Branch and Bound Optimization for Online Food Orders with Tiered Vouchers, Flash Sales, and Delivery Fees

Ishaq Irfan Farizal - 13524094
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: ishaqirfanfrzl@gmail.com , 13524094@std.stei.itb.ac.id

***Abstract*—Online food-order checkout is affected not only by selected menu prices, but also by package prices, flash-sale prices, voucher thresholds, delivery discounts, platform fees, and handling fees. These interactions can make locally attractive choices suboptimal, especially when a slightly different item combination activates a stronger voucher. This paper models online food checkout as a bounded combinatorial optimization problem whose objective is to minimize final checkout price while satisfying a target meal-value constraint, a main-dish requirement, quantity limits, and voucher rules. A Branch and Bound solver is implemented as the main exact method, with brute force used as a verifier for small cases and three greedy algorithms used as baselines. The controlled synthetic experiment uses 53 menu items, 9 vouchers, 1 fee profile, and 6 checkout scenarios. Branch and Bound matched brute force in all six 8-item validation cases and was then used as the exact reference for larger cases. The results show that greedy methods are much faster, but can produce higher checkout prices, with the largest observed greedy optimality gap reaching 30.77%. These findings support the use of Branch and Bound for controlled checkout optimization when voucher thresholds and package-price interactions affect the final price.**

***Keywords*—Branch and Bound, greedy algorithm, brute force, checkout optimization, voucher optimization, online food ordering.**

I. Introduction

Online food-ordering applications often present checkout decisions as simple menu selection. In practice, the final price can depend on several interacting components: food subtotal, package or combo prices, flash-sale prices, delivery fees, platform fees, handling fees, voucher minimum-spend thresholds, discount caps, and promotion eligibility rules. A customer who chooses only the cheapest-looking item may miss a voucher threshold, while another customer who adds a small side item may activate a larger discount and pay less overall. Official GoFood and Grab terms show that food-delivery transactions can include product prices, handling and delivery fees, promotion programs, voucher codes, and usage restrictions [7], [8].

This interaction creates an algorithmic optimization problem. The cheapest valid order is not always obtained by repeatedly taking the item with the best value-to-price ratio or the largest visible discount. Greedy methods are useful baselines because they are fast and simple, but their local decision rule does not generally guarantee a global optimum [2]. In a checkout setting, this weakness becomes more visible because item choices affect not only the subtotal but also voucher eligibility.

This paper studies a controlled online food checkout model for one restaurant and one platform. The goal is to find the minimum final checkout price while satisfying a target meal-value requirement, a main-dish requirement, item quantity limits, and voucher rules. The work is not a real food recommendation system. It does not attempt to predict taste, nutrition, customer satisfaction, or actual food-delivery market behavior. Instead, it isolates the algorithmic structure of checkout optimization under synthetic but realistic rules.

The main contribution is an implementation and experiment using Branch and Bound. Branch and Bound explores the search space exactly while pruning branches that cannot improve the current best solution [3], [4]. Brute force is used only for small validation cases because exhaustive enumeration becomes expensive as the number of items grows [1]. Three greedy baselines are implemented for comparison: value-per-price, discount-first, and voucher-threshold greedy. The implementation repository and reproducibility notes are listed in Appendix A.

The experiment uses 53 synthetic menu items, 9 vouchers, 1 fee profile, and 6 scenarios. The result shows that Branch and Bound matches brute force in small validation cases and produces lower or equal prices than the greedy baselines in the tested model. The largest observed greedy optimality gap is 30.77%, which occurs when a discount-first strategy selects a locally attractive package combination that is more expensive than the Branch and Bound solution after voucher effects are considered.

II. Theoretical Foundation

A. Brute Force

Brute force solves a problem by enumerating all possible candidate solutions and selecting the best valid candidate. For an optimization problem, this approach is exact because it does not skip any candidate in the search space [1]. Its main weakness is combinatorial growth. If there are n menu items and each item i can be selected from quantity 0 to q_i, then the number of quantity combinations is:

product_i (q_i + 1)

This grows quickly even when each item has a small quantity limit. In this paper, brute force is therefore used as a correctness verifier only for 8-item experiment cases. It is not used as the main algorithm for the full dataset.

B. Greedy Algorithms

A greedy algorithm constructs a solution by repeatedly making the locally best choice according to a selected criterion [2]. Greedy algorithms are attractive because they are usually fast and easy to implement. However, a greedy choice is only guaranteed to be optimal for problem classes that satisfy certain structural properties. The checkout model in this paper does not have such a guarantee because item choices can change voucher eligibility and final discount values.

Three greedy baselines are used: greedy_value_per_price, which selects items with high meal value per unit price; greedy_discount_first, which prioritizes items with large direct price reductions; and greedy_threshold, which prioritizes items whose prices are close to voucher thresholds. These strategies represent plausible local rules that a customer or simple program might use, but they do not perform global search.

C. Branch and Bound

Branch and Bound is an exact optimization method that represents candidate solutions as a state-space tree. The algorithm branches by expanding partial decisions, maintains the best complete solution found so far, and computes bounds to decide whether a partial branch can still improve the incumbent solution [3], [4]. If a branch cannot lead to a better valid solution, it is pruned.

For minimization, a lower bound estimates the smallest possible objective value that can still be reached from a partial state. If this lower bound is already greater than or equal to the current best final price, the branch can be discarded without losing the optimum. The method remains exact when the bound is safe, meaning that it never overestimates the best possible continuation of the branch.

D. Relation to Bounded Item Selection

The checkout problem resembles a bounded item-selection problem because each menu item has a maximum quantity and each selected quantity contributes price and value. This is related to bounded knapsack-style optimization, where a solution is built by choosing limited quantities of items under an objective and constraints [6]. The checkout model has an additional difficulty: the final price is not simply the sum of item prices. Vouchers introduce non-linear effects because crossing a subtotal threshold can reduce the final price.

III. Checkout Optimization Model

A. Menu Item Model

Each menu item is represented by:

Field	Meaning
`item_id`	Unique item identifier
`name`	Menu item name
`category`	`main`, `drink`, `side`, `snack`, `dessert`, or `combo`
`base_price`	Normal listed price
`flash_sale_price`	Effective checkout price
`value_score`	Synthetic meal/fullness value
`max_quantity`	Maximum selectable quantity

The `value_score` field is a synthetic measure of meal sufficiency. It is not taste, nutrition, or preference. Combo items are treated as atomic items because checkout platforms sell packages as one selectable menu entry. Their internal components are not split by the optimizer.

B. Voucher and Fee Model

Each voucher has a type, minimum food subtotal, discount type, discount value, discount cap, delivery discount cap, category restriction, and stack group. Food vouchers reduce eligible food subtotal. Delivery vouchers reduce delivery fees. Vouchers in the same stack group are mutually exclusive, so the implementation selects the best food voucher and the best delivery voucher separately.

The fee model contains:

Fee	Value
Base delivery fee	20000
Rush delivery fee	31000
Platform fee	3000
Handling fee	2000

The base delivery fee is used in normal scenarios. The rush delivery fee is used when a scenario marks `rush_hour = true`.

C. Objective Function

The objective is to minimize final checkout price:

```
final_price =
  food_subtotal
  - selected_food_voucher_discount
  + delivery_fee
  - selected_delivery_voucher_discount
  + platform_fee
  + handling_fee
```

The implementation computes `food_subtotal` using `flash_sale_price`, not `base_price`, because the flash-sale price is the effective checkout price.

D. Constraints

A solution is valid if it satisfies all of the following constraints:

- `total_value_score >= scenario.target_value`
- `quantity_i <= item_i.max_quantity for every item_i`
- `at least one main or combo item if requires_main_dish is true`
- `voucher minimum spend, caps, category restrictions, and stack groups are respected`

The main-dish requirement prevents orders that satisfy value only through drinks, sides, or desserts. Side items such as rice, tofu, and tempe are allowed as add-ons, but their value scores and quantity limits are kept low so they do not dominate the meal model.

E. Scope

The model is intentionally narrow: one restaurant, one platform, static platform and handling fees, base and rush delivery fee variants, synthetic voucher rules, and synthetic menu-value scores. The model excludes multiple restaurants, multiple platforms, real-time scraping, real customer preference, nutrition optimization, driver-distance modeling, and full real-world voucher edge cases.

IV. Algorithm Design

A. Checkout Evaluation

The checkout evaluator receives selected item quantities, voucher rules, fee data, and scenario settings. It computes the food subtotal, total base subtotal, total value score, applicable delivery fee, selected vouchers, discounts, and final price. Voucher selection is performed by stack group. For each group, the voucher with the largest applicable discount is selected.

The food-voucher discount is computed only when the food subtotal reaches the voucher minimum. If a voucher has a category restriction, only items in the eligible category contribute to the discount. Delivery-voucher discounts are capped by the delivery fee.

B. Brute Force Verifier

The brute-force solver recursively enumerates all possible item quantities from zero to each item's max_quantity. For each complete quantity assignment, it checks whether the order satisfies the scenario constraints. If valid, the checkout evaluator computes the final price. The best valid final price is returned.

This method is exact, but its search space is large. For this reason, it is used only for 8-item validation cases. The experiment runner fails if the Branch and Bound result differs from brute force in those validation cases.

C. Greedy Baselines

The implementation uses one shared greedy procedure with different item-ordering rules. The procedure keeps adding the currently preferred item until the order becomes valid or the item reaches its quantity limit. If the preferred ordering cannot produce a valid order, it falls back to the original item order to complete the solution.

The three greedy rules are:

Algorithm	Ordering rule
`greedy_value_per_price`	Highest `value_score / flash_sale_price` first
`greedy_discount_first`	Largest `base_price - flash_sale_price` first
`greedy_threshold`	Item price closest to a selected food-voucher threshold first

These algorithms are expected to be fast because they do not backtrack. Their weakness is that they commit to local choices and do not search alternative combinations after a valid order has been found.

D. Branch and Bound Solver

The Branch and Bound solver searches over item quantities exactly. Items are first sorted by a promising price-to-value ratio. The solver initializes its incumbent solution using `greedy_value_per_price`, then recursively visits item decisions. At each item index, it tries quantities from max_quantity down to zero.

The solver uses two pruning rules. First, it prunes branches that cannot satisfy the target value even if all remaining items are selected:

```
current_value + maximum_remaining_value < target_value
```

Second, it prunes branches whose optimistic lower-bound final price cannot beat the incumbent. The lower bound is voucher-aware. It considers the current subtotal, possible food-voucher thresholds, best delivery discount, delivery fee, platform fee, and handling fee.

This detail is important. A naive rule such as "prune when current subtotal already exceeds the best final price" is unsafe in this problem because adding an item can activate a voucher and lower the final checkout price. The implementation therefore uses a safe optimistic lower bound instead of pruning directly on current subtotal.

The solver returns the best checkout solution, selected quantities, explored-state count, pruned-state count, and runtime.

V. Experiment Setup

A. Dataset

The experiment uses controlled synthetic checkout data. Synthetic data is used because public food-delivery datasets rarely include complete voucher rules, discount caps, platform fees, and delivery-fee conditions. The purpose is to compare algorithm behavior under known rules, not to claim real-market accuracy. The menu and voucher presentation were manually observed from a GrabFood restaurant page as a practical reference, but the final dataset was simplified and should not be treated as an exact scrape of GrabFood data [9].

The dataset contains:

Component	Count
Menu items	53
Vouchers	9
Restaurant/platform fee profiles	1
Scenarios	6

The menu is based on one restaurant model. The voucher set includes food discounts with no minimum, 40000-minimum discounts, 100000-minimum discounts, and fixed delivery discounts. The delivery voucher V005 is frequently selected in the generated results, so delivery-voucher conclusions are treated cautiously.

B. Scenarios

The six scenarios are:

Scenario	Target value	Rush hour	Purpose
Normal Single Meal	12	false	Minimum valid one-person meal
Normal Two Person Meal	24	false	Two-person meal comparison
Voucher Threshold Trap	30	false	Tests whether extra items can unlock stronger food vouchers
Promo Package Trap	34	false	Tests discounted package interactions
Rush Hour Delivery Trap	24	true	Tests higher delivery fee effect
Group Order High Minimum	48	false	Tests high-minimum vouchers

All scenarios require at least one main or combo item.

C. Algorithms Compared

The experiment compares:

Algorithm	Role
`branch_and_bound`	Main exact solver
`brute_force`	Exact verifier for 8-item cases
`greedy_value_per_price`	Greedy baseline
`greedy_discount_first`	Greedy baseline
`greedy_threshold`	Greedy baseline

D. Experiment Scale

The item counts tested are:

```
8, 12, 16, 20, 24, 28, 32, 40, 53
```

Branch and Bound and all greedy baselines are run for every item count and every scenario. Brute force is run only for 8-item cases.

E. Metrics

The recorded metrics are final checkout price, total value score, food subtotal, food discount, delivery discount, selected food voucher, selected delivery voucher, selected items, explored states, pruned states, runtime, and optimality gap.

The greedy optimality gap is computed against the Branch and Bound result:

```
optimality_gap = (greedy_price - branch_and_bound_price)/
branch_and_bound_price * 100%
```

For 8-item cases, Branch and Bound is additionally checked against brute force.

VI. Results and Analysis

A. Summary by Algorithm

The experiment generated 222 result rows. Branch and Bound was run on 54 scenario-size cases. Brute force was run on 6 validation cases. Each greedy baseline was run on 54 cases.

Algorithm	Cases	Average gap (%)	Average runtime (ms)	Best case price	Worst gap (%)
branch_and_bound	54	0	233.870	36400	0
brute_force	6	0	806.453	42000	0
greedy_discount_first	54	18.12	0.023	46200	30.77

greedy_threshold	54	14.35	0.024	45000	26.37
greedy_value_per_price	54	6.52	0.030	42000	17.58

Branch and Bound has zero gap because it is used as the exact reference for the tested model. Brute force also has zero gap in the 8-item validation cases because it matched Branch and Bound in every validation scenario.

B. Brute-Force Validation

The 8-item cases validate the exact solver. In all six scenarios, Branch and Bound produced the same final price as brute force.

Scenario	Branch and Bound price	Brute force price	Match
Normal Single Meal	42000	42000	Yes
Normal Two Person Meal	64200	64200	Yes
Voucher Threshold Trap	64200	64200	Yes
Promo Package Trap	64200	64200	Yes
Rush Hour Delivery Trap	75200	75200	Yes
Group Order High Minimum	80400	80400	Yes

This validation does not prove correctness for every possible larger input, but it verifies that the Branch and Bound implementation matches exhaustive search on the small cases where exhaustive search is still practical.

C. Greedy Failure Case

The largest observed greedy gap occurs in `Normal Two Person Meal` with 20 items. The Branch and Bound solution selects two `Paket Katsu Saus Barbeque` items and obtains a final price of 54600. The discount-first greedy baseline selects two `Paket C` 4 items and obtains a final price of 71400, which is 30.77% higher.

Algorithm	Final price	Total value	Food subtotal	Food discount	Selected items
branch_and_bound	54600	24	56000	22400	2x Paket Katsu Saus Barbeque
greedy_value_per_price	64200	34	72000	28800	2x Paket B 5
greedy_discount_first	71400	36	84000	33600	2x Paket C 4
greedy_threshold	69000	36	80000	32000	2x Paket C 5

The discount-first solution receives a larger absolute discount than the Branch and Bound solution, but still pays more because its food subtotal is much higher. This illustrates the central weakness of local discount-based reasoning: a larger discount does not necessarily imply a smaller final price.

D. Full-Menu Results

For the full 53-item dataset, Branch and Bound continued to find lower or equal prices compared with greedy baselines. The following table shows Branch and Bound results for all six scenarios

Scenario	Final price	Total value	Food subtotal	Food voucher	Delivery voucher	Selected items
Normal Single Meal	36400	12	28000	V003	V005	1x Paket Katsu Saus Barbeque
Normal Two Person Meal	54600	24	56000	V002	V005	2x Nasi Putih; 1x Tahu; 1x Paket B 5
Voucher Threshold Trap	61800	30	68000	V002	V005	1x Paket Katsu Saus Barbeque; 1x Tahu; 1x Paket B 5

Promo Package Trap	64200	34	72000	V002	V005	2x Paket B 5
Rush Hour Delivery Trap	65600	24	56000	V002	V005	2x Nasi Putih; 1x Tahu; 1x Paket B 5
Group Order High Minimum	79300	49	106000	V001	V005	2x Paket B 5; 1x Pkt Komplit A1

Some optimal orders include cheap side items such as rice or tofu. In the current model, these items have low value scores and low quantity limits, so they act mainly as threshold helpers rather than full meal replacements.

E. Runtime and Pruning

Greedy methods are much faster than Branch and Bound. Their average runtime is about 0.023 to 0.030 ms in the generated run. Branch and Bound has an average runtime of 233.870 ms, and its hardest full-menu case, `Group Order High Minimum`, takes 9916.139 ms. This reflects the cost of exact search.

However, pruning significantly reduces the amount of search. Across Branch and Bound rows, the generated insight report records an average of 113410.6 pruned states over 149806.5 explored states. The hardest 53-item case explores 6081898 states and prunes 4608281 states. Without pruning, the bounded quantity search space would be much larger.

F. Interpretation

The results support the narrow claim that threshold discounts and package prices can make simple greedy checkout strategies suboptimal. Greedy algorithms are effective when speed is the only priority, but they can stop at a valid order before considering a cheaper combination that becomes visible only through global search.

The delivery-fee result should not be overstated. Although the experiment contains a rush-hour scenario, the generated rows usually select the same delivery voucher V005. Therefore, the strongest evidence in this experiment concerns food-voucher thresholds and package-price interactions, not delivery-voucher complexity.

The result is also model-dependent. It relies on synthetic value scores, simplified voucher rules, and one restaurant menu. The conclusion should therefore be read as an algorithmic result under controlled assumptions rather than as a claim about real food-delivery behavior.

VII. Conclusion

This paper modeled online food-order checkout as a bounded combinatorial optimization problem with meal-value constraints, item quantity limits, voucher thresholds, discount caps, delivery fees, platform fees, and handling fees. Branch and Bound was implemented as the main exact solver, while brute force was used as a verifier for small cases and three greedy algorithms were used as baselines.

The experiment shows that Branch and Bound matched brute force in all six 8-item validation cases. On larger cases, Branch and Bound was used as the exact reference and found lower or equal final prices compared with the greedy baselines. The largest observed greedy optimality gap was 30.77%, produced by the discount-first greedy baseline in the Normal Two Person Meal scenario with 20 items.

The main finding is that voucher thresholds and package-price interactions can make local greedy rules unreliable. A larger visible discount or a better local value ratio does not always minimize final checkout price. Branch and Bound can handle these interactions because it searches alternative item combinations and prunes only when a branch cannot improve the incumbent solution.

This work has several limitations. The dataset is synthetic, the value score is a simplified meal/fullness score, and the voucher model covers only a controlled subset of real promotion behavior. The experiment uses one restaurant and one platform fee profile. In addition, the delivery-voucher result is weak because one delivery voucher dominates the generated rows. Future work could use richer delivery-fee scenarios, real checkout data if available, multi-restaurant ordering, customer preference constraints, or nutrition constraints.

Appendix

A. Implementation Repository

The implementation repository for this paper is available at:

https://github.com/Res2dinv/makalah_stima

Acknowledgment

I would first like to thank God for giving me health and the change to write this paper.Next, i would like to express my sincere gratitude to Bapak Rilla Mandala and Bapak Rinaldi Munir for their guidance and knowledge in IF2211 Strategi Algoritma, and also to GPT-5.5 High for assisting me in refining the writing, structure, and formatting of this paper. I also wanna thank my friends, family, and Ex for their help,

References

[1] R. Munir, "Algoritma Brute Force (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/02-Algoritma-Brute-Force-%282026%29-Bag1.pdf

[2] R. Munir, "Algoritma Greedy (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available:

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/04-Algoritma-Greedy-%282026%29-Bag1.pdf

[3] R. Munir, N. U. Maulidevi, and M. L. Khodra, "Algoritma Branch and Bound (Bagian 1)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/17-Algoritma-Branch-and-Bound-%282026%29-Bagian1.pdf

[4] R. Munir, N. U. Maulidevi, and M. L. Khodra, "Algoritma Branch and Bound (Bagian 2)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2025-2026/18-Algoritma-Branch-and-Bound-%282026%29-Bagian2.pdf

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 4th ed. Cambridge, MA, USA: MIT Press, 2022.

[6] H. Kellerer, U. Pferschy, and D. Pisinger, Knapsack Problems. Berlin, Germany: Springer, 2004.

[7] Gojek, "GoFood Terms of Use," Gojek Terms and Conditions, updated Jan. 22, 2024. [Online]. Available: https://www.gojek.com/en-id/terms-and-condition/gofood

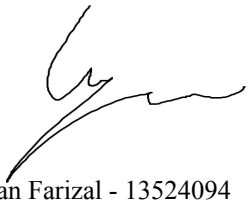
[8] Grab Indonesia, "Terms of Service and Policies: Transport, Delivery and Logistics." [Online]. Available: https://www.grab.com/id/en/terms-policies/transport-delivery-logistics/

[9] GrabFood Indonesia, "GrabFood Indonesia consumer food-ordering page." [Online]. Available: https://food.grab.com/id/id/. Accessed: Jun. 19, 2026.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026



Ishaq Irfan Farizal - 13524094