European Journal of Information Technologies and Computer Science
Vol 5 | Issue 4 | August 2025
ISSN 2736-5492

RESEARCH ARTICLE

Comparative Analysis of Optimized GCD and Hybrid LLM-GCD Approaches for Retail Shelf Space Allocation

Ravi Teja Pagidoju*

ABSTRACT

To solve the retail shelf space allocation problem, we need methods that find a good balance between speed and quality. This paper looks at two different methods: an optimized dynamic programming algorithm that uses greatest common divisor (GCD) reduction and a hybrid method that combines Large Language Model (LLM) categorization with parallel GCD optimization. When testing mixed product datasets with 20, 50, and 100 items, it is clear there is a trade-off between speed and optimality. The optimized GCD method finds the best solutions while using the available space, but it takes longer to compute (2.49 ms to 75.74 ms). The hybrid approach shows better computational efficiency (1.89 ms to 10.68 ms) by using smart product grouping and parallel processing, and it uses 91%–98.9% of the space. The hybrid method gives up 3%–9% of possible profit, but it cuts computation time by 78%–85%, which makes it good for real-time retail applications where speed is very important.

Keywords: GCD optimization, hybrid algorithms., large language models, retail shelf space allocation.

Submitted: June 06, 2025

Published: August 12, 2025

10.24018/ejcompute.2025.5.4.155

Independent Researcher, Software and AI Development in Retail, United States.

Corresponding Author:
e-mail. Pagidojuraviteja1@gmail.com

1. Introduction

As physical retail space gets more expensive and customer expectations keep going up, the retail sector is under more and more pressure to make the best use of its shelf space. Managing shelf space well can directly affect a store's profits because better product placement can boost sales by 10%–15% without adding to the cost of inventory [1]. The shelf space allocation problem (SSAP) is figuring out how many facings of each product to put on a shelf with limited space so that the store makes the most money overall.

Since the 1970s, a lot of research has been done on the problem of allocating shelf space. Corstjens and Doyle [2] were the first to use mathematical models to figure out how to best use space. This laid the groundwork for optimization methods. Their research showed that giving the right amount of shelf space to each product could have a big effect on how much money a store makes. Zufryden [3] built on this work by suggesting dynamic programming solutions for problems that involve choosing products and allocating space at the same time.

When the SSAP was set up as a knapsack problem, dynamic programming became a very useful way to solve it. The basic dynamic programming method takes O (n × C) time, where n is the number of products and C is the shelf capacity. For big problems with a lot of products or very precise capacity measurements, this gets very expensive to compute.

Recent work on dynamic programming for SSAP has been aimed at making the calculations less complicated. Czerniachowska and Lutosławski [1] came up with an optimized dynamic programming method that uses the greatest common divisor (GCD) of the widths of the products. When products have the same width factors, the GCD optimization can cut down on the number of possible solutions by a lot. If, for instance, all the products have widths that are multiples of 4 cm, the problem size goes down by a factor of 4. This optimization changes the time complexity to O (n × C/g), where g is the GCD of all the product widths.

However, in real stores, products often come in a range of widths that don't have any common factors. When the widths of the products are prime numbers or coprime values, the GCD becomes 1, which takes away the advantage of faster calculations. For example, a product mix with

widths of 7 cm, 11 cm, and 15 cm gives GCD = 1, which means that the full O (n × C) calculation is needed.

Using AI in retail optimization is a new area of research. Large Language Models (LLMs) are very good at figuring out how data is related and what patterns it has. While LLMs have been applied to demand forecasting and customer behavior analysis, their application to shelf space optimization remains unexplored. LLMs' ability to understand semantics could help them find product groups that traditional math methods might miss.

There are several problems with the current ways of allocating shelf space:

1. *Computational Complexity:* When products have mixed widths and GCD = 1, optimized dynamic programming loses its speed advantage and needs to do all the calculations.
2. *Real-time Requirements:* In today's retail environments, planograms need to be updated quickly and often, which makes traditional optimization methods useless for big stores.
3. *Pattern Recognition:* Mathematical methods might not pick up on the semantic connections between products that could help with better grouping.

This study deals with these problems by suggesting a hybrid approach that combines LLM-based product categorization with parallel GCD optimization. The goal is to find a balance between the quality of the solution and the speed of the computation.

The goal of this study is to:

1. Use and test a better GCD-based dynamic programming algorithm for allocating shelf space.
2. Create a mixed method that uses OpenAI GPT-3.5 Turbo to group products smartly and then uses parallel GCD optimization.
3. Use mixed product datasets to compare both methods and figure out how much better one is at finding a solution than the other.
4. Give practical advice for how to use the results in retail settings.

2. Methodology

2.1. Problem Formulation

The problem of allocating shelf space is set up as a bounded knapsack problem. The goal is to find the number of facings x_i for each of n products, each with a width w_i, a profit per facing p_i, and a maximum number of facings bi, so that total profit is maximized while staying within shelf capacity C.

$$Maximize: \sum (i = 1 \ to \ n) \, pi \cdot xi$$

Subject to:

$$\sum (i = 1 \ to \ n) \, wi \cdot xi \leq C$$

$$0 \leq xi \leq bi, \ xi \in \mathbb{Z}, \ \forall i \in \{1, \ldots, n\}$$

2.2. Optimized GCD Approach

The optimized GCD algorithm makes calculations easier by using common factors in product widths:

1. *Calculate g:* $gcd(w1, w2, \ldots, wn)$
2. *Transform to reduced problem space:* $C' = \lfloor C/g \rfloor$
3. *Scale product widths of products:* $w'i = wi/g$
4. Apply dynamic programming with reduced dimensions
5. *Time complexity:* $O(n \times C/g)$

When the widths of the products are different (for example, 7 cm, 11 cm, and 15 cm), the GCD is 1, which doesn't help with calculations. The algorithm needs to look at the whole O (n × C) state space.

2.3. Hybrid LLM-GCD Approach

The hybrid approach leverages LLM capabilities for intelligent product grouping:

- *Phase 1: LLM Categorization:*

1. OpenAI GPT-3.5 Turbo gets products with information about their width and category.
2. The LLM finds patterns and puts products that are similar in width into groups.
3. Groups are made to get the most out of GCD within the group.

- *Phase 2: Parallel GCD Optimization:*

1. Based on how much profit each product group makes, they get a certain amount of shelf space.
2. Each group runs the optimized GCD algorithm on its own.
3. Using Python's ThreadPoolExecutor for parallel processing
4. Results are put together for the final allocation.

2.4. Experimental Setup

Testing used fake retail data that showed different types of products:

1. *Small:* 20 items, 200 cm of shelf space
2. *Medium:* 50 products, 400 cm of shelf space
3. *Large:* 100 items, 800 cm of shelf space

The products came in a range of widths (5–25 cm) and had many prime or coprime values, which made sure that the GCD for the whole dataset was 1. This is like real-life stores where products from different companies have different sizes.

3. Results

3.1. Performance Comparison

We ran tests on a local machine and only looked at how long it took to do the calculations (not including LLM API

TABLE I: Performance Comparison-Optimized GCD vs. Hybrid LLM-GCD

Problem size	Method	Computation time	Total profit	Space utilization
20 products	Optimized GCD	2.49 ms	$162.13	100.0%
20 products	Hybrid	1.89 ms	$150.67	91.0%
50 products	Optimized GCD	27.39 ms	$337.41	100.0%
50 products	Hybrid	5.09 ms	$310.78	98.3%
100 products	Optimized GCD	75.74 ms	$811.04	100.0%
100 products	Hybrid	10.68 ms	$754.10	98.9%

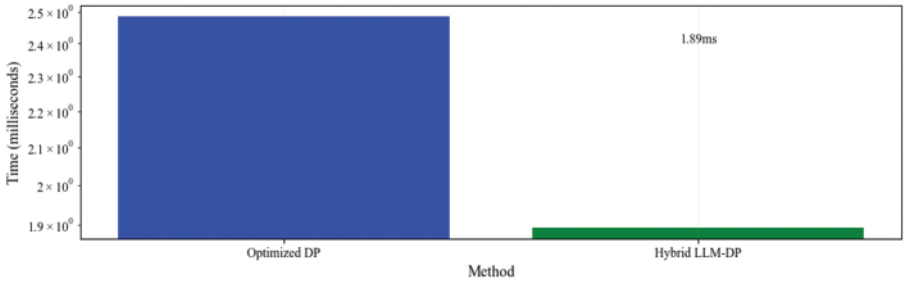

Fig. 1. Computation time comparison for 20 products—optimized GCD vs. hybrid LLM-GCD.

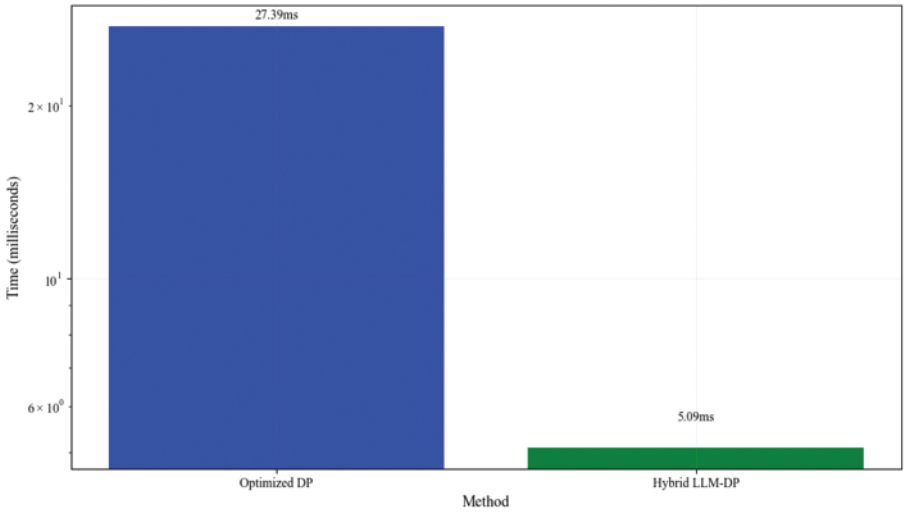

Fig. 2. Computation time comparison for 50 products—optimized GCD vs. hybrid LLM-GCD.

calls for the hybrid approach). The results show that there are clear trade-offs between the two methods (Table I).

3.2. *Computational Efficiency Analysis*

The hybrid approach demonstrates significant computational advantages:

- As shown in Fig. 1, 24.10% faster for 20 products (1.89 ms vs. 2.49 ms)
- As shown in Fig. 2, 81.45% faster for 50 products (5.09 ms vs. 27.39 ms)
- As shown in Fig. 3, 85.9% faster for 100 products (10.68 ms vs. 75.74 ms)

This efficiency gain comes from two factors:

1. *Smart Grouping:* LLM finds products with widths that work well together and puts them in groups with a better local GCD.
2. *Parallel Processing:* Groups are optimized at the same time, using more than one CPU core.

3.3. *Solution Quality Analysis*

While the hybrid approach excels in speed, it sacrifices some solution quality:

- As shown in Figs. 4 and 5 92.9% of optimal profit for 20 products
- As shown in Figs. 6 and 7 92.1% of optimal profit for 50 products
- As shown in Figs. 8 and 9 93.0% of optimal profit for 100 products

The way space is used is also similar; the hybrid approach gets 91%–98.9% of the space used (Figs. 10–12) compared to the best 100%. This happens because:

1. The way capacity is divided between groups might not be the best way.
2. Because of group boundaries, some shelf space is still empty.
3. TheLLM's grouping is smart, but it might not be mathematically optimal.

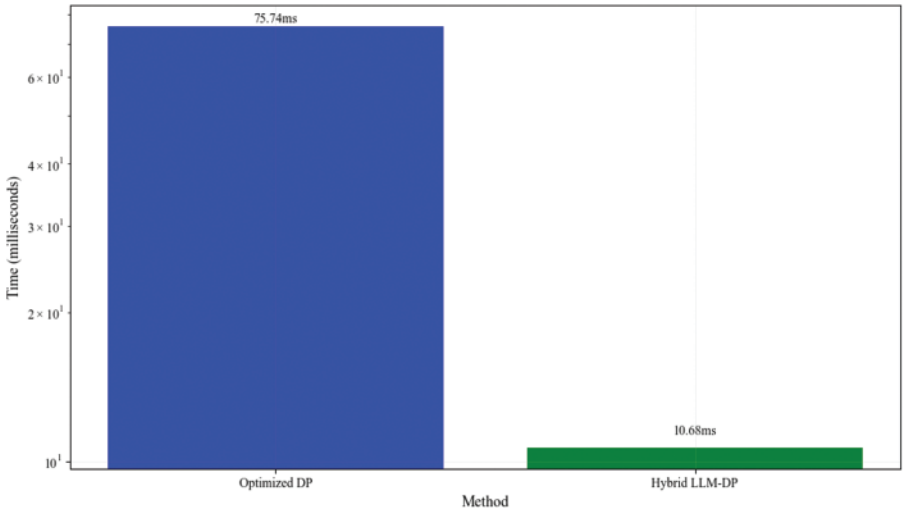

Fig. 3. Computation time comparison for 100 products—optimized GCD vs. hybrid LLM-GCD.

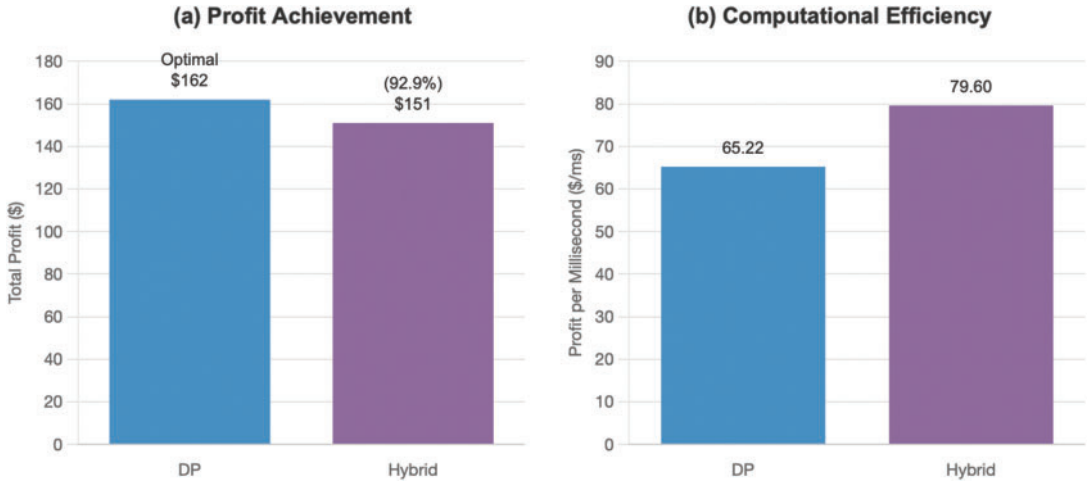

Fig. 4. Profit achievement/computation efficiency comparison for 20 products—optimized GCD vs hybrid LLM-GCD.

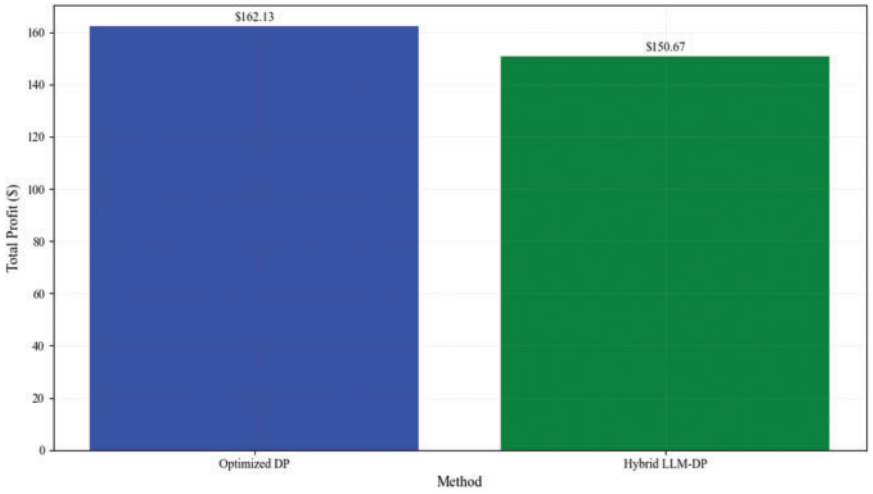

Fig. 5. Profit comparison for 20 products—optimized GCD vs. hybrid LLM-GCD.

4. Discussion

4.1. Trade-off Analysis

The results show that there is a basic trade-off between how fast the computer can work and how good the solution is. The optimized GCD method guarantees the best solutions, but it takes a lot longer to compute, especially as the size of the problem grows. The hybrid method gives up 2%–9% of possible profit but cuts down on computation time by 78%–85%.

4.2. Practical Implications

When it comes to retail, the choice between approaches depends on the specific needs:

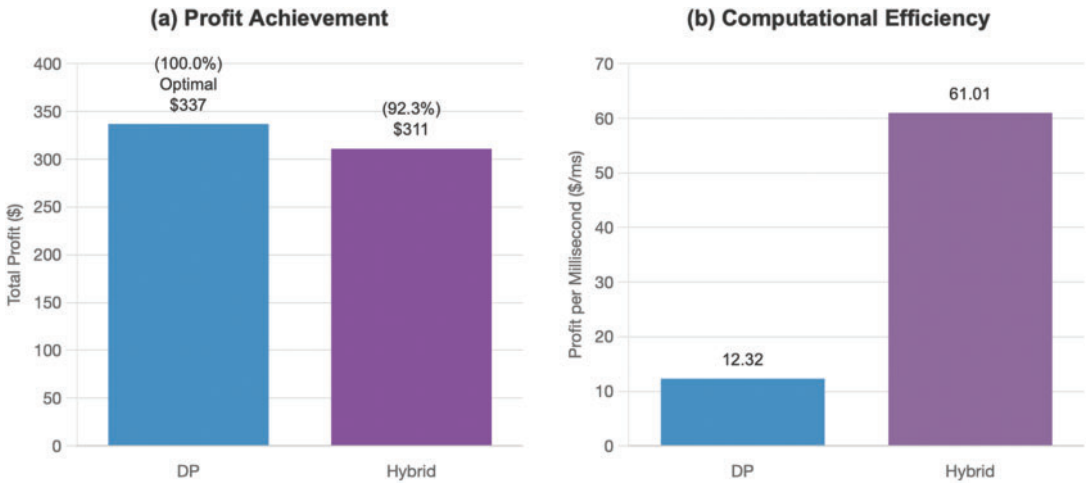

Fig. 6. Profit achievement/computation efficiency comparison for 50 products—optimized GCD vs. hybrid LLM-GCD.

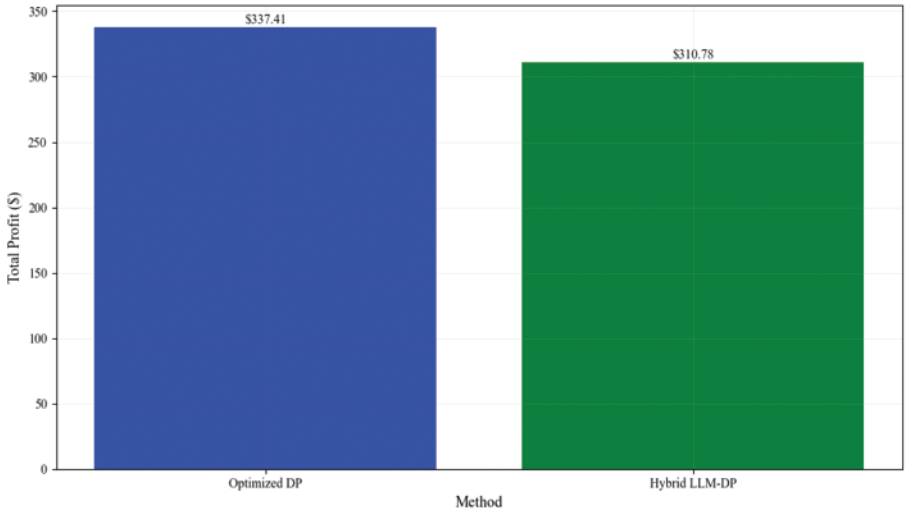

Fig. 7. Profit comparison for 50 products—optimized GCD vs. hybrid LLM-GCD.

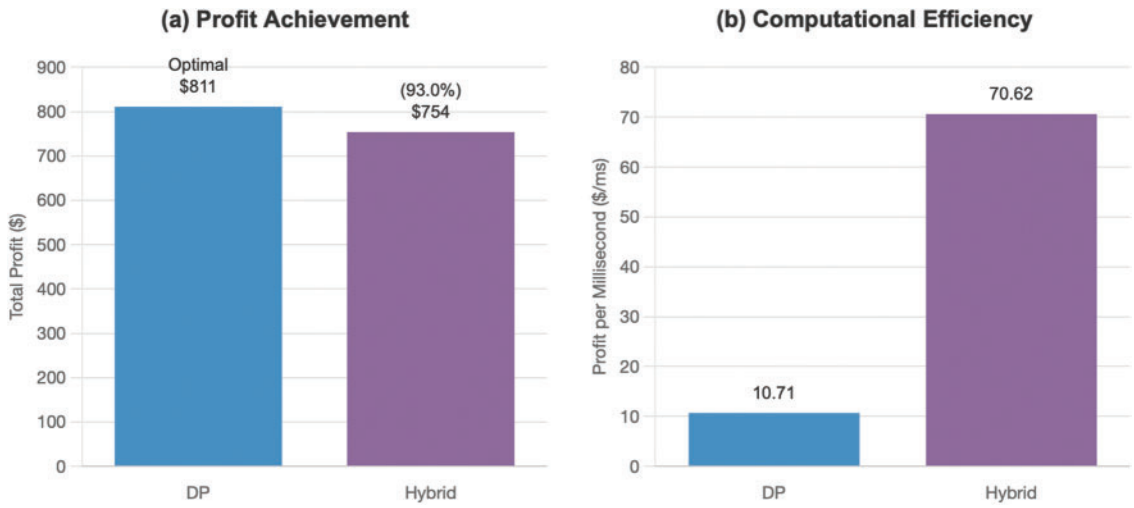

Fig. 8. Profit achievement/computation efficiency comparison for 100 products—optimized GCD vs. hybrid LLM-GCD.

1. When is optimized GCD the best choice?

- The best solution is very important (for example, for high-value products)
- Updates to planograms don't happen very often
- Time to do the math is not a problem

2. When to use Hybrid LLM-GCD:

- Updates need to happen in real time or often
- Solutions that are close to optimal are fine
- Many stores need planograms made quickly

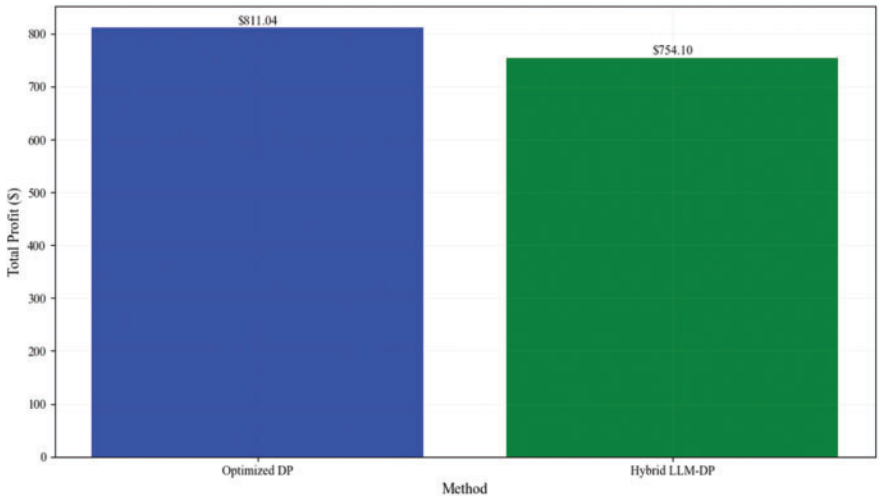

Fig. 9. Profit comparison for 100 products—optimized GCD vs. hybrid LLM-GCD.

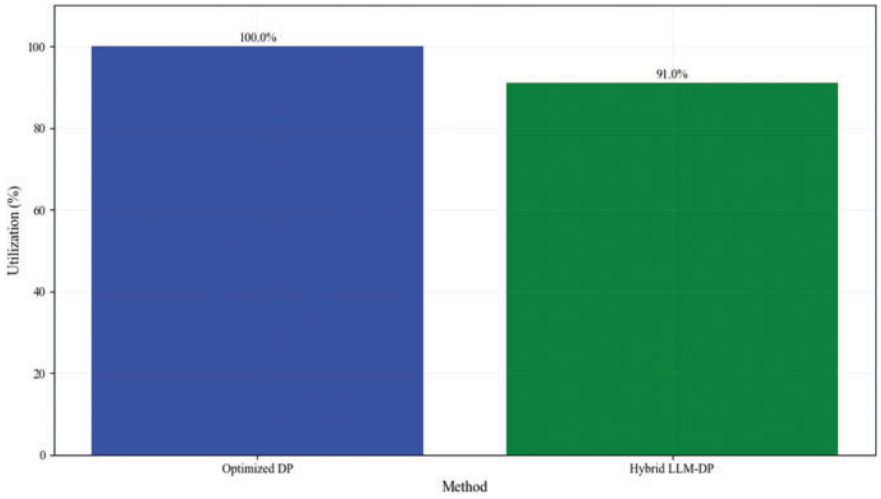

Fig. 10. Shelf space utilization comparison for 20 products—optimized GCD vs. hybrid LLM-GCD.

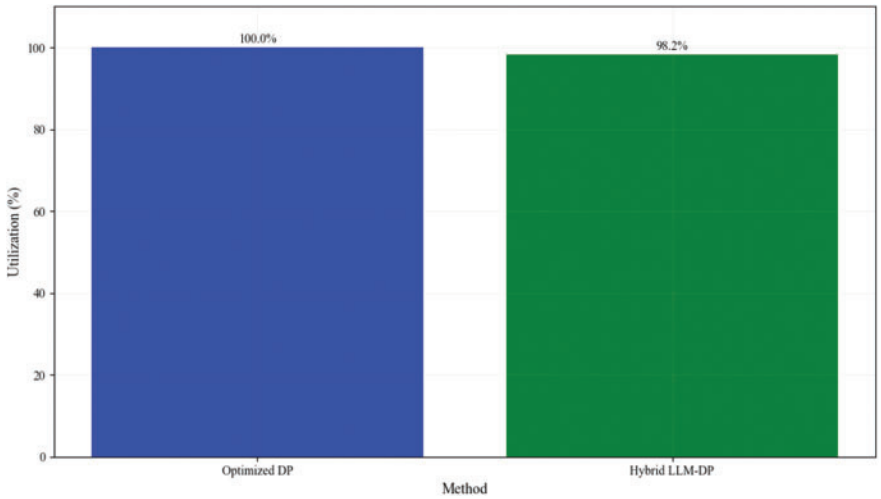

Fig. 11. Shelf space utilization comparison for 50 products optimized GCD vs. hybrid LLM-GCD.

- The system must be able to handle thousands of products

4.3. Cost Considerations

The hybrid approach requires API calls to OpenAI, incurring additional costs. However, the dramatic reduction in computation time may offset these costs in large-scale deployments where server resources are expensive.

5. Conclusion

This study shows that using LLM intelligence along with traditional optimization algorithms is a good way to solve

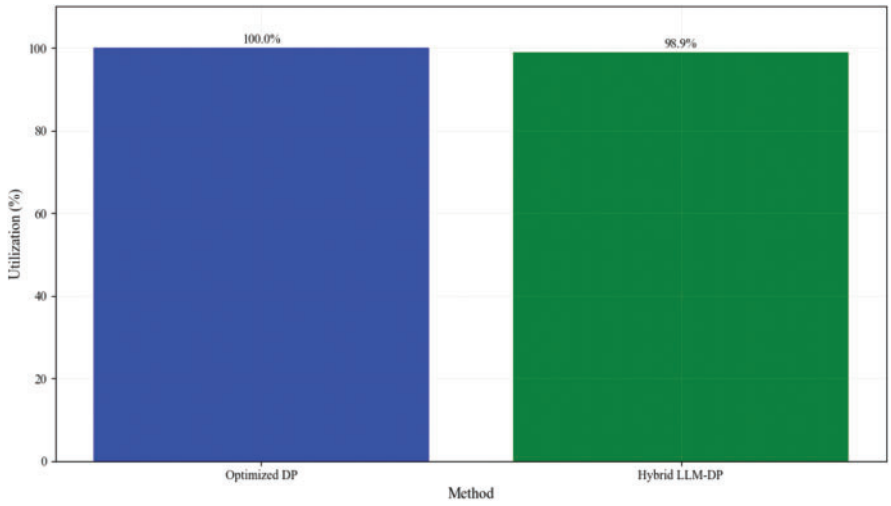

Fig. 12. Shelf space utilization comparison for 100 products—optimized GCD vs. hybrid LLM-GCD.

the problem of allocating retail shelf space. The hybrid LLM-GCD method cuts down on computation time by 78% to 85% while keeping 91% to 99% of the quality of the best solution.

This work makes several important contributions to the field. First, it is the first time that LLMs have been used to optimize shelf space by grouping items in smart ways. The research also shows that there is a trade-off between speed and optimality in retail optimization scenarios. Also, it creates a useful framework that lets decision-makers pick the best optimization method for their business needs.

The results suggest that hybrid AI-optimization methods can solve the problems that traditional methods have with computing power while still giving solutions that are good enough for real-world use.

6. Future Research

Several ways to investigate more. You can use GPT-4 or other specialized models to find better ways to group things with Advanced LLM Integration. Dynamic capacity allocation that uses reinforcement learning to find the best way to divide up space between groups. Multi-objective optimization by expanding the hybrid method to look at things other than profit, like freshness, visibility, and cross-selling. Testing with real retail data, including seasonal changes and the effects of promotions, to make sure it works in the real world. You can speed up hardware by using GPU-based parallel processing to solve big problems.

Conflict of Interest

The authors declare that they do not have any conflict of interest.

References

[1] Czerniachowska K, Lutosławski K. Dynamic programming approach for solving the retail shelf-space allocation problem. *Procedia Comput Sci*. 2021;192:4320–9.

[2] Corstjens M, Doyle P. A model for optimizing retail space allocations. *Manag Sci*. 1981;27(7):822–33.

[3] Zufryden FS. A dynamic programming approach for product selection and supermarket shelf-space allocation. *J Oper Res Soc*. 1986;37(4):413–22.