Lecture VI - Minimizing Split Orders

Applied Optimization with Julia

Dr. Tobias Vlćek
University of Hamburg - Fall 2025

Introduction

E-Commerce Trends

Question: What are current trends in e-commerce?

E-Commerce Sales

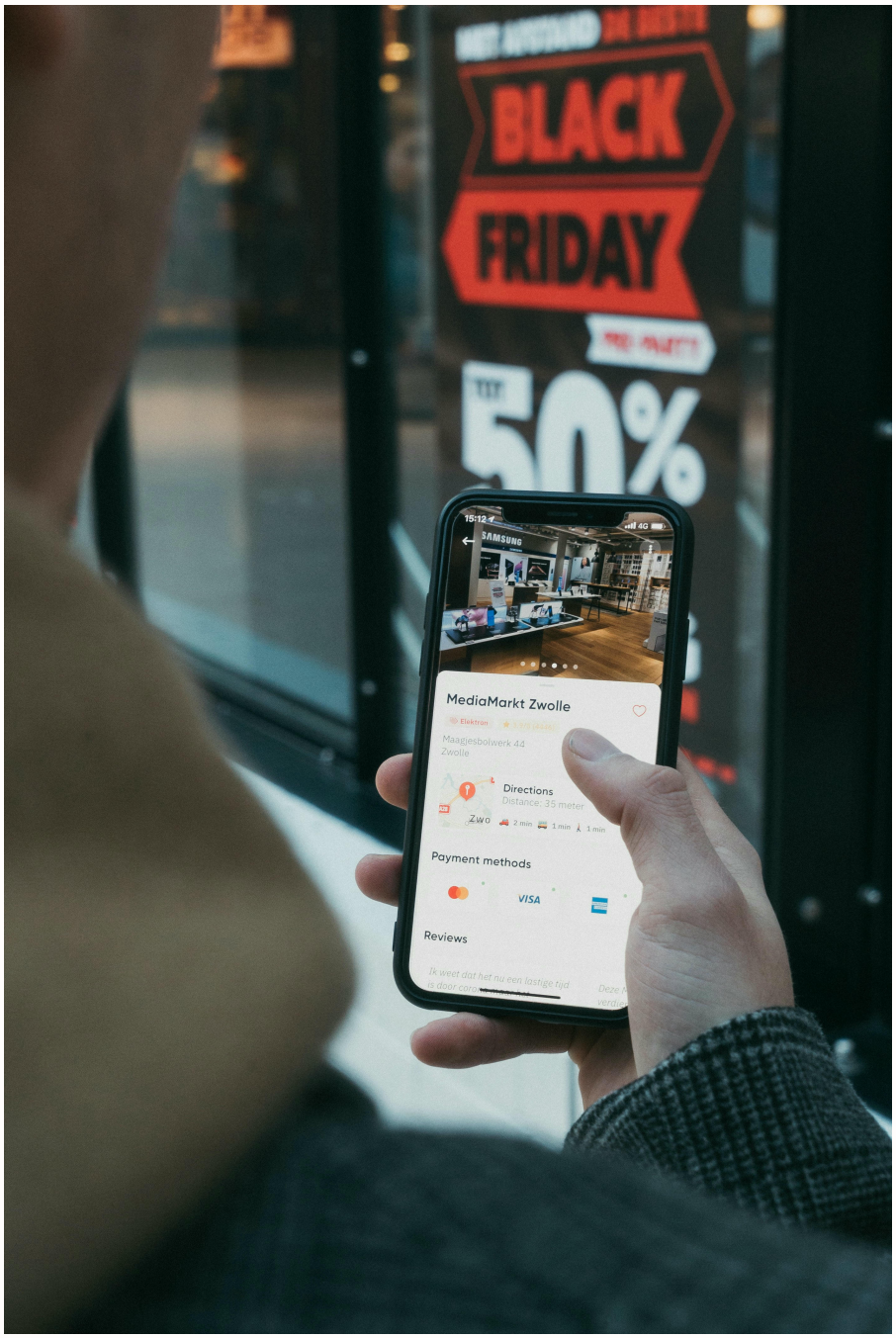
- E-Commerce sales are growing fast:
  - Products are no longer bound between borders
  - Product variety is rising
  - Consumer shopping patterns are shifting
  - Brick-and-mortar stores loose customers to the internet
  - Covid-19 accelerated this trend even more

Parcels Worldwide

- The number of parcels is rising:
  - 2014: 44 billion parcels [1]
  - 2019: 103 billion parcels [2]
  - 2026: 220 – 262 billion[1] [3]

Pressure on infrastructure

[1]Forecast, not actual number



- Consumers nowadays expect free, fast deliveries and returns
- Existing warehouses have to store an increasing range of products
- Better customer service requires faster deliveries
- Incurred fulfillment costs depend on the number of parcels

Pressure on the environment

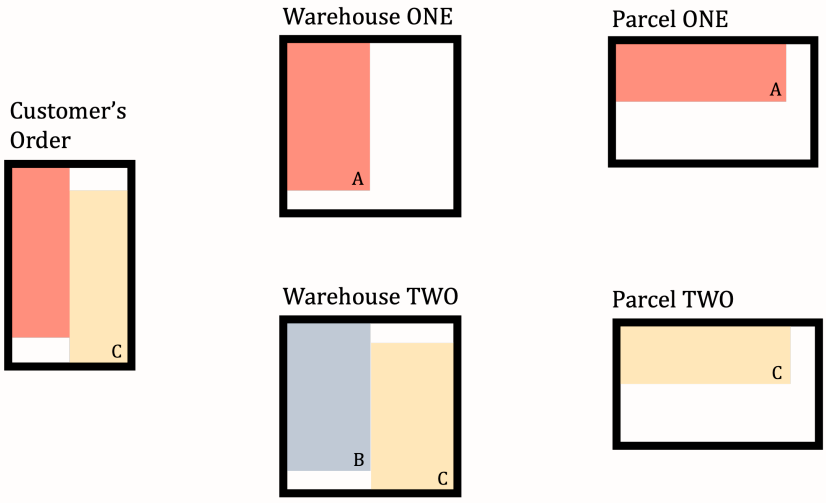
- Each parcel packaging consumes resources during production
- Every dispatched parcel to the customer causes CO_2 emissions
- In case of returns, more parcels cause more emissions

Problem Structure

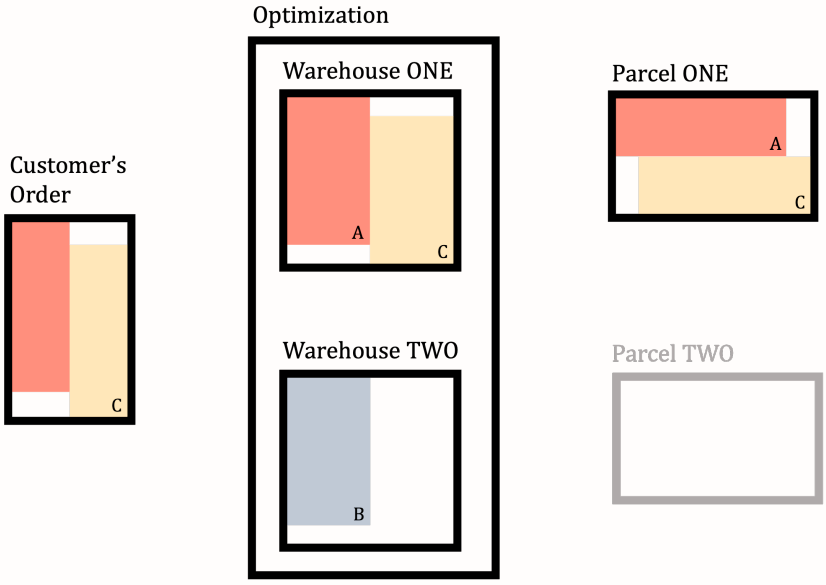
Split Order

Question: What is a split order?

. . .



No Split Order



Reason for Split Orders

Question: Why might they occur?

. . .

- Stock availability: Some products are out of stock at a warehouse and need to be fulfilled from another warehouse
- Capacity constraints: Some products are stored at different warehouses and need to be shipped from elsewhere

Impact of Split Orders

Question: What are the consequences?

. . .

- Higher shipping costs

- Increased packaging material
- More CO_2 emissions
- Higher operational complexity
- Lower customer satisfaction

Mitigations?

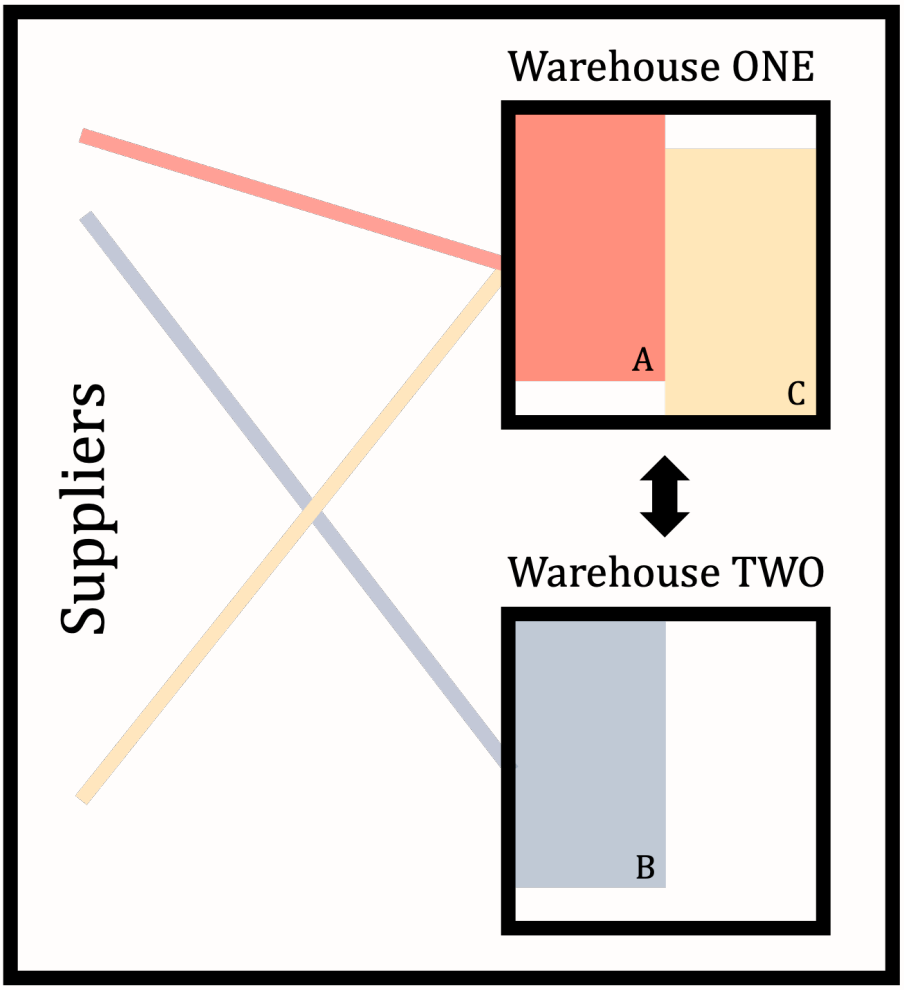
Question: What are possible mitigations?

. . .

- Consolidation: Ship to a central warehouse before dispatch
- Cross-docking: Ship directly from supplier to customer
- Transshipment: Ship between warehouses before delivery
- Co-allocation: Predict co-appearance of products and allocate them to the same warehouse

Problem Structure - Version 1

Optimizing Co-allocation



Question: What could be our objective?

We aim to improve the SKU[2]-warehouse allocation to minimize the number of split parcels resulting from SKUs being stored in different warehouses.

Available Sets

Question: What could be the sets here?

- $\mathcal{I}$ - Set of products indexed by $i \in \{1, 2, ..., |\mathcal{I}|\}$
- $\mathcal{K}$ - Set of warehouses indexed by $k \in \{1, ..., |\mathcal{K}|\}$
- $\mathcal{M}$ - Set of customer orders $m \in \{1, 2, ..., |\mathcal{M}|\}$

Available Parameters

Question: What are possible parameters?

- c_k - Storage space of warehouse $k \in \{1, ..., |\mathcal{K}|\}$
- $\boldsymbol{T} = \left(t_{m,i}\right)$ - Past customer orders for SKUs

...

Question: What could the transactional data look like?

Transactional Data

$t_{m,i}$	A	B	C	D
1	1	1	1	0
2	1	1	1	0
3	1	1	0	0
4	1	0	0	1
5	1	0	0	1
6	1	0	0	1
7	1	0	0	1
8	0	0	1	1

Past vs. Future

- The transactional data $\boldsymbol{T}$ is based on past orders
- It is a binary matrix of customer orders and SKUs
- We use this data to assume future co-occurrence
  - Past co-occurrence predicts future co-occurrence

...

Question: What is your opinion on the assumption?

Split-Order Minimization

Question: What could be our decision variable/s?

...

[2]SKU: Stock Keeping Unit

We have the following sets:

- $\mathcal{I}$ - Set of products indexed by $i \in \{1, 2, ..., |\mathcal{I}|\}$
- $\mathcal{K}$ - Set of warehouses indexed by $k \in \{1, ..., |\mathcal{K}|\}$
- $\mathcal{M}$ - Set of customer orders $m \in \{1, 2, ..., |\mathcal{M}|\}$

...

- $X_{i,k}$ - 1, if $i \in \mathcal{I}$ is stored in $k \in \mathcal{K}$, 0 otherwise
- $Y_{m,i,k}$ - 1, if SKU $i \in \mathcal{I}$ is shipped from warehouse $k \in \mathcal{K}$ for customer order $m \in \mathcal{M}$, 0 otherwise

Integer Programming Model

- A. Catalán and M. Fisher [4] created an integer model
- Number of SKUs of E-Commerce retailers can easily be between 10,000 - 100,000
- Number of customer orders necessary for “stable” results have to be higher in the order of 100,000 - 10,000,000

...

Question: Anybody an idea what this could mean?

Implementation Challenges

- Small instance with 10 SKUs and 1000 customer orders
- CPLEX 20.1.0 needs 3100 seconds to solve the problem
- Computation times scales exponentially
- $\rightarrow$ Not applicable in real world applications!

Any idea what

could be done?

Problem Structure - Version 2

Heuristic Approach

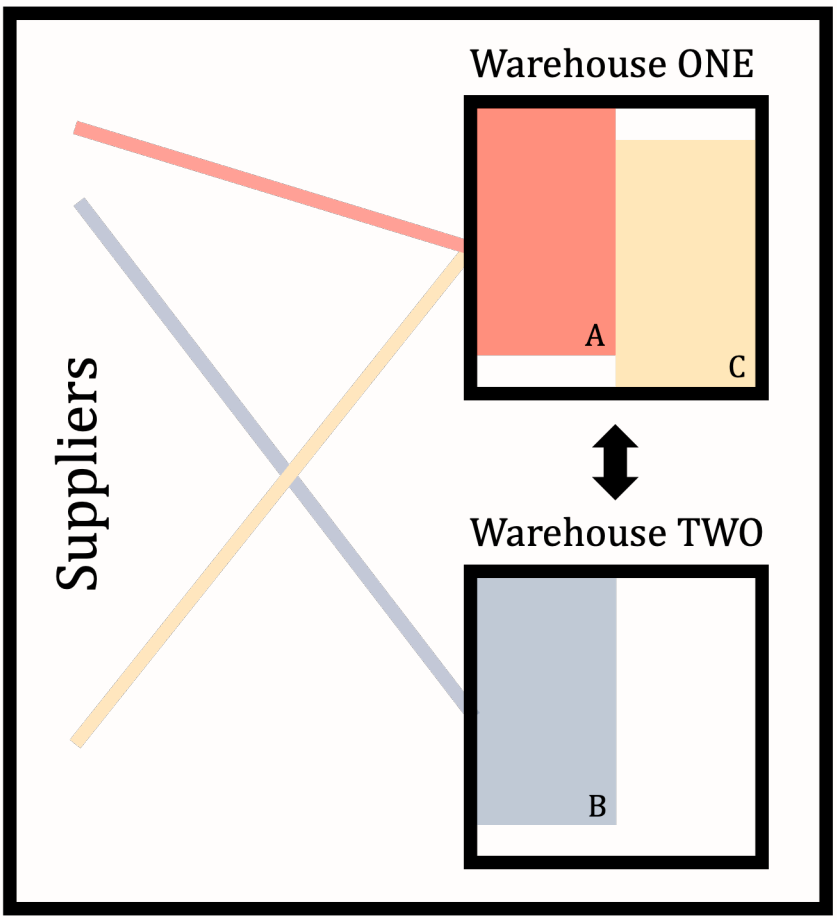
- Heuristic: Fast, but not necessarily optimal
- Approximation: Not guaranteed to be optimal, but close
- Computational Effort: Reasonable even for large instances

...

Different view on the problem

Focus on the warehouses and the co-appearance of SKUs! Discard the exact information about the customer orders.

Objective



Question: What could be the objective?

Maximize the coappearance of products that are often part of the same customer orders.

Transaction Matrix

```
T = [
    1 1 1 0;
    1 1 1 0;
    1 1 0 0;
    1 0 0 1;
    1 0 0 1;
    1 0 0 1;
    1 0 0 1;
    0 0 1 1
]

# Create the coappearance matrix
Q = T' * T
println("Coappearance matrix Q:")
display(Q)
```

```
Coappearance matrix Q:
```

```
4×4 Matrix{Int64}:
 7  3  2  4
 3  3  2  0
 2  2  3  1
 4  0  1  5
```

Coappearance Matrix

- $\boldsymbol{Q}$ is a symmetric matrix
- Proposed by A. Catalán and M. Fisher [4]
- $\boldsymbol{Q} = (\boldsymbol{T}^T \cdot \boldsymbol{T})$ where $\boldsymbol{Q} = \left(q_{ij}\right)_{i \in \{1,\ldots,\mathcal{I}\}, j \in \{1,\ldots,\mathcal{I}\}}$
- q_{ij} shows how often i and j appear in the same order

...

Question: What do the principal diagonal values tell us?

...

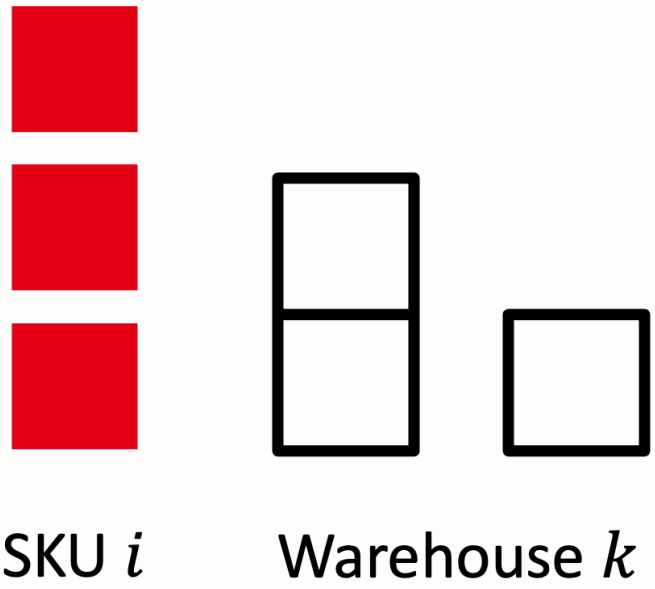
- How often each SKU appeared over all orders (binary!)

How to approach the problem?

- Greedy Heuristic[3]: Allocation based on matrix
- Mathematical Model[4]: Maximizes coappearance
- GRASP[5]: Good on small instances
- New: Max. coappearance with non-linear solver
- New: Heuristic based on Chi-Square Tests

[3] Simple and very fast, A. Catalán and M. Fisher [4]
[4] Computationally intensive with CPLEX, S. Zhu, X. Hu, K. Huang, and Y. Yuan [5]
[5] Greedy Randomized Adaptive Search Procedure, S. Zhu, X. Hu, K. Huang, and Y. Yuan [5]

Basic Setting



Available Data (Version 2)

Question: What could be the sets?

...

- $\mathcal{I}$ - Set of products indexed by $i \in \{1, 2, ..., |\mathcal{I}|\}$
- $\mathcal{K}$ - Set of warehouses indexed by $k \in \{1, ..., |\mathcal{K}|\}$

...

! No customer order information is needed!

We can focus on the SKUs and the warehouses, making the problem much smaller!

Available Parameters

Question: What are possible parameters?

- c_k - Storage space of warehouse $k \in \{1, ..., |\mathcal{K}|\}$
- $\boldsymbol{Q} = (q_{ij})_{i \in \{1,...,\mathcal{I}\}, j \in \{1,...,\mathcal{I}\}}$ - Coappearance matrix

...

! Transactional Data replaced

Instead of the transactional data, we just use the coappearance matrix in our model!

Model Formulation

Decision Variables?

> ℹ We have the following sets:
>
> - $\mathcal{I}$ - Set of products indexed by $i \in \{1, 2, ..., |\mathcal{I}|\}$
> - $\mathcal{K}$ - Set of warehouses indexed by $k \in \{1, ..., |\mathcal{K}|\}$

...

> ! Our objective is to:
>
> Maximize the coappearance of products that are often part of the same customer orders. In more mathematical terms: Maximize the sum of all unique pair-wise values $q_{i,j}$ of all SKUs stored in the same warehouse.

...

Question: What could be our decision variable/s?

Decision Variables

- $X_{i,k}$ - 1, if SKU $i \in \mathcal{I}$ is stored in $k \in \mathcal{K}$, 0 otherwise

...

> ! Only one variable per SKU and warehouse!
>
> As we don't need the customer order information, we only need to make a decision for each SKU and warehouse pair!

Decision Variable in Julia

Question: How could we formulate the variable in Julia?

```
import Pkg; Pkg.add("SCIP")
using JuMP, SCIP # SCIP is a non-commercial MIQCP solver

warehouses = ["Hamburg", "Berlin"] # Add warehouses as a vector
skus = ["Smartphone", "Socks", "Charger"] # Add SKUs as a vector

warehouse_model = Model(SCIP.Optimizer)
```

...

```
@variable(warehouse_model, X[i in skus, k in warehouses], Bin)
```

```
2-dimensional DenseAxisArray{VariableRef,2,...} with index sets:
    Dimension 1, ["Smartphone", "Socks", "Charger"]
    Dimension 2, ["Hamburg", "Berlin"]
And data, a 3×2 Matrix{VariableRef}:
 X[Smartphone,Hamburg]  X[Smartphone,Berlin]
 X[Socks,Hamburg]       X[Socks,Berlin]
 X[Charger,Hamburg]     X[Charger,Berlin]
```

Objective Function

i We need the following:

- $X_{i,k}$ - 1, if SKU $i \in \mathcal{I}$ is stored in $k \in \mathcal{K}$, 0 otherwise
- q_{ij} - Coappearance of SKU $i \in \mathcal{I}$ and $j \in \mathcal{I}$

! Our objective is to:

Maximize the sum of all unique pair-wise values $q_{i,j}$ of all SKUs stored in the same warehouse. Note, that this is a quadratic objective function!

...

Question: What could the objective function look like?

...

Quadratic Objective Function

$$\text{maximize} \quad \sum_{i=2}^{\mathcal{I}} \sum_{j=1}^{i-1} \sum_{k \in \mathcal{K}} X_{ik} \times X_{jk} \times q_{ij}$$

...

i This is a quadratic objective function!

The quadratic terms are $X_{ik} \times X_{jk}$. This objective function is based on the Quadratic Multiple Knapsack Problem (QMKP), formulated by A. Hiley and B. A. Julstrom [6].

Objective Function in Julia

Question: How could we formulate this in Julia?

...

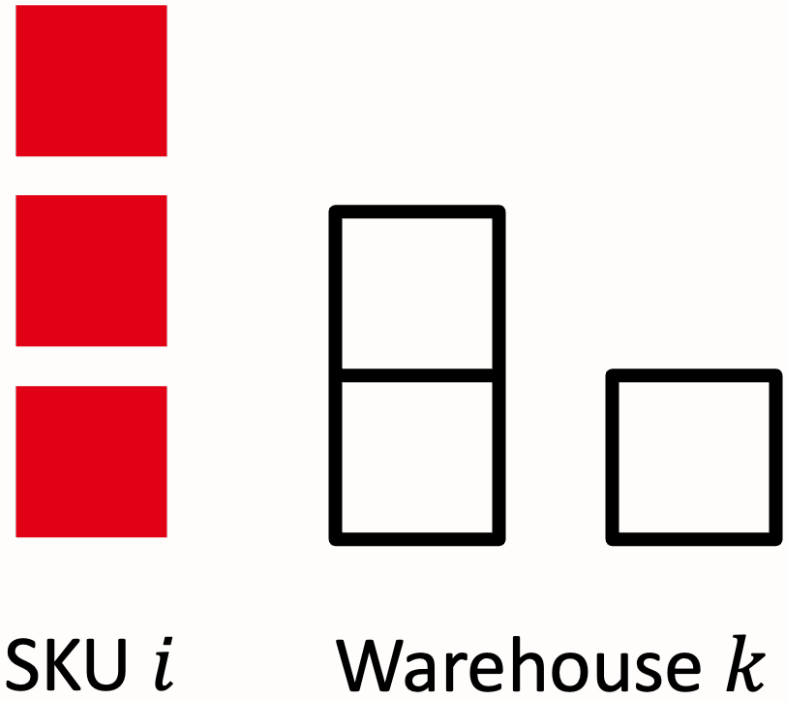
```
Q = [2 1 2; 1 2 1; 2 1 2]

@objective(warehouse_model,
    Max,
```

```
    sum(
        X[skus[i], warehouses[k]] * X[skus[j], warehouses[k]] * Q[i,j]
        for i in 2:length(skus)
        for j in 1:i-1
        for k in 1:length(warehouses)
    )
)
```

$ X_{Socks,Hamburg}X_{Smartphone,Hamburg} + X_{Socks,Berlin}X_{Smartphone,Berlin} + 2 X_{Charger,Hamburg}X_{Smartphone,Hamburg} + 2 X_{Charger,Berlin}X_{Smartphone,Berlin} + X_{Charger,Hamburg}X_{Socks,Hamburg} + X_{Charger,Berlin}X_{Socks,Berlin} $

Constraints

What constraints?



Question: What constraints?

- Allocate each SKU at least once
- Warehouses have a finite capacity
- Capacity is not exceeded

Single Allocation Constraint?

! The goal of this constraint is to:

Ensure that each SKU is allocated at least once.

...

i We need the following variable:

- $X_{i,k}$ - 1, if SKU $i \in \mathcal{I}$ is stored in $k \in \mathcal{K}$, 0 otherwise

...

Question: What could the constraint look like?

Single Allocation Constraint

$$\sum_{k \in \mathcal{K}} X_{ik} \geq 1 \quad \forall i \in \mathcal{I}$$

...

i Remember, this is the variable:

- $X_{i,k}$ - 1, if SKU $i \in \mathcal{I}$ is stored in $k \in \mathcal{K}$, 0 otherwise

...

Question: How could we change the constraint to ensure that each SKU is allocated only once?

...

Question: How could we add the constraint in Julia?

Single Allocation in Julia

```
@constraint(warehouse_model, single_allocation[i in skus],
    sum(X[i, k] for k in warehouses) >= 1
)
```

```
1-dimensional DenseAxisArray{ConstraintRef{Model,
MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64},
MathOptInterface.GreaterThan{Float64}}, ScalarShape},1,...} with index
sets:
    Dimension 1, ["Smartphone", "Socks", "Charger"]
And data, a 3-element Vector{ConstraintRef{Model,
MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64},
MathOptInterface.GreaterThan{Float64}}, ScalarShape}}:
 single_allocation[Smartphone] : X[Smartphone,Hamburg] +
```

```
X[Smartphone,Berlin] ≥ 1
 single_allocation[Socks] : X[Socks,Hamburg] + X[Socks,Berlin] ≥ 1
 single_allocation[Charger] : X[Charger,Hamburg] + X[Charger,Berlin] ≥ 1
```

Capacity Constraints?

! The goal of these constraints is to:

Ensure that the capacity of each warehouse is not exceeded.

...

i We need the following variables and parameters:

- $X_{i,k}$ - 1, if SKU $i \in \mathcal{I}$ is stored in $k \in \mathcal{K}$, 0 otherwise
- c_k - Storage space of warehouse $k \in \mathcal{K}$

...

Question: What could the second constraint be?

Capacity Constraints

$$\sum_{i \in \mathcal{I}} X_{ik} \leq c_k \quad \forall k \in \mathcal{K}$$

...

And that's basically it!

...

Question: How could we add the second constraint in Julia?

Capacity Constraints in Julia

```
capacities = Dict("Hamburg" => 2, "Berlin" => 1) # Add capacities

@constraint(warehouse_model, capacity[k in warehouses],
    sum(X[i, k] for i in skus) <= capacities[k]
)
```

```
1-dimensional DenseAxisArray{ConstraintRef{Model,
MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64},
MathOptInterface.LessThan{Float64}}, ScalarShape},1,...} with index sets:
    Dimension 1, ["Hamburg", "Berlin"]
And data, a 2-element Vector{ConstraintRef{Model,
MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64},
MathOptInterface.LessThan{Float64}}, ScalarShape}}:
 capacity[Hamburg] : X[Smartphone,Hamburg] + X[Socks,Hamburg] +
X[Charger,Hamburg] ≤ 2
```

```
 capacity[Berlin] : X[Smartphone,Berlin] + X[Socks,Berlin] +
X[Charger,Berlin] ≤ 1
```

QMK Model

$$\text{maximize} \quad \sum_{i=2}^{\mathcal{I}} \sum_{j=1}^{i-1} \sum_{k \in \mathcal{K}} X_{ik} \times X_{jk} \times q_{ij}$$

subject to:

$$\sum_{k \in \mathcal{K}} X_{ik} \geq 1 \; \forall i \in \mathcal{I}$$

$$\sum_{i \in \mathcal{I}} X_{ik} \leq c_k \forall k \in \mathcal{K}$$

$$X_{ik} \in \{0, 1\} \; \forall i \in \mathcal{I}, \forall k \in \mathcal{K}$$

QMK Model in Julia

```
set_attribute(warehouse_model, "display/verblevel", 0) # Hide solver output
optimize!(warehouse_model)

println("The optimal objective value is: ",
objective_value(warehouse_model))
println("The optimal solution is: ", value.(X))
```

```
The optimal objective value is: 2.0
The optimal solution is: 2-dimensional DenseAxisArray{Float64,2,...} with
index sets:
    Dimension 1, ["Smartphone", "Socks", "Charger"]
    Dimension 2, ["Hamburg", "Berlin"]
And data, a 3×2 Matrix{Float64}:
  1.0  0.0
 -0.0  1.0
  1.0  0.0
```

...

Question: What does this value tell us?

Model Characteristics

Characteristics

- Is the model formulation linear/ non-linear?
- What kind of variable domain do we have?
- Do we know the split-orders based on the objective value?
- Why couldn't we use HiGHS as solver?

Choosing a solver

- Identify problem structure, e.g. LP, MIP, NLP, QCP, MIQCP, ...

- What is the size of the problem?
- Is a commercial solver needed?

...

> **i** Commercial Solvers
>
> Commercial solvers are faster and more robust as open source solvers but also more expensive. During your studies, you can use most of them for free though! Nonetheless, we will only use open source solvers in this course.

Global vs Local Optimality

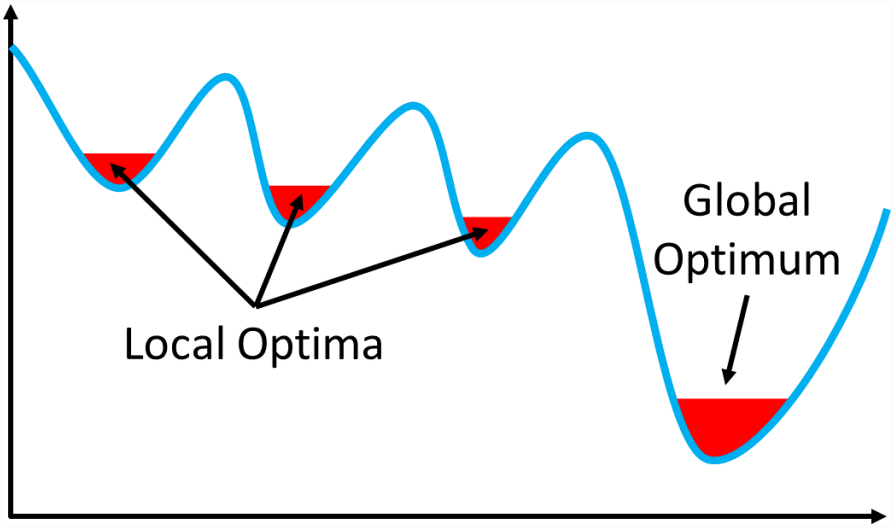


Figure 1: Local vs Global Optimum by Christoph Roser

Solver Comparison[6]

SKUs	SCIP	HiGHS	Gurobi
100	118s	X	~400s
1,000	1,011s (18%)	X	~200s (2%)
10,000	X	X	X

...

- SCIP: Best open-source for MIQCP, but limited scalability
- Commercial solvers: Better but still fail on 10,000+ SKUs
- Conclusion: Heuristics necessary for realistic problems!

Model Assumptions

Questions: On model assumptions

[6]Based on numerical experiments (QMK). The percentage in the table shows instances that could be solved within one hour and their average time (if solved).

- What assumptions have we made?
- Problem with allocating SKUs to multiple warehouses?
- What else might pose a problem in the real world?

Model Limitations

- Stock-outs: Model assumes perfect availability
  - Real stockouts create splits despite optimal allocation
- Storage costs: No differentiation between costs
- Demand dynamics: Weather, promotions, seasonality
- SKU sizes: Uniform capacity assumption unrealistic
- Shipping costs: No consideration of distance or weight

Practical Limitations

Allocation changes:

- Frequent reallocation disrupts operations
- Physical inventory movements costly
- Need for change minimization

Workload imbalance:

- Optimal allocation may overload some warehouses
- High-frequency SKUs cluster together
- Trade-off: splits vs. balance

Impact

Can this be

applied?

Problem Size is Crucial

- Up to 1,000 SKUs → commercial solvers
- More than 1,000 SKUs → heuristics
- For example, the CHI heuristic

. . .

> **i** CHI-Heuristic
>
> Detect dependencies between products and allocate them accordingly, as products within orders can have dependencies and products are bought with different frequencies!

CHI Heuristic

Statistical Independence

Question: When are two SKUs independent?

. . .

If purchasing one doesn't affect purchasing the other:

$$P(A \cap B) = P(A) \times P(B)$$

. . .

Example: If A appears in 10% of orders and B in 20%:

- Independent: Appear together in $0.10 \times 0.20 = 2\%$ of orders
- Dependent: Appear together in 5% of orders $\longrightarrow$ customers buy them together!

Chi-Square Test Intuition

Use chi-square tests to detect if SKUs are dependent:

- Calculate expected coappearances under independence
- Compare with actual coappearances
- If difference is significant $\longrightarrow$ SKUs are dependent
- Allocate dependent SKUs to same warehouse!

. . .

> Two-Phase Approach
>
> Phase 1: Build allocation using dependencies
> Phase 2: Refine with local search

. . .

Result: Works on large instances, 50,000 SKUs in 10 minutes!

Practical Implementation

Real-World Challenges

Question: What challenges might arise in practice?

- New products: Allocate SKUs without historical data?
- Dynamic inventory: SKUs appear and disappear
- Different sizes: SKUs consume varying storage space
- Computational limits: Problems with 50,000+ SKUs

Rolling Horizon Approach

- Use 5-week training window for allocation decisions
- Update allocations weekly based on recent patterns
- Balance stability vs. responsiveness

...

> Tip
>
> Grouped SKUs by category-brand combinations to reduce problem size while maintaining allocation quality. New SKUs inherit allocation from their cluster!

Case Study

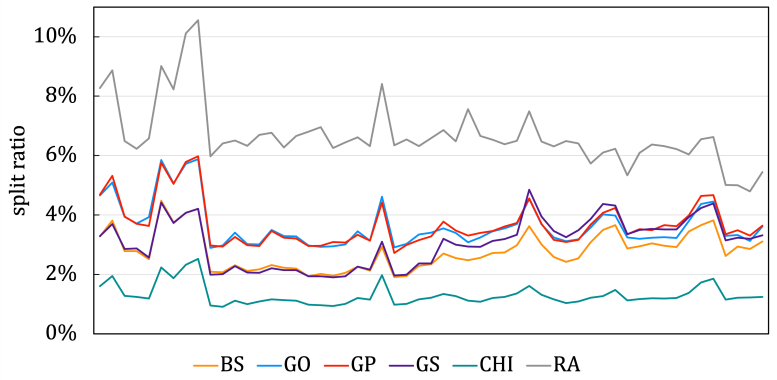
Problem characteristics:

- 100,000+ SKUs
- Several million orders
- Multiple warehouses
- Storage capacity constraints

Heuristics compared[7]:

- CHI: Chi-Square tests [7]
- GP, GO, GS, BS: Greedy [4]
- RA: Random allocation

...



Implementation Results

Theoretical improvements:

- CHI: 82.25% reduction
- BS: 62.63% reduction
- GS: 59.16% reduction
- vs. retailer: 6.95% split ratio

Conclusion

- Splits are of no benefit, except faster customer deliveries
- Increase workload, packaging and shipping costs
- Mathematical Optimisation of “full” problem not solvable

[7] QMKP is not applicable for instance in case study

- CHI Heuristic close to mathematical optimisation

...

> **i** And that's it for todays lecture!
>
> We now have covered the Quadratic Multiple Knapsack Problem and are ready to start solving some tasks in the upcoming tutorial.

Questions?

Literature

Literature I

For more interesting literature to learn more about Julia, take a look at the literature list of this course.

Bibliography

[1] Pitney Bowes Inc., "Pitney Bowes Parcel Shipping Index Reveals 48 Percent Growth in Parcel Volume since 2014." Accessed: May 27, 2021. [Online]. Available: https://www.businesswire.com/news/home/20170830005628/en/Pitney-Bowes-Parcel-Shipping-Index-Reveals-48

[2] Pitney Bowes Inc., "Pitney Bowes Parcel Shipping Index Reports Continued Growth Bolstered by China and Emerging Markets." Accessed: May 27, 2021. [Online]. Available: https://www.businesswire.com/news/home/20191010005148/en/

[3] Pitney Bowes Inc., "Pitney Bowes Parcel Shipping Index Reports Continued Growth as Global Parcel Volume Exceeds 100 billion for First Time Ever." Accessed: May 27, 2021. [Online]. Available: https://www.businesswire.com/news/home/20201012005150/en/

[4] A. Catalán and M. Fisher, "Assortment Allocation to Distribution Centers to Minimize Split Customer Orders," SSRN Electronic Journal, vol. 0, no. , p. , 2012, doi: 10.2139/ssrn.2166687.

[5] S. Zhu, X. Hu, K. Huang, and Y. Yuan, "Optimization of product category allocation in multiple warehouses to minimize splitting of online supermarket customer orders," European Journal of Operational Research, vol. 290, no. 2, pp. 556–571, 2021, doi: 10.1016/j.ejor.2020.08.024.

[6] A. Hiley and B. A. Julstrom, "The Quadratic Multiple Knapsack Problem and Three Heuristic Approaches to It," in Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, M. Keijzer, Ed., New York, NY: Association for Computing Machinery, 2006, pp. 547–552. doi: 10.1145/1143997.1144096.

[7] T. Vlćek and G. Voigt, “Optimizing SKU-Warehouse Allocations to minimize Split Parcels in E-Commerce Environments,” To be Submitted Soon, 2024.