Extended Capacity Warehouse Location Problem

Francisco Pinto[1], João Soares[1], and João Vieira[1]

1 Faculdade de Engenharia da Universidade do Porto, R. Dr. Roberto Frias, 4200-465 Porto, Portugal

Abstract. This study extends the capacitated warehouse location problem by incorporating practical constraints and compares Linear Programming and Constraint Programming approaches using IBM CPLEX and Google OR-Tools. Testing on OR-Library datasets reveals CPLEX's superiority in large-scale optimization, while OR-Tools excels in handling complex logical constraints. Our findings provide insights for selecting appropriate solution methodologies in warehouse allocation optimization.

Keywords: Warehouse · Solver · Optimization · CPLEX · OR-Tools.

1 Introduction

The classical capacitated warehouse location problem addresses how to satisfy customer demand at minimal cost by simultaneously selecting which warehouses to open and determining how to serve customers. While this problem is central to many real-world it also serves as a rich test bed for comparing solution methodologies in operations research.

This paper extends the traditional problem by incorporating practical constraints that often arise in real business contexts, such as minimum warehouse usage, restrictions preventing specific customers from sharing a warehouse, and operational dependencies between warehouses. These added layers of complexity emphasize key distinctions between Linear Programming (LP) and Constraint Programming (CP) approaches. In particular, they shed light on each method's strengths and limitations when handling a mixture of combinatorial and linear subproblems. By implementing and testing both LP and CP models with a range of datasets from the OR-Library, we provide detailed insights into solver behavior, solution quality, and computational performance. Additionally, we compare IBM CPLEX (via DOcplex) and Google OR-Tools to illustrate how different solvers respond to varied problem sizes and additional constraints.

Through this work, we demonstrate that LP solvers—particularly CPLEX—typically excel at finding optimal or solutions for large-scale problems, while Google OR-Tools showcased notable strengths in handling intricate logical constraints and dependencies. CPLEX proved superior in efficiently solving large-scale linear and mixed-integer problems, leveraging its advanced branch-and-bound algorithms and robust preprocessing techniques, whereas OR-Tools demonstrated distinct advantages in managing complex discrete constraints and dependencies.

2 The LP model

In this chapter, the problem is presented in mathematical form for the linear problem model, in the following components: objective function, decision variables, and restrictions.

2.1 Objective Function

The objective of the capacitated warehouse location problem is to minimize the total cost of operating the warehouse network, including:

- **Fixed Costs:** The cost of opening each warehouse.
- **Transportation Costs:** The cost of serving customers in warehouses.

The mathematical representation of the objective function is:

$$\text{Minimize} \quad \sum_{i=1}^{nW} \text{fixedCost}_i \cdot x_i + \sum_{i=1}^{nW} \sum_{j=1}^{nC} \text{transportCost}_{ij} \cdot \text{amountServed}_{ij}$$

Where:

- nW: Number of warehouses.
- nC: Number of clients.
- x_i: Binary decision variable, 1 if warehouse i is opened, 0 otherwise.
- $\text{amountServed}_{i,j}$: Amount of goods transported from warehouse i to customer j.
- fixedCost_i: Fixed cost of opening the warehouse i.
- $\text{transportCost}_{i,j}$: Cost of transporting one unit from warehouse i to customer j.

2.2 Decision Variables

The model uses the following decision variables:

- **Binary Variables:**
  - $x_i \in \{0, 1\}$: Indicates whether the warehouse i is open ($x_i = 1$) or closed ($x_i = 0$).
  - $y_{i,j} \in \{0, 1\}$: Indicates whether customer j is served by warehouse i.
- **Integer Variables:**
  - $\text{amountServed}_{i,j} \geq 0$: Amount of goods transported from warehouse i to customer j.

2.3 Constraints

The model includes the following constraints to ensure feasibility and problem requirements:

1. **Assignment of Customers to Warehouses:** Each customer must be served by at least one warehouse:

$$\sum_{i=1}^{nW} y_{ij} \geq 1 \quad \forall j = 1, \ldots, nC$$

2. **Demand Satisfaction:** The total amount served to each customer must meet their demand:

$$\sum_{i=1}^{nW} \text{amountServed}_{ij} = \text{demand}_j \quad \forall j = 1, \ldots, nC$$

3. **Capacity Constraints:** The total amount served by a warehouse cannot exceed its capacity if it is open:

$$\sum_{j=1}^{nC} \text{amountServed}_{ij} \leq \text{capacity}_i \cdot x_i \quad \forall i = 1, \ldots, nW$$

4. **Linking Transportation and Assignment:** The amount served to a customer from all the warehouses cannot exceed the customer demand:

$$\text{amountServed}_{i,j} \leq \text{demand}_j \cdot y_{i,j} \quad \forall i \in \text{Warehouses}, \forall j \in \text{Customers}$$

$$y_{i,j} \leq x_i \quad \forall i \in \text{Warehouses}, \forall j \in \text{Customers}$$

5. **Non-Negativity and Binary Constraints:**

$$x_i, y_{i,j} \in \{0, 1\}, \quad \text{amountServed}_{i,j} \geq 0$$

This formulation ensures that the solution is both cost-effective and feasible, respecting customer demands, warehouse capacities, and business rules.

3 The CP model

The CP model is very similar to the LP model, it has the same objective function, the same decision variables and apply the same constraints with some changes. The constraints of the CP model address the same fundamental aspects of the problem, such as customer assignment, demand satisfaction, warehouse capacity, and consistency between decisions and resource allocations. To avoid redundancy, the detailed model with the mathematical expressions can be found in appendix A.

4 Extended Constraints

4.1 Objective

Next it is described the actions taken to extend the warehouse optimization problem. The intent behind these changes is twofold:

1. To increase the complexity of the problem for the linear programming (LP) solver, thereby testing its robustness.
2. To provide an improved framework for constraint programming (CP) solvers to efficiently address the problem.

4.2 Overview of Modifications

To achieve the stated objectives, three additional constraints were introduced into the problem formulation. These constraints add layers of complexity to the problem while aligning with operational and practical considerations. Below is a detailed description of the newly added constraints:

Minimum Warehouse Usage This constraint ensures that any open warehouse must operate at a minimum of 80% of its capacity:

$$\sum_{j \in \text{Customers}} \text{amountServed}_{i,j} \geq \lceil 0.8 \cdot \text{capacity}_i \rceil \cdot x_i \quad \forall i \in \text{Warehouses}$$

Here:

- x_i is a binary variable indicating whether warehouse i is open.
- capacity_i represents the maximum capacity of warehouse i.
- $\text{amountServed}_{i,j}$ is the quantity served by warehouse i to customer j.

This constraint prevents underutilization of warehouse operations and adds complexity to the decision-making process for the LP solver. From a business perspective, it ensures operational efficiency by requiring warehouses to operate at a minimum of 80% capacity.

Prohibited Customer Pairs Certain pairs of customers cannot be served by the same warehouse:

$$y_{i,j_1} + y_{i,j_2} \leq 1 \quad \forall i \in \text{Warehouses}, \forall (j_1, j_2) \in \text{Prohibited\_Pairs}$$

Here:

- $y_{i,j}$ is a binary variable indicating whether customer j is served by warehouse i.
- Prohibited_Pairs is the set of customer pairs that cannot be served by the same warehouse.

This constraint introduces combinatorial complexity by limiting specific service pairings. In practical terms, it can be applied to ensure that certain customers are not served by the same warehouse operation, whether due to service level requirements or other business considerations.

Dependent Warehouses Dependencies between warehouses dictate that some warehouses can only operate if others are open:

$$x_i \leq x_j \quad \forall (i,j) \in \text{Dependent\_Warehouses}$$

Here:

- x_i and x_j are binary variables indicating whether warehouses i and j are open.
- Dependent_Warehouses is the set of warehouse pairs with operational dependencies.

This constraint creates interdependencies that add complexity to the optimization process. From a practical business perspective, it reflects scenarios where a cluster of warehouses is preferred or required for specific operational reasons.

4.3 Expected Outcomes

The introduction of these constraints is expected to result in:

1. **Extended solution times** for LP solvers due to the increased complexity.
2. **Enhanced solution quality** for CP solvers by aligning constraints with their problem-solving strengths.

These enhancements aim to create a balanced yet challenging optimization environment for LP and CP solvers.

5 Analysis of Results

This section analyzes the results obtained from solving the warehouse location problem using Linear Programming (LP) and Constraint Programming (CP). The study progressively adds complexity by exploring the effects of new constraints, varying the number of instances, execution times, and solver configurations. Additionally, a comparative analysis of the performance of IBM's CPLEX solver and Google's OR-Tools provides insights into their strengths and limitations in this context.

5.1 Differences between CP and LP and the Impact of Additional Constraints

The dataset for this analysis was sourced from the OR-Library, specifically using the file `cap44` for the initial tests. After defining the base parameters of the problem, we employed Python's `docplex` library to implement both LP and CP models via IBM's CPLEX solver.

The LP model successfully computed the optimal solution in 0.64 seconds, consistent with expectations as this problem predominantly involves optimization tasks that LP is designed to handle efficiently. In contrast, the CP model

struggled to converge, resulting in a gap of 98.35% after a 10-minute execution time. This disparity arises from the exhaustive search characteristic of CP, particularly when faced with a combinatorial problem where N customers can be served by X warehouses with Y capacities.

These findings highlight the strengths of LP in solving structured optimization problems quickly and the challenges CP faces with scalability under tight computational limits. Future iterations of this analysis could explore hybrid approaches or parallelization techniques to enhance CP's performance in such scenarios.

Table 1. Comparison of Results for CP and LP Approaches

Metric	LP (IBM Solver)	CP (IBM Solver)
Execution Time (seconds)	0.64	600.17
Optimal Result	1235500,45	1586247,50
Gap	0%	98.35%

Impact of Extended Constraints

The introduction of new restrictions aimed to improve the performance of the CP model while potentially reducing the efficiency of the LP model. However, our results indicate that the performance of the CP model did not improve; in most cases, it worsened due to the increased complexity introduced by the additional constraints. The performance of the LP model was also significantly negatively affected, particularly for a higher number of instances. This section sets the stage for a more detailed analysis of the impact of these restrictions, which will be discussed in the following sections.

5.2 Impact of Different Number of Instances

To evaluate the effect of varying the number of instances, we analyzed data from `cap44`, `cap92`, `cap123`, and `cap124`. These data sets differ primarily in the number of clients: `cap44` includes 16 clients, `cap92` has 25 clients, and both `cap123` and `cap124` include 50 clients.

The results reveal a consistent negative impact on the solver performance as the number of instances increased. Table 3 and Table 4 in Appendix B show that a higher number of instances result in longer execution times for the LP model and higher optimality gaps for the CP model, along with slower branch search speeds.

The new restrictions further exacerbated the resolution time of the LP model, with execution times increasing up to tenfold for larger instances, as we can see in Figure 1. Despite this slowdown, the LP model still significantly outperformed the CP model, underscoring its robustness for this type of problem.

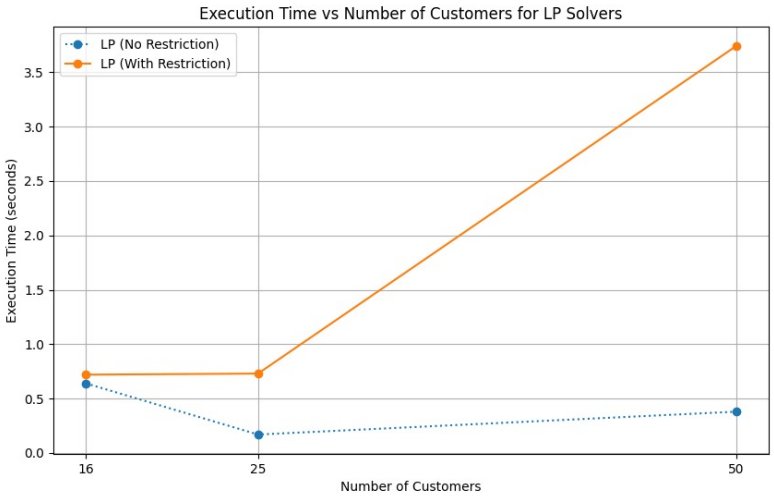

Fig. 1. Plot of the number of Execution Time by Number of customers of LP with and without restrictions

5.3 Comparative Analysis of Different Execution Times of CP

Considering the CP model of CPLEX could not find the optimal solution, we decided to investigate how varying time limits would affect the overall solution quality. We analyzed the file `cap44` using time limits of 2 minutes, 5 minutes, 10 minutes, 30 minutes, and 1 hour. The results indicated that up to 10 minutes, there was a notable improvement in the quality of the solution found, as well as a reduction in the optimality gap. However, beyond this point, the improvements became marginal, as shown in Table 5 in Appendix B. Additionally, we observed that increasing the time limit resulted in higher memory consumption without yielding significant benefits in the solution quality.

As illustrated in Table 5 in Appendix B, increasing the time limit positively impacts performance to some extent. However, it remains insufficient to match the efficiency and quality achieved by the LP solver.

5.4 Impact of CP Options

In this section, we explore how different configuration settings impact the performance of the CP model. For this analysis, we used the file `cap44` with a fixed time limit of 10 minutes. We focused on two primary settings: search type and log verbosity.

Search Type We compared several search strategies provided by the CP CPLEX solver: Depth First, Restart, Multipoint, Iterative Diving, Neighborhood, and Auto (default).

Depth First explores one branch of the search tree fully before backtracking, which often results in deeper exploration but failed to find a solution in this case. Neighborhood focuses on refining specific regions of the solution space but struggled with a low branch search speed of 293.5 branches per second and did not find a solution. Restart resets the search periodically to avoid getting stuck in poor-quality regions, achieving the highest branch speed among the methods. Multipoint explores multiple promising areas simultaneously, but while it found a solution, it was the least effective and used the most memory. Iterative Diving prioritizes diving quickly into feasible regions and provided the best solution among these last methods. Finally, Auto, the default search type, dynamically adjusts its strategy during runtime and delivered the overall best solution, balancing speed, memory usage, and solution quality effectively, as shown next in Table 2.

Table 2. Comparison of Search Strategies with Updated Time and Memory Sums

Search Strategy	**DepthFirst**	**Restart**	**Multipoint**	**IterativeDiving**	**Neighborhood**	**Auto (default)**
Time (seconds)	600.17s	600.18s	600.17s	600.21s	600.40s	600.37s
Best Solution Found	Not Found	1615092.60	1849364.34	1592999.19	Not Found	1452414.71
Gap	-	98.38%	100%	98.36%	-	98.20%
Nº Branches	6323802	38101122	16739693	19203337	176087	21947969
Nº Fails	3159049	8142085	6325995	4497714	-	5577895
Memory Usage	66.4 MB	234.3 MB	340.4 MB	261.4 MB	175.5 MB	283.1 MB
Search Sp. (br. /s)	10539.2	63486.4	27898.4	31996.0	293.5	36559.5

Log Verbosity We also analyzed the impact of different log verbosity levels on the performance of the model. The verbosity settings—Quiet, Terse, Normal (default), and Verbose—determined the amount of runtime information displayed during execution, ranging from minimal (Quiet) to extensive (Verbose). Although the overall results did not differ significantly, we observed a decrease in search speed as verbosity increased, as shown in Table 6 in Appendix B. This suggests that higher verbosity settings may introduce minor overheads, likely due to the additional computation required to generate and display detailed logs.

5.5 Comparison of OR-Tools and CPLEX

Google's OR-Tools is an open-source optimization suite that provides both Linear Programming (LP) and Constraint Programming (CP) solvers. Unlike a pure CP framework, OR-Tools' CP solver adopts a mixture of search methods, allowing it to tackle diverse problems efficiently. On the other hand, IBM's CPLEX is a proprietary framework that offers specialized LP and CP solvers with robust commercial support. In general, LP solvers in both frameworks are fast and precise, while CP solvers excel in scenarios with complex discrete constraints.

Detailed information of results in table format can be found in Appendix B tables 7 and 8.

Performance Overview Without Additional Restrictions.

- **LP OR-Tools** and **CP OR-Tools** both achieve optimal solutions, with LP usually faster for smaller problems.
- **LP CPLEX** remains optimal and is typically faster than OR-Tools methods.
- **CP CPLEX** exhibits suboptimal results, often stalling at high execution times.

Performance Overview With Additional Restrictions.

- **CP OR-Tools** outperforms **LP OR-Tools** in several cases, benefiting from its hybrid approach.
- **LP CPLEX** still achieves optimal solutions with lower computation times, remaining consistently faster than OR-Tools.
- **CP CPLEX** remains suboptimal in all tested scenarios.

Analysis of Solver Performance The results reveal expected trends and some surprising findings about solver performance:

1. **CPLEX Linear Programming Dominance**
   CPLEX Linear Programming excelled in both execution time and achieving optimality, leveraging advanced simplex and barrier methods optimized for large-scale linear problems, such as warehouse allocation.
2. **CPLEX Constraint Programming Limitations**
   CPLEX Constraint Programming performed poorly, likely due to a less mature CP implementation compared to its LP capabilities. While CP excels in combinatorial problems, it may underperform on problems better suited to LP formulations.
3. **Strong Performance of OR-TOOLS Constraint Programming**
   OR-TOOLS CP outperformed OR-TOOLS LP under additional constraints, a surprising result likely due to:
   - The new constraints introducing combinatorial complexities better handled by CP techniques.
   - OR-TOOLS CP's efficient domain filtering and constraint propagation algorithms.
   - The overhead of continuous LP relaxation slowing down OR-TOOLS LP in highly constrained scenarios.
4. **Consistent Superiority of CPLEX Linear Programming**
   Even with additional constraints, CPLEX LP remained the fastest and most robust solver, underscoring its advanced capabilities in solving complex linear optimization problems.

6 Conclusions

This paper introduced the Extended Capacity Warehouse Location Problem, enhancing a classic optimization model with constraints like minimum warehouse usage, prohibited customer pairs, and warehouse dependencies. Key findings from implementing and comparing LP and CP models with IBM CPLEX and Google OR-Tools include:

1. **LP Models Excel at Scale:** CPLEX LP consistently delivered optimal solutions efficiently, even with added complexities.
2. **CP Performance:** While CP handles combinatorial constraints effectively, CPLEX CP struggled with scaling, whereas OR-Tools CP performed unexpectedly well.
3. **Impact of Constraints:** Real-world constraints significantly increased computational complexity.
4. **Solver Insights:**
   - **CPLEX LP:** The most robust and efficient, handling added constraints with minimal performance loss.
   - **CPLEX CP:** Struggled under time limits, highlighting potential for CP-focused enhancements.
   - **OR-Tools LP:** Competitive for smaller instances but lagged behind for large-scale problems.
   - **OR-Tools CP:** Demonstrated notable efficiency, surpassing OR-Tools LP in certain scenarios.

In conclusion, by integrating real-world constraints into the traditional Capacitated Warehouse Location Problem, this paper underscores the substantial advantages of mature LP solvers (especially IBM CPLEX) for large-scale optimization and also gave some insight of OR-TOOLS CP that with its own particular approach demonstrated a surprising result.

Appendix A - The CP model

The model used for the problem of assigning warehouses to clients is formulated as a Constraint Programming problem in mathematical form. In the following, we detail the objective function, the decision variables, and the constraints of the model.

Objective Function

The objective of the model is to minimize the total cost, which includes the fixed costs of opening warehouses and the transportation costs of serving client demands. The objective function is the following.

$$\text{Minimize} \quad \sum_{i=1}^{nW} \text{fixedCost}_i \cdot x_i + \sum_{i=1}^{nW} \sum_{j=1}^{nC} \text{transportCost}_{ij} \cdot \text{amountServed}_{ij}$$

Where:

- nW: Number of warehouses.
- nC: Number of clients.
- fixedCost_i: Fixed cost of opening warehouse i.
- $\text{transportCost}_{ij}$: Transportation cost per unit from warehouse i to client j.
- x_i: Binary variable indicating whether warehouse i is open (1) or closed (0).
- amountServed_{ij}: Quantity of units supplied from warehouse i to client j.

Decision Variables

- $x_i \in \{0, 1\}$: Indicates whether the warehouse i is open (1) or closed (0).
- $y_{ij} \in \{0, 1\}$: Indicates whether the client j is served by the warehouse i (1) or not (0).
- $\text{amountServed}_{ij} \in [0, 15000]$: Quantity of units supplied from warehouse i to client j.

Constraints

The model includes the following constraints:

1. **Assignment of Customers to Warehouses:** Each customer must be served by at least one warehouse.

$$\sum_{i=1}^{nW} y_{ij} \geq 1 \quad \forall j = 1, \ldots, nC$$

2. **Demand Satisfaction:** The total quantity served to each client must equal their demand.

$$\sum_{i=1}^{nW} \text{amountServed}_{ij} = \text{demand}_j \quad \forall j = 1, \ldots, nC$$

3. **Capacity Constraints:** The total quantity served by each warehouse cannot exceed its capacity.

$$\sum_{j=1}^{nC} \text{amountServed}_{ij} \leq \text{capacity}_i \cdot x_i \quad \forall i = 1, \ldots, nW$$

4. **Consistency Between Quantity and Assignment:** The quantity supplied from warehouse i to client j must be zero if client j is not assigned to warehouse i.

$$\text{amountServed}_{ij} \leq \text{demand}_j \cdot y_{ij} \quad \forall i = 1, \ldots, nW,\ j = 1, \ldots, nC$$

5. **Assignment to Open Warehouses:** A client can only be assigned to an open warehouse.

$$y_{ij} \leq x_i \quad \forall i = 1, \ldots, nW,\ j = 1, \ldots, nC$$

6. **Binding Variables:** If a client is assigned to a warehouse, the warehouse must supply at least one unit.

$$\text{amountServed}_{ij} \geq y_{ij} \quad \forall i = 1, \ldots, nW,\ j = 1, \ldots, nC$$

This formulation ensures that the solution is both cost-effective and feasible, respecting customer demands, warehouse capacities, and business rules.

Appendix B - Tables and Metrics

Table 3. Comparison of LP Solvers with and without Restrictions Across Datasets

Dataset	Metric	LP (No Restriction)	LP (With Restriction)
cap44	Execution Time (seconds)	0.64	0.72
	Solution Found	1235500.45	1327496.93
cap92	Execution Time (seconds)	0.17	0.73
	Solution Found	855733.5	1080811.69
cap123	Execution Time (seconds)	0.38	3.74
	Solution Found	895302.32	1095811.69
cap124	Execution Time (seconds)	0.38	3.44
	Solution Found	946051.33	1118345.66

Table 4. Comparison of CP Solvers with and without Restrictions Across Datasets

Dataset	Metric	CP (No Restriction)	CP (With Restriction)
cap44	Execution Time (seconds)	600.17	600.37
	Solution Found	1586247.5	1452414.71
	Optimality Gap (%)	98.35%	98.20%
	Search Sp. (br. /s)	30962.3	36559.5
cap92	Execution Time (seconds)	600.14	600.19
	Solution Found	916489.13	1374624.01
	Optimality Gap (%)	98.39%	98.87%
	Search Sp. (br. /s)	23945.0	20403.8
cap123	Execution Time (seconds)	600.28	600.46
	Solution Found	1145804.16	1941252.85
	Optimality Gap (%)	98.47%	99.10%
	Search Sp. (br. /s)	14505.6	13330.3
cap124	Execution Time (seconds)	600.26	600.25
	Solution Found	1113363.93	1549509.59
	Optimality Gap (%)	97.75%	98.39%
	Search Sp. (br. /s)	12256.9	11319.7

Table 5. Solver Performance Across Time Limits

Time Limit	2 minutes	5 minutes	10 minutes	30 minutes	60 minutes
Time (seconds)	120.14s	300.15s	600.37s	1800.09s	3600.06s
Best Solution Found	1820190.39	1542333.29	1452414.71	1430627.43	1429537.83
Gap	98.56%	98.31%	98.20%	98.17%	98.17%
Nº Branches	2187509	10217278	21947969	56112605	95797503
Nº Fails	529694	2398765	5577895	16290684	31055462
Memory Usage	168 MB	226.6 MB	283.1 mb	380.9 MB	474.9 MB
Search Sp. (br. /s)	18213.6	34045.0	36559.5	31172.7	26610.2

Table 6. Performance Metrics by Output Verbosity Level

Verbosity Level	Quiet	Terse	Normal (default)	Verbose
Time (seconds)	-	600.36s	600.37s	600.42s
Lower Bound	-	26142.21	26142.21	26142.21
Best Solution Found	1485744.28	1451732.90	1452414.71	1452506.40
Gap	-	98.20%	98.20%	98.20%
Nº Branches	-	23138180	21947969	21493557
Nº Fails	-	5910487	5577895	5444298
Memory Usage (MB)	-	283.7	283.1	282.4
Search Speed (br. /s)	-	38542.9	36559.5	35799.9

Table 7. Performance Metrics for Different Methods and Scenarios without new restrictions

Method	cap44	cap92	cap123	cap124
Optimal Reference Value	1 235 500,45	855 733,50	895 302,33	946 051,33
LP - OR-Tools	**Optimal**	**Optimal**	**Optimal**	**Optimal**
Execution Time (s)	0,400	0,410	1,430	3,030
CP - OR-Tools	**Optimal**	**Optimal**	**Optimal**	**Optimal**
Execution Time (s)	0,359	0,325	3,452	4,593
LP - Cplex	**Optimal**	**Optimal**	**Optimal**	**Optimal**
Execution Time (s)	0,640	0,170	0,380	0,380
CP - Cplex	*Suboptimal*	*Suboptimal*	*Suboptimal*	*Suboptimal*
Difference (%)	28,4%	7,1%	28,0%	17,7%
Execution Time (s)	600,130	600,080	600,190	600,150

Table 8. Performance Metrics for Different Methods and Scenarios with new restrictions

Method	**cap44**	**cap92**	**cap123**	**cap124**
Optimal Achieved Value	1 327 373,35	1 080 811,69	1 095 811,69	1 118 311,69
LP - OR-Tools	**Optimal**	**Optimal**	**Optimal**	**Optimal**
Execution Time (s)	1,440	22,700	429,280	291,230
CP - OR-Tools	**Optimal**	**Optimal**	**Optimal**	**Optimal**
Execution Time (s)	3,639	14,49	112,01	92,87
LP - Cplex	**Optimal**	**Optimal**	**Optimal**	**Optimal**
Execution Time (s)	0,720	0,730	3,740	3,440
CP - Cplex	*Suboptimal*	*Suboptimal*	*Suboptimal*	*Suboptimal*
Difference (%)	9,42%	27,18%	77,15%	38,56%
Execution Time (s)	600,340	600,130	600,360	600,170