HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

Pareto Front Grid Guided Multi-objective Optimization in Dynamic Pickup and Delivery Problem Considering Two-Sided Fairness

Phan Duc Hung
hung.pd214903@sis.hust.edu.vn

Major: Data Science

Supervisor:	Assoc. Prof. Huynh Thi Thanh Binh	_____________
	Msc. Do Tuan Anh	Signature
Department:	Computer Science	
School:	School of Information and Communications Technology	

HANOI, 06/2025

Acknowledgement

First of all, I am grateful to express my deepest appreciation to Assoc. Prof. Huynh Thi Thanh Binh and PhD. Nguyen Thi Tam. Professor Binh provided a solid foundation of knowledge and invaluable advice throughout the completion of this thesis. Additionally, this thesis could not have been completed without the enthusiastic guidance of PhD. Tam. Both of them supported me wholeheartedly, even during their busiest times.

I would also like to extend my sincere thanks to the teachers, seniors, and friends in the MSO lab who accompanied and stood by me during this challenging and arduous scientific research journey. In particular, I will always remember the constructive feedback and enthusiastic help from Mr. Duc, Tung, and Ly, which significantly contributed to the completion of this thesis.

I am grateful to the friends who have been with me throughout my university years. You have been great companions, not only helping me to improve my knowledge but also sharing the worries and difficulties of life.

Lastly, I am deeply thankful for the endless support of my family. They have worked hard to provide me with the most comprehensive environment for development and have always been there for me, whether in success or failure. I love my family.

Abstract

The dynamic delivery problem poses a complex challenge with many practical applications in logistics and transportation. Unlike the static delivery problem, where all order details are known, the dynamic delivery problem deals with continuously evolving information, with only partial data about orders available at any given moment. This thesis presents the Multi-objective Dynamic Pickup and Delivery Problem with Time Windows framework, which integrates multiple objectives, including minimizing energy consumption, reducing waiting time, and ensuring fairness for both customers and vehicles. The goal is to minimize overall system costs while balancing the customer experience and the workload of service providers. Previous research has primarily focused on optimizing a single objective or converting other objectives into constraints, which can limit the flexibility and effectiveness of the solutions. Our approach tackles this challenge by introducing PFG-2F, a Pareto Front Grid-guided Multi-Objective Evolutionary Algorithm that incorporates two-sided fairness—ensuring equitable treatment for customers and service providers. The experimental results demonstrate that our method substantially outperforms existing multi-objective and single-objective algorithms specifically designed for the dynamic pickup and delivery problem on Hypervolume and Inverted Generational Distance metric.

Student

Phan Duc Hung

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	**Definition**
CVRP	Capacitated vehicle routing problem
CX	Cycle crossover
DPDP	Dynamic pickup and delivery problem
DPDPTW	Dynamic pickup and delivery problem considering time windows
DVRP	Dynamic vehicle routing problem
EVRP	Electric vehicle routing problem
GA	Genetic algorithm
GVRP	Green vehicle routing problem
HV	Hypervolume
IGD	Inverted generational distance
ILP	Integer linear programming
LERK	Leader-based random keys encoding scheme
MODPDPTW	Multi-objective dynamic pickup and delivery problem with time windows considering two-sided fairness
MOEA	Multi-objective evolutionary algorithm
MOEAD	Multi-objective evolutionary algorithm based on decomposition
MOO	Multi-objective optimization
OX	Order crossover
PDP	Pickup and delivery problem
PDPTW	Pickup and delivery problem with time windows
PFG	Pareto front grid
PFG-MOEA	Pareto front grid guided multi-objective evolutionary algorithm
PFG-2F	Pareto front grid guided MOEA considering two-sided fairness
PMX	Partially mapped crossover
SOO	Single-objective optimization
SVRP	Stochastic vehicle routing problem

Abbreviation	**Definition**
VRP	Vehicle routing problem
VRPTW	Vehicle routing problem with time windows

INTRODUCTION

The Dynamic pickup and delivery problem (DPDP) represents a significant and intricate extension of the well-known vehicle routing problem, with a primary focus on optimizing the transportation of goods across networked locations. In this context, each vehicle is required to fulfill several transport requests, where each request consists of a pickup point from one customer and a delivery point to another.

In the static variant of DPDP, all transportation requests are available during the planning stage, enabling route optimization prior to vehicle dispatch with the aim of minimizing travel costs and time. By contrast, the dynamic version introduces additional complexity, as order information becomes available incrementally over time. At any given point, only partial knowledge of customer requests is accessible, compelling the system to make rapid decisions based on incomplete data. As a result, routing strategies must be highly adaptive and capable of responding efficiently to continuous environmental changes. DPDP has widespread practical relevance, particularly in modern logistics applications such as express delivery, e-commerce distribution, and supply chain operations. Of particular interest in recent research is the Dynamic pickup and delivery problem considering time windows (DPDPTW), where each transport request must be satisfied within specific time windows. These constraints require that cargo pickups and deliveries occur within predefined intervals, thereby intensifying the complexity of vehicle scheduling and route planning.

Although most existing studies on the DPDP have focused on static scenarios where all requests are known in advance, comparatively little attention has been devoted to the dynamic counterpart. Moreover, solving the DPDP often entails addressing multiple objectives simultaneously, such as minimizing energy consumption and ensuring fairness in resource allocation. Reducing energy usage is critical for enhancing both environmental sustainability and operational cost-efficiency. Concurrently, fairness in service delivery, which ensures that no customer is disproportionately delayed or disadvantaged, is essential for maintaining equity and customer satisfaction. The integration of dynamic demands with multi-objective optimization makes the DPDPTW problem particularly challenging to solve. Algorithms designed for this problem must provide near-instantaneous responses to real-time requests while also maintaining a balanced trade-off among competing objectives.

The main contributions of this thesis are summarized as follows:

- Introduce the MODPDPTW, a multi-objective model for solving the dynamic pickup and delivery problem. This model addresses traditional logistics challenges while integrating considerations related to energy efficiency, customer fairness, and vehicle workload fairness. It aims to deliver equitable service across customers and to distribute operational duties evenly among vehicles, thus promoting both efficiency and sustainability.
- Propose PFG-2F, a novel algorithm designed to efficiently navigate the multi-objective solution space. The algorithm combines Pareto front grid (PFG) with random-key-based encoding and fairness-oriented heuristics, allowing for a more effective balance between exploration and exploitation.
- Experiments are conducted across diverse scenarios with varying customer numbers and spatial distributions. Numerical result evaluations demonstrate that PFG-2F outperforms state-of-the-art multi-objective approaches in terms of both HV and IGD. Additionally, PFG-2F surpasses several single-objective vehicle routing algorithms that incorporate fairness measures, highlighting the superior performance of the proposed approach.

Thesis outline

The structure is organized as follows:

- **Chapter 1** introduces the foundational concepts, including optimization, multi-objective optimization, and VRP.
- **Chapter 2** presents DPDP, reviews related works and recent approaches, and introduces the proposed model—Multi-objective dynamic pickup and delivery problem with two-sided fairness.
- **Chapter 3** proposes a novel method Pareto front grid-guided multi-objective optimization with two-sided fairness.
- **Chapter 4** presents the experimental results and performance evaluation. The proposed method is compared against state-of-the-art multi-objective algorithms using metrics such as hypervolume and inverted generational distance.

CHAPTER 1. BACKGROUND

This chapter focuses on explaining the fundamental concepts of optimization, beginning with the definition of optimization and the structure of an optimization problem, including objectives and constraints. Multi-objective optimization is introduced, providing definitions and discussing common approaches used to solve such problems. Finally, this chapter examines VRP, exploring their various variants and highlighting approaches for handling dynamic and real-time scenarios.

1.1 Optimization

Optimization is the process of finding the best solution to a problem from a set of feasible solutions. Formally, an optimization problem can be expressed as finding a decision vector $\mathbf{x}$ that minimizes (or maximizes) an objective function $f(\mathbf{x})$ subject to a set of constraints:

$$\text{minimize} \quad f(\mathbf{x}) \text{ subject to} \qquad \mathbf{x} \in X \ , \tag{1.1}$$

where $\mathbf{x}$ is a vector of decision variables and X is the feasible set defined by the problem's constraints. The function $f(\mathbf{x})$ maps a decision $\mathbf{x}$ to a real value representing the solution's quality. A solution $\mathbf{x}^* \in X$ is optimal if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in X$ (for a minimization problem). Key concepts include the feasible region (the set X of all $\mathbf{x}$ satisfying constraints), global optimum (an optimal solution with respect to all of X), and local optimum (a solution that is optimal within a neighboring subset of X but not necessarily globally).

In general, optimization problems are types of mathematical problems with goals are to find the best solutions from a solution searching space, these solutions may follow some criteria given by the problem definition. A simple optimization problem will require minimize or maximize a function, represented for cost, profit, time, distance, by constructing solutions subject to certain constraints. The solutions then are used to compute the value of the function. A general structure of an optimization problem includes:

- **Objective function**: a mathematical expression that quantifies the goal of the optimization. It defines the criterion to be minimized or maximized, such as cost, profit, time, or error. The objective function is expressed in terms of the decision variables and serves as the primary metric for evaluating the quality of a solution.

- **Decision variable**: the set of controllable inputs or parameters whose values are to be determined through the optimization process. These variables directly influence the value of the objective function and are subject to the problem's constraints. The solution to the optimization problem is defined by the specific values assigned to these variables.
- **Constraints**: a set of mathematical conditions, usually in the form of equalities or inequalities that define the feasible region within which the decision variables must lie. These conditions represent the problem's limitations or requirements, such as resource capacities, physical laws, or operational restrictions. A feasible solution must satisfy all specified constraints.

1.2 Multi-objective optimization

Many problems involve multiple objectives that must be optimized simultaneously. MOO, also known as multi-criteria or vector optimization, deals with problems where there are two or more conflicting objectives. Instead of a single objective function, it is replaced by a vector of objective functions. A general multi-objective optimization problem can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \mathbf{f}(\mathbf{x}) = \big(f_1(\mathbf{x}),, f_2(\mathbf{x}),, \dots,, f_m(\mathbf{x})\big) \\ \text{subject to} \quad & \mathbf{x} \in X \ , \end{aligned} \tag{1.2}$$

where X is the feasible set and $f_i(\mathbf{x})$ represents the i-th objective to be minimized (maximization objectives can be converted to minimization by applying a negative sign). In multi-objective optimization (MOO), the objectives $f_1, \dots, f_m$ are often in conflict, meaning that no single solution is best for all objectives simultaneously.

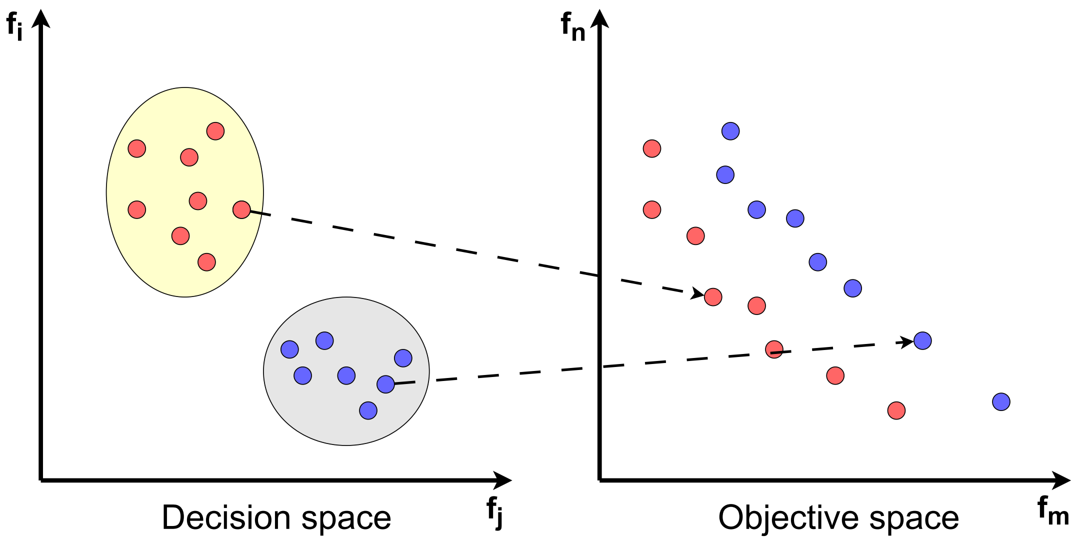


Figure 1.1: Multi-objective searching space.

1.2.1 Definitions

Due to trade-offs between conflicting objectives, the goal in multi-objective optimization is typically not to find a single optimal solution, but rather a set of *Pareto optimal* solutions. The concept of Pareto optimality is based on the notion of *dominance* between solutions.

A solution $\mathbf{x}^a \in X$ is said to *dominate* another solution $\mathbf{x}^b \in X$ (denoted $\mathbf{x}^a \prec \mathbf{x}^b$) if the following two conditions are satisfied:

$$f_i(\mathbf{x}^a) \leq f_i(\mathbf{x}^b), \quad \forall i = 1, \dots, m, \; f_j(\mathbf{x}^a) \; < f_j(\mathbf{x}^b), \quad \text{for at least one } j \in 1, \dots, m. \tag{1.3}$$

That is, $\mathbf{x}^a$ is no worse than $\mathbf{x}^b$ in all objectives and strictly better in at least one.

A solution $\mathbf{x}^* \in X$ is said to be *Pareto optimal* if there does not exist any $\mathbf{x} \in X$ such that $\mathbf{x} \prec \mathbf{x}^*$. Formally:

$$\nexists, \mathbf{x} \in X \text{ such that } \mathbf{x} \prec \mathbf{x}^{\cdot} \tag{1.4}$$

The set of all Pareto optimal solutions in X is called the *Pareto set*, and the corresponding set of objective vectors in the objective space is known as the *Pareto front* [1], as illustrated in Figure 1.2.

By definition, moving from one Pareto optimal solution to another on the Pareto front will improve some objectives but worsen at least one other objective. Decision makers often must choose from the Pareto front according to preference trade-offs.

Reference points in multi-objective optimization are important for measuring the effective of the multi-objective algorithms by evaluating the set of solution produced. Reference points includes:

- **Utopia Point** (also known as the ideal point) is a hypothetical point in objective space defined by $z^* = (z_1^*, \dots, z_m^*)$, where $z_i^* = \min_{x \in X} f_i(x)$ is the best achievable value for each objective individually. The utopia point is generally not attainable unless one solution simultaneously optimizes all objectives.
- **Nadir Point** $z^{nad} = (z_1^{nad}, \dots, z_m^{nad})$, is defined by $z_i^{nad} = \max_{x \in X_P} f_i(x)$, where X_P is the Pareto set. In other words, the nadir point components are the worst objective values among all Pareto-optimal solutions. The nadir point, when all objectives are to be minimized, gives an idea of the extent of the Pareto front.

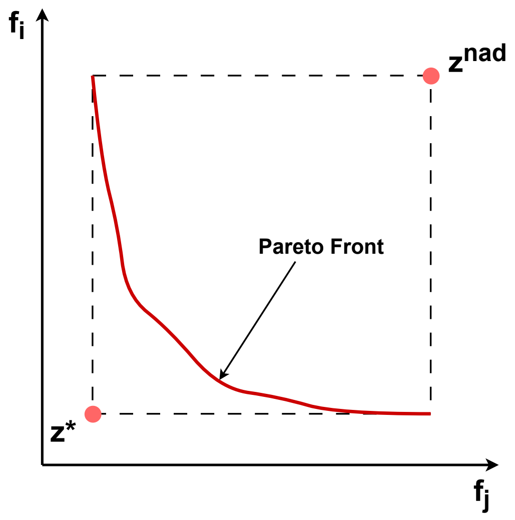


Figure 1.2: Pareto front and reference points.

- **Knee point** is a Pareto-optimal solution where a small improvement in one objective would lead to a large deterioration in at least one other objective. Knee points are often considered good compromise solutions because they exhibit a balanced trade-off among objectives.

1.2.2 Approaches

Solving a multi-objective problem means finding or approximating the Pareto front. If the problem is small and tractable (for example, a multi-objective linear or integer linear program), Pareto optimal solutions can be computed exactly by extending single-objective optimization techniques. For instance, a multi-objective ILP (integer linear program with multiple objective functions) can be tackled by algorithms that systematically explore trade-offs (e.g., using parametric linear programming or variants of branch-and-bound that handle multiple objectives) [2]. However, a common approach is to employ scalarization, converting the multi-objective problem into a series of single-objective problems whose solutions are Pareto optimal for the original problem [1], [2]. This thesis presents an overview of classical scalarization techniques and their corresponding strategies as follows:

- **Weighted Sum**: A weighted sum approach assigns a weight $w_i \geq 0$ to each objective and optimizes a linear combination of the objectives:

$$\min_{\mathbf{x} \in X} \sum_{i=1}^{m} w_i f_i(\mathbf{x}) \tag{1.5}$$

where $\sum w_i = 1$, $w_i \geq 0$. Weights reflect the relative importance of objectives [1]. In practice, solving the weighted sum formulation for different weight sets (e.g., using an ILP solver when f_i and X are linear) yields points along the Pareto front. This method is easy to implement but has limitations: if the true Pareto front is non-convex in objective space, the weighted sum method cannot obtain those Pareto-optimal solutions that lie on the non-convex parts of the front [2]. Additionally, choosing appropriate weights can be difficult without prior knowledge of the trade-offs.

- **Weighted Tchebycheff**: The weighted Tchebycheff approach is a more general scalarization that can obtain any Pareto optimal solution (given certain conditions) and is especially useful for finding extreme points on a non-convex Pareto front [1]. It uses a reference point, often the utopia point z^*, and minimizes the weighted maximum deviation from this reference point. A common formulation is:

$$\min_{\mathbf{x}\in X} \max_{1\leq i\leq m} \{w_i|f_i(\mathbf{x}) - z_i^*|\} \tag{1.6}$$

  where $w_i > 0$ are scaling weights. In words, this tries to minimize the worst-case (relative) objective value among all objectives. By adjusting the weights w_i, different Pareto solutions can be emphasized. The weighted Tchebycheff method can capture Pareto-optimal solutions that the simple weighted sum method might miss, and it provides a uniform spread of solutions under certain conditions. An augmented version of this method adds a small term $\epsilon \sum_i w_i f_i(\mathbf{x})$ to ensure the obtained solution is truly Pareto optimal (to avoid solutions that only minimize the maximum deviation but are dominated) [1].

- **Decomposition-based Methods**: These strategies involve decomposing the multi-objective problem into a set of simpler subproblems, each focusing on a particular region or trade-off of the Pareto front. For example, one can generate a set of N evenly distributed weight vectors $w^1, \ldots, w^N$ in the weight space and solve N single-objective problems (such as weighted sum or Tchebycheff scalarizations for each w^j). This produces a set of N Pareto-optimal solutions approximating the front. Decomposition strategies are at the core of certain evolutionary algorithms (discussed next) where multiple scalarized subproblems are tackled in parallel. The advantage of decomposition is that it can provide a well-spread approximation of the Pareto front by covering different preference regions or weight combinations. Benson's method [2] are examples of mathematical programming approaches that systematically generate Pareto fronts by solving a sequence of subproblems. In summary, scalarization and decomposition techniques convert the multi-objective task into a series of single-objective optimizations, enabling the use of classical solvers or guided search to find Pareto-optimal solutions.

While classical scalarization methods can find Pareto-optimal solutions, they often require solving many subproblems and may struggle with complex, non-linear, or combinatorial objectives. In practice, particularly for difficult problems like vehicle routing, evolutionary algorithms and other metaheuristics are popular for approximating Pareto fronts in a single run [3]. These algorithms evolve a population of candidate solutions and are naturally suited to exploring multiple trade-offs simultaneously. This thesis highlight some common algorithms and frameworks for multi-objective optimization:

NSGA Srinivas and Deb [4] introduced Non-dominated Sorting Genetic Algorithm (NSGA), one of the first genetic algorithms designed for multi-objective optimization. NSGA uses the concept of *non-dominated sorting*: the population of candidate solutions is ranked into tiers (“Pareto fronts”) based on dominance. All nondominated individuals are assigned to the first front (Pareto rank 1), then those dominated only by solutions in the first front form the second front (rank 2), and so forth. The GA then applies selection, crossover, and mutation biased toward lower-rank (better) solutions, along with a niche-sharing technique to maintain diversity along the Pareto front [4]. NSGA demonstrated the effectiveness of evolutionary approaches to obtain a spread of Pareto-optimal solutions in one run. However, the original NSGA had some drawbacks: it did not explicitly preserve elite solutions (the best found so far) and it required a sharing parameter to maintain diversity, which can be hard to set.

NSGA-II Deb et al. [5] proposed NSGA-II, which has become one of the most widely used multi-objective evolutionary algorithms. NSGA-II introduced several improvements over NSGA:

- **Elitism:** NSGA-II ensures that Pareto-optimal solutions found are carried over to the next generation. In each iteration, parent and offspring populations are merged and the best N individuals (for population size N) are selected for the next generation. This retains any superior solutions discovered.
- **Fast Non-dominated Sorting:** NSGA-II uses an efficient sorting algorithm to rank solutions into Pareto fronts with complexity $O(N \log N)$ per generation, improving on the $O(N^2)$ approach of NSGA.
- **Diversity Preservation:** Instead of the niche sharing parameter, NSGA-II uses the *crowding distance* metric. After ranking by Pareto dominance, solutions in

the last front that fits in the next generation are sorted by crowding distance (a measure of how isolated a solution is in objective space). Solutions with larger crowding distance (more isolated, contributing to diversity) are preferred. This helps maintain a well-spread Pareto front.

MOEA/D The Multi-objective evolutionary algorithm based on decomposition (MOEA/D), proposed by Zhang and Li [6], introduces a decomposition-based strategy for solving multi-objective optimization problems (MOPs). Unlike Pareto-dominance-based approaches, MOEA/D decomposes a MOP into a set of scalar optimization subproblems using aggregation functions, such as the weighted sum or the Tchebycheff approach.

Let $\mathcal{X}$ denote the decision space, and $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ represent the vector of m objective functions. MOEA/D defines N weight vectors $\{\boldsymbol{w}^1, \boldsymbol{w}^2, \ldots, \boldsymbol{w}^N\}$, where each $\boldsymbol{w}^j = (w_1^j, \ldots, w_m^j)$ satisfies $w_i^j \geq 0$ and $\sum_{i=1}^m w_i^j = 1$. Each weight vector corresponds to a scalar subproblem:

$$\min_{\mathbf{x} \in \mathcal{X}} \quad g^j(\mathbf{x} \mid \boldsymbol{w}^j, \boldsymbol{z}^*) = \max_{1 \leq i \leq m} \left\{ w_i^j \cdot |f_i(\mathbf{x}) - z_i^*| \right\}, \tag{1.7}$$

where $\boldsymbol{z}^* = (z_1^*, \ldots, z_m^*)$ is the reference point, typically the ideal point in the objective space.

MOEA/D maintains a population $\mathcal{P} = \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N$, where each solution $\mathbf{x}^j$ is associated with the weight vector $\boldsymbol{w}^j$. The algorithm proceeds as follows:

- **Neighborhood Definition:** For each subproblem j, define a neighborhood $\mathcal{N}(j)$ consisting of the indices of the T closest weight vectors to $\boldsymbol{w}^j$ based on Euclidean distance.
- **Variation:** For each subproblem j, select parent solutions from $\mathcal{N}(j)$ and apply genetic operators (e.g., crossover and mutation) to generate an offspring $\mathbf{y}$.
- **Update:** Evaluate $\mathbf{y}$ and update the solutions associated with the subproblems in $\mathcal{N}(j)$. Specifically, for each $k \in \mathcal{N}(j)$, if

$$g^k(\mathbf{y} \mid \boldsymbol{w}^k, \boldsymbol{z}^*) < g^k(\mathbf{x}^k \mid \boldsymbol{w}^k, \boldsymbol{z}^*),$$

  then set $\mathbf{x}^k = \mathbf{y}$.
- **Reference Point Update:** Update the reference point $\boldsymbol{z}^*$ if necessary, by setting

$$z_i^* = \min\{z_i^*, f_i(\mathbf{y})\}, \quad \text{for all } i = 1, \ldots, m.$$

- **External Population Maintenance:** Maintain an external population (EP) to store all non-dominated solutions found during the search process. When a new solution $\mathbf{y}$ is generated, it is compared against the current EP. If $\mathbf{y}$ is not dominated by any member of the EP, it is added to the EP, and any solutions in the EP dominated by $\mathbf{y}$ are removed.

The external population serves as an archive of elite solutions, providing a diverse approximation of the Pareto front. It is particularly beneficial for preserving diversity and guiding the search process, especially in many-objective optimization scenarios.

MOEA/D's decomposition strategy allows for parallel optimization of subproblems, facilitating scalability to problems with a large number of objectives. Its neighborhood-based update mechanism promotes information sharing among similar subproblems, enhancing convergence and diversity. Since its inception, various extensions and improvements have been proposed, incorporating adaptive weight adjustment, hybridization with other metaheuristics, and advanced archiving strategies.

1.2.3 Metrics

To evaluate the effectiveness of Pareto fronts generated by multi-objective optimization algorithms, two widely accepted performance metrics are employed: the HV and the IGD. These metrics serve as quantitative indicators for assessing both the convergence of the obtained solutions toward the true Pareto front and their distribution across the objective space—two key aspects of multi-objective optimization quality.

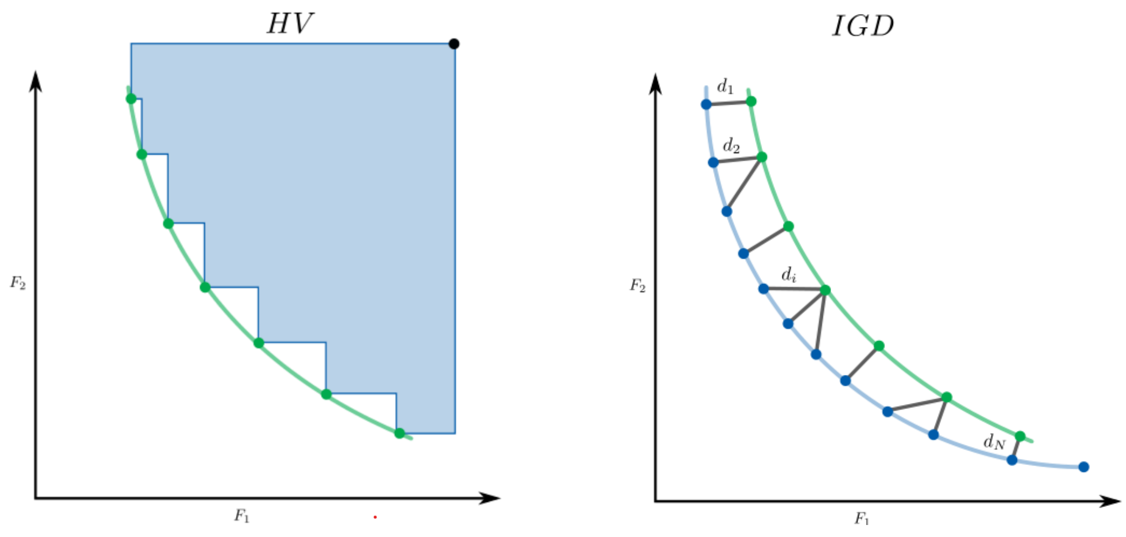
The HV metric measures the volume of the objective space that is weakly dominated by the non-dominated solutions in the obtained Pareto front, bounded from above by a reference point. It reflects both convergence (how close the solutions are to the true Pareto front) and diversity (how well they spread across the objective space). A higher HV value indicates a better-quality solution set, as it suggests that the algorithm has not only reached a region close to the optimal front but also captured a diverse range of trade-offs among the objectives. The formal expression of HV is shown in Equation 1.8, and an illustrative example is provided in Figure 1.3a [7].

$$HV = \int_{f_{\text{min}}}^{f_{z^n}} \prod_{i=1}^{m} (f_i^{z^n} - f_i)\, df, \tag{1.8}$$

In contrast, the IGD metric evaluates the average distance from a set of reference Pareto-optimal solutions to the closest solutions in the obtained front. It is used primarily to measure convergence, although it also indirectly captures distribution quality by penalizing regions of the true front that are poorly approximated. A lower IGD value indicates that the obtained solutions are closer to the reference front and thus reflect a higher-quality approximation. The formula for computing IGD is given in Equation 1.9, and an example visualization is presented in Figure 1.3b [7].

$$IGD = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \min_{p \in \mathcal{P}} \|q - p\|_2, \tag{1.9}$$

In this equation, $\mathcal{Q}$ denotes the set of reference Pareto-optimal solutions, $\mathcal{P}$ is the set of solutions obtained by the algorithm, and $\|\cdot\|_2$ represents the Euclidean distance. Together, HV and IGD provide a comprehensive framework for evaluating the quality of multi-objective optimization results, supporting both theoretical comparisons and practical performance analysis.



(a) Hypervolume **(b)** Inverted Generational Distance

Figure 1.3: MOO metrics.

1.3 Vehicle routing problem

VRP, first introduced by Dantzig and Ramser [8], has become a cornerstone of logistics optimization. Over the decades, numerous variants of the VRP have been developed to address the wide variety of real-world constraints and requirements that arise in distribution and transportation [9]. These problem variants extend the classical VRP formulation in order to model practical considerations such as vehicle capacity limits, customer service time windows, uncertainty in demands, and environmental sustainability goals. This thesis highlights several important modern

VRP variants below, focusing on their formulation and the real-world motivations behind them.

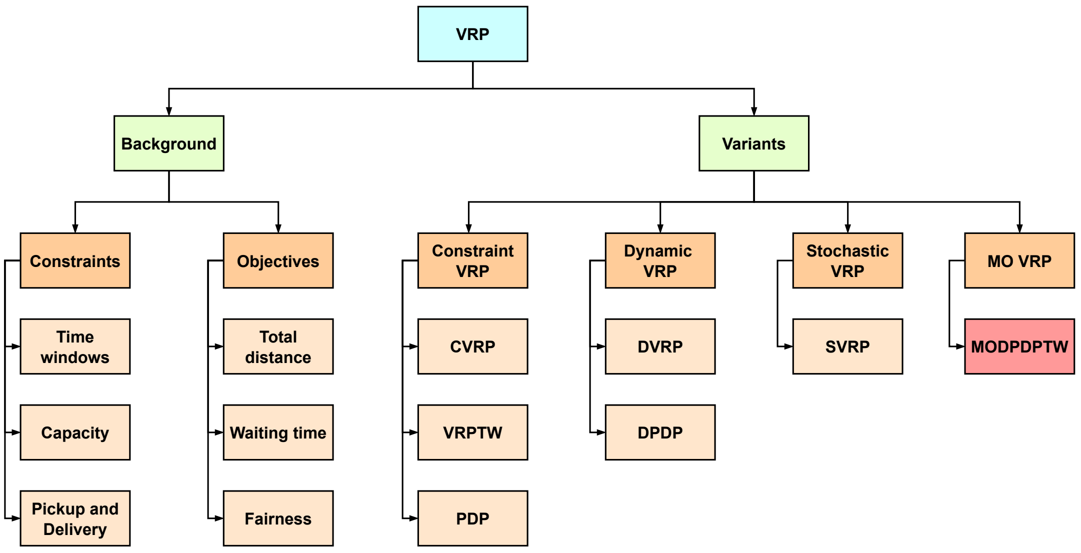


Figure 1.4: Survey on VRP.

1.3.1 Constraint vehicle routing problem

The Classical vehicle routing problem (VRP) is defined over a set of customers, each with a specific location and demand, and a fleet of vehicles based at a central depot. The goal is to design a set of routes—one for each vehicle—such that all customers are served exactly once, the total routing cost (e.g., distance or time) is minimized, and operational constraints are respected.

A key variant of this problem is CVRP, where each vehicle has a fixed carrying capacity. This reflects real-world limitations, such as delivery trucks being able to carry only a finite amount of goods, as referred in Figure 1.5. A feasible solution must ensure that the total demand on any route does not exceed the vehicle's capacity. This fundamental constraint, introduced in the original formulation by Dantzig and Ramser [8], remains central to modern routing problems.

CVRP captures the trade-off between route length and fleet size: while using fewer vehicles can reduce fixed costs, capacity limits prevent overloading, necessitating more or longer routes. This makes CVRP highly applicable to practical distribution tasks—for example, fuel delivery where tanker volume is fixed. It also forms the foundation for numerous advanced VRP variants, which build on the capacity constraint by introducing additional complexities such as time windows, stochastic demands, or environmental objectives.

One of the most prevalent constraints in VRP extensions is the inclusion of service **time windows**, leading to the VRPTW. In this variant, each customer is

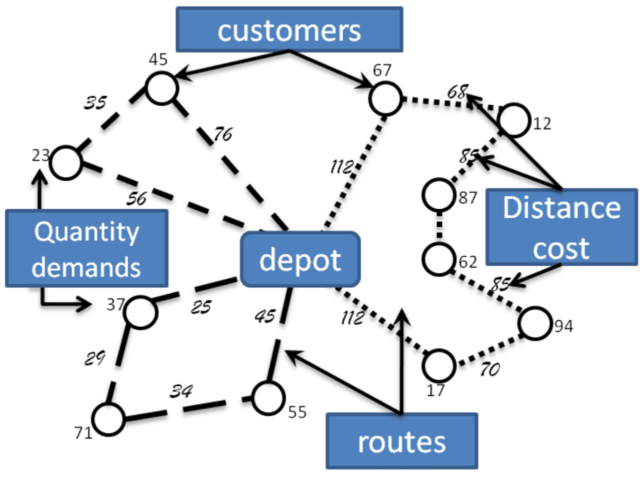


Figure 1.5: Example of CVRP [10].

associated with a permissible time interval—defined by an earliest and latest allowable arrival time—during which service must begin, as referred in Figure 1.6. The vehicle may arrive early and wait but cannot arrive late. This constraint reflects the practical necessity of synchronizing deliveries with customer availability or contractual service levels, such as in parcel delivery, home grocery delivery, and field service scheduling. Additionally, service durations at each customer must be accounted for. Solomon [11] introduced benchmark instances for VRPTW that significantly influenced research in this domain. These constraints make the solution space more restricted and complex, emphasizing timely service for scenarios involving perishable goods or restricted delivery hours.

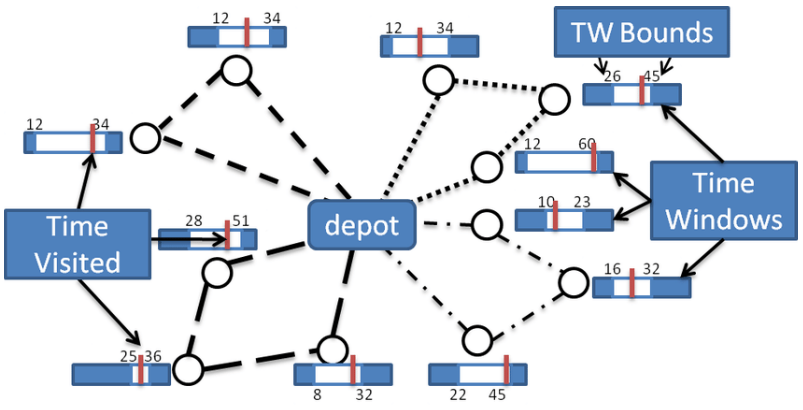


Figure 1.6: Example of VRPTW [10]

Another class of constraints is introduced in the Pickup and delivery problem (PDP). Here, each request consists of a pair of locations: a pickup point and a corresponding delivery point. The vehicle must serve both locations on the same route, with the pickup preceding the delivery. These **pairing and precedence constraints**, along with standard capacity and potentially time window constraints, create intricate routing requirements. PDP models practical scenarios such as courier services, ride-sharing systems, and less-than-truckload operations. Savelsbergh and Sol [12] formalized this problem, setting a foundation for many further variants including the dial-a-ride problem.

1.3.2 Stochastic vehicle routing problem

In the classical deterministic formulation of the VRP, all relevant problem parameters are assumed to be known in advance and remain fixed throughout the planning horizon. This includes customer demands, service times, travel times between locations, and the set of customers to be served. Based on this complete information, a solution is generated that specifies the optimal set of vehicle routes to minimize a given cost function—typically distance, time, or number of vehicles—while satisfying operational constraints such as capacity and service coverage. Deterministic VRP models are computationally tractable and widely studied, serving as the foundation for many algorithmic developments and industrial applications.

However, in real-world logistics and transportation systems, such certainty is rarely attainable. Operational environments are often dynamic and unpredictable: customer demands may fluctuate, traffic conditions may vary unexpectedly, new requests may appear, and some customers might cancel or reschedule. To address these challenges, the Stochastic vehicle routing problem (SVRP) [13], [14] introduces randomness into one or more elements of the problem formulation. In this setting, parameters such as customer demand, travel time, service duration, or customer presence are modeled as random variables, typically governed by known or estimated probability distributions.

The stochastic formulation more accurately captures the nature of real-world routing tasks, where planners must make decisions based on incomplete or probabilistic information. For example, a delivery company may need to dispatch vehicles before knowing the exact quantities that customers will request, or an emergency service provider must plan routes without knowing the exact locations and times of future calls. In these cases, relying solely on a deterministic plan can result in inefficiencies or failures to meet service requirements.

To cope with this uncertainty, SVRP models often incorporate **recourse strategies**—predefined rules or adaptive mechanisms that allow the solution to adjust in response to actual events during execution. A common example is allowing a vehicle to return to the depot for reloading if encountered demands exceed its remaining capacity. Other strategies include reassigning tasks, rerouting dynamically, or holding buffer capacity in anticipation of stochastic events. These mechanisms aim to enhance the robustness and flexibility of the solution, ensuring acceptable performance across a range of possible realizations of the uncertain parameters.

As highlighted by Oyola et al. [13], research into the SVRP has expanded significantly, resulting in a diverse array of models that differ based on the source and type of uncertainty, the assumptions about available information, and the complexity of the recourse actions allowed. Their survey classifies SVRP variants into categories such as demand uncertainty, travel time uncertainty, and customer presence uncertainty, and further distinguishes between offline (preplanned) and online (real-time adaptive) solution approaches. This growing body of work reflects the increasing need for vehicle routing systems that are not only cost-efficient but also reliable and adaptable in dynamic, uncertain environments.

As logistics operations become more reliant on real-time data, IoT integration, and responsive service delivery, the role of stochastic models in vehicle routing continues to grow. SVRP thus represents a critical evolution of the classical VRP, bridging the gap between theoretical optimization and operational resilience in uncertain and time-sensitive logistics contexts.

1.3.3 Dynamic vehicle routing problem

Many transportation optimization problems, including VRP, have traditionally been studied in a static context where all input data (e.g., customer demands, travel times) are known in advance. However, real-world operations often face uncertainty and change: new customer requests may arrive during the day, traffic conditions can vary, or orders may be cancelled. This gives rise to the Dynamic vehicle routing problem (DVRP) [15], a class of problems where information is revealed or updated in real-time and routes must be adjusted on-the-fly. In a DVRP, unlike the static VRP, decisions (such as which customer to serve next or which route a vehicle will follow) are made not just once at the start, but continuously as new information arrives over time. Formally, a dynamic routing problem can be seen as a combinatorial optimization problem on a timeline. Initially, a partial set of requests or tasks is known and must be scheduled into vehicle routes. Over time, new requests appear (or other changes occur), requiring the current solution to be updated. The objective is typically to optimize some metric of service (like total travel distance, operational cost, or response time) while feasibly incorporating all requests announced up to that time. Many dynamic problems also impose real-time constraints; for example, decisions may need to be recomputed within seconds each time an update occurs to be practically useful in dispatching vehicles. One important variant is DPDP [16], which is a dynamic version of the pickup-and-delivery problem. In DPDP, transportation requests consist of a pickup location and a delivery location (e.g., ride-sharing or courier requests) that can arrive dynamically during the routing period. Each new request must be assigned to a vehicle and sequenced into that vehicle's route, respecting any constraints such as capacity or time windows. The dynamic aspect greatly increases the complexity: the algorithm must decide how to insert the pickup and delivery of each new request into an existing set of routes without overly disrupting prior commitments.

Dynamic routing problems are inherently challenging because they combine the complexity of NP-hard routing optimization with the uncertainty of online decision-making. Nonetheless, a variety of strategies have been developed to handle DVRPs effectively:

- **Rolling Horizon / Periodic Reoptimization**: In this framework, time is divided into planning intervals. At each interval (say every few minutes), the system takes all information available up to that point (including newly arrived requests) and solves a VRP (or Pickup and delivery problem with time windows (PDPTW)) for those requests, treating any future requests as unknown. The solution provides updated routes for the vehicles. When the next interval arrives or a significant event happens, the process is repeated. This approach repeatedly solves static problems as new data comes in, effectively rolling the planning horizon forward [15]. It provides a structured way to balance solution quality and computational effort, as reoptimization can be scheduled at a fixed frequency. However, choosing the right reoptimization frequency is important: too frequent and the system may thrash with little improvement; too infrequent and it may react too slowly to new information.
- **Event-Driven Reoptimization**: Instead of a fixed schedule, some approaches trigger reoptimization immediately upon certain events, such as the arrival of a new request or a vehicle completing a service [15]. This is a purely reactive strategy: the moment something changes (e.g., a new customer calls in an order), the algorithm updates the vehicle routes to accommodate the change. This can minimize the delay in integrating new requests, potentially improving responsiveness. The challenge is ensuring the computation finishes fast enough to implement decisions in real time, especially if events happen in quick succession.
- **Heuristic Insertion Strategies**: A widely used class of methods for DVRP and DPDP is to insert new requests directly into existing routes using heuristics, rather than reoptimizing from scratch each time. For example, when a new

pickup-delivery request arrives, one can evaluate the cost of inserting that request at various feasible positions in each vehicle's route and choose the least-cost insertion [16]. This is akin to the insertion heuristics used in static route construction (like the savings algorithm or cheapest insertion), but applied online. In dynamic dial-a-ride problems, insertion heuristics with look-ahead can be used, where the algorithm considers not just the immediate cost but also future implications (sometimes by reserving slack for possible future requests). Insertion methods are typically fast and can handle high volumes of requests, though they may yield myopic decisions compared to a full reoptimization.

- **Metaheuristic Adaptation**: Many researchers have adapted metaheuristic algorithms to the dynamic context. For instance, Gendreau et al. [17] implemented a Tabu Search for a real-time VRP: their approach continuously improves the current set of routes and, when a new request arrives, they integrate it into the solution and then continue the Tabu Search from that new state. The search is designed to be fast and anytime, so that it can be interrupted to dispatch vehicles as needed. Similarly, genetic algorithms and ant colony optimization have been modified for dynamic VRPs by running a continuous optimization loop that refines routes and quickly adapts to changes. Montemanni et al. [18], for example, developed an Ant Colony System that repeatedly constructs vehicle routes and can add new requests as pheromone information is updated, making the algorithm "responsive" to real-time events. These metaheuristic approaches often maintain a population or a memory of solutions that they tweak as the problem data evolves, rather than restarting from scratch on each update.

- **Assignment and Dispatching Rules**: In some real-time systems, especially when decisions need to be extremely fast, simpler rule-based strategies are used. For instance, when a new request comes in, assign it to the nearest available vehicle (greedy choice) or a vehicle that will become free soonest. While such myopic policies might not yield globally minimal travel cost, they are easy to compute and can be effective when combined with periodic plan improvements. More sophisticated dispatching rules might take into account current route load, detour from current route, or balancing workload among vehicles. These can be seen as a form of rolling decision-making without heavy computation.

Managing dynamic routing in real time often requires careful handling of operational constraints. For example, when reoptimizing, one must consider that vehicles already en route to serve a customer cannot instantly change course to a totally different plan (especially if they have passengers on board in DPDP). Thus, many algorithms enforce route continuity or solution stability constraints, meaning changes to the plan should be feasible and not overly disruptive. One common practice is to “freeze” a portion of each vehicle’s route that is imminent (e.g., the next immediate service stops) and only allow changes to the part of the route beyond that [15]. This ensures that decisions that are in execution are not reversed by the optimizer, providing consistency in operations. Over the past decades, a number of academic works have provided frameworks and surveys for dynamic routing problems. Pillac et al. [15] present a comprehensive review of DVRP classifications and solution methods, highlighting the importance of information quality and timing. They categorize approaches by how and when they adapt to new information (periodic vs event-driven, myopic vs anticipatory). Similarly, Berbeglia et al. [16] survey dynamic pickup-and-delivery problems, discussing various models (e.g., allowing advance request announcements versus purely immediate requests) and algorithms applied in the literature. These works suggest that while no one method is universally best, successful dynamic routing systems often share traits: they are fast, flexible, and can handle incremental changes without completely rebuilding the solution from scratch. In summary, the dynamic routing problem extends traditional optimization into a time-sensitive, continually evolving setting. Whether it’s routing delivery trucks in a city with last-minute orders or dispatching ride-share drivers to ad-hoc passenger requests, dynamic VRPs require algorithms that balance solution quality with computational speed. Techniques range from fast heuristics for immediate decisions to sophisticated metaheuristics that continually improve routes as new data arrives. This thesis builds upon the concepts of multi-objective optimization and dynamic routing, aiming to address scenarios where multiple criteria (e.g., cost, service level) must be optimized in a real-time vehicle routing context.

1.3.4 Multi-objective vehicle routing problem

As concerns about climate change and environmental degradation continue to grow, reducing energy consumption and minimizing carbon emissions have become central objectives in modern logistics and transportation planning. In the context of vehicle routing, this shift has led to the development of environmentally focused problem variants that explicitly aim to lower the ecological impact of transportation operations. One key development is Green vehicle routing problem (GVRP), which incorporates vehicles powered by alternative fuels—such as electric or hybrid vehicles—into the routing model. These energy-constrained vehicles often have limited driving range and must plan visits to refueling or recharging stations. Erdogan and Miller-Hooks [19] formalized the GVRP to account for these constraints, while also considering the strategic placement of alternative energy infrastructure. Further developments such as Electric vehicle routing problem (EVRP) extend this framework by incorporating battery capacity limits, charging durations, and load-dependent energy consumption. The objective functions in these models often go beyond traditional cost or distance minimization to include metrics such as total energy usage or greenhouse gas emissions. These enhancements reflect a growing recognition that sustainable routing solutions must balance economic efficiency with environmental responsibility, especially as logistics providers adopt low-emission vehicle fleets to comply with urban regulations and corporate sustainability goals.

Beyond environmental considerations, sustainability in vehicle routing also involves ensuring fairness among the system's human stakeholders—both service providers and customers. Fairness is increasingly viewed as a vital component for the long-term viability of shared mobility and delivery platforms. Unfair treatment, whether it manifests as an uneven distribution of workload among drivers or excessively long wait times for certain customers, can result in dissatisfaction and attrition. To address this, recent research [20]–[26] has proposed various optimization models that integrate fairness metrics into the route planning process. These approaches aim to avoid scenarios where certain drivers are overburdened while others are underutilized, thereby promoting workforce retention and operational stability. Building on this foundation, Kang [27] introduced a two-sided fairness model for dynamic vehicle routing in ride-sharing systems. This framework simultaneously considers customer fairness—by minimizing average waiting times—and provider fairness—by balancing task assignments among drivers. By embedding fairness directly into the objective function, such models contribute to a more equitable and resilient mobility system, reinforcing the broader goals of sustainable urban logistics.

CHAPTER 2. MULTI-OBJECTIVE DYNAMIC PICKUP AND DELIVERY PROBLEM CONSIDERING TWO-SIDED FAIRNESS

This chapter presents the development of a model called the Multi-objective dynamic pickup and delivery problem with time windows considering two-sided fairness (MODPDPTW), which extends the classical Dynamic pickup and delivery problem (DPDP). Recent approaches to solving the DPDP are reviewed to provide context and highlight current research trends. Building upon these foundations, an Integer linear programming (ILP) formulation for the proposed model is introduced to capture the multi-objective nature and fairness considerations of the problem.

2.1 Dynamic pickup and delivery problem

This section review recent studies related to the DPDP. These studies are classified into two primary categories: exact methods and approximate methods [28]. Exact algorithms guarantee optimal solutions to the problem; however, their applicability to dynamic problems is often limited due to their high computational complexity. In contrast, approximate algorithms, while not ensuring optimality, can yield high-quality solutions within a reasonable computational time. Therefore, this work further categorize approximate methods into two main approaches: learning-based techniques and metaheuristic algorithms.

Learning-based algorithms for DPDP integrate machine learning techniques with optimization methods to allow systems to learn decision making strategies in dynamic environments, where customer demands and operational conditions evolve over time. In the research [29], the authors introduce the dynamic delivery problem with time constraints, specifically the bi-objective time-dependent dynamic pick up and delivery problem (TDPDPLP). This problem has two objectives: minimizing administrative costs and minimizing total late penalties. The study proposes a comprehensive multi-objective reinforcement learning method to solve the bi-objective TDPDPLP, reducing GPU usage during both training and deployment while producing models that can perform efficiently on various problem instances. In the study [30], the authors focus on minimizing the cost for DPDP. They introduce a hierarchical optimization framework to address large-scale DPDPs more effectively. An upper-level agent is designed to dynamically divide the DPDP into sub-problems of varying sizes, optimizing vehicle routes to achieve globally better solutions. A lower-level agent is also developed to solve each sub-problem efficiently by combining the advantages of classical operational research methods with reinforcement learning-based strategies. Li et al. [31] explore the DPDP with constraints on time, last-in-first-out (LIFO), and back-to-depot, to minimize transportation costs. They introduce a novel graph-based relational learning method for large-scale industry applications, where vehicle relationships in a dynamic dispatching environment are represented through attention-driven graph convolution. Additionally, they propose a constraint embedding technique to futher enhance the efficiency of inference time.

Metaheuristic algorithms have been widely applied to solve various combinatorial optimization problems. The main advantage of these methods is their ability to provide high-quality solutions in an acceptable time. The DPDP has made considerable progress, especially with the incorporation of autonomous delivery robots (ADRs) and electric vehicles to improve logistics efficiency and sustainability. Recent research, such as that by [32], has focused on DPDPs involving ADRs. Their study considers energy consumption and recharging strategies to address the unpredictable nature of order arrivals and peak demand periods. They introduce an e-assignment algorithm that reassigns previously scheduled orders when a new order arrives. In supply chain management, Zhou et. al [33] proposed a memetic algorithm combining genetic algorithms and local search strategies to solve real-world DPDP. Their extensive experiments demonstrated the effectiveness of this approach in minimizing delivery costs and handling periodic order releases. The authors in [34] introduce a decomposition-based multiobjective evolutionary algorithm to address the complexity of DPDPs with constraints like time windows and vehicle capacity. This approach is a hybrid of multiobjective optimization and tabu search to enhance solution diversity and avoid local optima.

The study [35] introduces a variant of the DPDP that incorporates electric vehicles and time windows, addressing issues like task allocation, routing, and queue scheduling at service sites. A mixed integer linear programming (MILP) model is developed to minimize travel distance and queue time. Additionally, the authors propose an adaptive hybrid neighborhood search algorithm to solve large-scale instances. The experimental results demonstrate the algorithm's effectiveness in finding competitive solutions. In [27], the authors focus on improve the fairness between service provider and customer in DVRP. However, they have main limitations as not considering multi-objective optimization to optimize effectively all the objectives, the encoding scheme and crossover, mutation is still simple, not focus on constrained problem.

In vehicle routing systems involving both providers and customers, fairness is a critical factor for ensuring system sustainability. A lack of fairness can lead to the departure of both providers and customers, thereby undermining the longevity of the system. To establish a balanced system that incorporates fairness, recent studies

[20]–[26] have developed methods aimed at optimizing utility while enhancing fairness among service providers. However, these studies have typically addressed fairness from a single perspective, focusing exclusively on either providers or customers. Recent research [27] has developed a system that considers two-sided fairness for both customers and providers by employing a Genetic algorithm (GA) specifically designed to optimize all relevant objectives. Despite this advancement, multi-objective approaches that leverage techniques such as Multi-objective evolutionary algorithm (MOEA) to enhance the search for optimal solutions have not been thoroughly explored. These approaches hold the potential to simultaneously optimize multiple conflicting objectives, thereby providing a more comprehensive solution that balances fairness and utility more effectively.

To address these limitations, the thesis introduce the MODPDPTW, wherein a vehicle system serves customer requests that arrive randomly and unpredictably. This problem encompasses multiple objectives, including minimizing consumption costs, improving service quality, and balancing demand among customers and vehicles, making it a nonconvex and NP-hard challenge. In this context, the proposed method employs a multi-objective framework that integrates fairness considerations for both providers and customers within the vehicle routing system. By utilizing advanced multi-objective optimization techniques, this work enhances the search process for optimal solutions, ensuring a more equitable distribution of resources and services. This approach overcomes the limitations of previous single-objective and two-sided fairness methods, providing a more robust and balanced solution for sustainable vehicle routing systems.

2.2 Formulation of multi-objective dynamic pickup and delivery problem considering two-sided fairness

This section formulates the MODPDPTW problem to schedule a fleet of vehicles to service customer demands in a dynamic environment where demands arrive randomly during the system's operation. The goal is to enhance the provider's revenue while ensuring fairness by balancing the workload among vehicles and waiting times among customers.

The real-world map, where a typical MODPDPTW operates, is represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V}$ denotes the set of all nodes (locations), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of directed edges (roads connecting different locations). A directed edge from position i to j, denoted as $(i, j) \in \mathcal{E}$, exists if there is a path enabling travel from i to j. Then, d_{ij} is the distance of the path that the vehicle travels from i to j. Let V be a set of K homogeneous vehicles, each with capacity

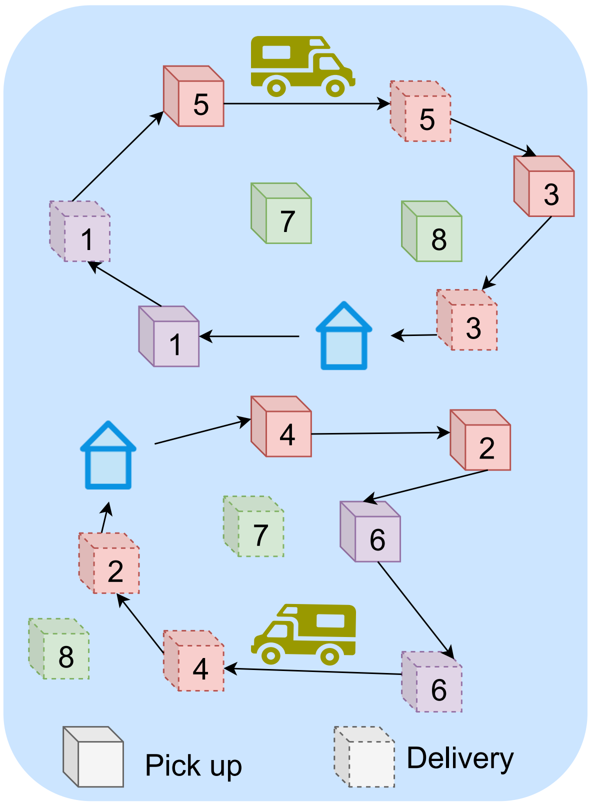


Figure 2.1: Example of solution to MODPDPTW.

Q, departing from and returning to a common depot daily. Denote T as the system's operation time. At any time $t \in T$, each vehicle $v \in V$ is defined by its location l_v^t, and a set of customer demands R^t arises at different locations. Each request $r \in R^t$ is represented by a tuple $< l_r^p, l_r^d, q_r, sp_r^b, sp_r^e, sd_r^b, sd_r^e >$, where l_r^p and l_r^d are the pickup and delivery locations, q_r is the load of the cargo, $[sp_r^b, sp_r^e]$ is the soft pickup time window, and $[sd_r^b, sd_r^e]$ is the soft delivery time window. Regarding the soft time windows, vehicles arriving early wait without penalty, but arriving after the window incurs a delay penalty. Once a vehicle picks up the cargo for request r, it must complete the corresponding delivery. However, vehicles can sequence multiple pickups before subsequent deliveries, provided the total load does not exceed their capacity. This part defines the following variables to monitor the scheduling of vehicles:

- x_r^v is a binary variable that equals 1 if demand r is served by vehicle v.
- $y_{r,ij}^v$ is a binary variable that equals 1 if vehicle v traverses arc (i, j) while serving demand r.
- τ_r^p represents the time when the cargo for demand r is picked up.

- τ_r^d represents the time when the cargo for demand r is fully delivered.

Energy consumption. The MODPDPTW involves a fleet of fuel-powered vehicles that depart from a depot to serve customers. Based on the mechanical power [36], the constant mechanical power P_{ij}^v for vehicle $v \in V$ traveling edge $(i, j) \in \mathcal{E}$ without considering the gradient of the terrain is calculated as follows:

$$P_{ij}^v = \frac{1}{2} \cdot c_d \cdot \rho \cdot S \cdot \gamma^3 + (m_v + \tilde{Q}_{ij}^v) \cdot g \cdot c_r \cdot \gamma, \tag{2.1}$$

where c_d is the aerodynamic drag coefficient, ρ is the air density, S is the frontal area of the vehicle, g is the gravitational constant, c_r is the rolling friction coefficient, m_v is the curb weight of vehicle, and γ is the speed of the vehicle. Additionally, $\tilde{Q}_{ij}^v$ is the load of the vehicle v in edge (i, j).

The fuel consumption of the vehicle $v \in V$ when traversing arc $(i, j) \in \mathcal{E}$ is denoted by L_{ij}^v and is calculated as follows:

$$L_{ij}^v = \frac{\xi}{\kappa \cdot \psi} \cdot \left(\omega \cdot R + \frac{P_{ij}^v}{\eta} \right) \cdot \frac{d_{ij}}{\gamma}, \tag{2.2}$$

where ξ is the fuel-to-air mass ratio, κ is the heating value of fuel, ψ is the fuel rate conversion factor, and ω is the engine friction factor. Additionally, R and η represent the technical parameters of the engine and the drive train efficiency, respectively.

Therefore, energy consumption costs are associated with fuel usage during pickups and deliveries. Based on the referenced energy consumption models, the energy consumption costs EC for the assigned vehicles are calculated as:

$$f_1 = \sum_{t \in T} \sum_{r \in R^t} \sum_{v \in V} \sum_{(i,j) \in \mathcal{E}} y_{r,ij}^v L_{ij}^v. \tag{2.3}$$

Waiting time. Furthermore, the quality of the service is assessed based on the response time to requests. The total waiting time for each demand is defined as follows:

$$WT_r = \max\left(\tau_r^p - sp_r^e, 0\right) + \max\left(\tau_r^d - sd_r^e, 0\right), \forall r \in R^t, t \in T, \tag{2.4}$$

where $\max\left(\tau_r^p - sp_r^e, 0\right)$ and $\max\left(\tau_r^d - sd_r^e, 0\right)$ represent the waiting time of the request before being picked up and delivered, respectively. The system's objective is to minimize the waiting time, ensuring that customers receive prompt attention immediately after making a request. This is particularly significant in contexts where the quality of goods deteriorates over time. Therefore, this research optimizes the maximum waiting time of a request, calculated as follows:

$$f_2 = \max_{r \in R^t, t \in T} WT_r. \tag{2.5}$$

Customer-based Fairness. Fairness is increasingly important in real-world vehicle routing and scheduling. Fairness is addressed from two perspectives: customer-based, focusing on balancing each customer's waiting time, and vehicle-based, aiming to distribute the workload among vehicles evenly. Each customer demand has two-time windows corresponding to the pickup and delivery moments. When a vehicle arrives at the pickup or delivery location before the start of the time window, it waits until the task can be performed. However, if the vehicle arrives after the end of the time window, the customers must wait, leading to a decrease in service quality. Then, customer-based fairness is quantified by the variability in wait times across all requests, as follows:

$$f_3 = \sqrt{\frac{1}{n} \sum_{t \in T} \sum_{r \in R^t} \left(WT_r - \frac{1}{n} \sum_{t \in T} \sum_{r \in R^t} WT_r \right)^2}, \tag{2.6}$$

where $n = \sum_{t \in T} |R^t|$ is the number of customer demands.

Vehicle-provider Fairness. The workload of each vehicle is calculated based on the total distance traveled by that vehicle. Let DT_v represent the workload of vehicle v, which is defined as follows:

$$DT_v = \sum_{t \in T} \sum_{r \in R^t} \sum_{(i,j) \in \mathcal{E}} y^v_{r,ij} d_{ij}, \forall v \in V. \tag{2.7}$$

Instead of focusing solely on profit or revenue, model aims to prevent any particular vehicle from having an excessive workload, thereby achieving a more balanced allocation of trips. Therefore, vehicle-based fairness is evaluated based on the disparity in the workload of the vehicles, calculated as follows:

$$f_4 = \sqrt{\frac{1}{K} \sum_{v \in V} \left(D_v - \frac{1}{K} \sum_{v \in V} D_v \right)^2}. \tag{2.8}$$

In summary, the MODPDPTW problem is modeled as follows:

$$\begin{aligned}
\text{minimize} \quad & \{f_1, f_2, f_3, f_4\}, && (2.9)\\
\text{subject to:} \quad & \sum_{v \in V} x_r^v = 1, \forall r \in R^t, t \in T, && (2.10)\\
& y_{r,ij}^v \leq x_r^v, \forall (i,j) \in \mathcal{E}, r \in R^t, t \in T, v \in V, && (2.11)\\
& \tilde{q}_{ij}^{v,t} \leq Q, \forall (i,j) \in \mathcal{E}, t \in T, v \in V, && (2.12)\\
& \tau_r^p \geq sp_r^b, \forall r \in R^t, t \in T, && (2.13)\\
& \tau_r^d \geq sd_r^b, \forall r \in R^t, t \in T. && (2.14)
\end{aligned}$$

The aim of (2.9) is to minimize energy consumption costs and the maximum time waiting among customers while maintaining a balance between customer waiting times and workload distribution across vehicles. Constraint (2.10) ensures that each customer is served exactly once. Constraint (2.11) stipulates that vehicle v only fulfills request r if r is assigned to v. Constraint (2.12) ensures that the load on each vehicle does not exceed its capacity at any point in time. Constraints (2.13) and (2.14) guarantee that service times, including pickups and deliveries, adhere to the beginning of each customer's designated time window.

CHAPTER 3. PARETO FRONT GRID GUIDED MULTI-OBJECTIVE OPTIMIZATION CONSIDERING TWO-SIDED FAIRNESS

This chapter presents an effective method for solving the MODPDPTW, named Pareto front grid-guided multi-objective optimization with two-sided fairness (PFG-2F). The approach integrates Pareto front grid (PFG) generation with fairness heuristics and an ordering rule based on time window urgency. To explore the solution space, it applies combinatorial operators including Partially mapped crossover (PMX) and Swap mutation.

3.1 Overview

The overview methods, including all components are described in Figure 3.1. To handle and optimize all dynamically appearing requests, each request is stored in a request database as it arrives. At every decision timestep, the system processes all currently unserved requests. The algorithm takes as input the pickup and delivery locations of these requests and initializes a population by randomly shuffling the request sequences. Each individual in the population is then evaluated, and non-dominated sorting is applied to extract a Pareto front for that generation.

If the stopping condition has not been met, the algorithm proceeds to generate a Pareto front grid (PFG) using the current ideal point z^* and nadir point z^{nad}. This grid guides the selection process, after which crossover and mutation operators—guided by selected heuristics—are applied to generate offspring. The resulting offspring are then added to the population for the next iteration.

Once the stopping condition is satisfied (i.e., max generation is reached), a representative solution is selected for execution. This solution typically balances multiple objectives—such as minimizing energy cost, reducing customer waiting time, or ensuring fairness among customers and service providers. In most cases, a knee point is selected as the final solution, as described in Subsection 1.2.1. The selected solution is then passed to the system controller, which routes the vehicles accordingly, completing the operations for the current timestep.

3.2 Individual representation

In [27], the Leader-based random keys encoding scheme (LERK) is introduced to represent complete solutions. Specifically, leader keys determine the sequence of service providers, and the subsequent assignment of all requests follows this ordering. This random key encoding strategy has been shown to be effective in generating diverse solutions within the objective space. In order to promote equitable

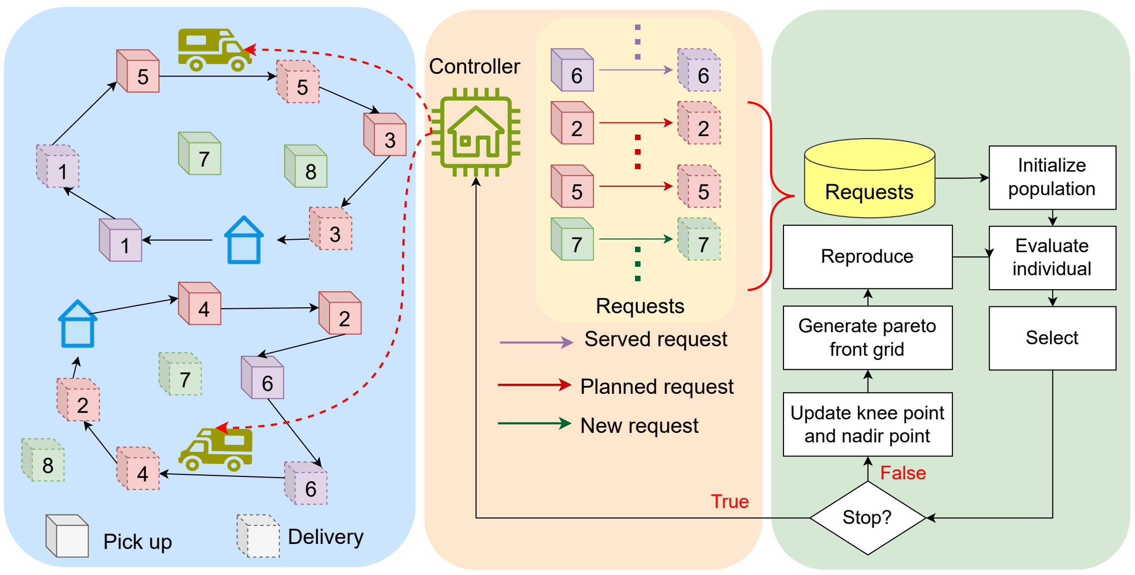


Figure 3.1: The overview of the proposed algorithm.

distribution of workload among vehicle providers, this thesis propose an enhanced version of LERK that incorporates heuristic initialization. Figure 3.2 demonstrates an example of individual representation. Our methodology comprises two main components: (1) *Fairness heuristics initialization* and (2) an *Urgent time ordering rule*. The former ensures a balanced allocation of requests across all vehicles, while the latter reorders requests based on the urgency of their time windows. By initially distributing requests equally among vehicles and then prioritizing more urgent time windows, the approach fosters fairness among providers and helps ensure that tightly constrained time windows are served promptly.

3.2.1 Fairness heuristics initialization

Consider a system with K vehicles, each operated by a distinct service provider and collectively responsible for fulfilling a set of customer requests. In multi-objective vehicle routing problems—especially those involving dynamic and evolving demand—ensuring fairness in workload distribution across vehicles is not only a practical concern but also a key sustainability objective. Unbalanced assignments, where some vehicles are overloaded while others are underutilized, can lead to provider dissatisfaction and reduce long-term system stability.

To address this challenge, the proposed method introduces a fairness-enhancing heuristic at the initialization stage of the algorithm. The core idea is to assign customer requests to vehicles in a balanced way, such that each vehicle is initially responsible for approximately the same number of customers or requests. This ensures a fair starting point before further refinement is performed through the evolutionary process.

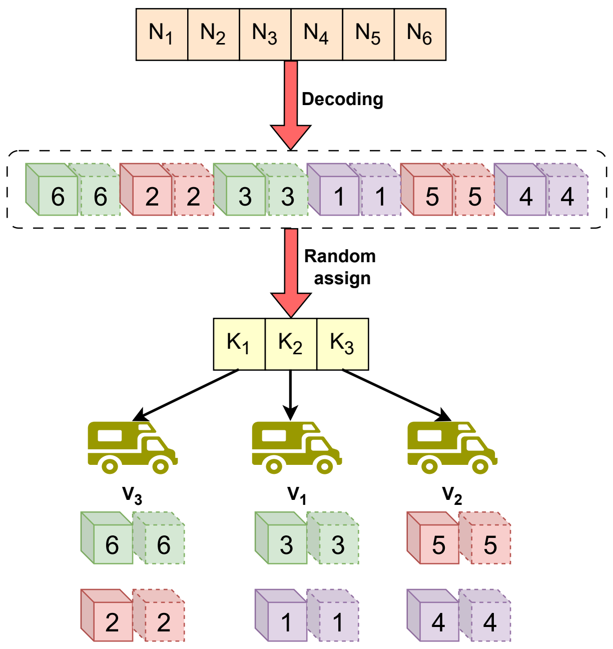


Figure 3.2: Individual representation.

Although this initial assignment aims to be equitable, it is not necessarily optimal in terms of routing efficiency or other objectives (e.g., energy consumption, waiting time). Therefore, the solution is further improved using evolutionary operations such as crossover and mutation, which explore the search space while preserving the fairness structure as much as possible.

The pseudocode in Algorithm 1 outlines the initial fairness-based assignment strategy.

Algorithm 1 Fair request assignment among vehicles

```
Require: Set of customer requests $\mathcal{R}$, number of vehicles $K$
Ensure: Initial assignment of requests to vehicle routes $\{V_1, V_2, ..., V_K\}$
1: Initialize empty vehicle routes: $\mathcal{V} \leftarrow [V_1, V_2, \ldots, V_K]$
2: Shuffle the request set $\mathcal{R}$ randomly to ensure fairness
3: Initialize counter $i \leftarrow 0$
4: for all $r \in \mathcal{R}$ do
5:     $k \leftarrow (i \bmod K) + 1$
6:     Assign request $r$ to vehicle route $V_k$
7:     $i \leftarrow i + 1$
8: end for
9: return $\mathcal{V}$
```

This heuristic ensures that each vehicle is initially assigned roughly $\lceil |\mathcal{R}|/K \rceil$ requests. Such an approach improves fairness among providers from the outset, laying a strong foundation for later optimization steps. As the evolutionary process proceeds, these assignments are modified using genetic operators such as crossover and mutation, which enable exploration of diverse routing configurations while maintaining workload balance constraints when necessary.

This fairness-first initialization plays a crucial role in the proposed multi-objective framework by improving both social equity among service providers and the quality of solutions obtained throughout the optimization process.

3.2.2 Urgent time ordering rule

In many vehicle routing scenarios, each vehicle $k \in K$ is assigned a subset of customer requests from the set R^t at time t. To prioritize more urgent requests, the algorithm applies a time-aware reordering heuristic to each vehicle's assigned requests. Specifically, for each vehicle k, the sequence of requests $\mathbf{p}_k$ is reordered in ascending order of urgency, determined by the closing time of the associated pickup or delivery window. The reordered sequence is denoted as $\tilde{\mathbf{p}}_k$, and computed as:

$$\tilde{\mathbf{p}}_k = \mathrm{ReOrder}\big(\mathbf{p}_k\big), \tag{3.1}$$

where $\mathrm{ReOrder}(\cdot)$ is a function that ranks requests based on the earliest end of their pickup or delivery time windows. This approach ensures that requests with tighter deadlines or narrower windows are serviced earlier, improving the feasibility of time-constrained routes.

Algorithm 2 ReOrder requests based on time window urgency

```
Require: Assigned request list $\mathbf{p}_k$ for vehicle $k$
Ensure: Reordered request sequence $\tilde{\mathbf{p}}_k$
1: Initialize empty list $\tilde{\mathbf{p}}_k \leftarrow []$
2: Define a comparator: for each request $r$, compute urgency score
3: urgency$(r) \leftarrow \min(sp_r^e, sd_r^e)$
4: Sort $\mathbf{p}_k$ in ascending order based on urgency$(r)$
5: Store result in $\tilde{\mathbf{p}}_k$
6: return $\tilde{\mathbf{p}}_k$
```

3.3 Pareto front grid generation

Pareto front grid guided multi-objective evolutionary algorithm (PFG-MOEA), first proposed in [37], has proven effective for identifying high-quality solutions in multi-objective optimization. The strategy leverages Pareto Front to improve the searching approach to guide the evolutionary process by partitioning the objective space into grids and retaining leading individuals within each grid cell. This method focuses the search on the most promising regions of the objective space, thereby enhancing both the quality of the solutions and the efficiency of the search.

The algorithm employs two well-established reference points in the multi-objective space: the ideal point z^* and the nadir point z^{nad}. Both points are updated at each evolutionary iteration. The ideal point represents the optimal values for each objective, highlighting solutions on the Pareto front that demonstrate significant trade-offs among objectives. In contrast, the nadir point corresponds to the worst objective values within the Pareto optimal set. These reference points are utilized in each generation to construct the PFG, as illustrated in Figure 3.3. The generation of PFG is detailed in Algorithm 3.

Algorithm 3 Generate PFG

```
Require: population pop, the number of segments GK, the number of objectives $m$, the idea point $z^* = [z_1^*, ..., z_m^*]$, the nadir point $z^n = [z_1^n, ..., z_m^n]$, a small positive value $\sigma$;
Ensure: Pareto front grid $\{PFG_j^i\}, \forall j = 1, ..., GK, i = 1, ..., m$;
 1: for $j \leftarrow 1$ to $m$ do
 2:     $d_j \leftarrow \left(z_j^n - z_j^* + 2\sigma\right)/\text{GK}$
 3: end for
 4: for all $x \in \text{pop}$ do
 5:     for $j \leftarrow 1$ to $m$ do
 6:         $\text{Grid}_j(x) \leftarrow \left\lceil \frac{f_j(x) - z_j^* + \sigma}{d_j} \right\rceil$
 7:     end for
 8: end for
 9: for $i \leftarrow 1$ to $m$ do
10:     for $j \leftarrow 1$ to GK do
11:         $S_i(j) \leftarrow$ individuals depending on the $j-$th segment of the $i$-th objective;
12:         $g_{\min} \leftarrow \min\{\text{Grid}(S_i(j))\}$
13:         for all $x \in S_i(j)$ do
14:             if $\text{Grid}_j(x) = g_{\min}$ then
15:                 $PFG_j^i \leftarrow PFG_j^i \cup \{x\}$
16:             end if
17:         end for
18:     end for
19: end for
```

3.4 Reproduction

PFG is employed to generate grids that partition the objective space, thereby grouping solutions into different regions. Within each grid, selection is conducted via non-dominated sorting, and the ideal point z^* is then computed. Any solutions lying farthest from the hyperplane defined by z^* are removed. Then, two solutions are randomly chosen from the mating pool P to serve as parents. To foster diversity both within the same PFG and across the entire population, crossover is governed by a probabilistic rate ϵ, as shown in Equation 3.3. With probability ϵ, the parents are drawn from consecutive grids (i.e., $PFG_i^j \cup PFG_{i+1}^j$); otherwise, they are taken from the entire population pop. Therefore, P is define as

$$P = \begin{cases} PFG_i^j \cup PFG_{i+1}^j, & \text{if } \mathrm{rand} < \epsilon, \\ \mathrm{pop}, & \text{otherwise}. \end{cases} \tag{3.2}$$

where $rand$ is a random variable following a uniform distribution with a range of [0, 1].

3.4.1 Selection

PFG is employed to generate grids that partition the objective space, thereby grouping solutions into different regions. Within each grid, selection is conducted via non-dominated sorting, and the ideal point z^* is then computed. Any solutions lying farthest from the hyperplane defined by z^* are removed. Then, two solutions are randomly chosen from the mating pool P to serve as parents. To foster diversity both within the same PFG and across the entire population, crossover is governed by a probabilistic rate ϵ, as shown in Equation 3.3. With probability ϵ, the parents are drawn from consecutive grids (i.e., $PFG_i^j \cup PFG_{i+1}^j$); otherwise, they are taken from the entire population pop. Therefore, P is define as

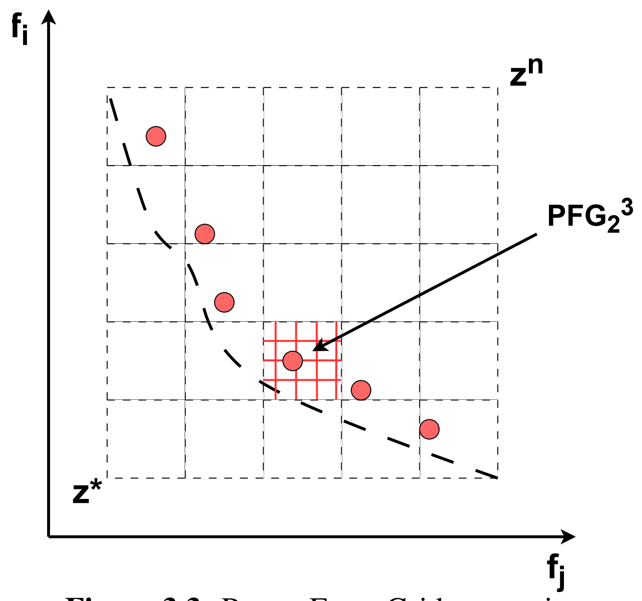


Figure 3.3: Pareto Front Grid generation.

$$P = \begin{cases} PFG_i^j \cup PFG_{i+1}^j, & \text{if } \mathrm{rand} < \epsilon, \\ \mathrm{pop}, & \text{otherwise}. \end{cases} \tag{3.3}$$

where $rand$ is a random variable following a uniform distribution with a range of [0, 1].

3.4.2 Crossover

This thesis use crossover PMX is a recombination operator introduced for genetic algorithms that work on permutation representations. It is particularly useful for problems where the solution can be represented as a sequence or permutation.

PMX works by selecting two parent solutions and exchanging a subset of genes between them while preserving the relative order and position of elements from the parents. The steps of the PMX operator are as follows:

1. Select two crossover points randomly along the length of the parents to define a matching section.
2. Copy the matching section from the first parent directly into the offspring.
3. Map the corresponding elements from the second parent into the offspring to resolve conflicts, ensuring no duplicate genes.
4. Fill in the remaining positions from the second parent, using the mapping to avoid conflicts and maintain a valid permutation.

Algorithm 4 Partially mapped crossover (PMX)

```
Require: Permutations P1, P2 of length $n$
Ensure: Offspring O
 1: Randomly choose crossover points $r_1 < r_2$
 2: Initialize O with None
 3: for $i = r_1$ to $r_2$ do
 4:     O[i] ← P1[i]
 5: end for
 6: for $i = r_1$ to $r_2$ do
 7:     if P2[i] not in O then
 8:         val ← P2[i], pos ← $i$
 9:         while P1[pos] in O do
10:             pos ← index of P1[pos] in P2
11:         end while
12:         O[pos] ← val
13:     end if
14: end for
15: for $i = 0$ to $n - 1$ do
16:     if O[i] is None then
17:         O[i] ← P2[i]
18:     end if
19: end for
20: return O
```

3.4.3 Mutation

Swap mutation is a simple yet effective genetic operator used to introduce variation in solution populations during the optimization process. In the context of MODPDPTW, each solution is typically represented as a sequence of pickup and delivery nodes that must respect precedence constraints—i.e., each pickup must occur before its corresponding delivery—and vehicle capacity limits.

The Swap mutation operator selects two distinct positions in the solution sequence and exchanges their values. However, due to the precedence requirements in modpdptw, a direct swap may result in infeasible solutions. To maintain feasibility, the mutation process is constrained as follows:

- A pickup node can only be swapped with another pickup node, and similarly, a delivery node with another delivery node.
- Swapping a pickup and its corresponding delivery is not allowed unless their relative order is preserved.
- After mutation, the solution is checked for precedence, capacity and time windows feasibility; if violated, the swap is discarded or repaired.

By applying these rules, Swap mutation helps explore the solution space without disrupting the problem's structural constraints. It enhances diversity in the population and aids in escaping local optima during the evolutionary search.

CHAPTER 4. SIMULATION EXPERIMENTS

In this chapter, experiments are conducted on multiple datasets, including large-scale instances with a high number of requests to simulate real-world scenarios. The proposed algorithm is compared against state-of-the-art MOO algorithms using the HV and IGD metrics. The results demonstrate the effectiveness and real-time adaptability of the proposed method.

4.1 Experimental settings

4.1.1 Datasets

The dataset is developed based on PDPTW [38], ensuring authenticity and diversity to reflect real-world scenarios. It consists of 180 instances tailored for time slot routing and is categorized into three types: 100 requests, 200 requests, and 400 requests. Each instance is named using the format `[dis][category]_-[type]_[index]`, where `dis` indicates the customer distribution type of the instance, including `R` for random, `C` for clustered, and `RC` for combined, `category` represents the difficulty level based on time windows (`1` for narrower and `2` for wider and more flexible), and `type` denotes the number of locations in the dataset (for example, a 100 request instance includes 200 locations). The number of vehicle-providers is scaled up to fit for each dynamic circumstances in the dataset. Hence, this work creates 18 different distribution to evaluate the effectiveness of all algorithms.

All parameters used in Section 2.2 are set up to be adapted to real world scenarios, this work uses the configuration to calculate energy consumption as the setup described in [36]. The problem parameters are set as follow: $c_d = 0.7$, $c_r = 0.01$, $\xi = 1$, $\eta = 0.36$, $m_v = 3.2\,\text{tons}$, $\psi = 737\,\text{liters/gram}$, $\gamma = 40\,\text{km/h}$, $\kappa = 44\,\text{kJ/gram}$, $g = 9.81\,\text{m/s}^2$, $\rho = 1.20\,\text{kg/m}^3$, $\omega = 0.2\,\text{kJ/(rev} \cdot \text{liter)}$, $S = 3.912\,\text{m}^2$, $R = 165\,\text{liters} \cdot \text{rev/s}$. These values are used consistently in all subsequent calculations.

4.1.2 Runtime environment

All experiments in this thesis were conducted using Kaggle's cloud-based computing environment. The notebook environment provided access to a CPU with 4 cores and 16 GB of RAM, which was sufficient for executing the proposed models and evaluating their performance on benchmark datasets. This setup allowed for reproducible and scalable experimentation without the need for dedicated local hardware resources.

4.1.3 Compared algorithms

This thesis evaluates the performance of PFG-2F against the original PFG-MOEA [37] and other state-of-the-art multi-objective baseline algorithms, including NSGA-II [5], MOEA/D [6], and MOEA/D-DE [39]. To assess the effectiveness of the Random-Key encoding scheme, the thesis develops a variant of PFG-2F named PFG-2F-PM, which employs a permutation encoding scheme combined with Partially Mapped Crossover (PMX) [40]. This variant is compared to the proposed algorithm, referred to as PFG-2F-RK, to determine the impact of the encoding strategy on overall performance.

To evaluate effectiveness, the HV and IGD metrics (as defined in Subsection 1.2.1) are computed separately for each instance and averaged over 10 independent runs. Specifically, for HV, the reference point is set as the worst objective values observed across all solutions for each instance. For IGD, the reference Pareto front is approximated by collecting all non-dominated solutions found by all algorithms across all runs.

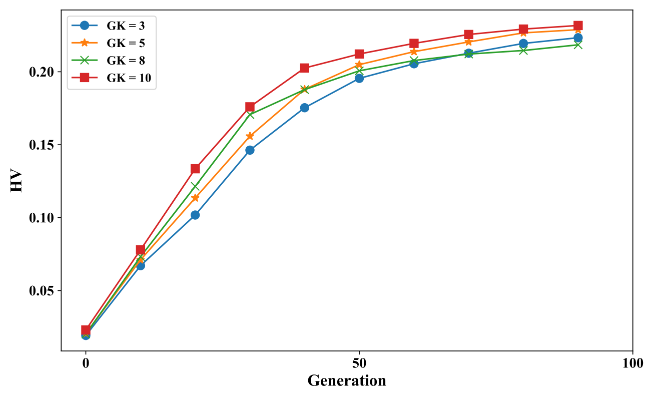
4.2 Experimental results

4.2.1 Parameter tuning

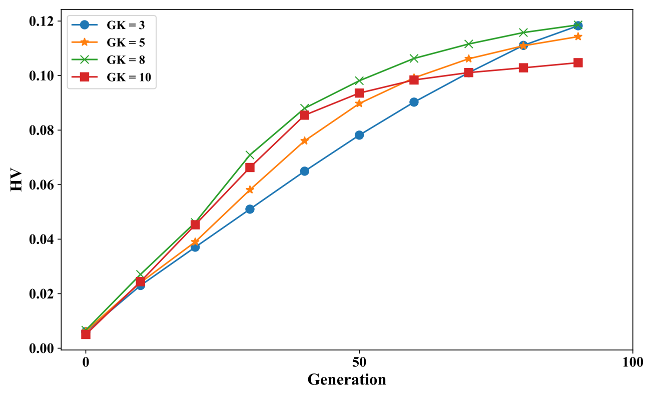
To investigate the impact of the parameter GK on the performance of PFG in the objective and solution space decomposition process, this section conduct a fine-tuning experiment. The parameter GK determines the number of grid segments used to partition the objective space, which directly affects how the PFG are constructed. Figure 4.1 presents the convergence trends in terms of hypervolume (HV) for different GK values. From the results, we observe that $GK = 5$ consistently yields the best overall performance across most test instances. This balance stems from the way GK influences the granularity of the decomposition:

When GK is set to a small value, the number of PFGs is limited, resulting in each grid region covering a broader portion of the objective space. Consequently, more solutions fall into each region, increasing the competition during environmental selection. Only a few solutions with superior convergence are retained, which reduces diversity across the population due to high selection pressure.

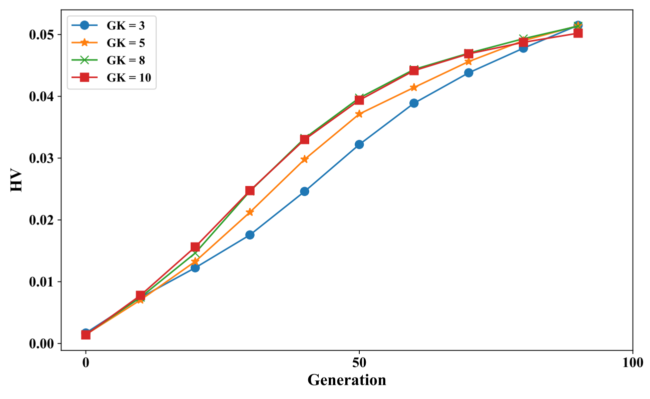
Conversely, using a large GK value results in a finer grid, generating a greater number of PFGs. Each region then contains fewer solutions, typically close to one another in objective space. Although this setup improves population diversity by spreading solutions across many small regions, it weakens convergence since many selected solutions may be similar and suboptimal in terms of quality.



(a) GK performance in 100 requests.



(b) GK performance in 200 requests.



(c) GK performance in 400 requests.

Figure 4.1: Parameter GK performance analysis.

Thus, selecting an appropriate GK is essential for maintaining a good trade-off between convergence and diversity. Our findings suggest that $GK = 5$ strikes this balance effectively, leading to improved overall performance.

4.2.2 Compare with MOO algorithms

Table 4.1 demostrates the average HV values of the evaluated algorithms across various scenarios. The results indicate that the proposed algorithm consistently outperforms existing approaches in nearly all instances. Among the compared methods, the algorithms incorporating the Pareto front grid and equal workload distribution among vehicles exhibit substantial improvements over the remaining approaches. Notably, PFG-2F with the Random-key encoding scheme achieves a marginal improvement over the Permutation encoding scheme. This improvement can be attributed to the inherent limitations of genetic operators in the Permutation encoding scheme, which are less effective in the reproduction process compared to the equal workload division strategy. In contrast, the Random-key encoding scheme offers greater flexibility and a more effective exploration of the solution space. Although PFG-MOEA follows an evolutionary trend guided by the Pareto front grid, fairness objectives are not explicitly considered. When the number of requests remains constant, PFG-2F-RK demonstrates the most significant improvement in scenarios where customer locations exhibit a clustered distribution. With customer locations are randomly distributed, the proposed algorithm continues to maintain superior performance relative to other approaches. However, when combining both random and clustered distributions, the fairness mechanism of PFG-2F-RK encounters challenges, leading to a reduction in performance gains. Despite these challenges, the proposed algorithm remains more effective across the majority of instances. Furthermore, as the number of requests increases, its performance remains stable despite capacity and resource constraints. This finding highlights the robustness and adaptability of PFG-2F-RK in handling complex and dynamic problem environments.

Table 4.1: HV values of algorithms

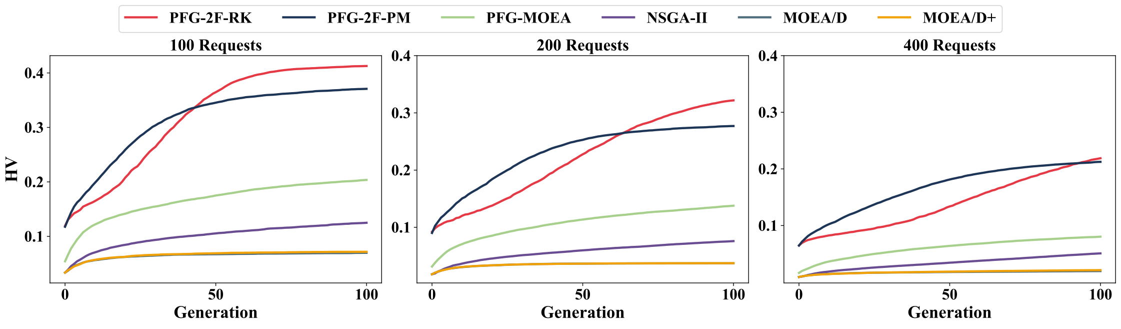
		NSGA-II	MOEA/D	MOEA/D-DE	PFG-MOEA	PFG-2F-PM	PFG-2F-RK
100	**C**	0.138 ± 0.03	0.077 ± 0.03	0.081 ± 0.02	0.209 ± 0.04	0.412 ± 0.05	**0.463 ± 0.06**
	R	0.081 ± 0.02	0.038 ± 0.01	0.038 ± 0.01	0.150 ± 0.03	0.287 ± 0.04	**0.325 ± 0.07**
	RC	0.043 ± 0.02	0.018 ± 0.01	0.020 ± 0.01	0.086 ± 0.02	0.212 ± 0.03	**0.223 ± 0.04**
200	**C**	0.113 ± 0.04	0.067 ± 0.03	0.064 ± 0.03	0.192 ± 0.05	0.331 ± 0.05	**0.370 ± 0.06**
	R	0.077 ± 0.02	0.037 ± 0.01	0.039 ± 0.01	0.132 ± 0.03	0.273 ± 0.03	**0.316 ± 0.02**
	RC	0.052 ± 0.02	0.018 ± 0.01	0.021 ± 0.01	0.074 ± 0.03	**0.211 ± 0.04**	0.203 ± 0.05
400	**C**	0.124 ± 0.04	0.066 ± 0.02	0.071 ± 0.02	0.209 ± 0.04	0.369 ± 0.06	**0.406 ± 0.06**
	R	0.070 ± 0.02	0.036 ± 0.01	0.036 ± 0.01	0.131 ± 0.03	0.271 ± 0.05	**0.324 ± 0.05**
	RC	0.057 ± 0.02	0.023 ± 0.01	0.022 ± 0.01	0.081 ± 0.03	0.215 ± 0.02	**0.231 ± 0.04**

Additionally, convergence of the algorithms is visualized in Figure 4.2a. Although PFG-2F-PM demonstrates superior exploration capability during the first 50 generations compared to the proposed algorithm, it quickly becomes trapped in local optima. In contrast, PFG-2F-RK continues to improve, although at a decreasing rate as it approaches convergence. The MOEA/D and MOEA/D-DE algorithms exhibit early convergence, often settling in local optima. Meanwhile, PFG-MOEA and NSGA-II continue to improve beyond the 100th generation; however, their

Table 4.2: IGD values of algorithms

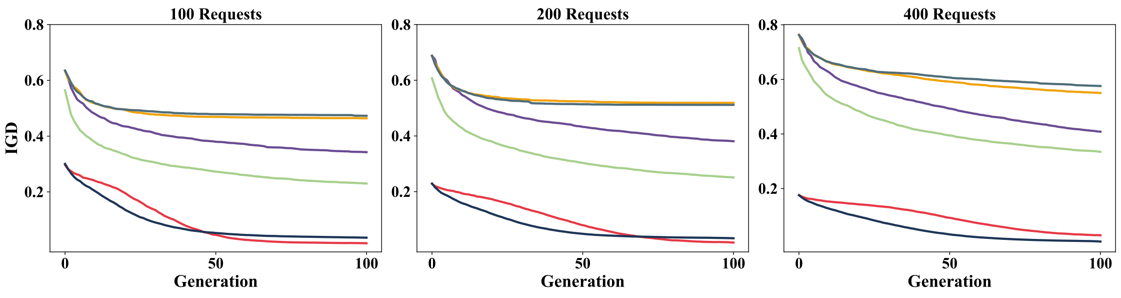
		NSGA-II	MOEA/D	MOEA/D-DE	PFG-MOEA	PFG-2F-PM	PFG-2F-RK
100	**C**	0.353 ± 0.05	0.481 ± 0.07	0.468 ± 0.06	0.258 ± 0.04	0.039 ± 0.03	**0.020** ± **0.02**
	R	0.369 ± 0.06	0.517 ± 0.07	0.523 ± 0.07	0.241 ± 0.05	0.033 ± 0.04	**0.017** ± **0.02**
	RC	0.435 ± 0.09	0.586 ± 0.12	0.558 ± 0.10	0.298 ± 0.06	**0.006** ± **0.01**	0.026 ± 0.01
200	**C**	0.325 ± 0.08	0.450 ± 0.08	0.454 ± 0.09	0.201 ± 0.07	0.030 ± 0.03	**0.010** ± **0.01**
	R	0.372 ± 0.07	0.505 ± 0.08	0.504 ± 0.08	0.247 ± 0.07	0.024 ± 0.02	**0.018** ± **0.01**
	RC	0.401 ± 0.10	0.590 ± 0.11	0.541 ± 0.10	0.370 ± 0.09	**0.004** ± **0.00**	0.035 ± 0.02
400	**C**	0.350 ± 0.06	0.488 ± 0.07	0.470 ± 0.07	0.232 ± 0.05	0.038 ± 0.04	**0.017** ± **0.02**
	R	0.402 ± 0.06	0.513 ± 0.08	0.530 ± 0.09	0.265 ± 0.06	0.040 ± 0.03	**0.017** ± **0.02**
	RC	0.389 ± 0.09	0.551 ± 0.08	0.552 ± 0.09	0.335 ± 0.08	**0.008** ± **0.01**	0.026 ± 0.02

progress remains marginal compared to the proposed algorithm in terms of both convergence speed and solution quality. Similarly, this part presents the average IGD values and convergence trends in Table 4.2 and Figure 4.2b, respectively. The improvement achieved by PFG-2F-RK follows a similar pattern to that observed for HV. However, PFG-2F-PM outperforms the proposed algorithm on datasets where customer distribution combines both random and clustered patterns.



S

(a) HV convergence trend of PFG-2F compared to other benchmarks.



(b) IGD convergence trend of PFG-2F compared to other benchmarks.

Figure 4.2: HV and IGD convergence across different instances.

Figure 4.3 illustrates the effectiveness of PFG-2F-RK across 18 scenarios. Each scenario combines 100, 200, or 400 requests with one of six distributions (`C1`, `C2`, `R1`, `R2`, `RC1`, `RC2`). As the number of requests increases, performance tends to decline because the algorithm must handle a larger search space with the same number of generations. Among all distributions, `C2` yields the most favorable outcomes, while those with narrower time windows exhibit more variability. The worst performance occurs under `R1`, primarily due to the randomness of the data and tighter time constraints.

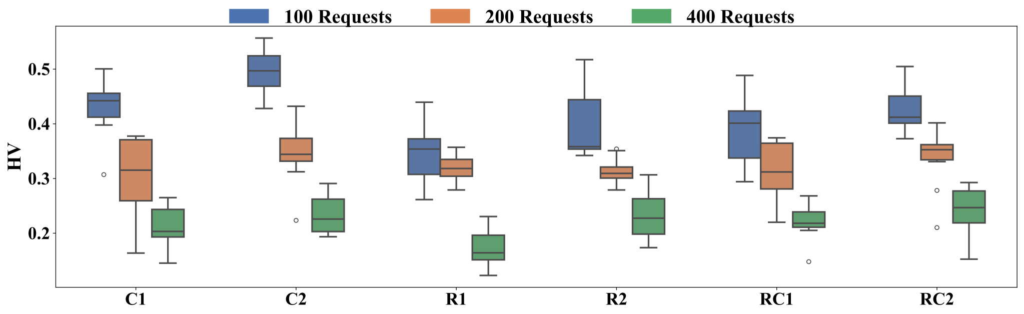


Figure 4.3: Effectiveness of PFG-2F over all instances.

4.2.3 Compare with SOO algorithms

To evaluate the effectiveness of PFG-2F in comparison to single-objective algorithms, this work benchmarks it against FairGA [27]. FairGA is specifically designed to optimize the total distance traveled while incorporating two-sided fairness considerations. It achieves this by integrating two fairness objectives related to reproduction alongside its primary distance optimization.

This thesis assesses improvement using Equation 4.1 across the four objectives: EC, VF, CF, and MWT. Since PFG-2F generates a set of non-dominated solutions due to its multi-objective nature, this work selects a single representative solution for comparison. The PFG-2F algorithm chooses the solution that corresponds to the intersection of the hyperplane created by the obtained Pareto front and the line connecting the ideal and nadir points. This solution is deemed effective for balancing the objective functions. If no such solution is found, the algorithm selects the solution nearest to the ideal point.

The improvement of algorithm A compared to algorithm B for instance i is $\%imp(A, B, i)$, which is calculated as follows:

$$\%imp(A, B, i) = \frac{f(B, i) - f(A, i)}{f(B, i)} \times 100\% \tag{4.1}$$

where $f(\cdot)$ is the considered objective value.

Table 4.3 presents the percentage improvements of PFG-2F over FairGA across different instances. The results indicate a significant enhancement in the total distance objective, which is the primary focus of FairGA. Additionally, PFG-2F demonstrates superior performance in the remaining three objectives: customer fairness (CF), vehicle fairness (VF), and time window constraint satisfaction (MWT).

Table 4.3: Improvement of PFG-2F Compared to FairGA

Dis	EC imp (%)	VF imp (%)	CF imp (%)	MWT imp (%)
C1	15.42	69.15	39.74	40.11
C2	18.74	58.04	45.75	36.18
R1	16.29	64.83	28.85	34.46
R2	14.56	59.95	58.11	49.19
RC1	17.62	64.90	31.35	33.60
RC2	15.85	64.40	58.87	52.21

These findings highlight the advantage of employing a multi-objective approach, as PFG-2F effectively balances and improves multiple objectives simultaneously, as evidenced by the increased HV and IGD.

CONCLUSIONS

This thesis introduces the Multi-objective dynamic pickup and delivery problem with time windows, which captures the complexities of real-world logistics systems operating under dynamic and uncertain conditions. Unlike static formulations, this problem requires vehicles to make routing decisions incrementally as new transportation requests arrive over time. To reflect realistic logistics objectives, the proposed framework incorporates multiple goals: minimizing energy consumption to promote sustainability, reducing customer waiting times to enhance service quality, and ensuring fairness in workload distribution across both customers and service providers. These objectives mirror key concerns in on-demand delivery services, ride-sharing, and urban distribution networks, where both efficiency and equity are critical.

To address these challenges, a novel algorithm called PFG-2F is proposed, which combines Pareto front grid generation with fairness-aware heuristics and random-key-based solution encoding. The algorithm is designed to efficiently explore the multi-objective solution space while remaining responsive to dynamic inputs. Experimental evaluations conducted across a variety of benchmark scenarios demonstrate that PFG-2F significantly outperforms existing state-of-the-art multi-objective algorithms, achieving better results in both hypervolume and inverted generational distance. These results highlight the algorithm's ability to generate diverse, high-quality solutions with superior convergence properties. Looking ahead, future research may investigate the integration of machine learning-based heuristics or predictive models to further enhance the adaptability and decision-making capabilities of the system, enabling even more efficient and intelligent vehicle-task allocation in complex, time-sensitive logistics environments.

Publications and related papers

The main research results and methodologies developed throughout this thesis have been peer-reviewed and officially accepted for presentation and publication at:

> **Phan Hung**, Bui Duc, Nguyen Tam and Huynh Thi Thanh Binh, "Pareto Front Grid Guided Multiobjective Optimization In Dynamic Pickup And Delivery Problem Considering Two-Sided Fairness" in The Genetic and Evolutionary Computation Conference (GECCO '25), 2025. (accepted, full paper, **rank A**)

REFERENCES

[1] K. Miettinen, *Nonlinear Multiobjective Optimization*. Springer, 1999.

[2] M. Ehrgott, *Multicriteria Optimization*, 2nd ed. Springer, 2005.

[3] K. Deb, “Multiobjective optimization using evolutionary algorithms. wiley, new york,” in Jan. 2001.

[4] N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. DOI: 10.1109/4235.996017.

[6] Q. Zhang and H. Li, “Moea/d: A multiobjective evolutionary algorithm based on decomposition,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007. DOI: 10.1109/TEVC.2007.892759.

[7] J. Mejía-de Dios, J. Carballido, D. Ruiz, V. Liern, and P. García-Sánchez, “Metaheuristics: A julia package for single- and multi-objective optimization,” *Journal of Open Source Software*, vol. 7, no. 78, p. 4723, 2022. DOI: 10.21105/joss.04723. [Online]. Available: https://doi.org/10.21105/joss.04723.

[8] G. B. Dantzig and J. H. Ramser, “The truck dispatching problem,” *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.

[9] T. Vidal, G. Laporte, and P. Matl, “A concise guide to existing and emerging vehicle routing problem variants,” *European Journal of Operational Research*, vol. 286, no. 2, pp. 401–416, 2020. DOI: 10.1016/j.ejor.2019.10.010.

[10] G. Team. “Vehicle routing problem.” Accessed October 29, 2019. (2019), [Online]. Available: https://gosmartlog.com/vehicle-routing-problem/ (visited on 10/29/2019).

[11] M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987. DOI: 10.1287/opre.35.2.254.

[12] M. W. P. Savelsbergh and M. Sol, “The general pickup and delivery problem,” *Transportation Science*, vol. 29, no. 1, pp. 17–29, 1995. DOI: 10.1287/trsc.29.1.17.

[13] J. Oyola, H. Arntzen, and D. L. Woodruff, “The stochastic vehicle routing problem, a literature review, part i: Models,” *EURO Journal on Trans-*

portation and Logistics, vol. 7, no. 3, pp. 193–221, 2018, ISSN: 2192-4376. DOI: `https://doi.org/10.1007/s13676-016-0100-5`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2192437620300649`.

[14] H. Lei, G. Laporte, and B. Guo, "The capacitated vehicle routing problem with stochastic demands and time windows," *Comput. Oper. Res.*, vol. 38, no. 12, 1775–1783, Dec. 2011, ISSN: 0305-0548. DOI: `10.1016/j.cor.2011.02.007`. [Online]. Available: `https://doi.org/10.1016/j.cor.2011.02.007`.

[15] V. Pillac, M. Gendreau, C. Guéret, and J.-Y. Potvin, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[16] G. Berbeglia, J.-F. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8–15, 2010.

[17] M. Gendreau, F. Guertin, J.-Y. Potvin, and É. Taillard, "Parallel tabu search for real-time vehicle routing and dispatching," *Transportation Science*, vol. 33, no. 4, pp. 381–390, 1999.

[18] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *Journal of Combinatorial Optimization*, vol. 10, no. 4, pp. 327–343, 2005.

[19] S. Erdogan and E. Miller-Hooks, "A green vehicle routing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 48, no. 1, pp. 100–114, 2012. DOI: `10.1016/j.tre.2011.08.001`.

[20] T. Suhr, A. J. Biega, M. Zehlike, K. P. Gummadi, and A. Chakraborty, "Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019. [Online]. Available: `https://api.semanticscholar.org/CorpusID:196184899`.

[21] N. S. Lesmana, X. Zhang, and X. Bei, "Balancing efficiency and fairness in on-demand ridesourcing," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019. [Online]. Available: `https://proceedings.neurips.cc/paper_files/paper/2019/file/3070e6addcd702cb58de5d7897bfdae1-Paper.pdf`.

[22] V. Nanda, P. Xu, K. Sankararaman, J. Dickerson, and A. Srinivasan, "Balancing the tradeoff between profit and fairness in rideshare platforms during high-demand hours," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 2210–2217, Apr. 2020. DOI: `10.1609/aaai.v34i02.5597`.

[23] Y. Xu and P. Xu, "Trade the system efficiency for the income equality of drivers in rideshare," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, ser. IJCAI'20, Yokohama, Yokohama, Japan, 2021, ISBN: 9780999241165.

[24] D. Shi, Y. Tong, Z. Zhou, B. Song, W. Lv, and Q. Yang, "Learning to assign: Towards fair task assignment in large-scale ride hailing," Aug. 2021, pp. 3549–3557. DOI: `10.1145/3447548.3467085`.

[25] N. Raman, S. Shah, and J. Dickerson, "Data-driven methods for balancing fairness and efficiency in ride-pooling," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed., Main Track, International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 363–369. DOI: `10.24963/ijcai.2021/51`. [Online]. Available: `https://doi.org/10.24963/ijcai.2021/51`.

[26] A. Gupta, R. Yadav, A. Nair, A. Chakraborty, S. Ranu, and A. Bagchi, "Fairfoody: Bringing in fairness in food delivery," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 11 900–11 907, Jun. 2022. DOI: `10.1609/aaai.v36i11.21447`.

[27] Y. Kang, R. Zhang, W. Shao, F. Salim, and J. Chan, "Promoting two-sided fairness in dynamic vehicle routing problems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '24, ACM, Jul. 2024, 759–767. DOI: `10.1145/3638529.3654207`. [Online]. Available: `http://dx.doi.org/10.1145/3638529.3654207`.

[28] J. Cai, Q. Zhu, Q. Lin, L. Ma, J. Li, and Z. Ming, "A survey of dynamic pickup and delivery problems," *Neurocomputing*, p. 126 631, 2023.

[29] G. Santiyuda, R. Wardoyo, R. Pulungan, and F. Y. Vincent, "Multi-objective reinforcement learning for bi-objective time-dependent pickup and delivery problem with late penalties," *Engineering Applications of Artificial Intelligence*, vol. 128, p. 107 381, 2024.

[30] Y. Ma, X. Hao, J. Hao, *et al.*, "A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems," *Advances in neural information processing systems*, vol. 34, pp. 23 609–23 620, 2021.

[31] X. Li, W. Luo, M. Yuan, *et al.*, "Learning to optimize industry-scale dynamic pickup and delivery problems," in *2021 IEEE 37th international conference on data engineering (ICDE)*, IEEE, 2021, pp. 2511–2522.

[32] J. Jeong and I. Moon, "Dynamic pickup and delivery problem for autonomous delivery robots in an airport terminal," *Computers & Industrial Engineering*, vol. 196, p. 110 476, 2024.

[33] Y. Zhou, L. Kong, L. Yan, Y. Liu, and H. Wang, "A memetic algorithm for a real-world dynamic pickup and delivery problem," *Memetic Computing*, pp. 1–15, 2024.

[34] J. Cai, Q. Zhu, Q. Lin, Z. Ming, and K. C. Tan, "Decomposition-based multiobjective evolutionary optimization with tabu search for dynamic pickup and delivery problems," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[35] S. Zhou, D. Zhang, W. Yuan, Z. Wang, L. Zhou, and M. G. Bell, "Pickup and delivery problem with electric vehicles and time windows considering queues," *Transportation Research Part C: Emerging Technologies*, vol. 167, p. 104 829, 2024.

[36] W. Chen, D. Zhang, T. Van Woensel, G. Xu, and J. Guo, "Green vehicle routing using mixed fleets for cold chain distribution," *Expert Systems with Applications*, vol. 233, p. 120 979, 2023.

[37] Y. Xu, H. Zhang, L. Huang, R. Qu, and Y. Nojima, "A pareto front grid guided multi-objective evolutionary algorithm," *Applied Soft Computing*, vol. 136, p. 110 095, 2023, ISSN: 1568-4946.

[38] H. Li and A. Lim, "A metaheuristic for the pickup and delivery problem with time windows," vol. 12, Dec. 2001, pp. 160–167, ISBN: 0-7695-1417-0. DOI: `10.1109/ICTAI.2001.974461`.

[39] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009. DOI: `10.1109/TEVC.2008.925798`.

[40] D. E. Goldberg, "Alleles, loci and the traveling salesman problem," 1985. [Online]. Available: `https://api.semanticscholar.org/CorpusID:204211074`.