Comparative Analysis of Heuristics for the Offline Food Delivery Problem on Starlike Graphs

Jose Alfonso Barreiro
Ateneo de Manila University
Quezon City, Philippines
jose.barreiro@student.ateneo.edu

Mikael Giannes Bernardino
Ateneo de Manila University
Quezon City, Philippines
mikael.bernardino@student.ateneo.edu

Erick Gabriel Lopez
Ateneo de Manila University
Quezon City, Philippines
erick.lopez@student.ateneo.edu

John Paul Vergara
Ateneo de Manila University
Quezon City, Philippines
jpvergara@ateneo.edu

ABSTRACT

This paper explores the offline version of the Food Delivery Problem (oFDP) on starlike graphs. In oFDP, customer requests are sent to the central hub/restaurant (depot) which has a single server that has to decide where and when to deliver customer requests (orders). The server is required to return to the depot to pick up new orders before serving requests. The objective is to minimize the maximum flow time, i.e., the maximum time between the submission and completion of a request. Thus, this problem attempts to reduce the longest waiting time that a single customer may experience. The researchers study the oFDP on star-like graphs to develop several heuristics and compare their performance using generated experimental data. Four heuristics—FIFO, MRT, SAT, and OAT—were evaluated and benchmarked. OAT consistently performed well across different scenarios while SAT showed higher average approximation ratios, especially on certain graph types. FIFO and MRT exhibited acceptable performance but were sensitive to graph density. OAT emerged as a strong contender, balancing competitiveness and efficiency. Additionally, two properties were established: one highlighting the efficiency gain from batch delivery options, and the other emphasizing the strategic advantage of allowing waiting.

KEYWORDS

Offline food delivery problem (oFDP), starlike graphs, maximum flow time, heuristics, simulation, scheduling

1 INTRODUCTION

In the past decade, the landscape of purchasing and consuming foods has witnessed a substantial transformation [5], propelled by the rapid growth of online food ordering platforms. This trend experienced an unprecedented acceleration, with the recent global pandemic exerting a profound influence on the relevance of these services. The pandemic catalyzed a shift in lifestyles, prompting individuals to embrace remote work and confinement to their homes, thereby elevating the appeal of convenient online food delivery[6]. In consequence, traditional avenues for purchasing meals have faced limitations within this evolving context[1]. This transformative shift has ushered in a paradigm where online food delivery has become a common choice, garnering favor among both consumers and businesses alike[6].

Nonetheless, this surge in demand for online food delivery services comes hand in hand with the potential challenge of optimizing delivery operations. The imminent risk of service overload necessitates a proactive approach to ensure seamless customer experiences. In this context, the need to manage delivery orders efficiently and mitigate potential delays assumes of utmost importance. Among the array of factors that collectively influence service quality of online food delivery, the duration elapsed between order placement and food delivery – commonly known as flow time – emerges as a critical dimension. This temporal aspect holds the potential to substantially shape the overall customer satisfaction for online food delivery. To further explore the complexities of optimizing delivery operations in the context of offline food delivery, this research aims to address the following questions: 1. What heuristics are applicable as solutions to the offline food delivery problem and how do they compare to one another in terms of efficiency, approximation ratio, robustness, and minimal traversal distance? 2. What properties can be derived from this problem, and how do they contribute to the performance of the algorithms?

In this study, we evaluate four heuristics in offline food delivery, comparing their efficiency and robustness. Additionally, we identify two key properties that contribute to their performance: the efficiency gain from batch delivery and the strategic advantage of allowing waiting. These findings advance understanding and optimization strategies in food delivery operations.

2 REVIEW OF RELATED WORK

In contrast to flow time scheduling problems that have primarily focused on preemptive solutions, the Food Delivery Problem presents a distinct scenario where preemption lacks practicality due to the preference for deploying additional servers over preempting those already in transit. This underscores the significance of the Food Delivery Problem (FDP) as an intriguing research area, driven not only by its real-world applicability, but also by its rarity as a flow time routing problem with the potential for non-trivial competitive algorithms[20].

The offline version of the food delivery problem (oFDP) shares close ties with numerous vehicle routing problems. For instance, the uncapacitated version (where $c = \infty$) with a single vehicle ($k = 1$) FDP, where all requests emerge at time 0, encompasses a variation of the path problem for the traveling salesman. In cases where the vehicle's capacity is finite ($c \neq \infty$), the previously mentioned FDP transforms into the Capacitated Vehicle Routing Problem (CVRP). A broader context is explored through the Dial-a-Ride Problem (DaRP), where not only the delivery destination but also individual pickup points can be specified by each customer. The vehicle is tasked with transporting the customer from their pickup location to their designated delivery spot. While much of the research on offline CVRP/DaRP has concentrated on the objective of minimizing total travel distance, these variations involve intricate problem nuances[21].

While the offline and online versions of the food delivery problem are closely linked, they have a clear distinction rooted in temporal dynamics. In the offline context, fixed parameters are known ahead, while the online scenario deals with incremental delivery requests over time. When deciding to start a delivery at time *t*, adherence to planned routes becomes crucial. This is formalized by scheduling *k* vehicle trips based solely on requests up to time t, connecting offline route design to online adaptability, where efficient decision-making in response to new requests is pivotal[20]. In the context of online food delivery optimization, Smith et al. proposed an innovative approach employing a capacitated multi pickup formulation and a branch-and-cut algorithm to efficiently determine least-cost vehicle routes, taking into account time windows, pickup and delivery constraints, as well as fleet capacity. Their method demonstrated significant success in solving benchmark problem instances with varying node sizes, addressing challenges previously reported as unsolved in the literature [22].

A similar problem to the OFDP is one referred to as the Online traveling salesman problem (Online TSP). In this problem, the server or salesman must navigate a graph to fulfill requests while trying to minimize the makespan all while only having limited information about future requests. Most solutions utilize a myopic approach to make dynamic decisions when creating the makespan. There is also a special case when the graph, which is a metric space, is on a real line and has received much attention in recent years [7, 10]. Several papers have observed that proper waiting could help improve the competitiveness of simple myopic algorithms that are implemented for Online TSP. Proper waiting is quite a useful concept as it allows for the mitigation of risks that arise from a myopic approach to decision making within the algorithm. The possibility of a better decision becoming available if a salesman were to merely wait for a given period of time is quite high yet only has minor overhead when utilized correctly [8, 14]. Currently, there are only competitive solutions for the minimal make-span variation of the Online TSP. To the best of our knowledge, if one were to change the objective to minimizing maximum flow time, there would be no algorithms currently that provide a competitive solution to the Online TSP with the goal of minimizing maximum flow time. This provides an opportunity for further studies to explore this variation.

A related (Offline) TSP problem is the deadline TSP in which every vertex *i* in the graph can only be accessed by the traveling salesman within given time-windows [R*i*, D*i*] of time that is set by the vertex [7]. These types of problems are quite difficult in general due to the constraints of the time-window as configuring a traveling sequence that matches the schedule while minimizing waiting times can be very complex. This principle can be applied to other problems that may recursively schedule trips by selecting longer trips ahead of time. The deadline TSP problem can also be applied to graphs that have a star or star-like topology which accurately represent the structure of companies that wish to deliver products to their clients with the guarantee of same day delivery [15, 16]. In this variation, a delivery driver is located at a single facility and must regularly return to the facility before completing delivery orders. Therefore, it closely mimics the problem in this paper but with additional restriction. Other studies have found that compiling multiple order requests on singular delivery trips within an arm of the star graph before returning to the depot is an effective measure at reducing the total delivery time; thus, ensuring that all orders are delivered [15,27].

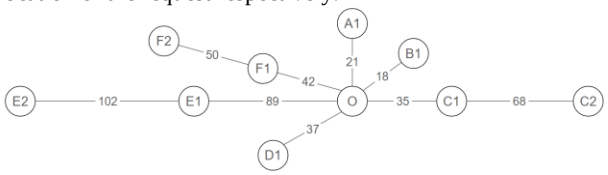
In addition, certain papers have expounded on these types of problems to focus on the aspect of fairness in real world systems as certain cities have different geographical areas with varying wealth and that greatly affects the performance of same day delivery to neighborhoods that are less affluent which causes political and class tension[9]. Researchers have attempted to use machine learning techniques to create a learning framework that can aid companies in adapting their delivery dispatching to have a fairer distribution of requests. This approach focuses on improving the regional service rate throughout the day by using Q-learning.

Another incentive for exploring the Offline Food Delivery Problem (FDP) emerges from an apparently unrelated challenge, the broadcast scheduling problem. In this distinct problem, a server maintains a collection of n pages, each with varying sizes, while requests for these pages are generated over time, each request being a query for one of the pages. The server can broadcast a page to all requests for that particular page, a process taking time equivalent to the size of the page. The overarching goal is to minimize the maximum flow time in this context. Intriguingly, the researchers can map the broadcast scheduling problem onto an uncapacitated single-vehicle FDP scenario on a star-shaped network, where each page's size in the broadcast scheduling problem corresponds to an edge in the FDP problem with a length of s/2. It's worth noting that the FIFO strategy is known to achieve O(1)-competitiveness for the broadcast scheduling problem[8, 11 , 15].

3. PRELIMINARIES

The problem addressed by this study is formally presented as the offline version of the Food Delivery Problem (FDP). Given a weighted star-like graph $G = (V, E)$ with edge lengths $\ell : E \rightarrow \mathbb{R}_{>0}$, where $\ell(e)$ denotes the time needed to traverse the edge e in either direction. There is a special vertex $o \in V$ called the depot,

which represents the restaurant. Located at depot o is a single server which has a capacity $c \in \mathbb{Z}_{>0} \cup \{\infty\}$. The function of this server is to deliver products from the depot to the customers located on the vertices along the different branches of the star-like graph. Customer orders come in the form of a set of $\mathbb{R}$ requests that are sent to the depot. Each request $\rho \in \mathbb{R}$ is denoted by $\rho = (r_\rho, v_\rho)$, where $r_\rho \in \mathbb{R} \geq 0$ and $v_\rho \in V$ are the arrival time and the delivery location of the request respectively.



The server has the limitation that it must travel back to the depot before fulfilling any successive customer requests. In addition, the server cannot preemptively halt an order mid trip. This variation of the problem is offline which means that the server is aware of all requests even before they are released when planning trips. The output of the offline food delivery problem contains k sequences of trips correspondent to the itinerary of the server. The following properties need to be satisfied. For every pair of adjacent trips in any of the k sequences, the starting time of the latter trip is at least the completion time of the former one. Moreover, each request in R is served by exactly one trip in the k sequences; in other words, the sets of served requests in all trips of the k sequences form a partition of R. Let $t\rho$ be the time that a request $\rho \in R$ is served (by the unique trip that serves it). The researchers define the flow time of ρ to be $t\rho - r\rho$. The goal of the problem is to minimize the maximum flow time, i.e, $max(\rho \in R(t\rho - r\rho))$ which is the longest waiting time across all requests that a customer will experience.

The scope of this study is limited to star and star-like graphs of the offline version of the Food Delivery Problem (FDP). The strategies deliberated in this paper will be evaluated with respect to the characteristics of this particular graph structure. Evaluations involving strategies within broader contexts are reserved for prospective research endeavors. Emphasis will be placed on empirically comparing the heuristics in this study, while theoretical proofs of competitiveness will be included whenever feasible.

3.1 Trip Selection Heuristics

3.1.1 FIFO(First in First out):

The heuristic that is most intuitive and commonly used, where the hub will always serve the nodes or customers based on the arrival of their request $(T_{arrival})$. The FIFO heuristic ensures that nodes or customers are serviced in the order of their arrival $(T_{arrival})$. Mathematically, for any two nodes i and j where i arrived before j (i.e., $(T_{arrival,\ i}\ T_{arrival,\ j})$, the service to node i will start before the service to node j.

Criteria 1 (Fairness): The FIFO heuristic maintains fairness in service provision, as it adheres to the principle of serving requests in the order they are received.

3.1.2. MRT (Min Return Time):

This heuristic will prioritize serving the customer node that will return the soonest with consideration with its priority value. Calculation for the estimated return time will be $max(T_{Current} - T_{arrival}) + 2d - max((T_{Current} - T_{arrival}), 0)$. Where "$max((T_{Current} - T_{arrival}), 0)$." pertains to the priority value assigned to each node to prevent starvation which is when some of the requests are ignored for an indefinite period of time while other requests are favored by the heuristic. On the other hand, the symbol d pertains to the time of travel to or from between the depot and the customer node.

Criteria 2 (Priority Calculation): The MRT heuristic calculates priority values for nodes based on the time they spend in the queue and the time needed to complete their service. The longer a node remains in the queue and the faster it can be completed, the higher its priority becomes. This ensures that nodes that are faster to complete and have been waiting longer are serviced earlier, preventing potential starvation and promoting efficiency.

3.1.3. SAT (Single Arm Trips):

This heuristic is based on the MRT but focuses on completing entire arm trips to minimize overhead costs. This heuristic will prioritize serving an entire arm that will result in the server returning the soonest with consideration of its priority value. Arrival time for the entire arm is the maximum arrival time of its nodes $T_{ArmArrival} = max(T_{NodeArr})$, and distance is the total distance from the hub to the leaf node in that arm $d = max(d_n)$. After calculating values for the whole arm, the calculation for the return time will be the same as MRT $max(T_{Curren} - T_{ArmArrival}) + 2d - max((T_{Curren} - T_{ArmArrival}), 0)$.

Criteria 3 (Overhead Reduction): SAT reduces overhead costs by completing arm trips in a manner that minimizes the total distance traveled from the hub to the leaf node in the arm, optimizing resource allocation and energy usage.

3.1.4. OAT (Opportunistic Arm Traversal):

OAT is a hybrid heuristic combining elements of FIFO, SAT and MRT. This heuristic prioritizes cost-efficient whole-arm trips like SAT but seamlessly switches to single-node trips during downtimes, Downtime is defined as the difference between the current time and the arrival time of the next arm $DT = T_{current} - T_{Arr,Arm+1}$. In sparse graphs, where total request costs are less than the time span between the earliest and latest arrivals, OAT employs FIFO for optimal performance. This adaptability optimizes overhead costs while addressing varying graph characteristics.

Criteria 4 (Adaptability): OAT exhibits adaptability by dynamically adjusting the traversal strategy based on the system's state, allowing for efficient use of resources while considering real-time dynamics.

3.1.5. BruteForce:

Brute Force is a straightforward approach where all possible combinations or sequences of servicing nodes are exhaustively examined without any specific heuristic or optimization. In the context of trip selection, this would involve considering every

possible order in which nodes or customers can be serviced, regardless of their arrival time, priority, or any other factors.

Criteria 5 (Exhaustive Search): The Brute Force approach ensures an exhaustive search through all possible combinations of servicing nodes. It explores every permutation without relying on specific heuristics, making it a comprehensive but computationally intensive method.

3.2 Testing and Evaluation of Heuristics

The research design adopted for this study employs a systematic approach to investigate, develop, and rigorously test new and existing heuristics for the offline version of the Food Delivery Problem (FDP). This study encompasses the evaluation of novel heuristics, as described in section 3.1, alongside a comprehensive analysis of other potential heuristics that could be applied to address the problem statement presented.

3.2.1 Data Collection Techniques

As this study tackles the offline version of the food delivery problem, a structured representation is used in the form of starlike graphs. It also uses comparative analysis to garner insights from preexisting studies involving the offline food delivery problem and its possible solutions. Such solutions introduce the usage of a number of heuristics which guide the selection of requests for the problem. After analyzing these performance metrics, optimization techniques are employed to minimize the maximum flow time.

To simulate the performance of these heuristics, the study conducts testing using test cases designed to represent different situations. The testing is divided into two segments based on the number of requests in the graphs: cases with 10 or fewer requests and cases with more than 10 requests. This differentiation is crucial, as it addresses the limitations of the exhaustive brute force approach, which becomes less effective with larger graphs due to its factorial time complexity.

In total, the study generates and tests 5000 cases, 1500 cases among which are used as default scenarios with 250 cases allocated for each graph type, explained in section 3.2.3, while the remaining 2000 cases are predefined test cases representing real-world scenarios and randomly generated cases with more than 10 requests. where the Brute-force approach is unable to provide timely solutions due to its factorial time complexity.

By systematically collecting and analyzing data through these techniques, the study aims to evaluate and compare the effectiveness of various heuristics in addressing the offline food delivery problem.

1. Default
2. Dense Graphs
3. Sparse Graphs
4. Graphs with Many Arms
5. Graphs with few arms
6. Dense Requests
7. Sparse Requests

3.2.2 Performance Metrics

This section outlines a structured approach for evaluating and assessing the effectiveness of the heuristics using the three performance metrics described below:

Approximation Ratio of Each Heuristic's Minimum Maximum Flow Time: This metric compares the solution produced by each heuristic to the optimal solution, providing a measure of how close each heuristic gets to the best possible outcome. The minimum maximum flow time is a critical factor in food delivery logistics, as it represents the time it takes for the last delivery to be completed, affecting overall customer satisfaction and operational efficiency.[4]

Time Complexity: This metric quantifies the computational resources required by each heuristic to find a solution. Lower time complexity indicates faster performance and more efficient resource utilization, which is desirable in real-world applications where time is a crucial factor. [25]

Total Traversal Distance: This metric measures the total distance traveled by delivery vehicles while fulfilling orders. Minimizing traversal distance is important for reducing fuel costs, vehicle wear and tear, and overall environmental impact. [24]

3.2.3 Generation of Test Cases

In addition to the predefined test cases that represent real world case scenarios, the research introduces the generation of additional test scenarios using a discrete probability distribution, specifically the Poisson distribution. This generation process comprises two elements: the graph and the requests.

The graph is generated using the following procedure:

1. Randomize the number of arms.
2. For each arm, randomize the number of nodes along it.
3. For each edge, assign a randomized value for distance.

Table 1: Graph Generation's Key Parameters per Graph Type

	Default	Sparse Request	Dense Request	Sparse Graphs	Dense Graphs	Few Arms	Many Arms
Minimum number of arms	2	2	2	2	2	2	5
Maximum number of arms	8	8	8	8	8	3	24
Average number of nodes per arm*	1.5	1.5	1.5	1.5	1.5	1.5	1.5
Average distance between nodes**	20	30	10	20	20	20	20
Average number of requests per arm*	1.5	1.5	1.5	0.125	10	1.5	1.5

** Randomized using Poisson distribution*

*** Distance is in 1 unit of time = 1 minute*

Numbers are approximated from real life delivery scenarios i.e. Shakeys

Table 1 enumerates several key parameters that influence each process of the graph generation procedure, which are modified to fit each test scenario. The number of arms is randomized within a range set by the minimum and maximum number of arms. In addition, the average number of nodes per arm is another key parameter which is used during the randomization of the number of nodes per arm. Finally, the weight of each edge is generated randomly which utilizes a given average distance between the nodes which affects the density of the generated graph.

Given this generated graph, the researchers can then generate a list of requests:

1. For each arm in the graph, use Poisson distribution to generate the number of requests along an arm
2. For each request, select a node and randomize the request arrival time within the set time interval

Similar to the graph generation procedure, there are key parameters responsible for influencing the generation of the requests on the graph. Maximum time is the total time span. 1 hour time intervals were selected as a standard for test scenarios for the research. Another key parameter is the average number of requests which is used to determine the number of requests per arm of the graph through the usage of Poisson distribution.

As shown in Table 1, graph types are classified according to the values of specific parameters. Sparse request graphs feature a low average distance between nodes, while graphs with dense requests feature a higher than default value. To generate sparse and dense graphs, the average number of requests per arm are altered from the default value. Graphs with few arms and many arms have proportional minimum and maximum numbers of arms. Given these parameters, certain heuristics tend to perform better depending on the graph type. For example, FIFO and MRT would be more efficient on a sparse request graph due to the low average distance between nodes, but SAT perform sub optimally as requests are less dense, making single-arm trips less efficient.

The modification of these key parameters allows for the researchers to generate test scenarios under the following different classifications:

4. EXPERIMENTAL RESULTS

As stated in section 3.2.1, the experimental results section is divided into two segments: the first pertains to graphs with 10 or fewer requests, while the second addresses cases involving more than 10 requests. This differentiation is essential as the Brute-force approach exhibits limited efficacy when confronted with graphs containing more than 10 requests, primarily attributed to its O(n!) time complexity. Consequently, the subsequent analysis and discussions will navigate these distinct scenarios to provide a nuanced understanding of the heuristic's performance under varying request loads.

Section 1: Approximation ratio in Minimizing Maximum Flow Time on 10 or fewer requests

Table 2: Heuristics as to Overall Approximation Ratio

	R_{Best}	R_{Ave}	R_{Worst}
FIFO	1.00	1.49	6.47
MRT	1.00	1.56	5.89
SAT	1.00	2.89	36.55
OAT	1.00	1.24	2.89
Bruteforce	1.00	1.00	1.00

Table 2 compares the approximation ratios of various heuristics for minimizing maximum flow time on 10 or fewer requests. FIFO and MRT achieve optimal ratios in the best-case scenario but perform less efficiently in average and worst cases. SAT exhibits good performance initially but suffers significantly in worst-case scenarios. OAT, on the other hand, maintains consistently good ratios across scenarios, making it a balanced choice among the 4. Bruteforce achieves optimal ratios but is computationally impractical due to its exhaustive nature.

Table 3: Heuristic as to average approximation ratio on Different Graph Types

	Sparse Request	Dense Request	Sparse Graphs	Dense Graphs	Few Arms	Many Arms
FIFO	1.23	1.69	1.16	1.57	1.82	1.31
MRT	1.34	1.73	1.27	1.64	1.91	1.35
SAT	4.72	1.53	7.74	2.4	1.56	2.1
OAT	1.15	1.25	1.12	1.31	1.27	1.19
Bruteforce	1.00	1.00	1.00	1.00	1.00	1.00

Table 3 outlines Approximation Ratios for various heuristics across different graph scenarios in the context of the Offline version of the Food Delivery Problem (FDP). SAT exhibits notably higher average approximation ratios across all scenarios, particularly on sparse graphs and on cases with sparse request distribution, indicating potential inefficiency in these specific situations. Conversely, OAT consistently maintains lower ratios across different graph types and request densities, suggesting its robust performance in various scenarios. Additionally, while FIFO and MRT perform adequately, they show increased ratios on dense graphs compared to sparse ones, highlighting the impact of graph density on the heuristic's efficiency.

Section 2: Approximation Ratio of Minimizing Maximum Flow Time on More Than 10 Requests

Table 4: Heuristics as to Overall Approximation Ratio relative to the Best output among all the heuristics

	R_{Best}	R_{Ave}	R_{Worst}
FIFO	1	2.05	5.84
MRT	1	2.06	5.84
SAT	1	2.04	18.47
OAT	1	1.02	1.81
Bruteforce	-	-	-

Table 4 highlights the relative performance of heuristics concerning their approximation ratios compared to the best performing heuristic at each particular case. OAT stands out by consistently maintaining approximation ratios close to the best case, indicating its robustness across different scenarios. Conversely, FIFO, MRT, and SAT show higher ratios relative to the best answer, particularly in worst-case scenarios, suggesting potential inefficiencies in these situations. This underscores the importance of algorithm choice, with OAT offering promising performance in minimizing maximum flow time.

Table 4: Heuristic's Performance Relative to Time Complexity

	$R_{Worst(<=10)}$	$R_{Worst(10+)}$	Time Complexity
FIFO	6.47	5.84	O(n)

MRT	5.89	5.84	$O(n^2)$
SAT	36.55	18.47	$O(n^2)$
OAT	2.89	1.81	$O(n^2)$
Bruteforce	1.00	-	O(n!)

As achieving a delicate equilibrium between competitiveness and computational efficiency is important, OAT stands out as the best and preferred choice. It maintains a worst case approximation ratio of 2.89 for smaller inputs and 1.82 for larger inputs, indicating effective performance with a quadratic time complexity ($O(n^2)$), making it more scalable than the factorial time complexity of the Brute force. But if scalability for considerably larger datasets is a primary concern, FIFO (First-In-First-Out) might be more suitable. Its approximation ratio of 6.47 for smaller inputs and 5.84 for larger inputs, paired with a linear time complexity (O(n)), indicates more efficient scaling with increasing input sizes compared to heuristics with quadratic or factorial complexities.

The time complexity of FIFO, denoted as O(n), reflects its straightforward and linear handling of requests. Each request is processed sequentially without any nested loops or additional calculations, leading to a direct scaling with the number of requests, denoted as 'n'. In contrast, MRT, SAT, and OAT exhibit quadratic time complexity, O(n^2), as their operations often involve nested loops or calculations that increase quadratically with the number of requests. These heuristics commonly integrate loops for priority calculation and starvation prevention, contributing to their increased computational load.

For the Brute force approach, characterized by a factorial time complexity of O(n!), exhaustive exploration of all possible combinations is employed by considering every permutation of the input elements. This exhaustive search strategy entails exploring 'n * (n-1) * (n-2) * ... * 1 = n!' possible combinations, where the number of possibilities decreases by one at each step. Consequently, the computational effort grows factorially with the number of requests, making it impractical for larger datasets and rendering it inefficient for many real-world applications.

5. PROPERTIES

In addition to heuristic evaluations, we established two properties to deepen our understanding of the problem:

Property 1: Batch Delivery Option Enhances Efficiency

This property highlights the efficiency gain from batch delivery options, affirming that batch delivery generally outperforms individual deliveries in terms of overall efficiency. Leveraging batch delivery strategies can reduce travel time and enhance resource utilization effectively.

Property 2: Option For Strategic Waiting Is Advantageous

This property emphasizes the strategic advantage of allowing waiting in decision-making processes. By demonstrating that there exists at least one solution for any graph G and set of request R where waiting improves decision quality, this property underscores the importance of flexibility in logistical operations. Incorporating waiting options can lead to better outcomes, particularly in scenarios requiring optimization of delivery schedules.

5.1 Elaboration on Key Properties

Property 1: Batch Delivery Option Enhances Efficiency. Improving Efficiency through the Batch Delivery Option. The availability of an option to perform batch delivery, particularly when serving multiple customers on the same arm simultaneously, enhances overall delivery efficiency by reducing travel time and optimizing resource utilization.

Proposition: For any given set of delivery requests *Rn*, the overall delivery efficiency achieved through the batch delivery option $E_{batch,Rn}$ is higher or equal to the efficiency of serving single customers individually $E_{single,Rn}$

$$E_{batch,Rn} \geq E_{single,Rn}$$

Definitions:

$E_{batch,Rn}$: Overall delivery efficiency when the batch delivery option is utilized for the set of requests Rn.

$E_{single,Rn}$: Overall delivery efficiency when single customers are served individually for the set of requests Rn.

T_{batch}: Total travel time for batch delivery.

T_{single}: Total travel time for individual deliveries.

Base Case: When addressing a single delivery request *(|R|=1)*, the delivery efficiency achieved through the batch delivery option is equivalent to serving a single customer individually $E_{batch,Rn} = E_{single,Rn}$. Thus, the base case holds.

Inductive Step: For any set of delivery requests R_{n+1}, let's assume the proposition holds true for R_n i.e., $E_{batch,Rn} \geq E_{single,Rn}$

Case 1: Batch Delivery Option

When the option to perform batch delivery is allowed for the set of requests R_{n+1}, let $|R_{n+1}|$=k, where k > 1. Denote the total travel time for batch delivery as T_{batch}. Then, the overall delivery efficiency with batch delivery is:

$$E_{batch,Rn+1} = \frac{k}{T_{Batch}}$$

Case 2: Serving Single Customers

If only serving single customers individually is considered, the overall delivery efficiency is determined based on individual deliveries: $E_{single,Rn+1}$ Without the option for batch delivery, the efficiency relies on the traditional approach of serving single customers separately. $\frac{1}{T_{single}}$

Comparing Cases:

To show that $E_{batch,Rn} \geq E_{single,Rn}$, we need to show

$$\frac{k}{T_{batch}} \geq \frac{1}{T_{single}}$$

Consider the time saved by batch delivery, denoted by $\Delta T = T_{single} - T_{batch}$. We can express T_{batch} in terms of T_{single} and ΔT as $T_{batch} = T_{single} - \Delta T$

Therefore, $E_{batch,Rn+1} = \frac{k}{T_{single} - \Delta T}$

Since ΔT represents time saved, $T_{single} - \Delta T \leq T_{single}$, which implies $\frac{1}{T_{single} - \Delta T} \geq \frac{1}{T_{single}}$

Thus, $E_{batch,Rn+1} \geq E_{single,Rn+1}$

Property 2: Option For Strategic Waiting Is Advantageous.
Proposition: Let G denote any graph and R denote any set of requests. Then, there exists a solution S, such that the decision quality when waiting is allowed denoted $q_{with\_waiting}$, is greater than or equal to the decision quality when waiting is not allowed, denoted $q_{without\_waiting}$.
In symbolic notation:

$$\forall G \forall R \exists S: q_{with\_waiting} \geq q_{without\_waiting}$$

This statement asserts that regardless of the specific graph or set of requests, there exists at least one solution where the decision quality achieved by waiting is greater than or equal to the decision quality obtained without waiting.
Proof: To prove this property, we need to show that there exists at least one solution where the decision quality achieved by strategic waiting is greater than or equal to the decision quality obtained without waiting.
Let's denote:

$Q_{no\_wait}(G,R)$ as the decision quality without waiting.
$Q_{with\_wait}(G,R,W)$ as the decision quality with strategic waiting, where W represents the waiting strategy.

The change in decision quality due to strategic waiting is given by:

$$\Delta Q(G,R,W) = Q_{with\_wait}(G,R,W) - Q_{no\_wait}(G,R)$$

Now, we need to prove that for any graph G and any set of requests R, there exists a solution S such that: $q_{with\_wait} \geq q_{no\_wait}$, where : q_{with_wait} and q_{no_wait} represent the decision qualities achieved with and without waiting, respectively. Mathematically, we can represent this as:

$$\forall G \forall R \exists S: Q_{with\_wait}(G,R,W) \geq Q_{no\_wait}(G,R)$$

To prove this statement, let's consider All possible cases:
Case 1: $Q_{with\_wait}(G,R,W) = Q_{no\_wait}(G,R)$. If the decision quality with waiting is equal to the decision quality without waiting, then the property holds trivially.
Case 2: $Q_{with\_wait}(G,R,W) > Q_{no\_wait}(G,R)$. If the decision quality with waiting is greater than the decision quality without waiting, then the property holds true.
Case 3: $Q_{with\_wait}(G,R,W) < Q_{no\_wait}(G,R)$. Let's denote the maximum flow time achieved with waiting as $Q_{max\_with\_wait}$ and without waiting as $Q_{max\_no\_wait}$. Then, we have:

$$Q_{max\_with\_wait} = \max_{\rho \in R}(t\rho - r\rho)$$

$$Q_{max\_no\_wait} = \max_{\rho \in R}(t\rho' - r\rho)$$

Where $t\rho$ is the time at which request ρ is served when waiting is allowed, and $t\rho'$ is the time at which request ρ is served when waiting is not allowed.

Given $Q_{with\_wait}(G,R,W) < Q_{no\_wait}(G,R)$.
we have: $Q_{max\_with\_wait} < Q_{max\_no\_wait}$

Let's consider the definition of $Q_{max\_with\_wait}$ and $Q_{max\_no\_wait}$ in terms of $t\rho$ and $t\rho'$. Since both $t\rho$ and $t\rho'$ are arrival times plus the time taken for service, we can express them as:

$$t\rho = r\rho + \text{service\_time}\rho$$
$$t\rho' = r\rho + \text{service\_time}\rho'$$

where service_timeρ and service_timeρ' are the service times for request ρ with and without waiting, respectively.
Since waiting allows the server to optimize its route and potentially reduce service times, we can express service_timeρ as:

$$\text{service\_time}\rho = \text{min\_service\_time}\rho$$

where min_service_timeρ is the minimum possible service time for request ρ when waiting is allowed. Therefore, we have:

$$t\rho = r\rho + \text{min\_service\_time}\rho$$

Now, let's examine $t\rho'$. Since waiting is not allowed, the service time service_timeρ' remains the same as the minimum service time when waiting is allowed. Hence:

$$t\rho' = r\rho + \text{min\_service\_time}\rho$$

Given these expressions for $t\rho$ and $t\rho'$, we can see that both $Q_{max\_with\_wait}$ and $Q_{max\_no\_wait}$ are equal, making the case of $Q_{with\_wait}(G,R,W) < Q_{no\_wait}(G,R)$ impossible.
Thus, the assumption that $Q_{with\_wait}(G,R,W) < Q_{no\_wait}(G,R)$ is invalid, and, therefore, $Q_{with\_wait}(G,R,W) \geq Q_{no\_wait}(G,R)$ holds true.

6. CONCLUSION

This study conducted a thorough evaluation and comparison of four distinct heuristics—FIFO, MRT, SAT, and OAT—against each other and the exhaustive brute force approach, as elaborated in Section 3.1.
The results reveal that SAT stands out with notably higher average approximation ratios across all scenarios, particularly on sparse graphs with many arms, suggesting potential inefficiency in these specific situations. Conversely, OAT consistently maintains lower ratios across different graph types and request densities, indicating its robust performance in various scenarios. Moreover, FIFO and MRT demonstrate acceptable performance, but they exhibit increased ratios on dense graphs compared to sparse ones,

highlighting the sensitivity of the heuristic efficiency to graph density variations.

In addition to heuristic evaluations, we established two properties to deepen our understanding of the problem: Property 1 highlights the efficiency gain from batch delivery options, affirming that batch delivery generally outperforms individual deliveries in terms of overall efficiency. Leveraging batch delivery strategies can reduce travel time and enhance resource utilization effectively. Property 2 emphasizes the strategic advantage of allowing waiting in decision-making processes. By demonstrating that there exists at least one solution for any graph G and set of request R where waiting improves decision quality, this property underscores the importance of flexibility in logistical operations. Incorporating waiting options can lead to better outcomes, particularly in scenarios requiring optimization of delivery schedules.

Overall, when considering both time complexity and average approximation ratio which are the key criteria in evaluating heuristics, the Opportunistic Arm Traversal heuristic emerges as a strong contender as it strikes a balance between competitiveness and computational efficiency, demonstrating effectiveness across different graph types and request densities.

ACKNOWLEDGMENTS

Dr. Andrei D. Coronel, PhD, Jose Alfredo A. de Vera III, Winfer Tabares, Felix II P. Muga: Our distinguished panelists whose critical role in shaping the quality of our work during defense sessions. We thank them for their insightful critiques and invaluable guidance, which significantly contributed to the refinement of our research.

Ateneo de Manila University: We extend our sincere appreciation to Ateneo for providing the conducive environment and resources that allowed this research to flourish. Family, Friends, Classmates, and Teachers: To our support network, we express gratitude for your unwavering encouragement, serving as a constant source of motivation throughout this journey. This study is a collaborative effort, and each acknowledgment represents a vital piece of the puzzle that has culminated in its completion. To everyone who played a part in this journey, we extend our deepest thanks.

REFERENCES

[1] Aday, S., & Aday, M. S. (2020). Impact of COVID-19 on the food supply chain. Food Quality and Safety, 4(4), 167-180. https://doi.org/10.1093/fqsafe/fyaa024

[2] Agnets, A., Cosmi, M., Nicosia, G., & Pacifici, A. (2023). Two is better than one? Order aggregation in a meal delivery scheduling problem. Computers & Industrial Engineering, 183, 109514. doi:10.1016/j.cie.2023.109514

[3] ANON. (2023). Poisson Distribution. Retrieved September 24, 2023 from https://www.sciencedirect.com/topics/mathematics/poisson-distribution

[4]Ausiello, G., Paschos, V. Th. (2006). Reductions, completeness and the hardness of approximability. European Journal of Operational Research, 172(3), 719-739. https://doi.org/10.1016/j.ejor.2005.06.006

[5]Axon, S., Lent, T., Njoku A. (2023). Shifting sustainable lifestyle practices and behaviour during times of pandemic disruptive change: Implications for on-going socio-technical transitions, Energy Research & Social Science, 102, https://doi.org/10.1016/j.erss.2023.103188.

[6] Balieva, G. (2023). Online food purchasing during COVID-19 pandemic.
Scientific Papers Series Management, Economic Engineering in Agriculture and Rural Development,
23(2), 51-58. https://managementjournal.usamv.ro/

[7] Bansal, N., Blum, A., Chawla, S., & Meyerson, A. (2004). Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing (STOC '04) (pp. 1–10). doi:10.1145/1007352.1007385

[8] Bansal, N., Charikar, M., Khanna, S., & Naor, J. (2005). Approximating the average response time in broadcast scheduling. In Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (pp. 215–221). Philadelphia, PA: Society for Industrial and Applied Mathematics.

[9] Bienkowski, M., Kraska, A., & Liu, H.-H. (2021). Traveling repairperson, unrelated machines, and other stories about average completion times. arXiv preprint arXiv:2102.06904.

[10] Bjelde, A., et al. (2000). Tight bounds for online TSP on the line. In Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (pp. 994–1005). New York: ACM.

[11] Borodin, A., & El-Yaniv, R. (2005). Online computation and competitive analysis. Cambridge, MA: Cambridge University Press.

[12] Chang, J., Erlebach, T., Gailis, R., & Khuller, S. (2011). Broadcast scheduling: Algorithms and complexity. ACM Transactions on Algorithms, 7(4), 1–14. doi:10.1145/2000807

[13] Chen, P.-C., Demaine, E. D., Liao, C.-S., & Wei, H.-T. (2019). Waiting is not easy but worth it: The online TSP on the line revisited. arXiv preprint arXiv:1907.00317.

[14] Chen, X., Wang, T., Thomas, B. W., & Ulmer, M. W. (2023). Same-day delivery with fair customer service. European Journal of Operational Research, 308(2), 738–751. doi:10.1016/j.ejor.2022.12.009

[15] Chekuri, C., Im, S., Moseley, B., Fiat, A., & Sanders, P. (2009). Minimizing maximum response time and delay factor in broadcast scheduling. In Algorithms - ESA 2009 (pp. 444–455). Berlin, Heidelberg: Springer.

[16] Cosmi, M., Oriolo, G., Piccialli, V., & Ventura, P. (2019). Single courier single restaurant meal delivery (without routing). Operations Research Letters, 47(6), 537–541. doi:10.1016/j.orl.2019.09.007

[17] Feuerstein, E., & Stougie, L. (2001). On-line single-server dial-a-ride problems. Theoretical Computer Science, 268(1), 91–105. doi:10.1016/S0304-3975(00)00261-9

[18] Fiat, A., & Woeginger, G. J. (1998). Competitive analysis of algorithms. In Online algorithms: The state of the art (pp. 1–12). Berlin, Heidelberg: Springer.

[19]Juris Hartmanis and Richard E. Stearns (1965). On the computational complexity of algorithms. Transactions of the American Mathematical Society, 117(1965), 285-306.

[20] Guo, X., Li, S., Luo, K., & Zhang, Y. (2021). Online food delivery to minimize maximum flow time. arXiv preprint arXiv:2110.15772.

[21] Guo, X., Luo, K., Tang, Z. G., & Zhang, Y. (2022). The online food delivery problem on stars. Theoretical Computer Science, 928, 13–26. doi:10.1016/j.tcs.2022.06.007

[22] Kohar, A., & Jakhar, S. K. (2021). A capacitated multi pickup online food delivery problem with time windows: A branch-and-cut algorithm. Annals of Operations Research. doi:10.1007/S10479-021-04145-6

[23] Lopez, J. (2023). Examining the online food delivery problem on starlike graphs.

[24] Nicola, D., Vetschera, R., & Dragomir, A. (2019). Total distance approximations for routing solutions. Computers & Operations Research, 102, 67-74. https://doi.org/10.1016/j.cor.2018.10.008

[25] Reyes, D., Erera, A. L., & Savelsbergh, M. W. P. (2018). Complexity of routing problems with release dates and deadlines. European Journal of Operational Research, 266(1), 29–34. doi:10.1016/j.ejor.2017.09.020

[26] Russell, S. J., Norvig, P., & Chang, M.-W. (2023). Artificial intelligence: A modern approach (4th ed.). Pearson.

[27] Ulmer, M., Nowak, M., Mattfeld, D., & Kaminski, B. (2020). Binary driver-customer familiarity in service routing. European Journal of Operational Research, 286(2), 477–493. doi:10.1016/j.ejor.2020.03.037