPE_{site}.



Heuristics

Construct an initial feasible solution using a greedy algorithm:

- Eligible orders are first allocated to F1 until fulfilling F1's capacity.

- Allocate the remaining eligible orders to F2 until its capacity is reached.

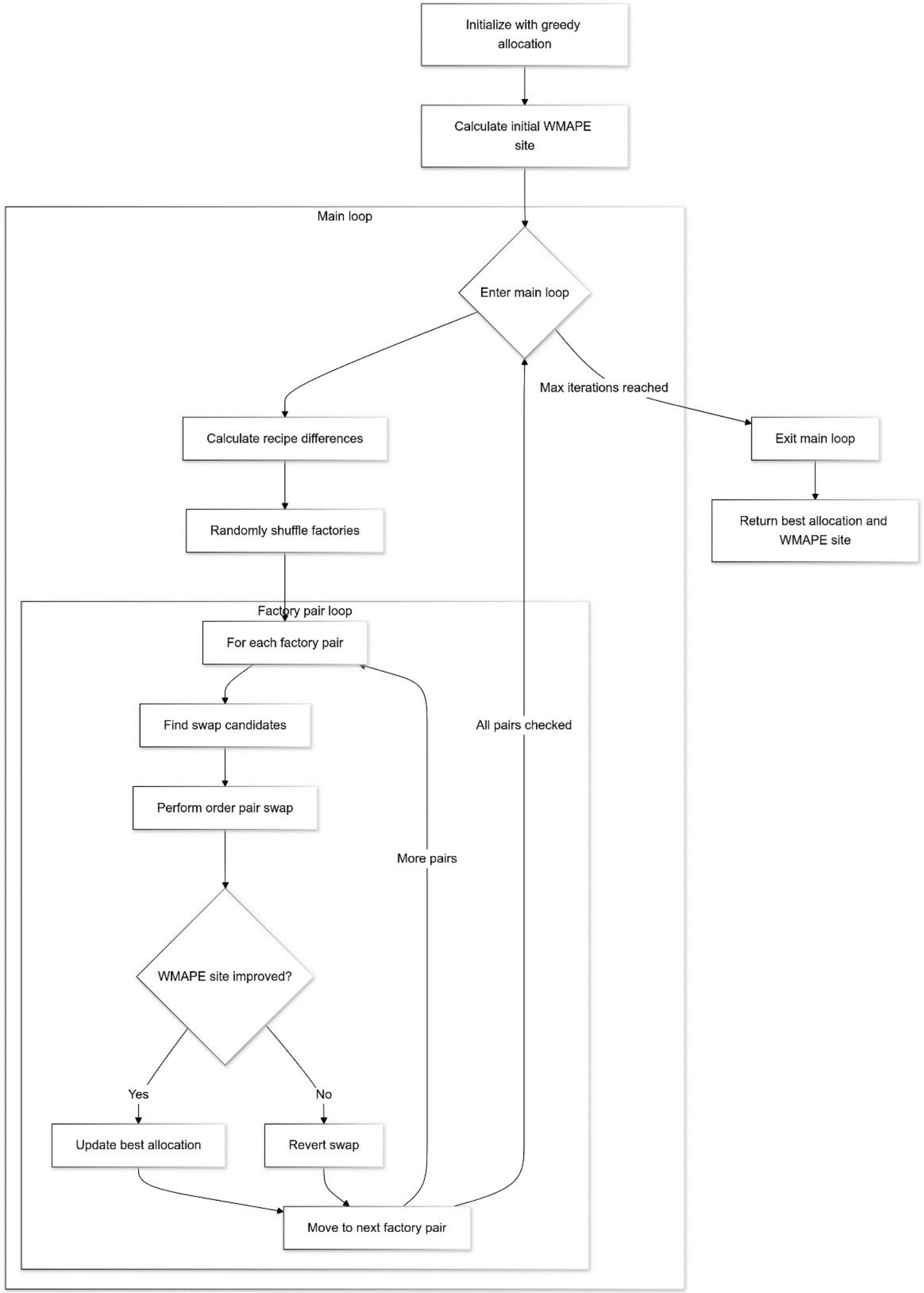
- All unallocated orders are assigned to F3 (catch-all factory).

⇨ Apply 2 proposed heuristics to improve this initial solution.

Iterative targeted pairwise swap (ITPS)

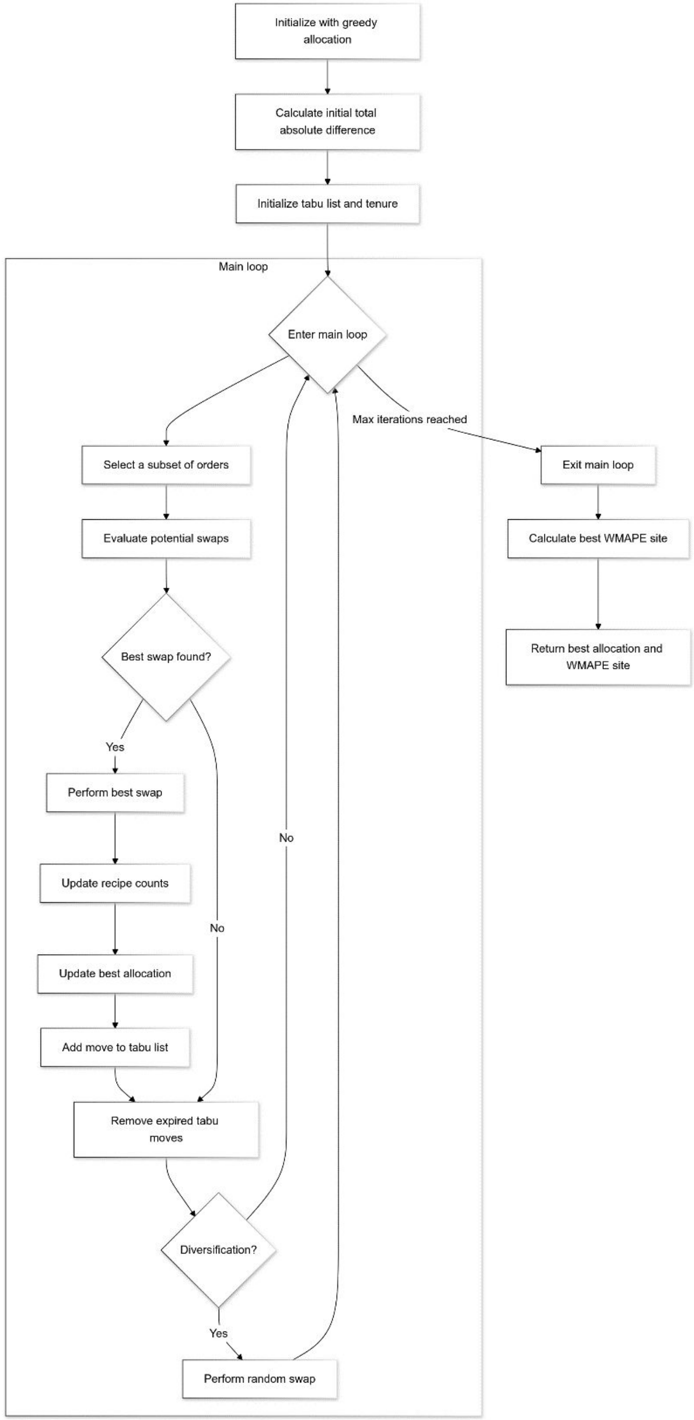
- Local search, inspired by 2-opt.

- Find beneficial swaps, then exchange 2 orders between 2 factories.



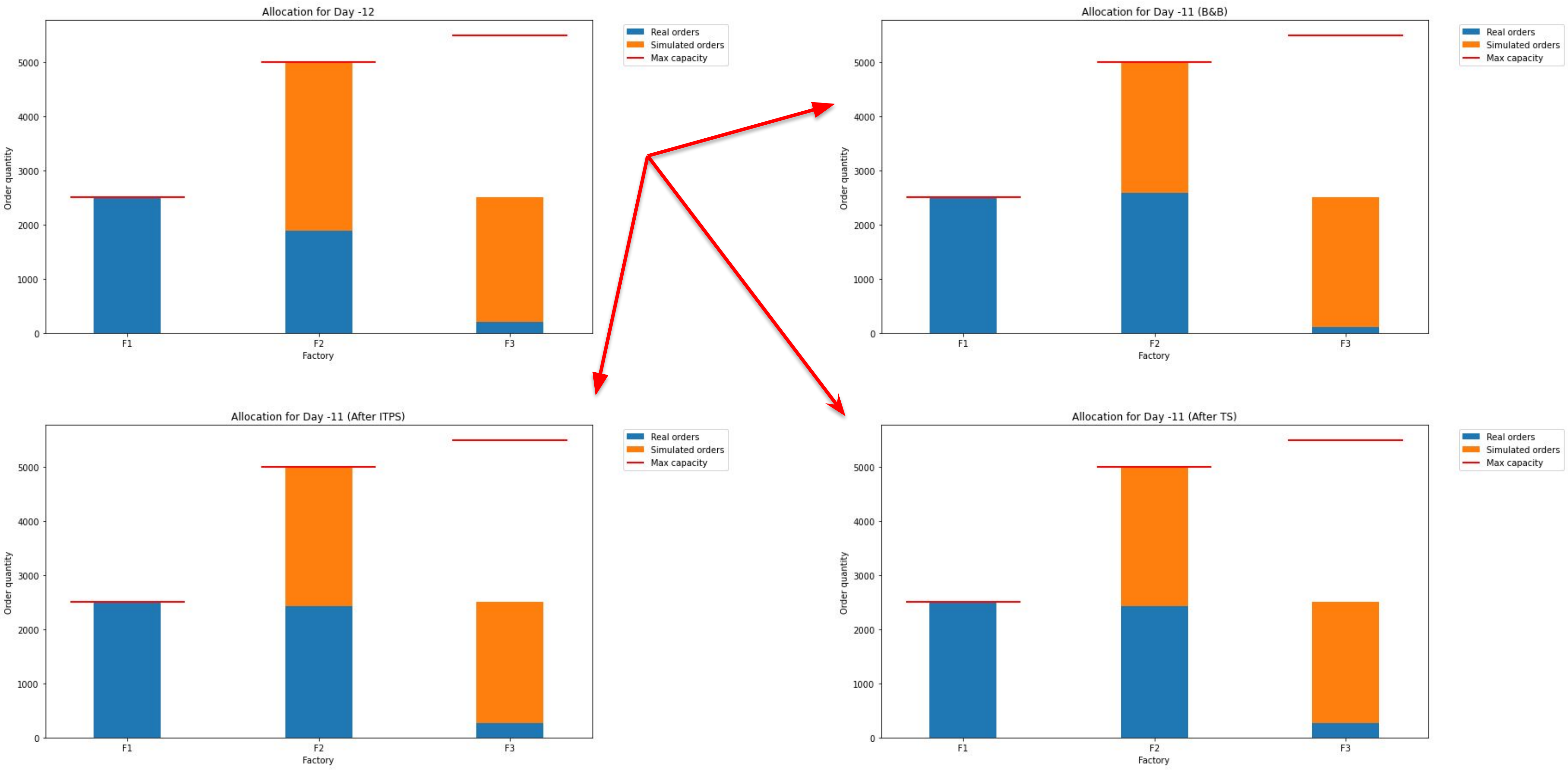
Tabu search (TS)

- Metaheuristic
- Uses adaptive memory to track previous moves and prevent revisits.
- Combines strategic exploitation (local search) and exploration.



Contents

Benchmark test *(LD12: 46% real ; LD11: 52% real)*



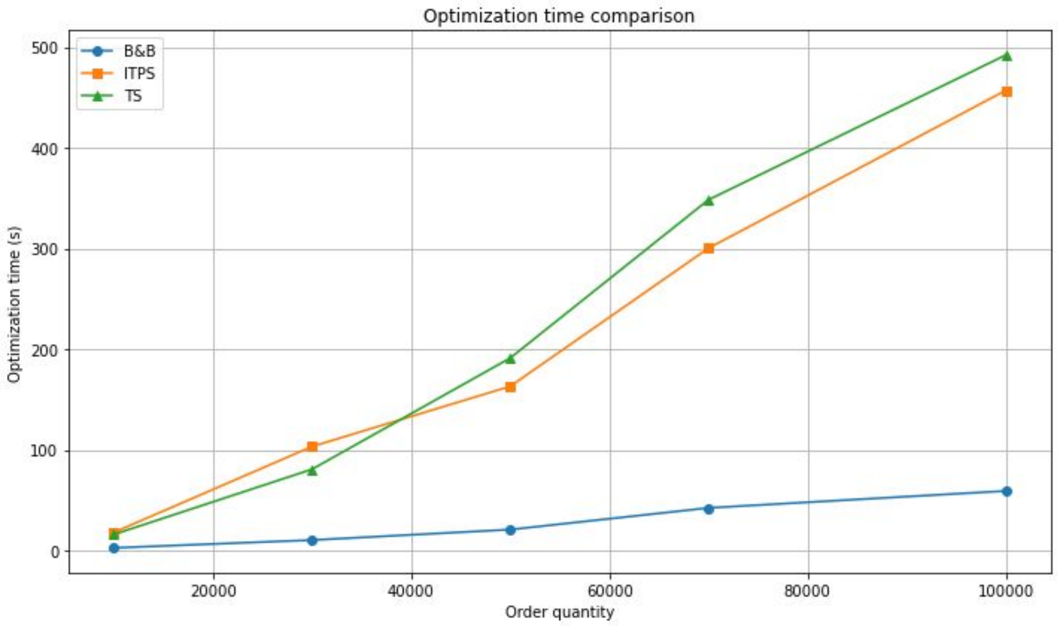
Benchmark test

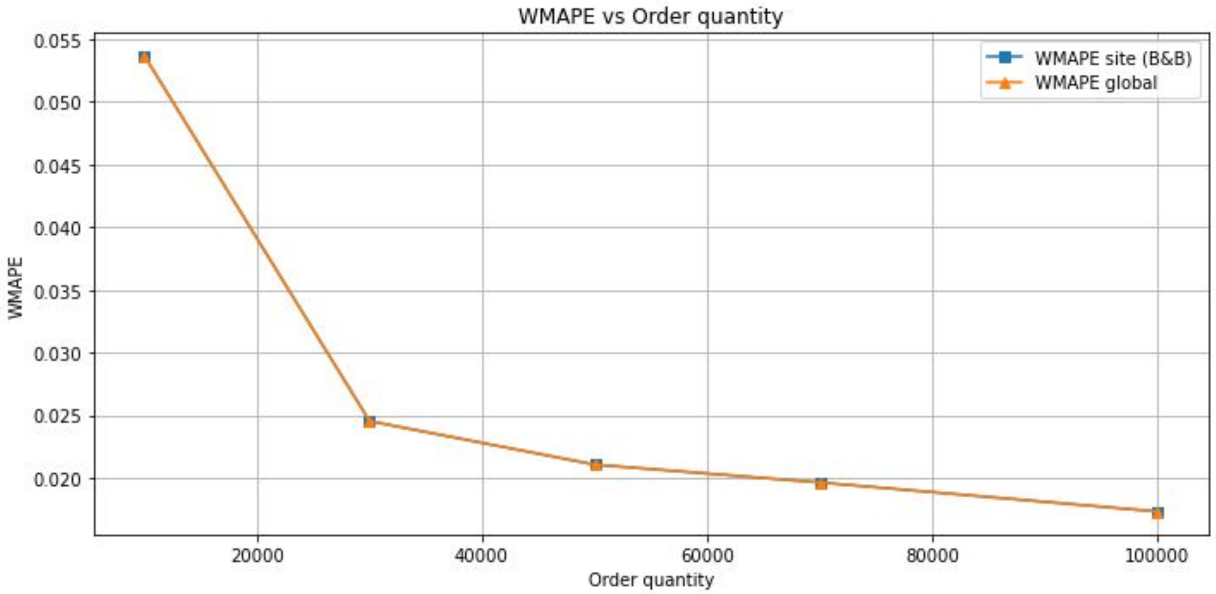
UNIVERSITY OF Southampton

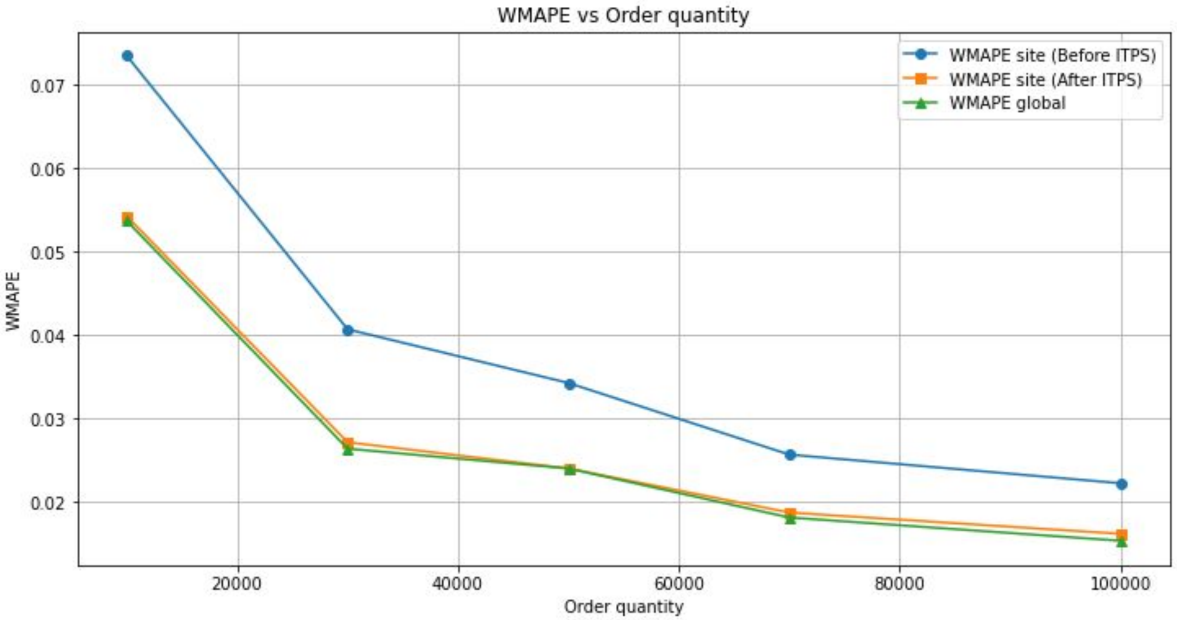
B&B	
LD12	
F1	2500 orders, 2500 real
F2	5000 orders, 1890 real
F3	2500 orders, 210 real
LD11	
F1	2500 orders, 2500 real
F2	5000 orders, 2593 real
F3	2500 orders, 107 real
WMAPE site	0.054
WMAPE global	0.054
Optimization time	2.89 seconds

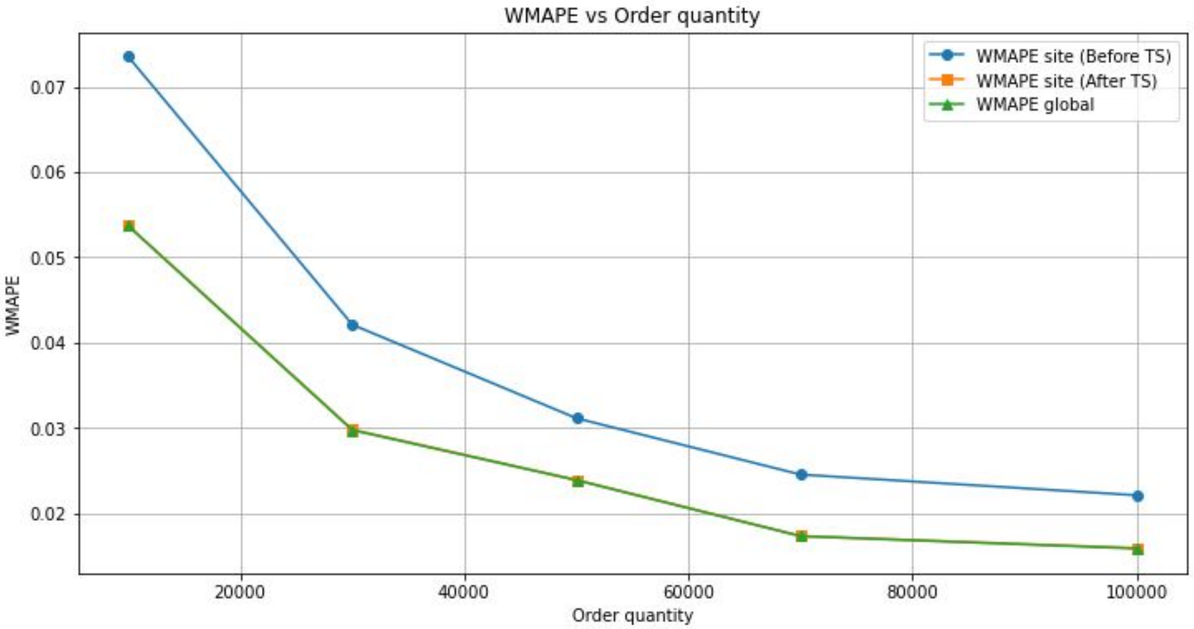
	ITPS	TS
LD12		
F1	2500 orders, 2500 real	
F2	5000 orders, 1890 real	
F3	2500 orders, 210 real	
LD11 (Before)		
F1	2500 orders, 2500 real	
F2	5000 orders, 2422 real	
F3	2500 orders, 278 real	
LD11 (After)		
F1	2500 orders, 2500 real	2500 orders, 2500 real
F2	5000 orders, 2436 real	5000 orders, 2429 real
F3	2500 orders, 264 real	2500 orders, 271 real
WMAPE site		
Before	0.074	0.074
After	0.054	0.054
Improvement	26.21%	26.97%
WMAPE global	0.054	
Optimization time	19.45 seconds	15.35 seconds

Scalability test









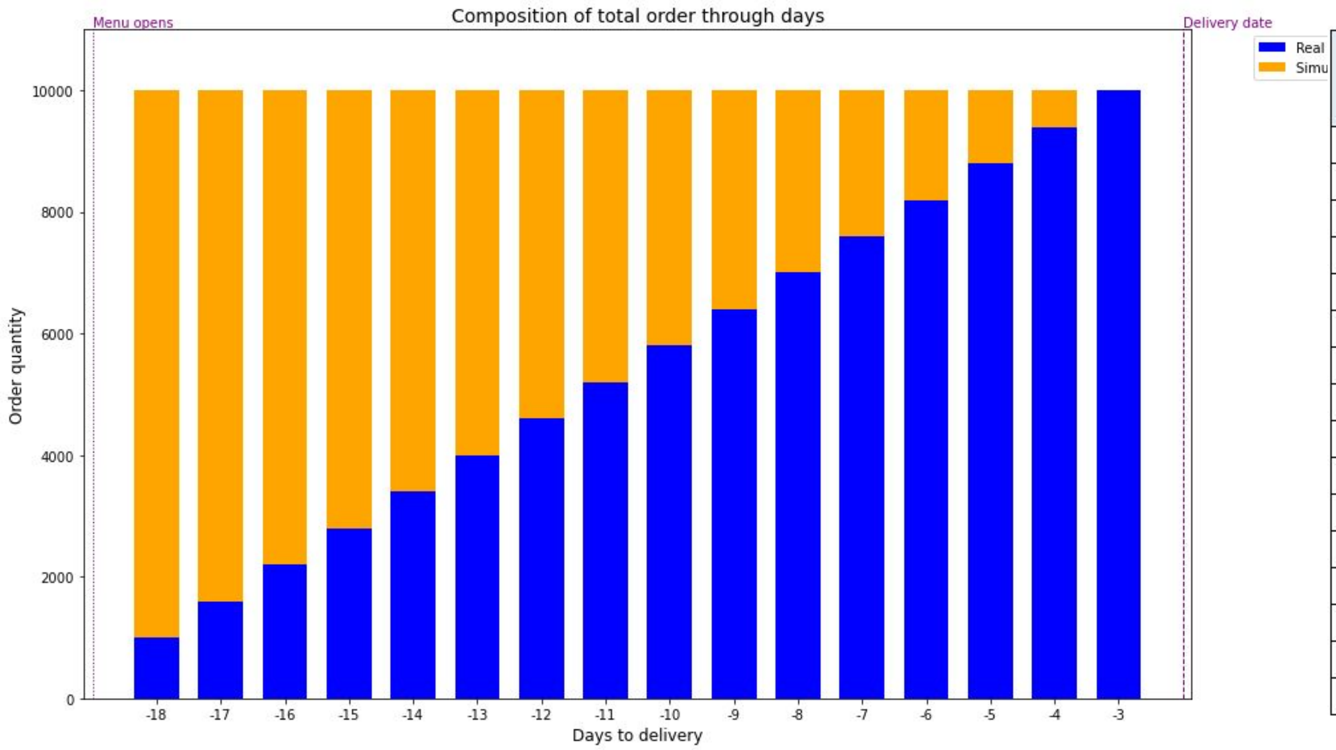
Aggregate forecasting principle

- Principle: “Forecasts are more accurate for groups or families of items rather than for individual items” *(Reid and Sander, 2012).*

- Reasons:
  - Individual variability cancels out in larger groups.
  - Larger quantities reduce the impact of individual fluctuations.

- Gousto application: lower WMAPE when order volumes increase.

Temporal test - No changes



Day	Real orders proportion	WMAPE site (B&B)	WMAPE site (Greedy)	WMAPE global	WMAPE site vs LD3	WMAPE global vs LD3
-18	10%	N/A	N/A	N/A	0.093	0.049
-17	16%	0.054	0.079	0.052	0.083	0.054
-16	22%	0.060	0.086	0.058	0.075	0.054
-15	28%	0.055	0.074	0.054	0.069	0.050
-14	34%	0.053	0.074	0.052	0.068	0.055
-13	40%	0.053	0.071	0.053	0.057	0.045
-12	46%	0.051	0.067	0.051	0.059	0.047
-11	52%	0.046	0.064	0.046	0.056	0.048
-10	58%	0.045	0.068	0.045	0.047	0.038
-9	64%	0.044	0.068	0.044	0.045	0.039
-8	70%	0.034	0.053	0.034	0.042	0.035
-7	76%	0.036	0.057	0.036	0.040	0.035
-6	82%	0.031	0.054	0.031	0.033	0.029
-5	88%	0.028	0.052	0.028	0.024	0.022
-4	94%	0.023	0.046	0.023	0.017	0.017
-3	100%	0.017	0.033	0	0	0

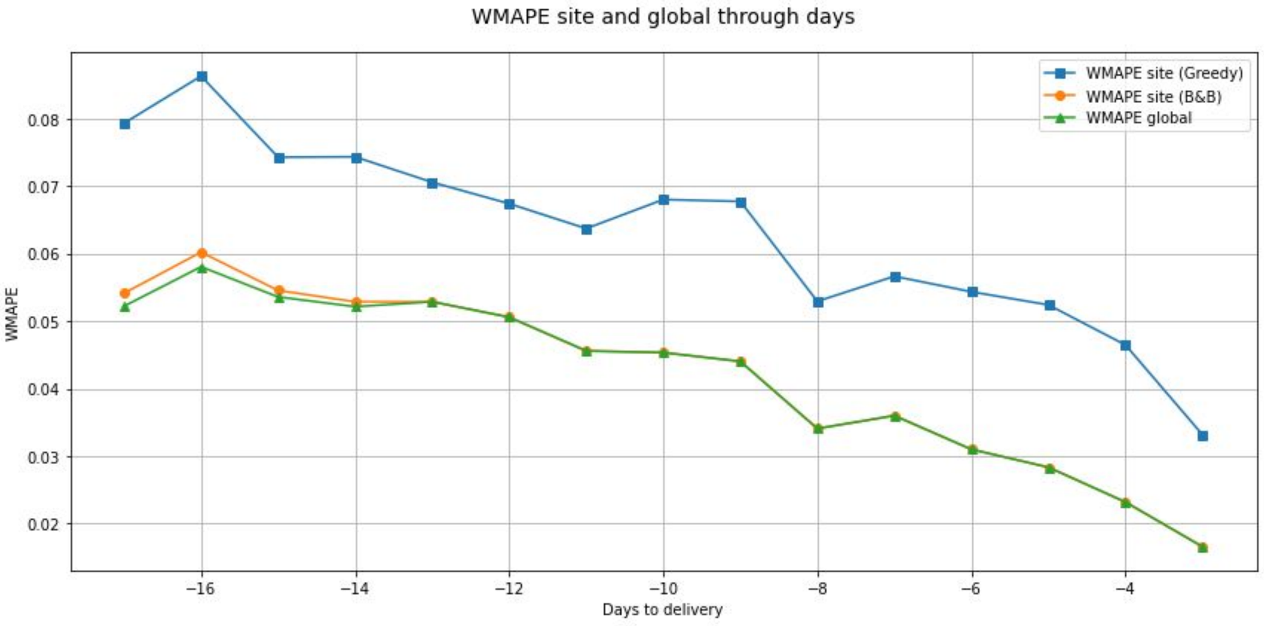
Temporal test - No changes

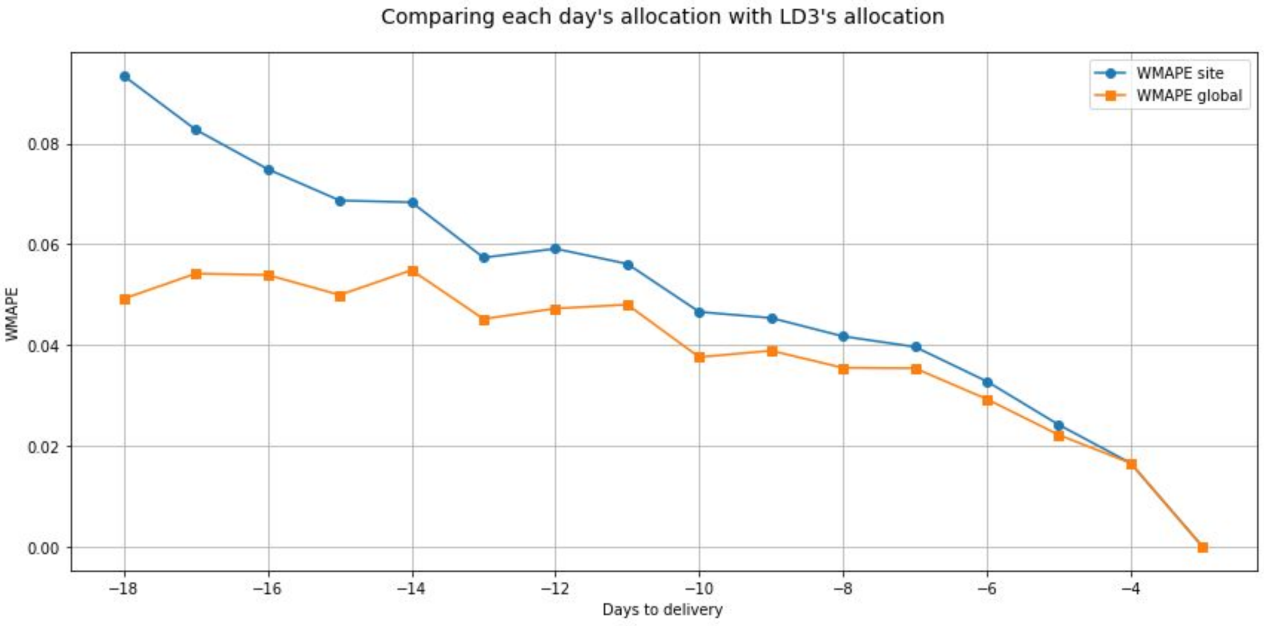
Minimize errors between consecutive days:

LD15's error is achieved by minimizing difference in recipe count with LD16 (previous day).

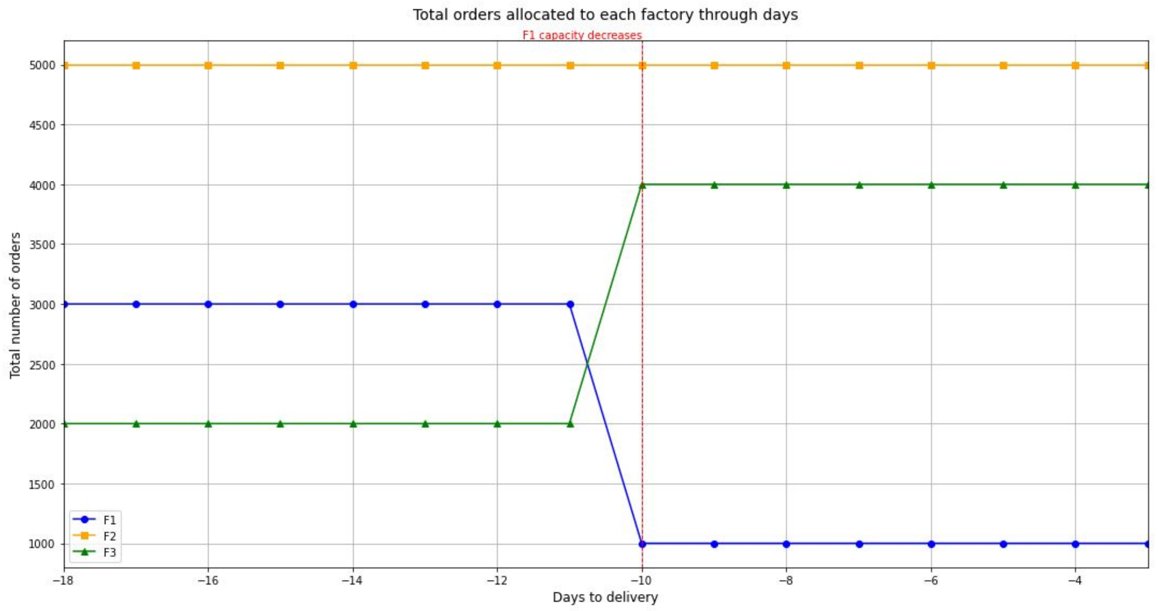
Compare actual allocation of all days with final day's allocation (B&B method):

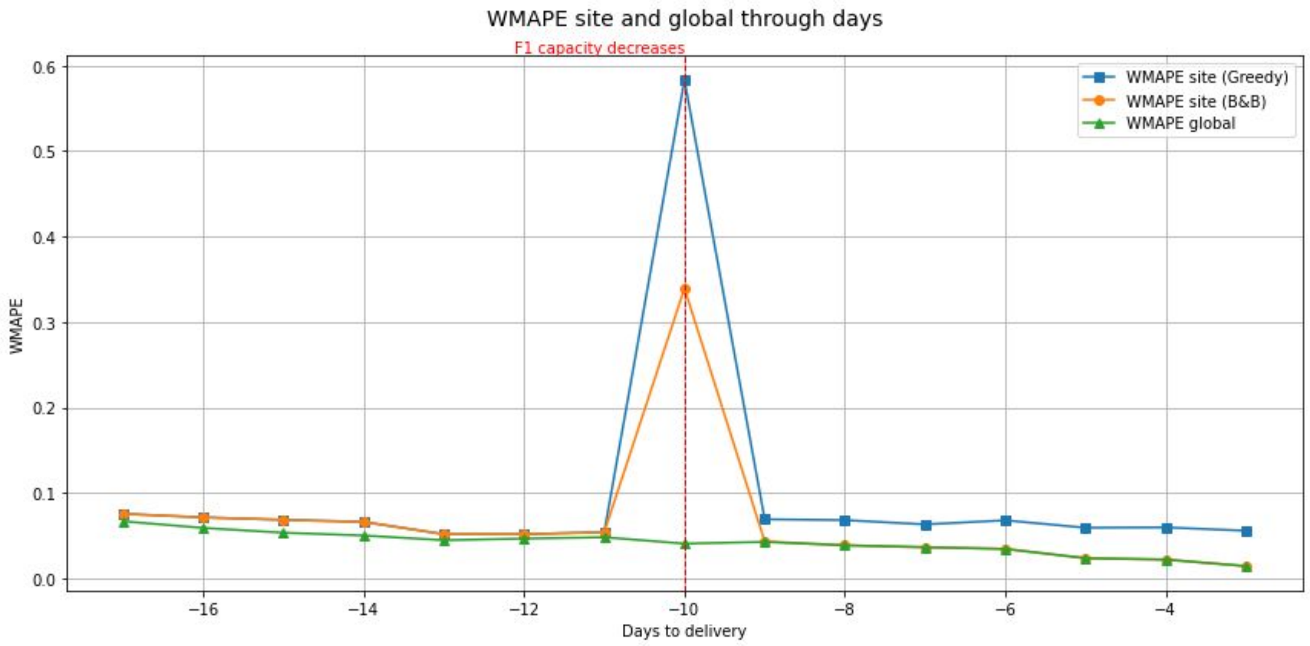
- LD18's recipes are compared with LD3
- LD17's recipes are compared with LD3
- ....

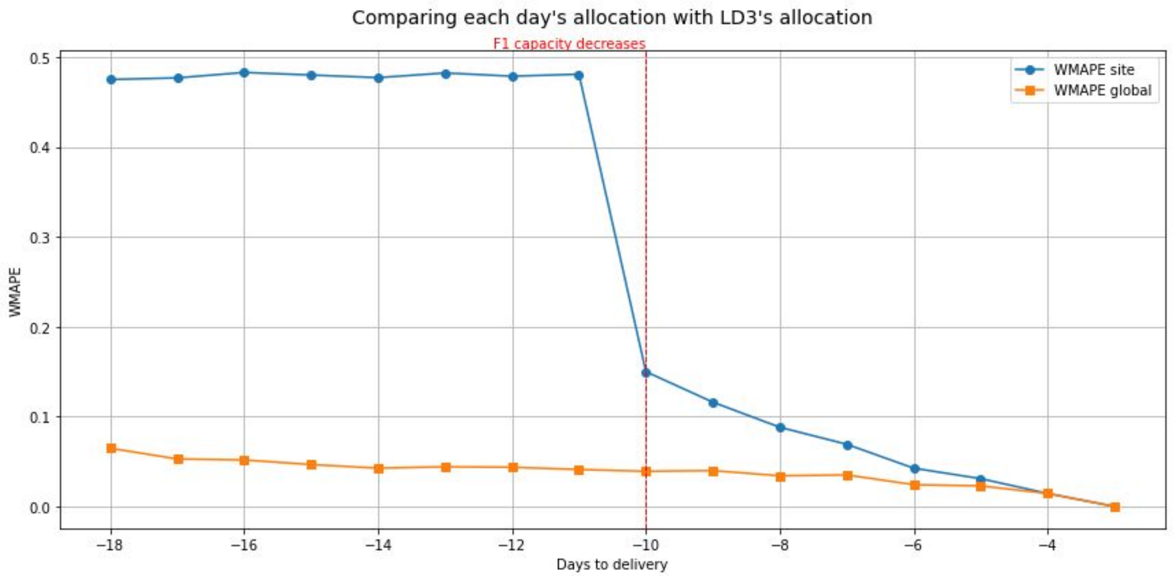


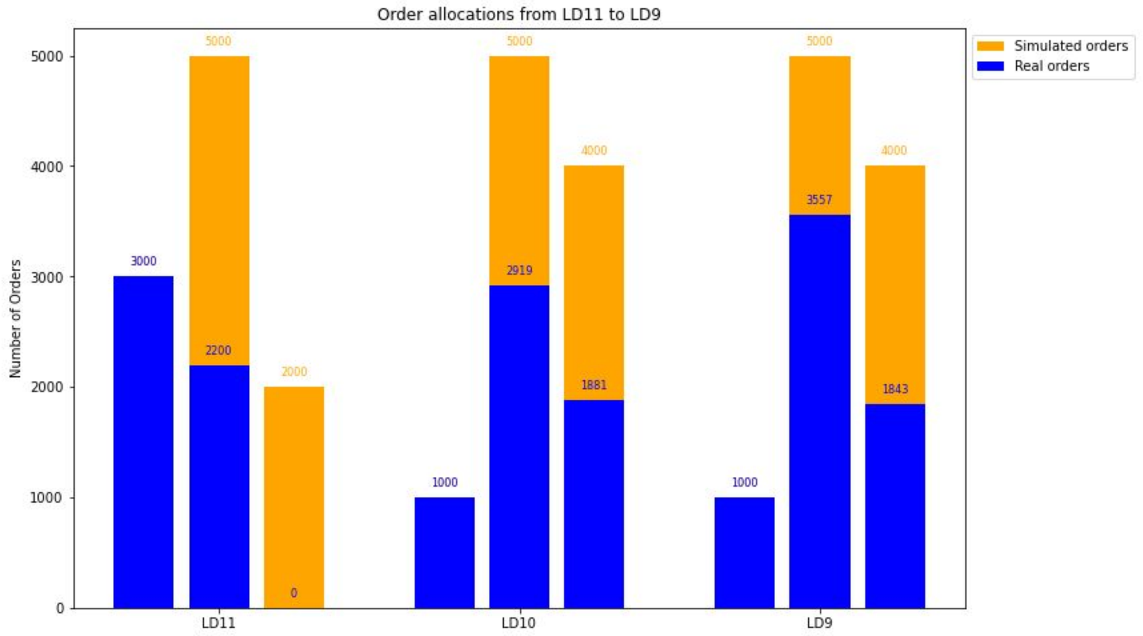


Temporal test - Capacity change

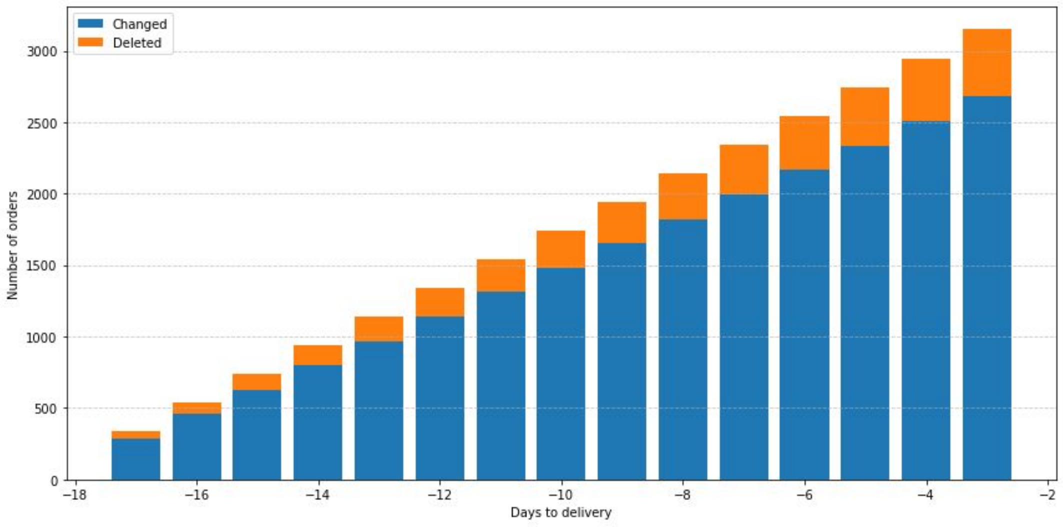




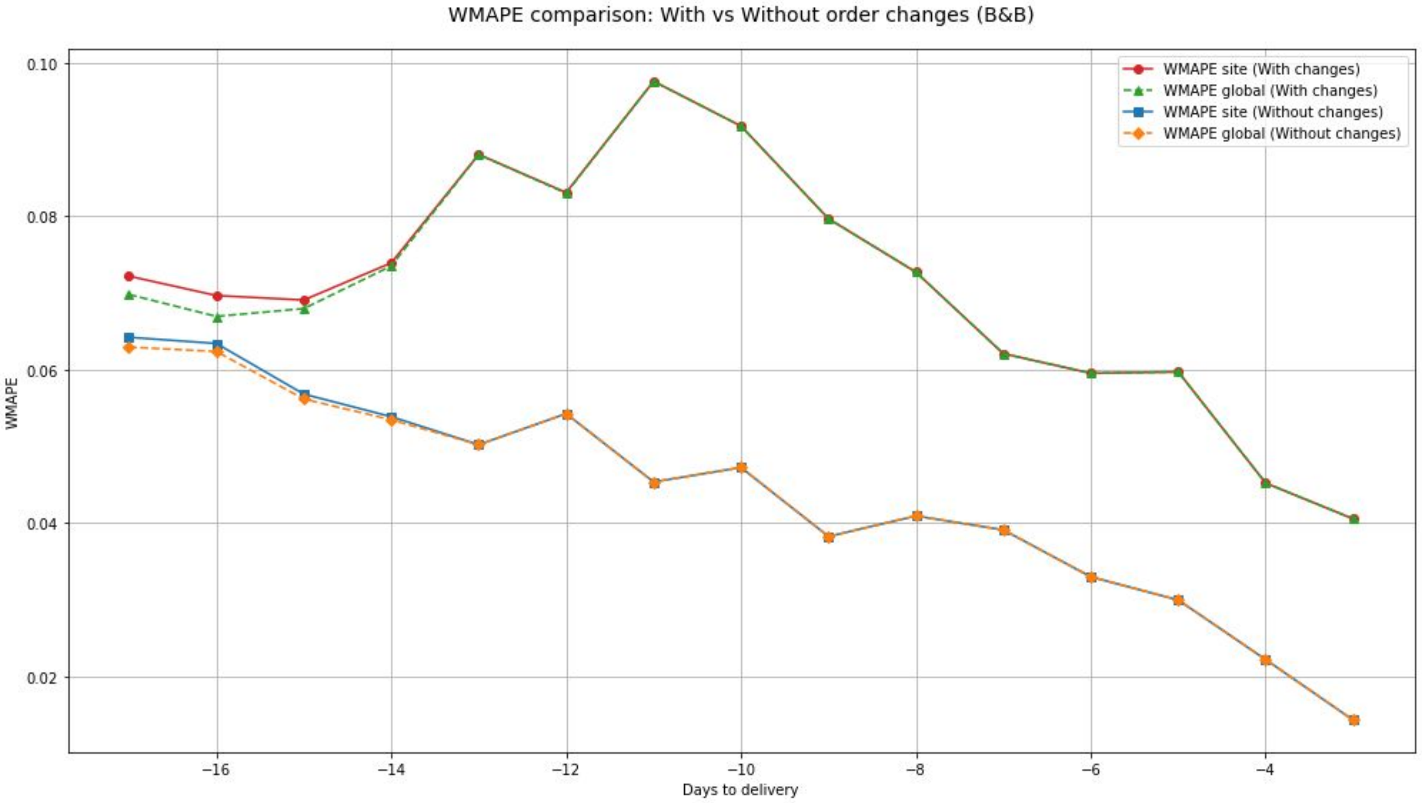




Temporal test - Order changes



Real orders: 5% deleted, 30% changed recipes



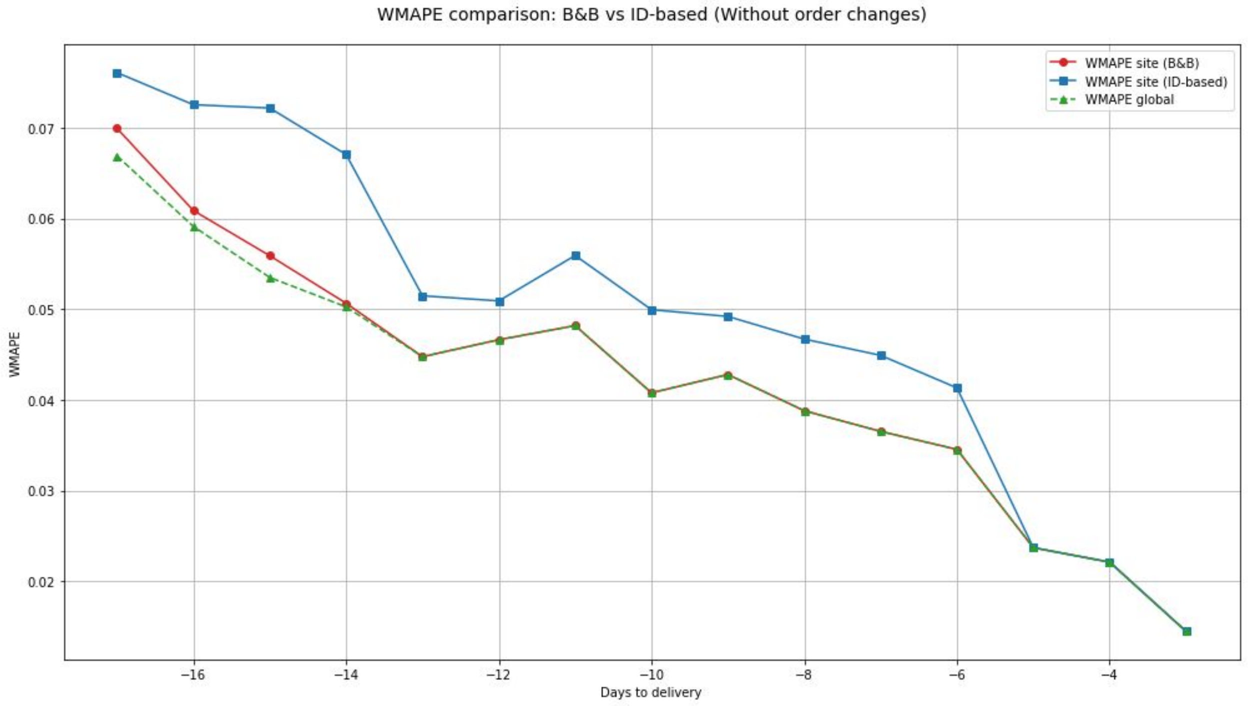
B&B versus ID-based allocation method

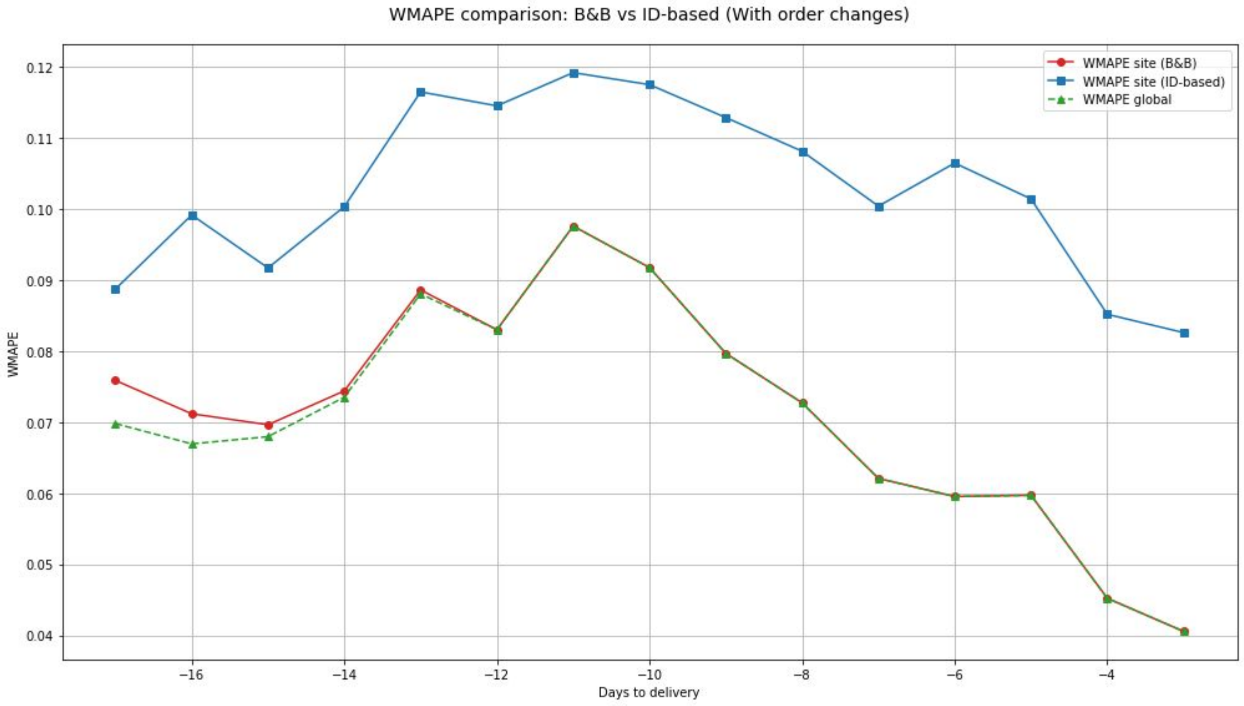
Example of ID-based method:

– LD14:

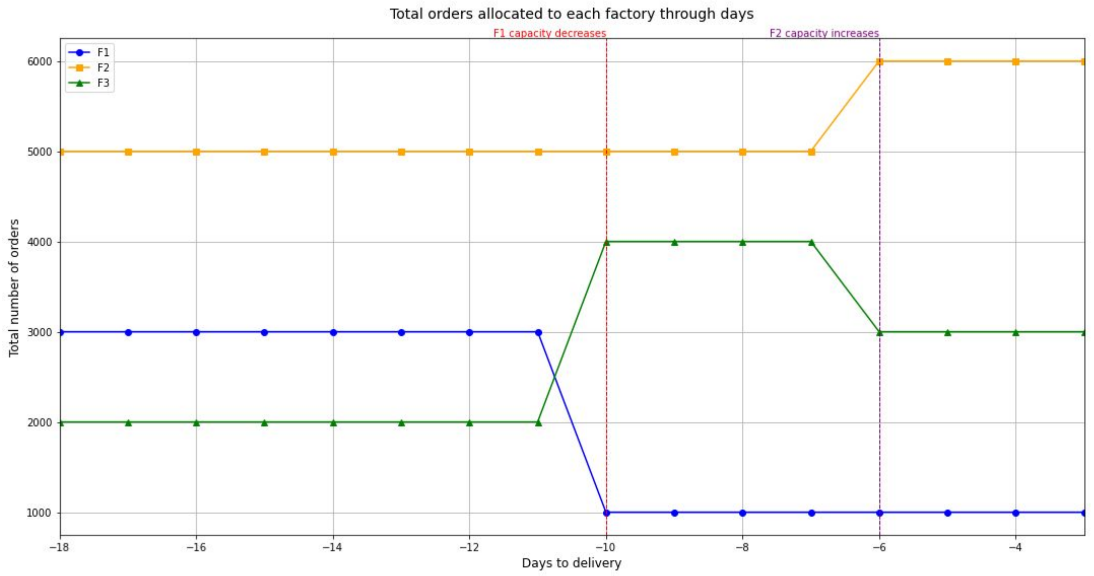
Real order ID	Allocated factory
R1	F1
R2	F2
R3	F2
R4	F3

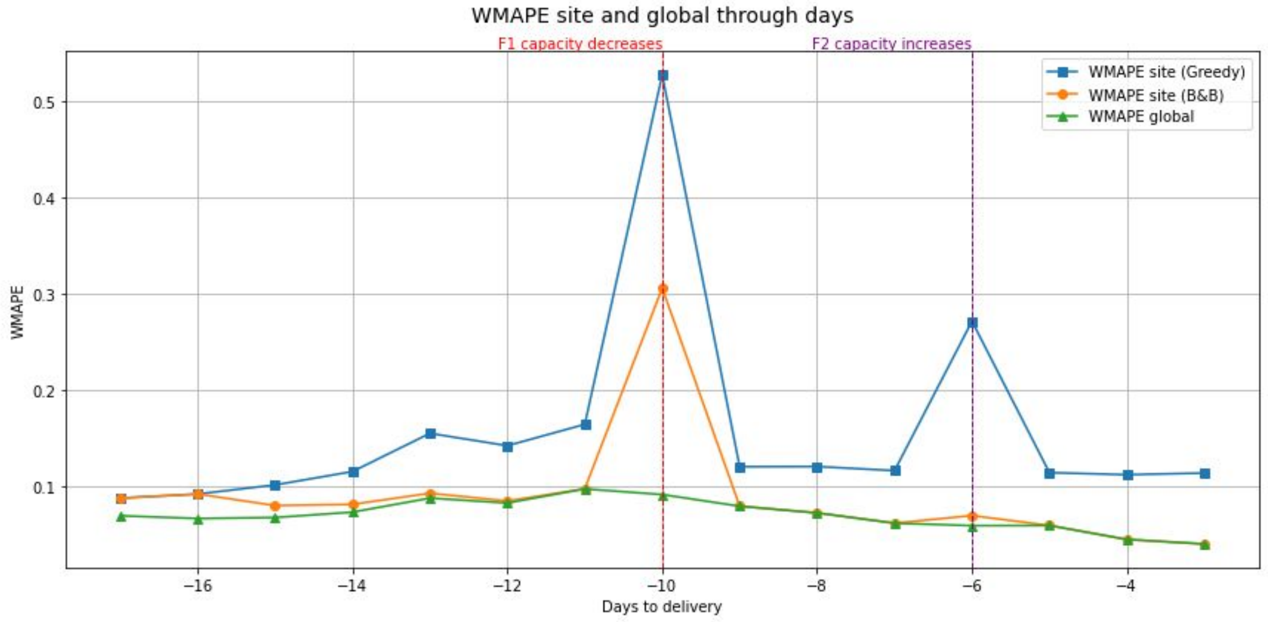
– LD15: The previous allocation decisions for real orders are reused to ensure these orders are consistently sent to the same factory, and minimize changes.

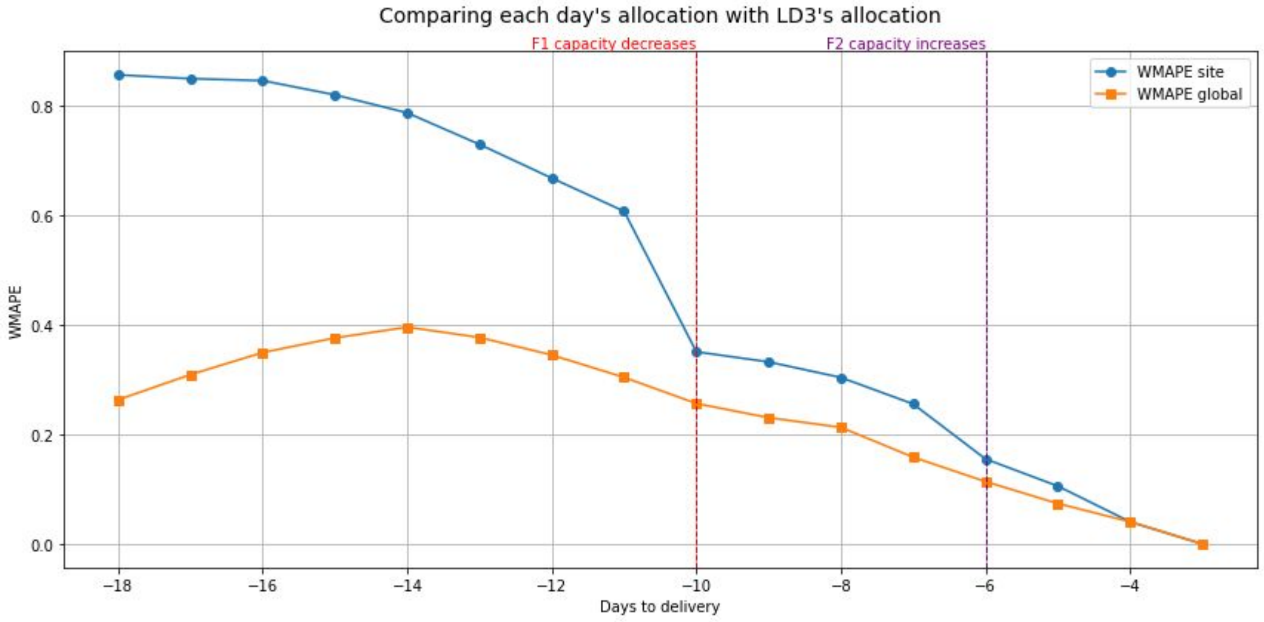




Temporal test - Both capacity and order changes







Contents

University of Southampton

Limitations and future directions

☹️ Limitations

☺️ Suggestions

Limitations	Suggestions
Test on simulated data	Validate effectiveness with real data
Separate single-day optimization	Investigate optimization approach that consider the entire planning period
Use single-swap heuristics	Try larger moves (LNS, GA,…)

Comments and Questions

Thank You!

Appendix

Codes: https://github.com/35436506/MSc-dissertation

ITPS

```
target_allocation = allocation_t_minus_1    # Allocation of previous day's orders (Input)
current_allocation = allocation_t           # Greedy allocation of current day's orders (Input)
best_allocation = current_allocation
best_wmape = calculate_WMAPE(current_allocation, target_allocation)

While not reached max_iterations do
    recipe_diff = calculate_recipe_differences(current_allocation, target_allocation)
    factories = randomly_shuffle(factories)
    For each source_factory in factories do
        For each target_factory in factories do
            If source_factory ≠ target_factory then
                Find swap_candidates based on recipe_diff
                If swap_candidates is found then
                    Perform_swap(swap_candidates)
                    new_wmape = calculate_WMAPE(current_allocation, target_allocation)
                    If new_wmape < best_wmape then
                        best_wmape = new_wmape
                        best_allocation = current_allocation
                    Else
                        Revert_swap(swap_candidates)
                    End If
                End If
            End If
        End For
    End For
End While
Return best_allocation, best_wmape        # Output
```

TS

```
target_allocation = allocation_t_minus_1        # Allocation of previous day's orders (Input)
current_allocation = allocation_t               # Greedy allocation of current day's orders (Input)
best_allocation = current_allocation
current_total_abs_diff = calculate_total_abs_diff(current_allocation, target_allocation)
best_total_abs_diff = current_total_abs_diff
tabu_list = {}                                  # Initialize empty tabu list
tabu_tenure = 20                                # Set the number of iterations a move remains in the tabu list

While not reached max_iterations do
    best_move = None
    best_move_diff = 0

    orders_to_consider = randomly_select_subset_of_orders(current_allocation)

    For each order1 in orders_to_consider do
        factory1 = find_current_factory(order1)
        For each factory2 in factories do
            If factory1 ≠ factory2 then
                For each order2 in randomly_select_subset_of_orders(factory2) do
                    If is_swap_eligible(order1, factory2) and is_swap_eligible(order2, factory1) then
                        move = (order1.id, factory1, order2.id, factory2)
                        move_impact = calculate_move_impact(order1, factory1, order2, factory2)
                        If move not in tabu_list or satisfies_aspiration_criteria(move, current_total_abs_diff, best_total_abs_diff) then
                            If move_impact < best_move_diff then
                                best_move = (order1, factory1, order2, factory2)
                                best_move_diff = move_impact
                    End If
                End If
            End For
        End If
    End For
End For

    If best_move is found then
        Perform_swap(best_move)
        current_total_abs_diff += best_move_diff

        If current_total_abs_diff < best_total_abs_diff then
            best_total_abs_diff = current_total_abs_diff
            best_allocation = current_allocation.copy()

        Add_to_tabu_list(best_move, current_iteration, tabu_tenure)

    Remove_expired_tabu_moves(tabu_list, current_iteration)

    If current_iteration % 100 == 0 then
        Perform_diversification_move()

End While

best_wmape_site = calculate_wmape(best_allocation, target_allocation)
Return best_allocation, best_wmape_site          # Output
```