Journal of Optimization in Industrial Engineering

Vol.18, Issue 2, Summer & Autumn 2025, 107-137

Research Article

A template-based hybrid large neighborhood search for the consistent vehicle routing problem with time-dependent travel times

Hossein Nikdel [1], **Mohammad Mahdi Nasiri** [2,*] **Seyyed Mohammad Javad Mirzapour Al-e-Hashem** [3]

1. Department of Industrial Engineering, Alborz Campus, University of Tehran, Tehran, Iran
2. School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
3. Department of Industrial Engineering & Management Systems, Amirkabir University of Technology, Tehran, Iran

https://doi.org/10.71720/joie.2025.1210053

Received: 20 June 2025
Revised: 29 August 2025
Accepted:18 September 2025

Abstract

Customer satisfaction attracts increasing attention in competitive environments. The consistent vehicle routing problem (ConVRP), introduced in recent years, incorporates customer satisfaction into VRP. In ConVRP, vehicle routes must be designed for multiple periods, and each customer must be visited by the same driver at roughly the same time on each period. Previous ConVRP research models travel times as constant and based only on distance. This is unrealistic for urban areas, where travel times vary dynamically with factors like congestion and time of day. The time-dependent VRP (TDVRP) incorporates time-varying travel times. In this paper, the ConVRP is considered with time-dependent travel times to integrate the TDVRP and ConVRP models. A mixed-integer linear programming (MILP) model is proposed for the new problem, termed the consistent TDVRP (ConTDVRP). We extend the ConVRP benchmark instances from the literature by incorporating time-dependent travel times. The model is solved using a solver for small-scale instances. Since the new problem -an extension of the two aforementioned models- is NP-hard, we propose a template-based hybrid large neighborhood search (THLNS) algorithm that incorporates variable neighborhood search (VNS) to solve it. An iterative procedure is also presented to modify a heuristic departure-time adjustment in the literature to be used with time-dependent travel times. Computational experiments and sensitivity analysis are performed on new extended instances to evaluate the efficiency of the proposed algorithm. Three presented methods in ConVRP literature are adapted to solve ConTDVRP and the results of proposed approach compared with them for three time-dependent speed profiles on extended instances. The results demonstrate that the proposed method not only achieves consistent solutions with reduced computation time but also delivers solutions with 13.38%, 10.61%, and 35.67% lower average travel times compared to the three alternative methods. Departure-time adjustment also results in 17.48% lower average travel times and 5.91% better time consistency across all benchmark instances and speed profiles.

Keywords:
Consistent Vehicle Routing Problem;
Template;
Large Neighborhood Search;
Variable Neighborhood Search;
Time-Dependent Travel Times;
Departure-Time Adjustment

Citation:
Nikdel, H., Nasiri, M.M.. & Mirzapour Al-e-Hashem, M. J. (2025). A template-based hybrid large neighborhood search for the consistent vehicle routing problem with time-dependent travel times. *Journal of Optimization in Industrial Engineering*, 18(2), 107- 137. https://doi.org/10.71720/joie.2025.1210053

* **Corresponding Author:**
Mohammad Mahdi Nasiri
School of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
E-Mail: **mmnasiri@ut.ac.ir**

1. Introduction

The classical vehicle routing problem (VRP), studied for over 65 years, aims to minimize the total cost or distance traveled by a homogeneous fleet under given constraints. While traditionally focused on fleet efficiency, recent research has shifted toward customer-related factors like service quality and satisfaction. Businesses now prioritize consistent service delivery—often more valuable than marginal cost savings—as a key competitive advantage, especially in parcel delivery industries where reliability enhances perceived service quality. This leads to the introduction of the consistent vehicle routing problem (ConVRP) (Groër, Golden, & Wasil, 2009).

In the ConVRP framework, consistency constraints ensure that the same driver visits the same customers (named driver consistency) at roughly the same times across different planning periods (named arrival time consistency). Driver (or service provider) consistency improves service quality in home healthcare, where assigning the same personnel to patients enhances care through better communication between staff and patients, which in turn reduces service time. Similarly, in other applications like small-package delivery or transportation for the elderly and disabled people, limiting customers per driver eliminates the need to learn new routes or adjust to new customers. This increases driver productivity, service quality, and finally leads to greater customer satisfaction. In business applications—such as wholesale-to-retail distribution, reverse logistics (pickup and delivery), and restaurant supply chain food distribution—as well as personal customer services like home healthcare, consistent arrival times enable customers to plan service receptions more efficiently and foster long-term relationships with the company. This builds customer loyalty and enhances satisfaction.

This study presents the first integration of time-dependent VRP (TDVRP) with the ConVRP, which focuses on customer satisfaction. Unlike prior ConVRP research, which relied on unrealistic constant travel times, this research uses time-dependent functions to accurately reflect real-world variables like traffic congestion. This allows for precise calculation of customer arrival times, ensuring both driver and visit-time consistency to enhance practical applicability and customer satisfaction. While this approach better balances service quality and operational costs, it also increases computational complexity due to the dynamic travel time calculations.

We propose a novel hybrid approach called template-based hybrid LNS (THLNS), which integrates variable neighborhood search (VNS) into a large neighborhood search (LNS) framework, preserving the template concept. The proposed approach employs a novel search space filtering mechanism to skip the evaluation of unpromising solutions and estimates time-dependent travel times to deliver consistent satisfactory solutions with reduced computational time. We investigate the impact of modeling time-dependent travel times on ConVRP's performance metrics. Prior studies have proposed departure-time adjustment heuristics to improve arrival time consistency (measured by l-max index). We modify an existing sophisticated heuristic (Kovacs, Golden, Hartl, & Parragh, 2015) by incorporating time-dependent travel times.

The LNS metaheuristic and its adaptive counterpart, the ALNS, have been effectively used to solve numerous VRP variants, especially the ConVRP. (Voigt, 2025) classify ALNS operators using unified terminology, evaluate their performance and provide guidelines for future use. Existing ConVRP heuristic approaches have predominantly employed the template concept to maintain driver consistency and satisfy precedence principle, typically implementing LNS frameworks. While VNS has demonstrated strong performance for ConVRP in terms of solution quality (Xu & Cai, 2018). We conduct an extensive evaluation comparing our method with state-of-the-art approaches, regarding the runtime and solution quality metrics using newly developed benchmark instances. This study makes following main contributions:

- Develops a model for the new problem and proposes an efficient solution approach by integrating the most effective existing ConVRP methods.
- Incorporates time-dependent travel time estimation functions and introduces a novel search space filtering technique to significantly reduce computational requirements.
- Adapts an existing heuristic for departure-time adjustment to accommodate time-dependent travel times.

The article's structure is as follows: Section 2 presents an in-depth review of relevant ConVRP and TDVRP literature to identify research gaps. Section 3 describes the new problem along with key assumptions and notation, then formulates it as a mixed-integer linear programming (MILP) model. Section 4 presents our proposed solution framework and the modified departure-time adjustment heuristic. Section 5 presents computational experiments analyzing the approach's efficiency through comparisons with CPLEX solver solutions and existing ConVRP methods, including sensitivity analyses on generated benchmark instances. Finally, Section 6 presents concluding remarks along with suggestions for future studies.

2. Literature Review

This study bridges the ConVRP and TDVRP frameworks to enhance the practical applicability of ConVRP models. We systematically review both literatures to identify the critical research gap at their intersection.

2.1. ConVRP Related Research

As an extension of the classical VRP, the ConVRP falls into the category of NP-hard problems. Since its introduction, various versions suitable for different real-world applications have been explored in previous studies. (Kovacs, Golden, Hartl, & Parragh, 2014) conducted a systematic review of vehicle routing problems incorporating consistency considerations. Due to the problem's high complexity, researchers have proposed different approaches to derive near-optimal solutions of high quality within acceptable computation times, with most employing metaheuristics or hybrid heuristic methods. A few numbers of studies have developed specialized exact methods to determine optimal solutions for larger-scale problems more effectively (Goeke, Roberti, & Schneider, 2019; Subramanyam & Gounaris, 2016, 2018).

(Shaw, 1998) first proposed LNS algorithm to solve VRP. Since then, the LNS and its adaptive version (ALNS (Ropke & Pisinger, 2006)) has effectively solved many VRP variants specially the ConVRP. (Groër et al., 2009) developed a mixed-integer programming (MIP) model and a record-to-record heuristic (ConRTR) for the ConVRP. The key feature of their approach is enforcing the precedence principle to ensure consistent customer sequencing across days. Their two-step method involves creating template routes from "frequent customers" (those with multi-day demand) and building daily routes by adjusting these templates, adding or removing "non-frequent customers" with single-day demand. They generated new test instances from classic VRP benchmarks and evaluated their algorithm by comparing its results against a non-consistent version of the RTR method. (Tarantilis, Stavropoulou, & Repoussis, 2012) introduced a template-based tabu search (TTS) algorithm for the ConVRP. Similar to ConRTR, TTS operates at both master and daily levels: it first generates template routes, then resolves and improves daily routes. Template feasibility is evaluated by checking the feasibility of corresponding daily routes. Finally, a tabu search is applied to each daily route to further improve them through neighborhood search.(Kovacs, Parragh, & Hartl, 2014) developed a template-based adaptive large neighborhood search (TALNS) for the ConVRP. The algorithm iteratively improves an initial template using adaptively selected removal and repair operators, accepting new templates via simulated annealing. A key insight was their heuristic for optimizing depot departure times, which proved critical in avoiding significant costs when stricter service time consistency was required.(Xu & Cai, 2018) developed a VNS algorithm for the ConVRP. Their two-step method first uses a "shaking" procedure to diversify the search with a template (which may be infeasible), and then a local search to optimize it. They used three neighborhood structures (relocation, exchange, reverse) and a "near points" technique to improve efficiency by skipping unpromising operations. Infeasible templates were evaluated with penalty costs for violations. The second step was only performed on higher-quality templates to achieve feasibility and further improve the solution.

(Feillet, Garaix, Lehuédé, Péton, & Quadri, 2014) introduced "time classes" as a new measure for time consistency, grouping customers with service start times within a defined sensitivity threshold. The objective is to minimize the total number of time classes, formulated as a graph coloring problem. Their model generalizes the ConVRP (Groër et al., 2009), which corresponds to the special case of a single time class. The authors solved the model using a LNS method.(Smilowitz, Nowak, & Jiang, 2013) examined how workforce management strategies affect transportation firms' competitive positioning. The study established two workforce management metrics: (1) driver-customer consistency, quantified by repeated service encounters, and (2) driver-area consistency, measured by frequency of serving specific geographic zones. (Yu, Hu, & Wu, 2024) proposed a ConVRP framework that integrates two key objectives: 1. Workload Equity: Enforcing a maximum daily workload difference between drivers. 2. Route Consistency: Promoting the use of familiar routes by applying reduced costs for travel time on them. They developed an ALNS algorithm with new operators to solve this problem. The algorithm uses remove and repair operators to generate new solutions. For feasible solutions that meet all constraints, departure schedules are further adjusted to enhance time consistency. (Mancini, Gansterer, & Hartl, 2021) investigated a collaborative ConVRP where multiple companies share customers to maximize collective profit. Their model enforces service consistency by requiring the same company (but not necessarily the same driver) to serve a customer throughout the period. They formulated the problem with a mathematical model enhanced by valid inequalities and developed a matheuristic (MH) to solve large instances, evaluating its performance against an iterative local search (ILS) method

(Kovacs, Golden, et al., 2015) introduced the generalized ConVRP (GenConVRP), which limits the number of drivers per customer and penalizes service time variations. Departing from template-based methods, they employed a flexible LNS applied directly to all daily routes. Their algorithm also incorporated adjustable route departure times to improve arrival time consistency and used a greedy method to reduce the l-max. (Luo, Qin, Che, & Lim, 2015) studied a multi-period VRP with time windows (VRPTW) where customer visits are limited to a few vehicles. They formulated a MIP model and solved it using a three-step heuristic: first generating initial solutions via decomposition, then minimizing fleet size with a tree-search repair mechanism, and finally applying TS to reduce total travel distance.

(Kovacs, Parragh, & Hartl, 2015) later proposed a multi-objective GenConVRP (MoGenConVRP), treating routing costs, driver consistency, and arrival time consistency as conflicting objectives. They developed exact ϵ-constraint algorithms and a multi-directional large neighborhood search that combined multi-directional local search (MDLS) with LNS to solve large-scale instances. (Lian, Milburn, & Rardin, 2016) enhanced the MDLS method for the multi-objective ConVRP. Their approach integrated LNS to identify non-dominated solutions iteratively, under the assumption that all vehicles depart the depot at time zero in every period.

(Sungur, Ren, Ordóñez, Dessouky, & Zhong, 2010) addressed a stochastic courier delivery problem (CDP) with uncertain demands and service times. Using stochastic and robust optimization, their goal was to maximize coverage and efficiency rather than enforce fixed assignments. Their approach generated a master schedule for overall planning and adaptable daily plans optimized for coverage, route consistency, travel cost, and on-time delivery. They developed a two-part heuristic and a TS algorithm to solve large-scale instances. (Alvarez, Cordeau, & Jans, 2024) addressed a ConVRP with uncertain customer presence and demand. Using a two-stage stochastic model, their approach first plans routes while penalizing consistency violations. In the second stage, it minimizes actual routing and penalty costs after uncertainties are realized. They solved the problem via sample average approximation (SAA), employing exact algorithms to handle sampled scenarios iteratively.

(Subramanyam & Gounaris, 2016) proposed a branch-and-cut algorithm for the consistent traveling salesman problem (ConTSP), a special case of the ConVRP with a single uncapacitated vehicle. This was the first exact solution method for consistency-based routing problems. Their approach, which included three MIP models compared via branch-and-cut, can serve as either a component in metaheuristic hybrids or an exact decomposition method for the ConVRP. (Subramanyam & Gounaris, 2018) developed an exact method for the ConTSP that incorporates vehicle waiting times. Their approach decomposed the problem into periodic time-windowed TSPs within a branch-and-bound framework, accounting for AM/PM time windows and variable depot departure schedules.(Goeke et al., 2019) introduced the first exact solution method for the ConVRP. They found standard column generation ineffective due to weak linear relaxations caused by consistency constraints. Instead, their novel approach used column generation with variables representing a vehicle’s complete multi-period route sequence. A modified Clarke-Wright algorithm (Clarke & Wright, 1964) generated initial solutions, while LNS provided upper bounds for larger instances. Driver consistency was prioritized before addressing arrival time consistency.

(Braekers & Kovacs, 2016) studied a dial-a-ride problem (DARP) with driver consistency for specialized transit services. Their model included precedence constraints between pickup and drop-off locations. They proposed two formulations and solved them using a branch-and-cut approach enhanced with techniques to reduce model size and strengthen constraints. (Ulmer, Nowak, Mattfeld, & Kaminski, 2020) studied a dynamic, stochastic multi-period routing problem where driver-customer familiarity reduces service times after initial contact. Daily revealed demands necessitate sequential decisions modeled as a Markov decision process (MDP). The goal was to evaluate the strategic benefit of long-term driver-customer relationships. (Jost, Jungwirth, Kolisch, & Schiffels, 2022) tackled a specialized transportation problem for football players with prioritized passenger demands. They introduced an iterative, template-based heuristic to maximize demand priority and routing consistency. An ϵ -constraint mechanism balanced these objectives under fleet capacity constraints

(Zhen, Lv, Wang, Ma, & Xu, 2020) introduced a new variant combining ConVRP with the VRP with simultaneous pick-up and delivery (VRPSPD), termed ConVRPSPD (or ConVRPSDC). They formulated it as a MIP and solved medium to large instances using template-based methods: RTR travel, LNS-enhanced local search (LSVNS), and TTS. (Stavropoulou, 2022) studied a heterogeneous fleet ConVRP that jointly optimizes fleet composition and consistent routing to minimize total costs (fixed and variable) under vehicle availability constraints. A hierarchical tabu search (HTS) algorithm was used, where the upper level selects the fleet mix and the lower level employs variable neighborhood descent (VND) to optimize routes. (Stavropoulou, Repoussis, & Tarantilis, 2019) studied a VRP combining profit maximization and service consistency. The model included mandatory regular customers and optional profitable customers. Routes were designed to maximize profit under capacity, tour-length, and time consistency constraints, using an adaptive TS algorithm with long and short-term memory for effective exploration.

(Nolz, Absi, Feillet, & Seragiotto, 2022) introduced the CEVRP-BCM, integrating electric vehicle routing with backhauls, charging, and consistency. Their hybrid approach combined template-based ALNS with constraint programming for charging and quadratic optimization for pickups/deliveries. A backhaul policy mandated deliveries before pickups. The objective function penalized violations of arrival time and driver consistency, promoting equitable service. The TALNS method used worst-case demands to generate templates.

Table 1
The related literature of ConVRP

Authors	Solution Method	Objective Function	e	Travel Time	Fleet	Consistency	Constraints

	Hybrid	Metaheuristic	Heuristic	Exact	Characteristics		Constant	Time-Dependent	Heterogeneous	Homogeneous	Driver	Arrival Time	Tour-Length	Capacity
(Groër et al., 2009)	•				Minimizing total travel time in all periods		•			•	•	•	•	•
(Sungur et al., 2010)			•		Maximizing the count of served customers and minimizing travel time and lateness/earliness penalties		•				•		•	
)Tarantilis et al., 2012(	•				Minimizing total travel time in all periods		•			•	•	•	•	•
(Smilowitz et al., 2013)		•			Minimizing the total distance and maximizing the driver-to-customer and driver-to-service region familiarity		•			•	•			•
(Kovacs, Parragh, et al., 2014)	•				Minimizing total travel time in all periods	•	•			•	•	•	•	•
(Feillet et al., 2014)	•				Minimize total travel time across all periods by limiting the maximum count of time classes per customer		•			•		•		•
(Kovacs, Golden, et al., 2015)		•			Minimizing the weighted aggregation of overall travel time and l-max	•	•			•	•		•	•
(Luo et al., 2015)			•		First minimizing the utilized fleet size then minimizing the overall travel time	•	•			•	•		•	•
(Kovacs, Parragh, et al., 2015)	•			•	Minimizing the vector of overall travel time, maximum count of assigned drivers across all customers and l-max	•	•			•	•	•	•	•
(Lian et al., 2016)		•			Minimizing the vector of total distance traveled, maximum number of drivers per customer and maximum arrival time difference		•			•	•	•	•	•
(Subramanyam & Gounaris, 2016)				•	Minimizing total travel time in all periods		•					•		
(Braekers & Kovacs, 2016)		•		•	Minimizing the total travel cost in all periods	•	•			•	•		•	•
(Xu & Cai, 2018)		•			Minimizing total travel time in all periods		•			•	•	•	•	•
(Subramanyam & Gounaris, 2018)				•	Minimizing total travel time in all periods		•					•	•	
(Stavropoulou et al., 2019)		•			Maximizing the overall obtained profit minus the overall travel cost		•			•	•	•	•	•
(Goeke et al., 2019)		•		•	Minimizing total travel time in all periods		•			•	•	•	•	•
(Zhen et al., 2020)	•				Minimizing total travel time in all periods		•			•	•	•	•	•
(Ulmer et al., 2020)			•		Minimizing expected cost including service and routing costs in all periods		•				•		•	
(Mancini et al., 2021)			•		Maximizing the overall revenue minus the overall travel cost in all periods		•			•	•	•	•	•
(Jost et al.,		•			Maximizing the priorities of players then		•		•		•		•	•

2022)					minimizing total travel time in all periods									
(Stavropoulou, 2022)	•				Minimizing the total cost including vehicles fixed costs and routing cost		•		•		•	•	•	•
(Nolz et al., 2022)	•				Minimizing the total cost including fixed costs, travel time and consistency violation costs	•	•			•	•	•	•	•
(Yu et al., 2024)		•			Minimizing the overall travel time including the time discounts resulted from traveling familiar routes in all periods	•	•			•	•	•	•	•
(Alvarez et al., 2024)	•				Sum of the penalty costs of driver consistency violation and expected routing costs and penalty costs of non-serving customers in all scenarios		•			•	•			•
This study	•				Minimizing total travel time in all periods	•	•	•		•	•	•	•	•

2.2. Relevant Literature on TDVRP

The study of TDVRP has its own rich history since its initial introduction. (Gendreau, Ghiani, & Guerriero, 2015) performed the systematic review of TDVRP, establishing key classification frameworks. (Adamo, Gendreau, Ghiani, & Guerriero, 2024) later synthesized methodological advances in TDVRP (2015–2022), highlighting emerging machine learning applications and unresolved challenges in routing.

(Beasley, 1981) first presented a time-dependent travel time model with an algorithm for a two-interval planning period including distinct travel times. (Ahn & Shin, 1991) studied the time-dependent VRP with time windows (TDVRPTW), introducing the key concept of arrival time monotonicity. This property simplifies computations, enables efficient feasibility checks, and reduces the computational burden. Their work demonstrated that with this property, solving the TDVRPTW is only marginally harder than the standard VRPTW. (Malandraki & Daskin, 1992) formulated MILP models for the time-dependent TSP and VRP using step-function travel times. They proposed nearest-neighbor heuristics for both problems and a cutting-plane method for the TDTSP. Their heuristics were also adaptable to continuous travel time functions. (Hill & Benton, 1992) introduced a modeling framework employing node-based time-varying step functions for speed, with edge travel times computed from the mean speed of adjacent nodes. (Fleischmann, Gietz, & Gnutzmann, 2004) studied a static TDVRP, introducing a parametric method to smooth travel time functions under the first-in-first-out (FIFO) principle. They also proposed a route-based time window concept to derive feasibility conditions for path concatenation operations.

(Jung & Haghani, 2001) developed a genetic algorithm (GA) for a dynamic TDVRP where new demands and changing travel times occur after vehicle departure. They categorized vehicles as used or unused. (Haghani & Jung, 2005), in later work, established solution lower bounds and provided simulation results on a large network. Early models, however, violated the FIFO principle by allowing later departures to sometimes result in earlier arrivals. (Ichoua, Gendreau, & Potvin, 2003) introduced the foundational IGP speed model for TDVRPs, which guarantees FIFO compliance by using interval-specific travel speeds. Key contributions include a method (Algorithm A.1 in Appendix A) to compute FIFO-preserving travel time functions (illustrated in Fig. A.1 in Appendix A), dynamic speed adjustments at interval boundaries, and a parallel TS algorithm with an approximated evaluation for computational efficiency. Validated on Solomon benchmarks (Solomon, 1987), the IGP model has become a cornerstone in TDVRP research.

(Donati, Montemanni, Casagrande, Rizzoli, & Gambardella, 2008) solved the TDVRPTW using a multiple ant colony system (MACS). The approach employed two hierarchical colonies: ACS-VEI to minimize the number of vehicles and ACS-TIME to minimize travel time. They evaluated the algorithm on modified Solomon benchmarks (Solomon, 1987) against five speed models across four time intervals. (Hashimoto, Yagiura, & Ibaraki, 2008) solved the TDVRPTW with an iterative local search that uses dynamic programming (DP) to efficiently optimize route schedules and a filtering mechanism to prune low-potential neighborhoods. (C. Liu et al., 2020) developed an enhanced ant colony algorithm (ACA) for the same problem, incorporating congestion avoidance through modified pheromone updates to prevent congested routes. (Balseiro, Loiseau, & Ramonet, 2011) developed a hybrid ant colony optimization (ACO) for TDVRPTW that combats infeasible solutions by integrating insertion heuristics. It used three constructive heuristics for initialization and a local search phase employing Fleischmann’s route time windows (Fleischmann et al., 2004) to efficiently verify feasibility during customer sequence insertions. (Maden, Eglese, & Black, 2010) studied TDVRPTW with departure-time scheduling, using a parallel insertion method and TS algorithm, validated on a UK distribution case. (Figliozzi, 2012) provided a general framework for generating

TDVRP instances and a heuristic for hard/soft time windows, employing a generalized nearest neighbor heuristic (GNNH). (Gmira, Gendreau, Lodi, & Potvin, 2021) developed a TS method for TDVRPTW with segment-based speeds including an approximate evaluation method and time-dependent Dijkstra's algorithm, testing it on NEWLET benchmarks with up to 200 nodes. (Ticha, Absi, Feillet, & Quilliot, 2017).

(Dabia, Ropke, Van Woensel, & De Kok, 2013) introduced the first exact branch-and-price algorithm for TDVRPTW, using column generation and a labeling algorithm for the time-dependent pricing subproblem. (Soler, Albiach, & Martínez, 2009) transformed TDVRPTW into an equivalent asymmetric capacitated VRP (ACVRP) via graph reduction techniques. (Sun, Veelenturf, Dabia, & Van Woensel, 2018) studied profitable TDVRPTW with precedence constraints, developing a modified labeling algorithm and generating new benchmarks. (Sun, Veelenturf, Hewitt, & Van Woensel, 2018) in subsequent work, proposed an exact branch-and-price method for the time-dependent pickup and delivery problem with time windows (TDPDPTW) with profits.

The green vehicle routing problem (GVRP) is a recent variant of the VRP, closely related to the TDVRP. Its primary objective is to incorporate environmental aspects, such as minimizing greenhouse gas (GHG) emissions. The classic version of the GVRP assumes constant travel speeds for vehicles. (Sharafi & Bashiri, 2016) developed two MIP models for the GVRP that include social factors, such as fair workload distribution for drivers. They also proposed a genetic algorithm for large-scale problems. (Manavizadeh, Farrokhi-Asl, & WT Lim, 2020) proposed a mathematical model for the GVRP that incorporates a bi-fuel mixed fleet and refueling options using a comprehensive fuel consumption function. They linearized the model and introduced valid inequalities to calculate the fuel consumption of the bi-fuel vehicles. The model's validity was demonstrated by solving a small-scale example. (Shahrabi, Nasiri, & Al-e, 2024) proposed a sustainable VRP model integrated with cross-docking to enhance efficiency. The model minimizes costs, GHG emissions, maximum driver working hours (for social equity), and ensures high product freshness. A hybrid GA-MIP algorithm was developed for large instances, with results validated against CPLEX and a case study.

Several studies have been conducted on time-dependent GVRP (TDGVRP). (Alinaghian & Naderipour, 2016) created a detailed fuel model and solved the problem with an enhanced firefly algorithm. (Soysal & Çimen, 2017) modeled congestion and solved their problem by converting it into a TSP solved with restricted dynamic programming (RDP). (Fan, Zhang, Tian, Lv, & Fan, 2021) used trigonometric speed functions and a hybrid GA for a time-dependent problem with time windows. (Y. Liu et al., 2023) also addressed this with an ALNS heuristic featuring a time discretization search (TDS). (Ulsrud, Vandvik, Ormevik, Fagerholt, & Meisel, 2022) developed a MIP model and ALNS for weather-dependent vessel routing, allowing for unmet or delayed demand.

(Mancini, 2017) studied TDVRP without time windows, proposing a two-step heuristic method. The first step generates initial solutions using a multi-start random constructive heuristic (MRCH), then the second step includes these solutions in a set partitioning problem formula. (Huang, Zhao, Van Woensel, & Gross, 2017) introduced the TDVRP considering path flexibility (TDVRP-PF), including decisions for selecting the proper path in TDVRP. They modeled TDVRP-PF under both traffic situations with deterministic and stochastic congestion conditions.

(R. Zhang, Guo, & Wang, 2020) studied a time-dependent electric vehicle routing problem with time windows (TDEVRPTW) that includes congestion tolls for peak travel. They formulated it as a MIP model and solved it using an ALNS algorithm. (Lu, Chen, Hao, & He, 2020) studied TDEVRP, enhancing route planning for vehicles by optimizing departure schedules and speeds across all route segments using an iterative VNS (IVNS) algorithm that combines VND for node sequencing with specialized optimization for departure times and speed variables. (Xiong, Xu, Yan, Guo, & Zhang, 2024) enhanced electric vehicle routing models with drivetrain loss considerations under traffic congestion, using real-time congestion coefficients and presented an ALNS with capacity-aware initial solutions.

(Pan, Zhang, & Lim, 2021) introduced the multi-trip TDVRPTW (MTTDVRPTW). They solved it using a hybrid ALNS-VND algorithm, which featured a segment-based method to efficiently check route feasibility, (Zhao, Poon, Tan, & Zhang, 2024) presented a GA hybridized with time-dependent split algorithm (TD-SPA) for MTTDVRP. The TD-SPA was devised to split a tour into multiple routes and GA was used to generate these tours. Monotone queue optimization (MQO) was used to speed-up the TD-SPA.

(Kok, Hans, & Schutten, 2012) tested congestion protocols using Dijkstra's algorithm and an RDP heuristic. (T. Zhang, Chaovalitwongse, & Zhang, 2014) developed a hybrid ACS and TS algorithm for a time-dependent vehicle routing problem with simultaneous pickup and delivery (TDVRPSPD). (Rincon-Garcia, Waterson, Cherrett, & Salazar-Arrieta, 2020) used an LNS algorithm with a scheduling component to adhere to driving time regulations. (Cai, Lv, Xiao, & Xu, 2021) presented a linearized model for connected and automated vehicle (CAV) routing, solved with a particle swarm optimization (PSO) enhanced by VNS. (Jie, Liu, & Sun, 2022) incorporated stochastic factors (such as weather and traffic conditions) into TDVRP with soft time windows, solving it with a hybrid sweep algorithm and improved PSO (IPSO). (Zhou, Li, Bian, & Zhang, 2024) introduced two-echelon TDVRP with simultaneous pickup-delivery and satellite synchronization (2E-TDVRPSPDSS), solved by a memetic algorithm (MA) featuring self-adaptive operators and specialized local search.

Table 2
An overview of the most significant literature of TDVRP

Authors	Solution Method				Objective Function Characteristics	Model		Waiting at Customers' Location		Fleet		Constraints		
	Hybrid	Heuristic	Metaheuristic	Exact		Dynamic	Static	Not-Allowed	Allowed	Heterogeneous	Homogeneous	Tour-Length	Capacity	Time Windows
(Malandraki & Daskin, 1992)		•			Minimizing total travel time		•		•	•			•	•
(Ichoua et al., 2003)			•		Minimizing the weighted aggregation of the overall travel time and delay time in serving all customers	•	•		•			•		•
(Fleischmann et al., 2004)		•			Minimizing overall travel time and the fraction of customers with time window violations		•		•			•		•
(Haghani & Jung, 2005)			•		Minimizing the total cost including vehicle fixed costs, routing costs and penalty cost of time window violations	•			•	•		•	•	•
(Soler et al., 2009)		•			Minimizing total travel time including transportation and waiting times		•		•		•	•	•	•
(Figliozzi, 2012)	•				First minimizing the route number then minimizing the overall travel time		•		•		•	•	•	•
(Dabia et al., 2013)				•	Minimizing overall travel time		•		•		•	•	•	•
(Alinaghian & Naderipour, 2016)			•		Minimizing fuel consumption		•	•			•	•	•	
(Sun, Veelenturf, Dabia, et al., 2018)		•		•	Maximizing the earned profit minus the overall travel time		•		•			•	•	•
(Sun, Veelenturf, Hewitt, et al., 2018)				•	Maximizing the earned profit minus the sum of travel time and fixed costs		•		•		•	•	•	•
(Pan et al., 2021)	•				Minimizing total distance traveled		•		•		•	•	•	•
(Cai et al., 2021)	•				Minimizing fuel consumption and greenhouse gas generation		•		•		•		•	•
(Gmira et al., 2021)			•		Minimizing total travel time including transportation,		•		•		•	•	•	•

					waiting and service times								
(Fan et al., 2021)	●				Minimizing the total cost including fixed costs, fuel consumption and penalty costs of time windows violations		●		●		●	●	●	●
(Ulsrud et al., 2022)			●		Minimizing the total cost including the vessels travel costs plus the cost of renting new vessels and penalty costs of back-ordering some demands		●			●			●	
(Jie et al., 2022)	●				Minimizing total cost including distance traveled costs, fixed costs and penalty costs of time window violations		●		●	●		●	●	●
(Y. Liu et al., 2023)			●		Minimizing greenhouse gas generations		●		●		●	●	●	●
(Zhao et al., 2024)	●				Minimizing the sum of fixed cost and overall travel time costs		●	●			●	●	●	
(Xiong et al., 2024)			●		Minimizing fuel consumption		●	●			●	●	●	
(Zhou et al., 2024)	●				Minimizing the sum of fixed costs, routing costs, loading, inventory and fuel consumption costs		●		●		●	●	●	●

As shown in Table 1, all existing ConVRP research assumes constant travel times, and none has considered time-dependent travel times in their models. Additionally, Table 2 reveals that the existing TDVRP literature has not yet explored consistency considerations for customers. This indicates a significant research gap in the ConVRP literature, where all studies assume constant travel times—an assumption far removed from real-world applications. Moreover, most ConVRP studies do not allow flexible depot departure times—a feature demonstrated in the literature to enhance arrival time consistency. To address this gap, we incorporate both time-dependent travel times and flexible vehicle departure times into the ConVRP model, with constant travel times becoming a special case of the newly proposed model.

3. Problem Description and Mathematical Modeling

In this section, we extend the ConVRP by assuming time-dependent travel times instead of distance or constant times used in all previous studies of ConVRP. We call the new problem as consistent time-dependent vehicle routing problem (ConTDVRP). The ConTDVRP is characterized as follows. We have a fleet of at most k identical vehicles positioned at a single depot, each with a fixed capacity Q. These vehicles must depart from and return to the depot after completing their routes. The problem spans d days (or periods) with each customer requiring service on a specific pre-determined day(s) and can be served at most once per day from any vehicle. All routes must finish by time T. The service time and demand for each customer on their requested day are known in advance with specific values. The same driver services each customer at approximately the same time each day throughout the planning period, ensuring the maximum difference between the latest and earliest arrival times (called l-max) never exceeds the permitted maximum L. In this problem, initial vehicle departure from the depot is synchronized to occur at time zero, (i.e., from the start of working day) and vehicles are prohibited from idle waiting at customer sites. The travel time between any two locations is derived from time-dependent speed profiles which is a piece-wise linear function. The model's objective function seeks to minimize total travel time (including travel times and service times) across all routes in all days of the planning period. One application of this problem is providing services to disabled and elderly people because consistency between service providers and customers (driver consistency) is important in these cases. Additionally, consistency in service times for these customers (arrival time consistency) must be maintained, which is why arrival time consistency is defined as a constraint in the model. The other assumptions of modelling the new problem are described in details as follows:

- All demands in all periods must be fully met.
- The number of available vehicles is unlimited (matching the total count of customers in the proposed model).
- There are no time windows for customer demands.
- The speed profiles between two nodes are time-dependent and defined as stepwise functions for every pair of nodes in the entire network.

• The continuous piece-wise linear travel time functions associated with each pair of nodes are derived from the time-dependent stepwise speed functions; therefore, the FIFO property always holds.
• The fastest routes connecting all nodes are precomputed for the entire network and do not change.
• While traveling, a vehicle's speed changes along the remaining edge distance when the time interval of the speed profile changes.
• All customers have delivery demands.

Notation
The following notation is used in the mathematical modeling of ConTDVRP:

Sets		**Description**
$D = \{1,2, \dots\dots, d\}$		set of planning periods.
$K = \{1,2, \dots\dots, k\}$		set of available vehicles in the fleet.
$M_{ij} = \{1,2, \dots\dots, m_{ij}\}$	$\forall i,j \in N$	set of time slots in the travel time function for edge (i,j).
$N = \{0,1,2, \dots\dots, n\}$		set of customer nodes plus depot. (where depot is node 0).
$N' = \{1,2, \dots\dots, n\}$		set of customers.
Parameters		
bp_{ijm}	$\forall i,j \in N,\ \forall m \in M_{ij}$	breakpoint of time slot m in the travel time function for edge (i,j).
d		count of planning periods.
k		count of available vehicles.
L		maximum arrival time difference across all customers.
M'		a big positive value.
m_{ij}	$\forall i,j \in N$	the count of breakpoints in the travel time function for edge (i,j).
n		count of customers.
Q		maximum capacity of each vehicle.
q_{id}	$\forall i \in N', \forall d \in D$	demand of customer i in period d.
s_{id}	$\forall i \in N', \forall d \in D$	service time of customer i in period d.
T		latest allowed return time to depot (maximum tour-length).
$w_{id} \in \{0,1\}$	$\forall i \in N', \forall d \in D$	1 if customer i requests demand in period d, 0 otherwise.
θ_{ijdm}	$\forall i,j \in N, \forall d \in D, \forall m \in M_{ij}$	gradient of the travel time function for edge (i,j) in time slot m of period d.
ω_{ijdm}	$\forall i,j \in N, \forall d \in D, \forall m \in M_{ij}$	intercept of the travel time function for edge (i,j) in time slot m of period d.
Decision Variables		
t_{ijdmk}	$\forall i,j \in N,\ \forall d \in D,\ \forall m \in M_{ij},\ \forall k \in K$	departure time of vehicle k from customer i to customer j in time slot m of period d.
t_{ikd}	$\forall i \in N,\ \forall k \in K,\ \forall d \in D$	departure time of vehicle k from customer i in period d.
$x_{ijdmk} \in \{0,1\}$	$\forall i,j \in N,\ \forall d \in D,\ \forall m \in M_{ij}, \forall k \in K$	1 if vehicle k travels from i to j in time slot m of period d, 0 otherwise.
$z_{ikd} \in \{0,1\}$	$\forall i \in N,\ \forall k \in K,\ \forall d \in D$	1 if vehicle k visits customer i in period d equals 1, 0 otherwise.

MILP Model

$$OF = Min \sum_{i=0}^{n} \sum_{j=0}^{n} \sum_{k=1}^{k} \sum_{d=1}^{d} \sum_{m=1}^{m_{ij}} \theta_{ijdm} * t_{ijdmk} + \omega_{ijdm} * x_{ijdmk} \tag{1}$$

$$z_{0kd} = 1 \qquad \forall k \in K,\ \forall d \in D \tag{2}$$

$$t_{0kd} = 0 \qquad \forall k \in K,\ \forall d \in D \tag{3}$$

$$\sum_{k=1}^{k} z_{ikd} = w_{id} \qquad \forall i \in N', \forall d \in D \tag{4}$$

$$\sum_{i=1}^{n} q_{id} * z_{ikd} \leq Q \qquad \forall k \in K,\ \forall d \in D \tag{5}$$

$$\sum_{i=0}^{n} \sum_{m=1}^{m_{ij}} x_{ijdmk} = \sum_{i=0}^{n} \sum_{m=1}^{m_{ji}} x_{jidmk} = z_{jkd} \qquad \forall j \in N, \forall k \in K,\ \forall d \in D \tag{6}$$

$$w_{id} + w_{id'} - 2 \leq z_{ikd} - z_{ikd'} \leq -(w_{id} + w_{id'} - 2) \qquad \forall i \in N',\ \forall k \in K,\ \forall d,d' \in D | d \neq d' \tag{7}$$

$$t_{ikd} = \sum_{j=0}^{n} \sum_{m=1}^{m_{ij}} t_{ijdmk} \qquad \forall i \in N',\ \forall k \in K,\ \forall d \in D \tag{8}$$

$$t_{ikd} + \theta_{ijdm} * t_{ikd} + \omega_{ijdm} - (1 - x_{ijdmk}) * M' \leq t_{jkd} - s_{jd} \qquad \forall i \in N,\ \forall j \in N',\ \forall m \in M_{ij},\ \forall k \in K,\ \forall d \in D \tag{9}$$

$$t_{ikd} + \theta_{ijdm} * t_{ikd} + \omega_{ijdm} + (1 - x_{ijdmk}) * M' \geq t_{jkd} - s_{jd} \qquad \forall i \in N,\ \forall j \in N',\ \forall m \in M_{ij},\ \forall k \in K,\ \forall d \in D \tag{10}$$

$$t_{ikd} + (\sum_{m=1}^{m_{ij}} (\theta_{i0dm} * t_{ikd} + \omega_{i0dm} * x_{i0dmk})) * w_{id} \leq T * w_{id} \qquad \forall i \in N',\ \forall k \in K,\ \forall d \in D \tag{11}$$

$$t_{ikd} + (\sum_{m=1}^{m_{ij}} (\theta_{i0dm} * t_{ikd} + \omega_{i0dm} * x_{i0dmk})) * w_{id} \geq 0 \qquad \forall i \in N',\ \forall k \in K,\ \forall d \in D \tag{12}$$

$$-L + T * (w_{id} + w_{id'} - 2) \leq t_{ikd} - t_{ikd'} \leq L - T * (w_{id} + w_{id'} - 2) \qquad \forall i \in N',\ \forall k \in K,\ \forall d,d' \in D | d \neq d' \tag{13}$$

$$bp_{ijm-1} * x_{ijdmk} \le t_{ijdmk} \le bp_{ijm} * x_{ijdmk} \quad \forall i,j \in N,\ \forall\ 2 \le m \le M_{ij},\ \forall k \in K,\ \forall d \in D \qquad (14)$$

$$0 \le t_{ijd1k} \le bp_{ij1} * x_{ijd1k} \quad \forall i,j \in N,\ \forall k \in K,\ \forall d \in D \qquad (15)$$

$$t_{ikd} \ge 0 \quad \forall i \in N,\ \forall k \in K,\ \forall d \in D \qquad (16)$$

$$t_{ijdmk} \ge 0 \quad \forall i,j \in N,\ \forall m \in M_{ij}, \forall k \in K,\ \forall d \in D \qquad (17)$$

$$x_{ijdmk} \in \{0,1\} \quad \forall i,j \in N,\ \forall m \in M_{ij}, \forall k \in K,\ \forall d \in D \qquad (18)$$

$$z_{ikd} \in \{0,1\} \quad \forall i \in N,\ \forall k \in K,\ \forall d \in D \qquad (19)$$

Eq. 1 defines the objective function, which minimizes the overall travel time for all vehicles across every day. Eq. 2, 3 ensure that every vehicle must visit the depot each day and start their routes at time zero. Eq. 4 ensures each customer is visited by only one vehicle on all days requiring service and Eq. 5 enforces the vehicle capacity constraint, limiting loads to Q units for each vehicle per day. Eq. 6, specifies that each visited customer must have only one predecessor and one successor. Eq. 7 maintains driver consistency, requiring the same driver for each customer across all service days. Eq. 8 defines the departure time of the given vehicle visiting each customer on requested day. Eq. 9, 10 determine the departure times of successive customers visited on each route and prohibit waiting at customer locations. Eq. 9 also serves as sub-tour elimination constraints in the individual daily routes. Eq. 10 could be eliminated to permit drivers to wait at a location before proceeding to the next customer. The vehicle tour-length limit is defined by Eq. 11, 12. Eq. 13 bounds the arrival time consistency, limiting the maximum difference between visit times for every customer across any pair of days to L time units. Eq. 14, 15 constrain departure times between consecutive customers to fall within the appropriate time slot of the piecewise travel time function. The decision variable domains are defined in Eq. 16 to Eq. 19.

4. Solution Methodology

We design a hybrid metaheuristic that embeds a local search stage in a templated-based LNS algorithm. Similar to (Xu & Cai, 2018) our template consists of all customers who demand service in one period or more. First, an initial template generation procedure generates a feasible template meeting both capacity and tour-length limitations. Arrival time consistency isn't considered in generating initial template. Then, the LNS algorithm starts with the initially generated template, and a pair of remove-repair operators is selected randomly. The number of removed customers is selected randomly from the predefined interval in each iteration of LNS. The selected pair of remove and repair operators destroys the template by removing the specified number of customers from the template and re-inserting them into the partial template routes respectively, creating a new template solution. In the repair operators, feasibility checking for template routes considers only artificial capacity limit. Because verifying tour-length feasibility with time-dependent travel times is computationally expensive, and because the resulting template might still produce infeasible daily routes after resolution, we omit this check during the template phase. Also artificial capacity concept is defined similar to (Kovacs, Parragh, et al., 2014) to resolve and obtain daily solutions from generated templates in shorter time. The artificial capacity limit Q_a is chosen randomly within the range [Q, Q_a -UB], where Q represents the actual capacity and Q_a -UB is the predefined upper bound. Q_a-UB is calculated as the sum of maximum customer demand across all periods divided by the maximum number of routes required based on capacities among all periods. Eq. 20 illustrates how Q_a-UB is computed.

$$Q_a - UB = \sum_{i=1}^{n} max_{\forall d \in D}\{q_{id}\} / max_{\forall d \in D}\left\{\sum_{i=1}^{n} \frac{q_{id}}{Q}\right\} \qquad (20)$$

Then, the sum of all travel times and the maximum divergence in arrival times (l_max) are approximated for the new solution, and simulated annealing determines whether to accept it. If accepted, the approximate l_max is verified against the threshold (i.e. l_max_constant * L). If below the threshold, the template is resolved into daily routes to verify feasibility against actual capacity and tour-length constraints. If template is feasible and approximate l-max is lower than L, then l-max is calculated exactly for resolved daily routes to check arrival time consistency. Else, if the routes are marginally feasible, the general improvement procedure executes to simultaneously reduce l_max and travel times. Else general repair procedures (including load- and time-repair operators) are performed to convert the infeasible daily routes to feasible ones. Subsequently, if the feasible solution satisfies arrival time consistency, its travel time is evaluated for potential selection as the best-found solution; otherwise, general improvement is applied to enhance the feasible solution satisfying all constraints. At this time, the first stage of generating a new template and choosing an accepted template as the appropriate one to repair and performing improvement is finished and the second stage is started. The second stage includes performing a local search procedure on the new template obtained from chosen repair operator of LNS. Local search is done for specific neighborhood structures of a template considering special requirements for decreasing neighborhood search space and computational times. As the obtained solution of local search may not satisfy the constraints on vehicle capacity and tour-length, the repair and general improvement procedures is performed similar to the first stage to first, convert the obtained solution to a feasible one and then improve it. Local search and general improvement procedures are described below in details. We refer to the proposed template-based hybrid LNS approach as THLNS.

Algorithm 1. Pseudo code of the proposed THLNS

```
Input: l_max_constant, l_max_penalty, ol_penalty, ot_penalty, FC, n_e_s, $p_{worst}$, $Q_a$-UB, travel time functions, initial
feasible template, l_max_penalty_accept, local_search_counter_limit, repair and remove operators, $w_{t^\wedge}$, c
Output: best_solution, best_found_objective, l_max of best solution
local_search_counter = 0
iteration = 0
current template = initial feasible template
while local_search_counter <= local_search_counter_limit:
    if iteration % 50 = = 0 and local_search_counter = = 0:
        l_max_constant += 0.2
    iteration += 1
    if iteration > 2000:
        choose q randomly from [min (0.2* $n$, 30),  min (0.4* $n$, 60)]
    else:
        choose q randomly from [min (0.1* $n$, 30),  min (0.2* $n$, 60)]
    choose y randomly from [0,1]
    $Q_a$ = Q + y * ($Q_a$-UB – Q)
    select a pair of remove and repair operators randomly
    apply selected pair on current template to obtain a candidate template
    resolve candidate template and calculate objective and l_max approximately
    if l_max > L:
        cand objective with penalty = candidate objective + (l_max – L) * l_max_penalty_accept
    else:
        cand objective with penalty = candidate objective
    if (accept && l_max <= l_max_constant * L) || (iteration > 2000 && l_max <= l_max_constant * L):
        check feasibility status of the candidate template
        if l_max <= L and status = = “feasible”:
            calculate new objective and l_max exactly
            update best_feasible_objective
        else:
            perform First Stage on the candidate template
        perform Local Search (The Second Stage) on the candidate template to obtain a new template
        resolve the new template and check feasibility status
        perform First Stage on the new template
end while
return best_solution, best_found_objective, l_max of best solution
```

4.1. LNS Components

4.1.1. Initial template generation

An initial feasible template is generated with respect to capacity and tour-length constraints. First, all customers are unassigned at the empty solution. Initial template generation takes the empty solution and uses a greedy heuristic method for sequentially adding customers in feasible positions with a minimum added travel time across all routes. Each position is checked for feasibility by resolving the template to daily routes and checking feasibility on each day. Then, the increase in travel time for each position is computed for each day and added to obtain total inserting cost for each position. The daily routes are feasible, if they satisfy load capacity and tour-length limits. Arrival time consistency is not considered in initial template generation. When no feasible placement exists in current routes, a new empty route is added to assign other customers. The template generation is complete when all customers are assigned to a template route.

4.1.2. Remove and repair operators

We used three remove and five repair operators as follows. Removal and repair operators include random, worst, and related for removals and greedy, and regret (four versions) for repairs. Appendix B explains each operator in details.

4.1.3. Acceptance criterion

The algorithm employs simulated annealing to determine if a new template τ producing solution s should replace the existing incumbent template τ. The new template τ is adopted when its associated solution yields a better (lower) objective value $f(s')$ than the current incumbent solution, $f(s)$. We calculate the *l_max* of candidate template's corresponding solution approximately. If *l_max* > *L*, then the penalty is also added to candidate objective and $f(s')$ (candidate objective with penalty) is computed as follows:

$$f(s') = candidate\ objective + (l\_max - L) * l\_\max\_penalty\_accept \qquad (21)$$

l_max_penalty_accept in Eq. 21 is the penalty assigned to each unit of arrival time consistency violation and defined as follows:

$$\left(\frac{\sum_{i=1}^{n}\sum_{d=1}^{d} w_{id}}{D*n}\right) * \left(\frac{f(s)}{\max\{1, l\_max - L\}}\right) / \left(\max_{\forall d\in D}\left\{\frac{MST_{cost} + \sum_{i=1}^{n} s_{id} + \min_{\forall i\in N'}\{distance[0,i]\}}{T}\right\} + \max_{\forall d\in D}\left\{\sum_{i=1}^{n}\frac{q_{id}}{Q}\right\}\right) \qquad (22)$$

In Eq. 22 *f(s)* and *l_max* were calculated for initial feasible template exactly. The MST_{cost} is the cost of minimum spanning tree of the customers requesting demand on the given day. *Distance [0, i]* is the Euclidean distance of depot to customer *i*. In Eq. 22, *l_max_penalty_accept* is calculated according to characteristics of each problem instance. Thus, it is computed for each problem independently and is used as input parameter for the solution approach. This way, we reduce the number of parameters needed to be tuned before running algorithm. It is defined in a way that the more difficult to satisfy arrival time consistency in the problem the less acceptance penalty will be assigned for violation.
The algorithm permits acceptance of worse solutions with probability $e^{\frac{-(f(s')-f(s))}{t^{\wedge}}}$ where $t^{\wedge}$ represents the current temperature which is initially set to:

$$t^{\wedge} = -\left(\frac{w_{t^{\wedge}}}{\ln 0.5}\right) * f(S) \qquad (23)$$

We configure $t^{\wedge}$ such that solutions worse by $w_{t^{\wedge}}$ % have a 50% acceptance probability, with $w_{t^{\wedge}}$ being a tunable parameter. $f(S)$ is the initial feasible template objective which is computed approximately. The geometric cooling is applied to decrease the temperature, expressed as $t^{\wedge} = t^{\wedge} * c$, where c is the cooling rate parameter. We used the same values for parameters as the same applied in (Kovacs, Parragh, et al., 2014) ($w_{t^{\wedge}}$=0.01, *c*=0.9999).

4.1.4. Selection and stopping criterion

During every iteration, the algorithm randomly chooses a pair of removal and repair operators. The THLNS terminates when reaching the predefined local search iteration limit (local_search_iteration_limit).

4.2. The first stage

Algorithm 2. Details of the first stage used in algorithm 1

```
if status = = “feasible”:
    perform General Improvement on candidate template and obtain new solution
    compute l_max and objective of new solution exactly
    if l_max of new solution <= L:
        update best feasible objective
else:
    perform Load and Tour-Length Time Repairs procedure to obtain new feasible template
    resolve new feasible template to obtain new solution
    compute l_max and objective of new solution exactly
    if l_max <= L:
        update best feasible objective
    else:
        perform General Improvement on candidate template and obtain new solution
        compute l_max and objective of new solution exactly
        if l_max of new solution <= L:
            update best feasible objective
```

4.2.1. Load and Tour-Length Time Repairs

First, we repair the template to make it load-feasible. For routes violating their capacity limits, we shift candidate points to other routes while preserving load-feasibility constraints. All relocations of a point must maintain route (driver) consistency across all days, choosing insertion positions that minimize overall travel time throughout the planning horizon. Travel times are computed using estimated travel time functions. The shift must not worsen the new route's overload. The procedure favors minimal template perturbation, because the current template has been accepted as new incumbent template in LNS or locally optimized by the local search stage. Thus, we prioritize: (1) single-shift repairs to execute relocations that resolve route violations through single-shift corrections and (2) shifts minimizing total travel time. If no appropriate points are found, we implement the shift yielding the minimal total cost:

$$Total\ cost = total\ travel\ time + ol\_penalty * \ total\ overload \qquad (24)$$

Then, the repair process of this route is complete. Total overload is determined by summing daily overloads across all routes. *ol_penalty* represents the overload penalty factor. Again, total travel times are computed using estimated travel time functions. We then seek to find points capable of resolving the route through one more relocation. Iterations continue until achieving load-feasibility. When no valid shifts exist, the solution creates a new empty route to be replicated daily. Time feasibility is then addressed through a process comparable to load repair: for each route violating time constraints, carefully selected points are shifted to other routes without creating new capacity violations or increasing overtime on the destination route.

Here, the total cost is computed similarly as follows:

$$Total\ cost = total\ travel\ time + ot\_penalty * \ total\ overtime \qquad (25)$$

We approximate overtime using estimated travel times for each route, then total overtime is obtained by adding overtime of all routes over all days. Total travel time is also computed the same as load repair and *ot_penalty* is the penalty coefficient per overtime.

4.2.2. General improvement

As mentioned before, the general improvement procedure is performed on solutions that satisfy both capacity and tour-length limitations. The main structure is the same as the local search procedure, i.e., three operations introduced above are applied during every iteration and the solution minimizing the total cost across three neighborhoods is determined, but there are some differences. First, three operations are applied only on the same route to maintain the driver consistency of the current solution. Moreover, feasibility checking is performed with respect to tour-length constraints and only feasible movements are evaluated to obtain an improved feasible solution. It is worth noting that because operations are only applied on the same route, capacity constraints cannot be violated and thus need not be checked. Thus, total cost change is only consists of travel time change of the operation in the selected day and the change of total time difference excess. Also, no filtering is done and all feasible movements of operations are evaluated. Another difference is that operations are performed on daily routes and are compared among all routes of all periods. Finally, as indicated in (Kovacs, Parragh, et al., 2014), the reverse operation is restricted to apply on sequences with maximum length of three nodes here to avoid large increases in l-max and also to help solve larger instances in relatively reduced computational time. All other details of general improvement are the same as local search.

4.3. Local search (the second stage)

We use three well-known neighborhood search operators to define the neighborhood search structures. Each of these structures is searched to improve the total cost of the given template solution which is used as the lowest cost in each iteration and is initialized with C_0. (Xu & Cai, 2018) used a near concept to restrict the neighborhood search structures. In this concept if an operator could not create connections between a predefined fraction of near customers, this operator would be skipped and not be evaluated (Xu & Cai, 2018). The near concept was only based on distances between customers. As we study the problem under time-dependent conditions here, we apply a different filtering method to restrict the search space for each operator.

4.3.1. Filtering mechanism

Estimation methods were proposed in (Gmira et al., 2021; Ichoua et al., 2003) to approximately evaluate neighboring solutions for solving the TDVRPTW using the TS algorithm. In (Ichoua et al., 2003), interpolation was used to approximate the travel time of new solutions in the neighborhood. In the approximate evaluation used in (Gmira et al., 2021), the delay in the departure time of the subsequent node (which is affected by the operation) is calculated and multiplied by its penalty value. This penalty is itself derived from the delay propagated to the next node when the current node's departure is delayed by one unit. We employ this delay concept as a filtering mechanism for each operation. In Fig. 1, nodes 5 and 0 represent the subsequent affected nodes along routes a and b, respectively, after applying the reverse operation. The following filtering condition is applied:

$$Total\ departure\ time\ delay\ of\ the\ subsequent\ affected\ nodes \geq FC * sum\ of\ the\ early\ departure\ times\ of\ these\ nodes \qquad (26)$$

Each node's departure time delay is the difference between its departure time after and before applying the operation. By summing the delays at nodes 5 and 0, we obtain the total delay. Early departure times mean the departure times of these two nodes in the current solution before applying the operation. If the inequality defined in Eq. 26 holds, then this reverse operation move will be skipped and will not be evaluated. Otherwise, the cost change caused by the operation is evaluated. In this way, we skip the operations without significant potential to improve the solution and reduce the computational time. We realized that FC = -0.1 would be a good value to sufficiently reduce the search space for solving large instances in reasonable times.

We compute changes in: (1) travel time (C_t), (2) overtime (C_{ot}), and (3) overload (C_{ol}) on each day to obtain the cost change. However, because of time-varying travel times, the calculation of C_t and C_{ot} is not straightforward as done in (Xu & Cai, 2018). Whereas (Xu & Cai, 2018) used distance-based calculations, we compute changes in time-dependent travel times per operation on each day. Estimated travel time functions are computed to approximately calculate the change in travel times. These estimated functions obtain the travel time with the distance between nodes divided by the relevant speed of the time interval in which the departure time is positioned. Summing C_t across all days yields C_{tt}..The change of overtime for each day (C_{ot}) is also calculated based on C_t of each affected route and then total overtime change (C_{tot}) is calculated similarly. Calculating the change in total overload (C_{tol}) is computationally inexpensive and direct, providing a cost change without change in total time difference excess (C_{cwTotd}) as C_{cwTotd} = C_{tt} + *ol_penalty* * C_{tol} + *ot_penalty* * C_{tot} = *total cost change* (T_{cc}) – *l_max_penalty* * C_{totd}, in which C_{totd} denotes the change in total time difference violation. *Ol_penalty*, *ot_penalty*, and *l_max_penalty* are penalty factors associated to load, tour-length and time difference violations. We need to determine C_{totd} to calculate T_{cc}. Given the time-intensive nature of C_{totd} computation, we instead compute *operation_cost* = C_0 – *l_max_penalty* * *t_otd_initial* + C_{cwTotd}, in which *t_otd_initial* and C_0 are the total time difference violation and total cost of current template respectively. If *operation_ cost* ≥ C_0, since the template's total time difference violation remains non-negative after the operation, the new template's cost cannot be lower than C_0. Therefore, C_{totd} computation is unnecessary. However, if this condition isn't met, we compute C_{totd} to determine T_{cc}. It is worth to note that C_{totd} is also computed using estimated travel times to keep the computational time as short as possible. As noted in (Gmira et al., 2021; Ichoua et al., 2003) a predefined number of better solutions regarding approximated evaluations were kept and then the total cost of these solutions were computed exactly and the best solution was obtained. The input parameter showing the number of better solutions to keep for exact evaluation is indicated as *n_e_s* in our algorithm. In early stages of designing our algorithm we realized that *n_e_s* = 30 seems appropriate to keep the operations exact enough and simultaneously decreasing computational time. Three neighborhood structures are generated by three operators namely, relocation, exchange and reverse operators. Each of these operators can be performed within a single route or across multiple routes. The relocation operation moves a point to a different position, while the exchange operator interchanges the positions of two points, finally, Reverse operator reverses parts of the selected routes. In order to prevent increasing in reverse operations of local searches in larger instances, the reverse is restricted to apply on parts with maximum length of eight nodes. This value showed satisfying results in our preliminary stages of designing algorithm regarding computational efficiency and solution optimality. During every local search iteration, these three neighborhood structures of a template are searched through implementation of the three operations. In order to more decrease the computational time of local search in solving large instances, the local search iteration count is limited to eight iterations before finding an arrival-time consistent solution and if a consistent solution is found, this limit will decrease to three iterations. The lowest-cost template solution found across all three neighborhoods is determined. If the total cost of this template is lower than the current template and iteration limit is not reached, the search continues with new template as the current template. Otherwise, local search is complete. Fig. 1 shows how the three operations are applied on template routes

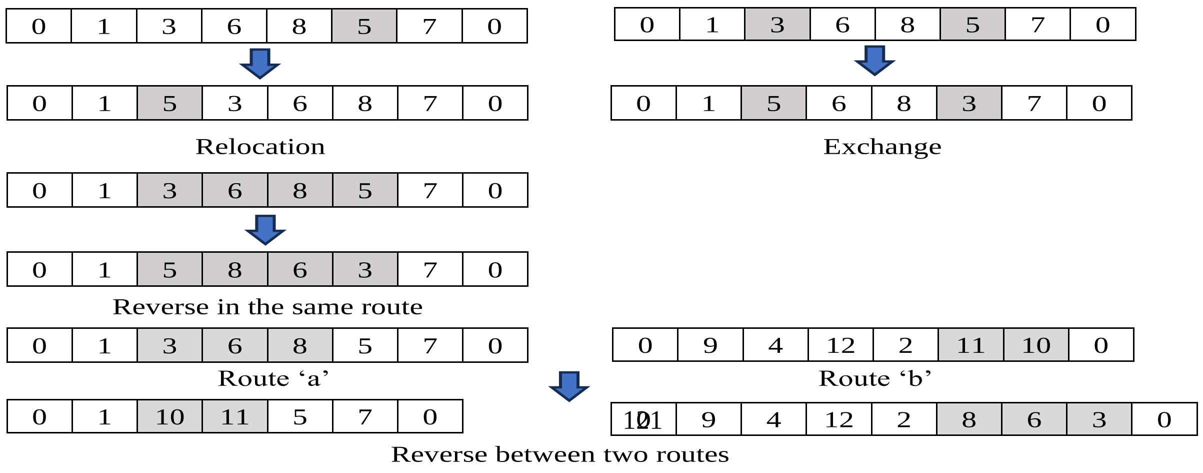


Fig 1. Three types of operations used in Local Search

4.4. Departure-time adjustment

(Kovacs, Parragh, et al., 2014) proposed the exact and heuristic methods for adjusting the departure times of vehicles from depot. They indicated that delaying vehicle departure times from depot can considerably improve arrival time consistency without changing in total travel times. (Kovacs, Golden, et al., 2015) embedded a new sophisticated heuristic for adjusting the departure times in their proposed LNS. These approaches are presented for ConVRP in which travel times are assumed to have constant values not dependent on departure times. Thus, delaying departures from the depot doesn't have any impact on travel times. The heuristic approach presented in (Kovacs, Golden, et al., 2015) cannot be used directly for adjusting departure times in ConTDVRP, so we modify and extend this heuristic approach to be used in our problem. Algorithm A.2 in appendix A indicates the pseudocode of their approach. (More details of their algorithm are explained in (Kovacs, Golden, et al., 2015)). According to the heuristic approach, the maximum push forward *pf (j,k)* and pull backward *pb (j,k)* of customer *j*'s route on the relevant day(s) is determined in relation to all other customers k sharing that route. In ConTDVRP, delaying departure time of a route from depot affects travel times between customers of the route. So, the computation cannot be straightforwardly done as in (Kovacs, Golden, et al., 2015), because delaying departure times of a route from depot may not lead to the same delay in arrival times of customers visited in the route. Thus, we modify their algorithm by proposing an iterative method for computation of *pf (j,k)* and *pb(j,k)* values in ConVRP with time-dependent travel times. Algorithm 3 shows the pseudocode of our proposed iterative approach. We implement an enhanced version of the THLNS approach in which when the obtained solution fails to satisfy arrival time consistency constraint, the extended heuristic for adjusting departure times is executed to check if it can improve l-max to find feasible solution. In the next section, we solve each problem instance by two versions of the proposed THLNS

Algorithm 3. Pseudocode of the proposed iterative approach to compute *pf (j,k)* in Algorithm A.2

```
Input: customers j and k, arrival times of all customers in days, solution routes, day, customer j's route on the given day, current delayed departure of routes in all days, speed profile (SP) with corresponding breakpoints (bp), α
Output: delay value in departure time (dt) of customer j's route on the given day
if customer j's route on the given day is existed in current delayed departure of routes:
    candidate dt of customer j's route on the given day = its former dt + (last bp of SP- its former dt)/2
else:
    candidate dt of customer j's route on the given day = last bp of SP/2
candidate dt list = empty list
add candidate dt to candidate dt list
while True:
    set all departure times to zero
    set departure time of customer j's route on the given day to the candidate dt
    if customer j's route is feasible after setting its departure time on the given day:
        compute arrival times of customers j and k in all days
        compute the arrival time difference of customers j and k (ATD_of_j, ATD_of_k)
        if ATD_of_j < ATD_of_k:
            if || candidate dt list || <=1
                candidate dt = candidate dt – (candidate dt – former dt) / 2
                add candidate dt to candidate dt list
            elseif absolute (candidate dt – the last but one element of candidate dt list) > α:
                candidate dt=candidate dt -absolute (candidate dt – a last but one element of candidate dt list) /2
                add candidate dt to candidate dt list
            else:
                break
        else:
            if || candidate dt list || <=1
                candidate dt = candidate dt + (last bp of SP – candidate dt) / 2
                add candidate dt to candidate dt list
            elseif absolute (candidate dt – the last but one element of candidate dt list) > α:
                candidate dt=candidate dt +absolute (a last but one element of candidate dt list -candidate dt) /2
                add candidate dt to candidate dt list
            else:
                break
    else:
        if || candidate dt list || <=1
```

```
            candidate dt = candidate dt – (candidate dt – former departure time) / 2
            add candidate dt to candidate dt list
        **elseif** abs (candidate dt – the last but one element of candidate dt list) > α:
            candidate dt = candidate dt – absolute (candidate dt – the last but one element of candidate dt list) /2
            add candidate dt to candidate dt list
        **else**:
            **break**
**end while**
**return** absolute (candidate dt – former dt of customer j's route on the given day)
```

We found that α = 0.01 and ϵ = 0.01 are proper values to converge fast and exactly. The calculation of *pb (j,k)* can be done similarly using analogous logic. According to (Kovacs, Golden, et al., 2015), calculation of *pb (j,k)* is done when pushing forward is not possible. Then, the previously pushed departure times become candidates to see if they can be pulled backward. Tour-length feasibility remains guaranteed because: the former feasible departure times will be decreased in this case and also the FIFO property holds on, and it is not needed to check the feasibility after setting the departure time on the given day.

5. Computational Experiments

In this section, time-dependent travel time functions were computed in MATLAB 2020, while the proposed approach and all benchmark algorithms were implemented in Python 3.11. All experiments were conducted on a system featuring an Intel Core i7 2.6 GHz CPU and 16 GB of RAM. We extend the ConVRP benchmark instances introduced by (Groër et al., 2009) and generate new instances to analyze the efficiency of the presented approach in solving ConTDVRP. They created a straightforward method for randomly producing a ConVRP benchmark derived from the classical VRP benchmarks, generating a unified five-day ConVRP benchmark from problems 1-12 with *p = 0.7* daily service probability. In generating these problems, they defined the travel times (in minutes) between any two customers as identical to their Euclidean distance. (Groër et al., 2009) reported l-max of each problem obtained by their proposed algorithm. Then, later research used these l-max values as the arrival time difference limit (L) in solving these instances. Table B.1 of Appendix B represents characteristics of the ConVRP large instances. We developed their benchmark instances by introducing three speed profiles. These profiles are derived from (Figliozzi, 2012) which for the first time tried to define standard TDVRPTW instances that can be used repetitively to evaluate other TDVRPTW solution approaches. These benchmark instances incorporated two peak congestion intervals within the depot's operating hours. The depot operational time (T) was partitioned into five equal time intervals, with specific travel speeds defined for each interval. Since our problem lacks time windows, we adopt their three speed profiles which presented for TDVRPTW with soft time windows because in this case vehicle would travel the same distance with the same average speed as in the classical VRP instances until depot closing time with speed variability in periods. Therefore, these profiles can be used to generate ConTDVRP instances without time windows. Similar to (Figliozzi, 2012), the depot operational time (i.e. [0,T]) is partitioned into five equal intervals and the associated travel speeds are defined for each profile as follows:

$$\begin{aligned} Profile\_1 &= [1.1, 0.85, 1.1, 0.85,\ 1.1] \\ Profile\_2 &= [1.2,\ 0.8,\ 1,\ \ 0.8,\ 1.2] \\ Profile\_3 &= [1.2,\ 0.7,\ 1.2,\ 0.7,\ 1.2] \end{aligned} \tag{27}$$

For example, for the tenth problem of ConVRP instances with T=200, the working time is divided to the five equal periods as [0, 40), [40, 80), [80, 120), [120, 160), [160, 200]. Therefore, three instances are generated regarding the tenth ConVRP instance and above three profiles. Finally, we have 36 instances of ConTDVRP by extending 12 ConVRP instances. Similar to former ConVRP research, we assumed the l-max values reported by (Groër et al., 2009) as l-max limit (L) values. The ch-3 problem with profile 2 is chosen as the middle instance to perform design experiments. We tune four parameters including *l-max-penalty*, *ot-penalty*, *ol-penalty* and *l-max-constant* by Taguchi method. Three levels are tested per parameter and L-9 array is used to perform 9 experiments with proposed levels. Each experiment is repeated five runs on ch-3 with profile 2. The response variable is computed by Eq. 28 and defined such that the relative percentage violation of l-max in compare to L is added to the best-found travel time as penalty.

$$Response = Travel\ time * (1 + l\_\max\_violation/L) \tag{28}$$

Table B.2 of Appendix B shows the results of tunning parameters and Figs. B.1.a-B.1.b display the main effect plots for Means and SN ratios respectively. Then the best levels of parameters are determined based on SN ratios as follows: *l-max-penalty* = 75, *ot-penalty* = 10, *ol-penalty* = 10 and *l-max-constant* = 1.75. These parameter settings are used in performing all numerical experiments of this section. Totally, 12 instances are solved with proposed THLNS for each speed profile and solving each instance is repeated five runs. Table 3 shows the obtained results. In this table ch-1 to ch-12 indicates 12 benchmark instances of ConVRP. There are five columns reported for each speed profile. Avg.TT and Avg l-max indicates averages of the travel time (in minutes) and l-max of each instance over five runs. Similarly, Min TT and Min l-max shows minimum travel time and minimum l-max of each problem among five runs respectively. Avg CPU time represents the average processor runtime (in seconds) for solving each instance. As shown in the Avg. TT column of Table 3, the average travel time across all instances remains nearly unchanged between Profile 1 (6,978.7) and Profile 2 (6,930.5), with a marginal decrease of 0.69%. However, it increases significantly by 4.98% from Profile 2 (6,930.5) to Profile 3 (7,275.45). Average l-max increases by about 5.39% from profiles 1 to 2 and decreases only 1.22% from profiles 2 to 3. The average CPU time increases considerably from profiles 1 to 2 and 2 to 3 (3.78% and 9.1% respectively). it indicates that obtaining solutions with satisfying l-max becomes difficult by increasing the speed variation between periods. (from profiles 1 to 2) and obtaining solutions with less travel time becomes difficult by more increasing in speed variation (from profiles 2 to 3). Also, the CPU runtime of the proposed algorithm increases with greater variations in the speed profile.

Table 3
Results for ConTDVRP instances solved by THLNS without departure-time adjustment

ID	Profile 1					Profile 2					Profile 3				
	Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time	Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time	Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time
Ch-1	2560.33	30.19	2514.29	23.04	256.07	2587.35	29.01	2330.31	23.63	275.16	2607.39	30.22	2364.03	25.62	310.70
Ch-2	4231.41	29.45	4081.86	28	529.61	4277.20	31.92	4131.51	30.44	549.12	4195.17	47.77	3996.67	29.83	857.01
Ch-3	4180.08	22.50	4099.3	22.21	2198.47	4074.68	30.95	3938.46	22.37	3198.40	4380.86	27.32	4330.75	21.54	3159.52
Ch-4	5934.86	29.59	5825.97	26.44	8789.10	6134.98	26.92	5811	26.79	7793.79	6427.19	27.01	6141.46	26.38	9295.81
Ch-5	8710.42	25.71	8370.79	24.04	16020.71	8036.07	25.04	7634.67	24.13	19765.97	8990.23	24.48	8641.11	23.49	18564.67
Ch-6	4564.68	60.48	4497.72	56.94	90.17	4799.10	55.71	4694.14	48.68	117.95	5147.91	55.46	5054.69	53.87	120.29
Ch-7	7621.47	77.43	7349.64	72.5	291.50	7872.98	77.77	7613.68	72.47	322.75	8036.68	71.72	7784.81	68.24	280.48
Ch-8	8409.34	69.96	8324.01	65.05	931.68	7979.71	72.19	7533.1	71.76	1103.91	8406.35	69.53	8240.35	68.32	927.72
Ch-9	11937.39	74.29	11812.1	66.7	2562.50	11947.33	81.90	11609.58	72.53	2212.37	12405.31	92.60	12098.53	74.6	3569.95
Ch-10	14924.51	78.52	14864.02	70.01	18868.65	15070.18	85.02	14733.72	79.02	16994.68	15623.62	68.86	15321.01	60.09	20545.73
Ch-11	5965.49	15.36	5677.58	14.67	4614.21	5871.00	17.50	5623.39	15.65	5585.79	6482.56	17.68	6203.2	15.61	5118.72
Ch-12	4704.39	18.41	4691.65	17.04	2708.59	4515.39	26.62	4362.67	11.78	2129.74	4602.07	21.08	4356.71	17.05	2762.57
Average	6978.7	44.32	6,842.41	40.55	4,821.77	6,930.5	46.71	6,668.02	41.6	5004.14	7,275.45	46.14	7044.44	40.39	5,459.43

We also solve above instances by including extended heuristic for adjusting departure times in the proposed approach. The obtained results are represented in Tables 4-6.

Table 4
Obtained results of proposed THLNS with departure-time adjustment for profile 1

ID	Profile 1					(k, d) = departure time of route k in day d (in minutes after working start time) and departure times of all remaining routes of other days equals zero (i.e. the starting time of depot).
	Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time	
Ch-1	2067.65	25.17	1925.2	20.65	283.16	(5,2) = 14.59, (4,2) = 4.9, (1,1) = 11.53, (1,3) = 11.53
Ch-2	3602.25	31.17	3586.21	27.51	585.08	(10,3) = 34.61, (3,4) = 23.2, (4,1) = 0.2
Ch-3	3207.14	20.32	3179.08	18.08	1952.31	(2,2) = 9.89, (7,1) = 5.13, (2,3) = 9.93, (4,3) = 0.67, (6,3) = 2.04
Ch-4	4922.25	26.28	4661.18	25.05	5990.06	(1,2) = 16.23, (11,5) = 8.29, (4,3) = 7.77, (8,4) = 2.55
Ch-5	6616.72	23.85	6362.17	22.58	15846.19	(4,4) = 4.13, (2,1) = 9.23, (11,2) = 2.97
Ch-6	3945.99	60.06	3800.1	58.87	103.59	(3,5) = 25.45, (1,3) = 2.67, (6,4) = 27.26
Ch-7	6791.98	68.48	6277.42	62.93	275.75	(1,5) = 22.8
Ch-8	6879.16	68.62	6810.64	63.42	910.92	(9,5) = 3.7

Ch-9	10899.54	92.06	9472.26	73.61	2459.83	(3,4) = 108.51, (12,2) = 57.12, (12,4) = 45.72, (12,5) = 57.12, (5,1) = 48.79 (5,2) = 26.53, (5,3) = 71.64, (5,5) = 48.79, (9,1) = 1.09, (9,2) = 28.6 (9,3) = 41, (9,5) = 75.18
Ch-10	12262.19	51.71	12026.84	49.19	16464.46	(17,5) = 22.6, (1,3) = 15.45, (11,3) = 23.94, (8,1) = 30.73, (12,2) = 23.21 , (9,5) = 13.31
Ch-11	4712.95	15.54	4570.08	14.93	4524.55	(6,5) = 2.22, (2,4) = 7.21, (1,1) = 6.13, (4,2) = 5.04
Ch-12	3676.80	24.84	3441.66	17.46	2038.83	(3,2) = 6.55, (2,4) = 0.49
Average	5,798.72	42.34	5,509.4	37.86	4,286.23	

Table 5
Obtained results of proposed THLNS with departure-time adjustment for profile 2

ID	Profile 2					(k, d) = departure time of route k in day d (in minutes after working start time) and departure times of all remaining routes of other days equals zero. (i.e. the starting time of depot).
	Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time	
Ch-1	2284.17	31.50	2064.85	23.81	274.79	(3,1) = 3.74, (2,2) = 3.41, (1,5) = 6.34, (4,5) = 7.62
Ch-2	3429.78	30.39	3303.21	24.19	628.42	(3,4) = 21.2, (7,2) = 3.23, (8,1) = 5.38, (4,2) = 18.11
Ch-3	3413.94	36.02	3311.27	22.09	3016.03	(5,1) = 5.27, (4,3) = 1.2, (3,2) = 2.42, (1,2) = 17.28, (2,1) = 1.4
Ch-4	4983.21	26.20	4652.63	25.81	6712.92	(11,5) = 13.46
Ch-5	6677.46	24.11	6339.33	22.69	16593.30	(18,1) = 13.89, (3,3) = 20.12, (11,2) = 2.28, (14,2) = 9.54
Ch-6	3900.32	50.86	3792.55	49.98	129.15	(5,5) = 16.57, (4,4) = 10.05
Ch-7	6295.96	69.11	6244.67	62.53	330.78	(6,4) = 19.27, (10,5) = 23.43, (1,3) = 28.96, (1,5) = 50.44
Ch-8	6825.32	65.56	6710.29	61.75	1144.77	(7,5) = 28.01, (1,1) = 2.06
Ch-9	10447.45	96.70	9435.13	79.52	2599.83	(9,4) = 5.76
Ch-10	12623.81	53.00	12260.82	50.65	12611.39	(14,5) = 33.48, (17,4) = 35.68, (11,2) = 32.29, (6,1) = 19.61, (10,3) = 20.11 (2,3) = 45.27, (16,1) = 23.32, (9,1) = 20.93, (8,5) = 24.3, (12,3) = 2.58 (2,1) = 2.61, (12,1) = 42.83
Ch-11	5251.63	26.10	4775.68	13.95	3826.13	(1,2) = 11.4, (5,2) = 5.33, (2,2) = 5.18, (4,2) = 7.4, (4,4) = 6.49
Ch-12	3674.09	16.01	3376.92	14.45	2018.39	(4,5) = 0.69
Average	5,817.26	43.8	5,522.28	37.62	4,157.16	

Table 6
Obtained results of proposed THLNS with departure-time adjustment for profile 3

ID	Profile 3					(k, d) = departure time of route k in day d (in minutes after working start time) and departure times of all remaining routes of other days equals zero. (i.e. the starting time of depot).
	Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time	
Ch-1	2099.07	26.38	1976.7	23.98	398.24	(1,1) = 13.36, (4,5) = 3.86, (4,1) = 2.86, (3,4) = 5.69
Ch-2	3532.49	45.35	3206.66	25.94	796.13	(7,2) = 1.7
Ch-3	3318.93	29.77	3271.9	20.44	3394.12	(7,4) = 2.41, (4,3) = 11.83, (2,4) = 6.9, (4,4) = 12.85, (3,2) = 3.83
Ch-4	4952.92	25.74	4825.05	25.42	8198.00	(4,2) = 15.68, (8,4) = 7.46, (9,5) = 4.24
Ch-5	6659.22	22.22	6119.33	19.39	17225.56	(17,5) = 13.93, (2,2) = 7.71, (14,2) = 10.65
Ch-6	3983.76	57.25	3833.09	51.78	136.45	(2,5) = 34.23, (1,2) = 6.24
Ch-7	6753.92	69.28	6685.31	65.93	234.39	(10,4) = 131.25, (3,2) = 81.69, (5,3) = 75.23, (12,2) = 8.53, (12,3) = 10.07, (12,4) = 10.07, (12,5) = 10.07
Ch-8	7001.64	59.81	6916.04	55.24	1082.88	(6,3) = 43.75, (5,1) = 15.39, (2,5) = 56.37
Ch-9	10341.50	73.69	9916.52	63.24	4162.14	(9,4) = 36.94, (13,2) = 36.66, (10,3) = 15.36
Ch-10	12794.15	56.98	12697.02	54.35	15393.22	(16,4) = 20.59, (4,3) = 7.1, (11,2) = 1.23, (19,5) = 12.41
Ch-11	5354.50	22.88	4743.74	15.01	4962.14	(2,5) = 10.65, (5,5) = 8.21, (3,4) = 0.52
Ch-12	3585.17	25.85	3374.15	13.77	2226.94	(5,5) = 13.76, (7,2) = 8.87, (6,2) = 5.55, (1,4) = 3.33, (2,5) = 8.31
Average	5864.77	42.93	5630.46	36.21	4850.85	

As shown in Tables 3-6 the average travel time of all instances decreases for all profiles by including departure time adjustment in proposed approach. The average l-max and average CPU time of all instances also decreases for all profiles. The average travel time of profiles 1 to 3 between all instances decreases by 16.9%, 16.06%, and

19.39% respectively compared to the baseline THLNS without adjusting departure times. It seems an interesting result, because although adjusting departure times are performed to decrease only l-max value without considering travel times, but it simultaneously achieves significant reductions in travel times for all defined speed profiles. Similarly, the average l-max decreases by 4.47%, 6.23%, 6.96% respectively which shows that although the departure-time adjustment is not performed for l-max-feasible solutions, it already can decrease the average l-max of best-found solutions. Also, there is one instance for each profile in which THLNS without departure time adjustment cannot find arrival time consistent solution among five runs of the algorithm (i.e. Min l-max > L), but the proposed approach including departure-time adjustment can find arrival time consistent solutions of all instances for each defined profile. Average CPU times of all instances decreases by 11.1%, 16.93%, 11.15% for profiles 1-3. It is an expected result, because by adjusting departure times, algorithm can find consistent solution earlier and then iteration count limit of local search in the algorithm reduces from 8 to 3 which can greatly reduce the computational time of local search.

All instances are also solved with constant travel times in which the speed equals one. Similarly, we perform five runs per instance. The results are collected in Table B.3 of Appendix B. The results demonstrate that the algorithm have found consistent solution for instances with tight time-consistency constraints (small L values) and constant speed profile which may not find consistent solutions in time-dependent profiles for them. Figure 2 compares the results for each instance across the constant profile and three other profiles. The results show that the constant profile generally yields lower average travel time, l-max, and CPU runtime.

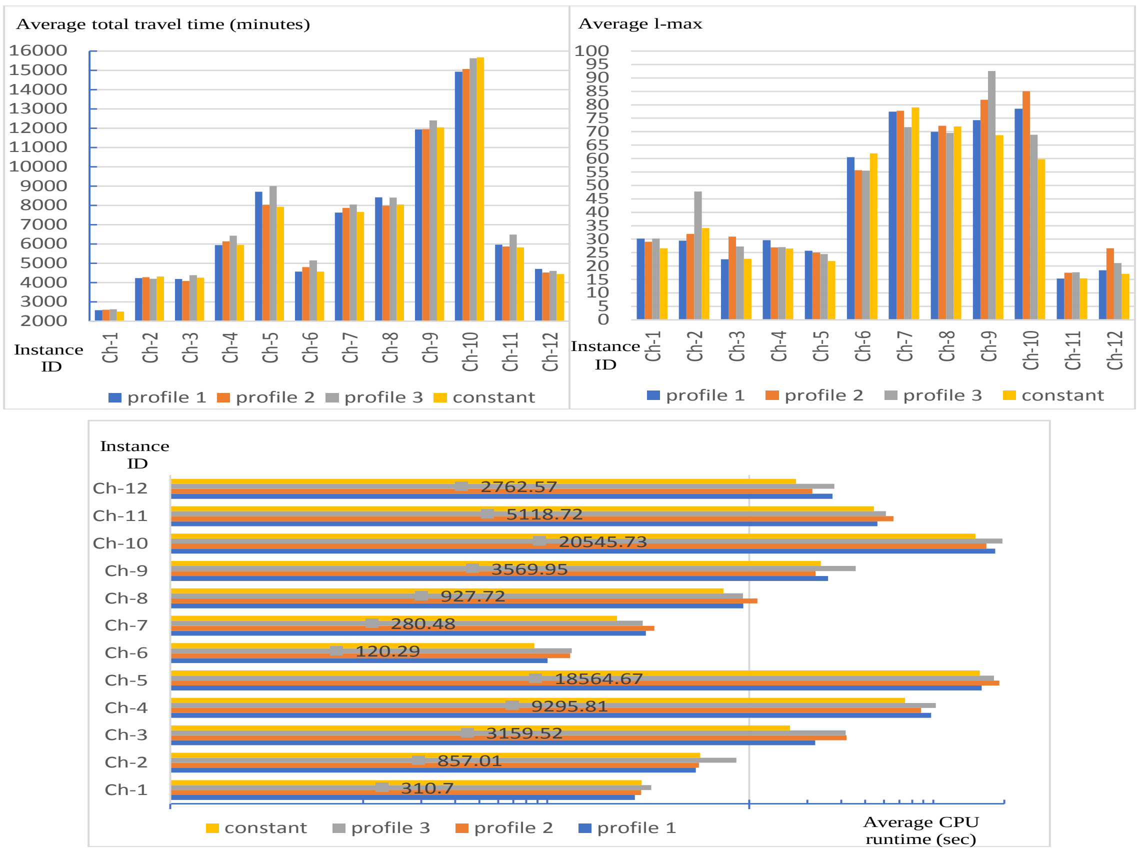


Fig. 2. Comparison of the THLNS results between profiles

Small instances were presented in (Groër et al., 2009) to solve ConVRP to optimality. Similarly, we extend these small problems with predefined speed profiles for evaluating the MIP model. Table B.4 of Appendix B indicates the detailed information of these small instances. Finding optimal solutions for these instances may take many days. To address this, we supposed the above extended problems as our medium-sized instances and created five smaller ConVRP instances (indicated as 6-1 to 6-5) by retaining only the first half nodes of instances 12-1 to 12-5 while keeping all other parameters unchanged. Table 7 compares the optimal solutions for new small instances (under Profiles 1–3) with the results of five runs from THLNS. Here, TT* represents the optimal travel times for each profile, while the Avg gap measures the percentage variation across Avg TT and TT*.

Table 7
Comparing the results of THLNS without departure-time adjustment to CPLEX optimal solutions for new small instances

ID	Profile Type	THLNS					CPLEX		Avg gap (%)
		Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time (sec)	TT*	CPU Time (sec)	
6-1	Profile_1	101.65	4.69	100.6	3.47	0.75	87.97	585.86	15.55
	Profile_2	107.45	3.34	107.45	3.34	0.11	89.63	327.47	19.88
	Profile_3	105.62	3.48	105.62	3.48	0.69	88.74	423.83	19.02
6-2	Profile_1	58.4	4.76	58.4	4.76	0.05	50.397	9.19	15.88
	Profile_2	58.75	4.53	58.75	4.53	0.05	50.755	2.11	15.75
	Profile_3	58.8	4.76	58.8	4.76	0.07	50.798	19.64	15.75
6-3	Profile_1	90.3	4.89	90.3	4.89	0.09	76.303	310.41	18.34
	Profile_2	92.22	4.75	90.93	3.91	0.75	78.136	679.25	18.02
	Profile_3	94.76	4.71	94.76	4.71	0.12	78.529	516.3	20.67
6-4	Profile_1	136.06	0.85	136.06	0.85	0.07	115.554	2450.02	17.75
	Profile_2	137.98	0.89	137.98	0.89	0.07	117.757	524.56	17.17
	Profile_3	136.77	0.66	136.77	0.66	0.07	116.849	243.64	17.05
6-5	Profile_1	84.34	2.31	84.34	2.31	0.1	70.338	1801.47	19.91
	Profile_2	86.57	2.00	85.9	0.93	0.15	71.903	363.14	20.4
	Profile_3	89.42	1.01	85.99	0.58	0.16	71.986	2312.95	24.22
Average		95.94	3.175	95.51	2.938	0.22	81.043	704.656	18.58

Table 8 demonstrates the numerical results of running medium instances with THLNS for profiles 1-3. The MIP model of these instances is implemented by CPLEX solver. The maximum allowed runtime is configured as 3600 seconds for all instances. The gap column measures the relative difference between Avg TT and Upper Bound if available.

Table 8
Comparison of THLNS without departure-time adjustment and CPLEX on medium instances

ID	Profile Type	THLNS					CPLEX			Avg gap (%)
		Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time (sec)	Upper Bound	Lower Bound	CPU Time (sec)	
10-1	Profile_1	134.8027	2.8449	132.0457	2.5986	0.304	173.2276	69.2414	3600	-22.18
	Profile_2	121.4004	3.1904	109.7841	2.6533	0.352	156.478	65.9568	3600	-22.42
	Profile_3	127.7065	3.9018	124.4387	2.6333	0.326	166.3702	65.69	3600	-23.24
10-2	Profile_1	133.2339	2.1935	128.8691	1.6192	0.328	NA	59.7555	3600	NA
	Profile_2	126.2172	3.8888	126.2172	3.8888	0.524	140.1082	54.584	3600	-9.91
	Profile_3	123.0142	3.8292	113.6534	2.6925	0.548	159.0368	49.6378	3600	-22.65
10-3	Profile_1	144.7637	3.2168	144.5736	2.0669	0.444	132.4553	60.6938	3600	9.29
	Profile_2	139.1806	4.4981	138.7734	4.0838	0.626	184.3733	57.1707	3600	-24.51
	Profile_3	141.4415	4.0518	138.6834	2.6675	0.45	171.6666	55.634	3600	-17.61
10-4	Profile_1	146.3757	3.8678	145.2353	3.8678	0.304	193.3243	69.137	3600	-24.28
	Profile_2	142.3056	4.4032	140.0943	4.4032	0.338	186.4076	63.5727	3600	-23.66
	Profile_3	143.4217	3.4040	140.1200	1.7143	0.284	167.9169	64.7899	3600	-14.59
10-5	Profile_1	133.8742	2.7648	129.2655	0.6190	0.446	134.1546	66.1048	3600	-0.21
	Profile_2	129.4069	3.6138	121.9890	1.6550	1.24	149.5773	64.5501	3600	-13.48
	Profile_3	126.4124	3.3760	124.3068	1.5796	1.102	140.2731	59.5455	3600	-9.88
12-1	Profile_1	165.3900	3.6287	161.8618	3.3329	0.814	NA	82.4684	3600	NA
	Profile_2	155.5854	4.1498	144.4618	3.9031	0.64	363.8623	73.7669	3600	-57.24
	Profile_3	162.2788	4.0004	155.7395	3.5272	0.604	NA	72.1691	3600	NA
12-2	Profile_1	125.0532	3.4863	123.0339	2.1266	0.426	130.067	55.3939	3600	-3.85
	Profile_2	113.4390	3.1311	107.8172	2.7000	0.74	140.6775	49.9921	3600	-19.36
	Profile_3	116.8819	4.4716	116.6014	3.7408	1.386	NA	47.9792	3600	NA
12-3	Profile_1	155.1394	2.5853	149.1467	1.0809	0.626	181.3129	55.0591	3600	-14.44
	Profile_2	136.9676	4.4560	128.9389	3.4452	2.718	NA	50.0187	3600	NA
	Profile_3	138.7748	2.8330	134.4852	2.5428	0.958	198.6782	48.7897	3600	-30.15
12-4	Profile_1	169.0764	3.4706	165.2609	2.8511	0.664	NA	69.4519	3600	NA
	Profile_2	168.2808	4.4154	154.8830	4.0701	1.37	180.526	64.1517	3600	-6.78
	Profile_3	156.316	4.5511	155.0342	4.2365	3.132	246.2437	63.6587	3600	-36.52
12-5	Profile_1	141.5695	3.6904	138.8055	3.0832	1.156	NA	54.77	3600	NA
	Profile_2	137.3213	3.5858	131.9158	3.0775	1.072	NA	54.8066	3600	NA
	Profile_3	133.2511	3.6156	125.4192	2.5375	1.088	NA	49.5064	3600	NA

We adapted the three approaches presented in literature for ConVRP including TALNS (Kovacs, Parragh, et al., 2014), LNS (Kovacs, Golden, et al., 2015) and VNS (Xu & Cai, 2018) via replacing the constant travel times by piece-wise linear travel time functions and solved the extended large instances for profiles 1 to 3. All of these approaches are implemented using their tuned parameter values from the main work. Each instance is solved five times and average observed l-max and travel time values are reported. Table 9 compares the performance between these approaches for profile 3. To standardize comparisons, we use the average CPU time from Table 3 (Profile 3) as the runtime limit for all algorithms per instance. (Similarly, Tables B.5 and B.6 in Appendix B compare these results for profiles 1 and 2.) Since VNS does not consider departure-time adjustment, and since the departure-time adjustment methods of the other two approaches cannot be applied directly to ConTDVRP, we compare the base versions of THLNS, TALNS (Kovacs, Parragh, et al., 2014), and LNS (Kovacs, Golden, et al., 2015) (all without departure-time adjustment) for a fair comparison. An approach is superior if it yields a consistent solution with the lowest Avg TT (where Avg l-max ≤ L) or the lowest values for both metrics. Superior results are bolded; otherwise, the best-performing metric is highlighted in bold. THLNS outperforms others in five instances, TALNS in four, and neither dominates in Ch-1, Ch-3, or Ch-11. (Table B.6 shows the same result for Profile 2, while Table B.5 demonstrates the superiority of THLNS in seven instances for Profile 1.) For Profile 3, the proposed method yielded 11.82%, 12.21%, and 34.52% lower average travel times than TALNS, LNS, and VNS, respectively. The corresponding improvements for Profiles 1 and 2 were also significant: 11.83%, 9.43%, 35.91% and 16.48%, 10.18%, 36.57%. Thus, THLNS performs best under identical computational limits.

Table 9
Comparison of THLNS (without departure-time adjustment) with three literature approaches for large instances (Profile 3)

ID	THLNS		TALNS (Kovacs, Parragh, et al., 2014)		LNS (Kovacs, Golden, et al., 2015)		VNS (Xu & Cai, 2018)		l-max Limit (L)	CPU Runtime Limit (sec)
	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max		
Ch-1	2607.39	**30.22**	3127.84	64.02	**2522.70**	76.60	3773.79	118.57	24.38	310.70
Ch-2	**4195.17**	**47.77**	5992.32	87.24	5740.27	196.75	6621.35	108.55	34.26	857.01
Ch-3	4380.86	**27.32**	5460.48	72.87	**3530.80**	106.99	6505.84	88.60	22.87	3159.52
Ch-4	**6427.19**	**27.01**	7241.14	51.51	8645.66	260.79	10323.26	80.35	27.53	9295.81
Ch-5	**8990.23**	**24.48**	9822.75	56.82	11254.77	253.73	13168.70	77.33	26.93	18564.67
Ch-6	5147.91	55.46	**4361.78**	**63.36**	5493.52	162.23	6546.80	87.68	63.47	120.29
Ch-7	8036.68	71.72	**7294.19**	**74.04**	8510.25	127.84	10192.87	60.60	83.96	280.48
Ch-8	**8406.35**	**69.53**	9653.05	70.92	8478.89	170.75	12790.37	128.73	73.04	927.72
Ch-9	12405.31	92.60	**12378.57**	**64.45**	13164.54	172.11	19040.19	104.61	106.43	3569.95
Ch-10	15623.62	68.86	**16489.42**	**58.92**	17089	172.25	24603.10	115.26	60.17	20545.73
Ch-11	6482.56	**17.68**	7312.16	163.45	**5313.46**	52.75	7312.16	163.45	16.1	5118.72
Ch-12	**4602.07**	**21.08**	6729.97	88.75	6544.39	256.04	8228.81	78.79	17.58	2762.57
Average	7044.44	40.39	7988.64	76.36	8024.02	167.40	10758.94	101.04	46.39	5459.43

To examine the impact of varying L parameters on the performances of THLNS with and without departure time adjustment, we selected instance ch-3 and increased the L values by 0% to 80% with 20% step, then each algorithm variant executed three runs per L value and profile. It should be noted that because the algorithm may find a consistent solution earlier by increasing L values, we don't decrease the iteration count limit of local search after finding the first consistent solution and the local search continues with 8 iterations to the end of approach. This approach neutralizes the effect of decreasing L values on the obtained travel time and CPU times with respect to local search iterations and we have a fair comparison. Figs.3 and 4 show the obtained results of above changing on the proposed approach without and with departure time adjustment respectively. We selected ch-1 to analyze the effects of increasing local_search_counter_limit on solution metrics of each profile. This limit was increased from 10 to 90 counts. Fig. 5 indicates the obtained results. The FC parameter range which impacts on filtering neighborhood search space of a template was varied from -0.1 to 0.3 for ch-1 and ch-8 with profile 3. Results are shown in Figs. 6 and 7.

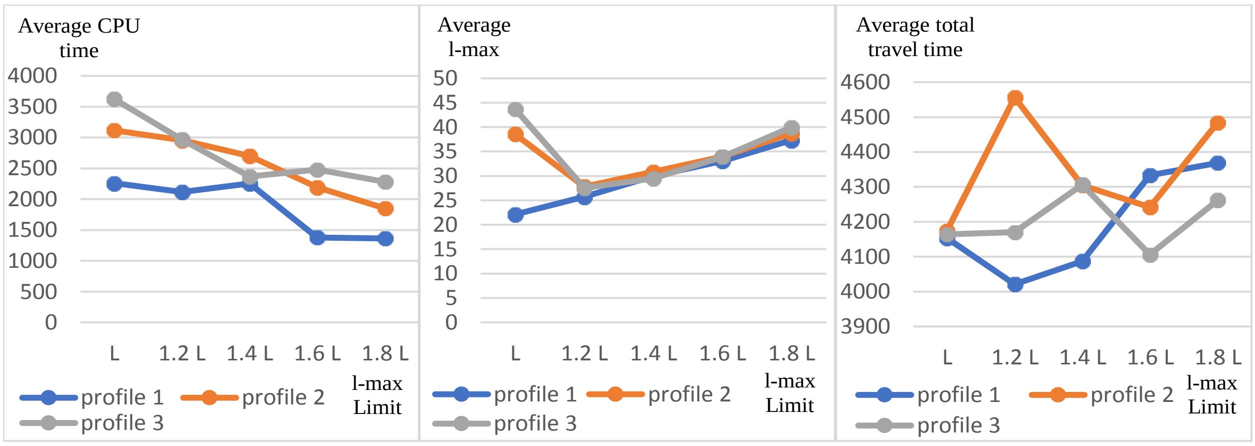


Fig. 3. Effect of changing L on results of ch-3 without departure-time adjustment

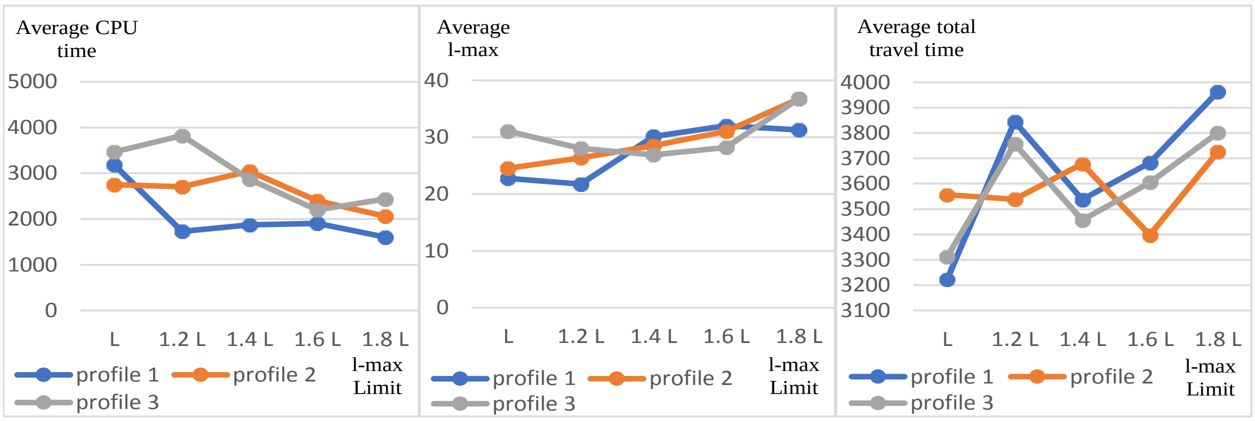


Fig. 4. Effect of changing L on results of ch-3 with departure-time adjustment

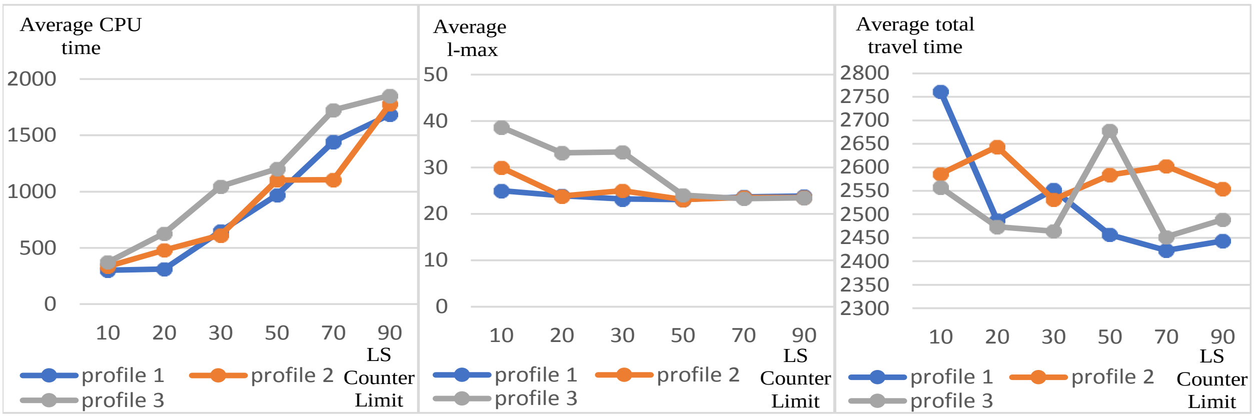


Fig. 5. Effect of increasing local_search_counter_limit on results of ch-1 without departure-time adjustment

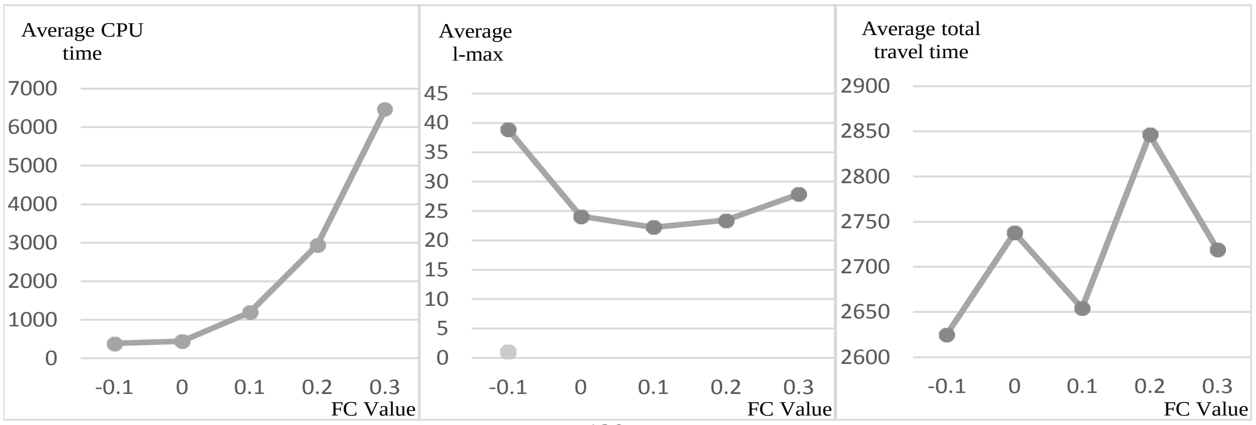


Fig. 6. Effect of increasing FC parameter on results of ch-1 for profile 3 without departure-time adjustment

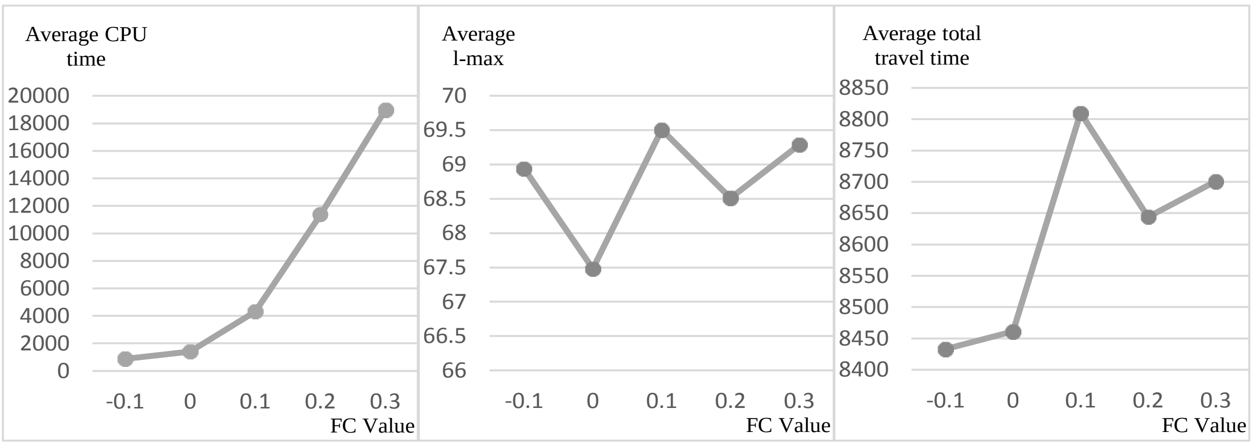


Fig. 7. Effect of increasing FC parameter on results of ch-8 for profile 3 without departure-time adjustment

As shown in Figs. 3 and 4, the average runtime indicates a decreasing trend approximately as L increases for both versions of the proposed approach. Although the approach is modified and local search iterations don't change to have a fair comparison in this case, CPU time decreases in both versions for all profiles because by increasing L value the candidate solutions found by LNS component can be accepted and chosen earlier, so performing the first and second stages of the proposed approach can be started faster. Average l-max of the solution in the first version of THLNS (i.e. without adjusting departure times) increases for profile 1. For profiles 2, 3 in the first version of algorithm, average l-max exceeds the limit L, but when the limit increases by 20 % (i.e. 1.2 L), average l-max of both profiles becomes very close to the 1.2 L limit. It shows that in profiles 2, 3 with higher speed variation, when the l_max limit is increased by 20%, the algorithm can find solutions with l-max values that are very close to or under the l-max limit. After that by more increasing of l-max, both profiles indicate similar trend as the first profile. As expected, in the second version of the algorithm with adjusting departure times, profiles demonstrate different behaviors and l-max indicates an increasing trend for profile 2 as well as profile 3 approximately. It indicates that adjusting departure time leads to the solutions with satisfying or very close to l-max limit for profiles 1, 2 and has been effective on decreasing remarkable values of l-max. The average travel time doesn't indicate a monophonic and constant trend for any profile in the two versions of the algorithm. With the local search limit fixed at 10 (to ensure consistent feasible solutions within reasonable time), increasing l-max limit has negligible impact on travel times. The proposed algorithm is designed such that obtaining solutions with less travel times is expected by increasing l-max limit and local search counts simultaneously. Fig. 5 shows that average CPU time of the first version for all profiles is increased by increasing local search counts as expected. Average l-max is decreased in all profiles which indicates that increasing local search counts has great impact on obtaining consistent solutions of all profiles specially for profiles 2, 3 with higher degrees of speed variation. Also, the average total travel time of profile 1 shows a decreasing trend approximately but other profiles don't indicate a specific trend. It implies that increasing local search count in profile 1 can simultaneously result in finding solutions with less travel times and less l-max with the given l-max limit (L), but for profiles 2, 3 it should be required to increase l-max limit in addition to perform more numbers of local searches so that consistent solutions with less total travel times can be found. Figs. 6 and 7 show that by increasing FC parameter in the first version of the algorithm, the average CPU time increases with similar trend for both instances of ch-1 and ch-8 in profile 3. The average l-max first decreases greatly by increasing FC to 0 in ch-1 and then continue decreasing until FC=0.1 and increasing FC above 0.1 doesn't decrease l-max anymore. In ch-8 which has very higher l-max limit (L=73.04), the algorithm has found consistent solution for all FC values and increasing FC doesn't have impact on decreasing l-max anymore because the proposed algorithm designed such that inconsistent solutions are penalized and once it reaches a consistent solution it doesn't give any reward to decrease l-max values anymore. The average travel time shows an approximately increasing trend with increasing FC for both instances of ch-1 and ch-8. This reveals an important result: the FC parameter can decrease the travel time of obtained solutions and computational time of algorithm simultaneously if properly set. FC=-0.1 has the minimum average travel time as well as CPU time among other values.

5.1. Managerial insight

The current research provides an efficient approach for managers to obtain consistent solutions in real-world cases where travel times can fluctuate because of changing factors like traffic congestion in urban environments. The research presents an approach which incorporates a filtering mechanism as well as estimating travel times to reduce the computational times, so managers can use it to find solutions which simultaneously focus on customer satisfaction via consistency considerations and cost efficiency by maintaining moderate travel times in real-world environments like urban regions with reduced computational time.

Results indicated that by incorporating dynamic conditions of real-world practice through modeling time-dependent travel times, finding consistent solutions becomes more difficult. This suggests managers should account for dynamic conditions by implementing time-dependent travel times, enabling them to: (1) satisfy customer consistency requirements while (2) maintaining lower travel times in practice.

The incorporation of the departure-time adjustment procedure into the algorithm reduced travel times and improved consistency without increasing computational time. Moreover, thanks to the specialized design of THLNS, the proposed procedure further reduced computational time, as the algorithm could identify consistent solutions earlier. The results demonstrate that the proposed approach with departure-time adjustment could find consistent solutions for all ConVRP instances in the literature with introduced time-dependent travel times. Managers can apply this adjustment to large-scale cases in practice to meet customer consistency requirements more efficiently in real urban environments.

6. Conclusion

In recent years, the primary focus of VRP has shifted from fleet cost optimization to greater emphasis on customer-related factors. Concentrating more on customer satisfaction through improved service levels and quality is crucial to remaining competitive in today's business environment. Providing consistent services, using the same provider at roughly the same times during customers' demanding periods, is a key aspect of high-quality service in many applications such as home healthcare services, parcel delivery and retail distribution systems. It can help improve customer loyalty and maintain long-term relationships with the company. The ConVRP represents the initial VRP variant that places primary emphasis on ensuring customer satisfaction. All previous research on the ConVRP considered deterministic, time-independent travel times in the transportation network. These models may fail to reflect real-world scenarios, where travel times depend on departure times due to time-varying factors like traffic congestion, especially in urban environments. Thus, they cannot ensure arrival time consistency but can only guarantee driver consistency. Moreover, these models' total travel time estimates may not reflect reality, leading to suboptimal or even infeasible solutions in practice.

This study is the first to model time-dependent travel times in the ConVRP by integrating TDVRP and ConVRP frameworks, enhancing practicality. A template-based hybrid approach was proposed combining VNS within a LNS framework. The method incorporates an efficient search-space filtering mechanism and travel-time estimation to identify consistent solutions with reduced computational effort. Additionally, a modified heuristic was developed to adjust depot departure times, improving arrival-time consistency. Results demonstrated that the proposed THLNS algorithm finds consistent solutions within reasonable computational times. The small ConVRP instances of the literature were also run by THLNS and CPLEX with 3600 seconds runtime limit. These results indicated the superiority of the proposed approach for finding consistent solutions with far lower computation times. The proposed approach was also compared to three established methods from the ConVRP literature. The results demonstrate its superiority, yielding consistent solutions with 13.38%, 10.61%, and 35.67% lower average travel times than the alternative methods, within equal computation times. The results of sensitivity analysis indicated that the CPU time was decreased and l-max generally increased with increasing parameter L. By increasing local search counter, the CPU time was increased but average l-max decreased for all profiles. The average travel time demonstrated a decreasing trend for profile 1. The results of increasing FC parameter which filters the solution space showed that average l-max was decreased greatly by a little increase in FC. Increasing FC simultaneously increased the travel time and CPU time. This indicated the efficiency of filtering mechanism for skipping low-potential solutions in the proposed approach.

Future research should focus on a multi-objective model that optimizes cost (travel time), service consistency, and sustainability, incorporates live traffic data to handle unpredictable traffic patterns dynamically, and robust optimization or stochastic programming to deal with uncertainty in customer demands or service times, uses machine learning to predict travel time variability based on historical or real-time data and integrates green logistics with consistency.

References

Adamo, T., Gendreau, M., Ghiani, G., & Guerriero, E. (2024). A review of recent advances in time-dependent vehicle routing. *European Journal of Operational Research*.

Ahn, B.-H., & Shin, J.-Y. (1991). Vehicle-routeing with time windows and time-varying congestion. *Journal of the Operational Research Society, 42*(5), 393-400.

Alinaghian, M., & Naderipour, M. (2016). A novel comprehensive macroscopic model for time-dependent vehicle routing problem with multi-alternative graph to reduce fuel consumption: A case study. *Computers & Industrial Engineering, 99*, 210-222.

Alvarez, A., Cordeau, J.-F., & Jans, R. (2024). The consistent vehicle routing problem with stochastic customers and demands. *Transportation Research Part B: Methodological, 186*, 102968.

Balseiro, S. R., Loiseau, I., & Ramonet, J. (2011). An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. *Computers & Operations Research, 38*(6), 954-966.

Beasley, J. (1981). Adapting the savings algorithm for varying inter-customer travel times. *Omega*, *9*(6), 658-659.

Braekers, K., & Kovacs, A. A. (2016). A multi-period dial-a-ride problem with driver consistency. *Transportation Research Part B: Methodological, 94*, 355-377.

Cai, L., Lv, W., Xiao, L., & Xu, Z. (2021). Total carbon emissions minimization in connected and automated vehicle routing problem with speed variables. *Expert Systems with Applications, 165*, 113910.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations research, 12*(4), 568-581.

Dabia, S., Ropke, S., Van Woensel, T., & De Kok, T. (2013). Branch and price for the time-dependent vehicle routing problem with time windows. *Transportation Science, 47*(3), 380-396.

Donati, A. V., Montemanni, R., Casagrande, N., Rizzoli, A. E., & Gambardella, L. M. (2008). Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research, 185*(3), 1174-1191.

Fan, H., Zhang, Y., Tian, P., Lv, Y., & Fan, H. (2021). Time-dependent multi-depot green vehicle routing problem with time windows considering temporal-spatial distance. *Computers & Operations Research, 129*, 105211.

Feillet, D., Garaix, T., Lehuédé, F., Péton, O., & Quadri, D. (2014). A new consistent vehicle routing problem for the transportation of people with disabilities. *Networks, 63*(3), 211-224.

Figliozzi, M. A. (2012). The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. *Transportation Research Part E: Logistics and Transportation Review, 48*(3), 616-636.

Fleischmann, B., Gietz, M., & Gnutzmann, S. (2004). Time-varying travel times in vehicle routing. *Transportation Science, 38*(2), 160-173.

Gendreau, M., Ghiani, G., & Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & Operations Research, 64*, 189-197.

Gmira, M., Gendreau, M., Lodi, A., & Potvin, J.-Y. (2021). Tabu search for the time-dependent vehicle routing problem with time windows on a road network. *European Journal of Operational Research, 288*(1), 129-140.

Goeke, D., Roberti, R., & Schneider, M. (2019). Exact and heuristic solution of the consistent vehicle-routing problem. *Transportation Science, 53*(4), 1023-1042.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & service operations management, 11*(4), 630-643.

Haghani, A., & Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research, 32*(11), 2959-2986.

Hashimoto, H., Yagiura, M., & Ibaraki, T. (2008). An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. *Discrete Optimization, 5*(2), 434-456.

Hill, A. V., & Benton, W. C. (1992). Modelling intra-city time-dependent travel speeds for vehicle scheduling problems. *Journal of the Operational Research Society, 43*(4), 343-351.

Huang, Y., Zhao, L., Van Woensel, T., & Gross, J.-P. (2017). Time-dependent vehicle routing problem with path flexibility. *Transportation Research Part B: Methodological, 95*, 169-195.

Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research, 144*(2), 379-396.

Jie, K.-W., Liu, S.-Y., & Sun, X.-J. (2022). A hybrid algorithm for time-dependent vehicle routing problem with soft time windows and stochastic factors. *Engineering Applications of Artificial Intelligence, 109*, 104606.

Jost, C., Jungwirth, A., Kolisch, R., & Schiffels, S. (2022). Consistent vehicle routing with pickup decisions-Insights from sport academy training transfers. *European Journal of Operational Research, 298*(1), 337-350.

Jung, S., & Haghani, A. (2001). Genetic algorithm for the time-dependent vehicle routing problem. *Transportation Research Record, 1771*(1), 164-171.

Kok, A. L., Hans, E. W., & Schutten, J. M. (2012). Vehicle routing under time-dependent travel times: the impact of congestion avoidance. *Computers & Operations Research, 39*(5), 910-918.

Kovacs, A. A., Golden, B. L., Hartl, R. F., & Parragh, S. N. (2014). Vehicle routing problems in which consistency considerations are important: A survey. *Networks, 64*(3), 192-213.

Kovacs, A. A., Golden, B. L., Hartl, R. F., & Parragh, S. N. (2015). The generalized consistent vehicle routing problem. *Transportation Science, 49*(4), 796-816.

Kovacs, A. A., Parragh, S. N., & Hartl, R. F. (2014). A template- based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks, 63*(1), 60-81.

Kovacs, A. A., Parragh, S. N., & Hartl, R. F. (2015). The multi-objective generalized consistent vehicle routing problem. *European Journal of Operational Research, 247*(2), 441-458.

Lian, K., Milburn, A. B., & Rardin, R. L. (2016). An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. *Iie Transactions, 48*(10), 975-992.

Liu, C., Kou, G., Zhou, X., Peng, Y., Sheng, H., & Alsaadi, F. E. (2020). Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach. *Knowledge-Based Systems, 188*, 104813.

Liu, Y., Roberto, B., Zhou, J., Yu, Y., Zhang, Y., & Sun, W. (2023). Efficient feasibility checks and an adaptive large neighborhood search algorithm for the time-dependent green vehicle routing problem with time windows. *European Journal of Operational Research, 310*(1), 133-155.

Lu, J., Chen, Y., Hao, J.-K., & He, R. (2020). The time-dependent electric vehicle routing problem: Model and solution. *Expert Systems with Applications, 161*, 113593.

Luo, Z., Qin, H., Che, C., & Lim, A. (2015). On service consistency in multi-period vehicle routing. *European Journal of Operational Research, 243*(3), 731-744.

Maden, W., Eglese, R., & Black, D. (2010). Vehicle routing and scheduling with time-varying data: A case study. *Journal of the Operational Research Society, 61*(3), 515-522.

Malandraki, C., & Daskin, M. S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science, 26*(3), 185-200.

Manavizadeh, N., Farrokhi-Asl, H., & WT Lim, S. F. (2020). A New Mathematical Model for the Green Vehicle Routing Problem by Considering a Bi-Fuel Mixed Vehicle Fleet. *Journal of Optimization in Industrial Engineering, 13*(2), 165-183.

Mancini, S. (2017). A combined multistart random constructive heuristic and set partitioning based formulation for the vehicle routing problem with time dependent travel times. *Computers & Operations Research, 88*, 290-296.

Mancini, S., Gansterer, M., & Hartl, R. F. (2021). The collaborative consistent vehicle routing problem with workload balance. *European Journal of Operational Research, 293*(3), 955-965.

Nolz, P. C., Absi, N., Feillet, D., & Seragiotto, C. (2022). The consistent electric-Vehicle routing problem with backhauls and charging management. *European Journal of Operational Research, 302*(2), 700-716.

Pan, B., Zhang, Z., & Lim, A. (2021). Multi-trip time-dependent vehicle routing problem with time windows. *European Journal of Operational Research, 291*(1), 218-231.

Rincon-Garcia, N., Waterson, B., Cherrett, T. J., & Salazar-Arrieta, F. (2020). A metaheuristic for the time-dependent vehicle routing problem considering driving hours regulations–An application in city logistics. *Transportation Research Part A: Policy and Practice, 137*, 429-446.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science, 40*(4), 455-472.

Shahrabi, F., Nasiri, M. M., & Al-e, S. M. J. M. (2024). Vehicle routing problem with cross-docking in a sustainable supply chain for perishable products. *journal of Optimization in Industrial Engineering, 17(2), 259-278*.

Sharafi, A., & Bashiri, M. (2016). Green vehicle routing problem with safety and social concerns. *Journal of Optimization in Industrial Engineering, 10*(21), 93-100.

Shaw, P. (1998). *Using constraint programming and local search methods to solve vehicle routing problems.* Paper presented at the International conference on principles and practice of constraint programming.

Smilowitz, K., Nowak, M., & Jiang, T. (2013). Workforce management in periodic delivery operations. *Transportation Science, 47*(2), 214-230.

Soler, D., Albiach, J., & Martínez, E. (2009). A way to optimally solve a time-dependent vehicle routing problem with time windows. *Operations Research Letters, 37*(1), 37-42.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research, 35*(2), 254-265.

Soysal, M., & Çimen, M. (2017). A simulation based restricted dynamic programming approach for the green time dependent vehicle routing problem. *Computers & Operations Research, 88*, 297-305.

Stavropoulou, F. (2022). The consistent vehicle routing problem with heterogeneous fleet. *Computers & Operations Research, 140*, 105644.

Stavropoulou, F., Repoussis, P. P., & Tarantilis, C. D. (2019). The vehicle routing problem with profits and consistency constraints. *European Journal of Operational Research, 274*(1), 340-356.

Subramanyam, A., & Gounaris, C. E. (2016). A branch-and-cut framework for the consistent traveling salesman problem. *European Journal of Operational Research, 248*(2), 384-395.

Subramanyam, A., & Gounaris, C. E. (2018). A decomposition algorithm for the consistent traveling salesman problem with vehicle idling. *Transportation Science, 52*(2), 386-401.

Sun, P., Veelenturf, L. P., Dabia, S., & Van Woensel, T. (2018). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research, 264*(3), 1058-1073.

Sun, P., Veelenturf, L. P., Hewitt, M., & Van Woensel, T. (2018). The time-dependent pickup and delivery problem with time windows. *Transportation Research Part B: Methodological, 116*, 1-24.

Sungur, I., Ren, Y., Ordóñez, F., Dessouky, M., & Zhong, H. (2010). A model and algorithm for the courier delivery problem with uncertainty. *Transportation Science, 44*(2), 193-205.

Tarantilis, C. D., Stavropoulou, F., & Repoussis, P. P. (2012). A template-based tabu search algorithm for the consistent vehicle routing problem. *Expert Systems with Applications, 39*(4), 4233-4239.

Ticha, H. B., Absi, N., Feillet, D., & Quilliot, A. (2017). Empirical analysis for the VRPTW with a multigraph representation for the road network. *Computers & Operations Research, 88*, 103-116.

Ulmer, M., Nowak, M., Mattfeld, D., & Kaminski, B. (2020). Binary driver-customer familiarity in service routing. *European Journal of Operational Research, 286*(2), 477-493.

Ulsrud, K. P., Vandvik, A. H., Ormevik, A. B., Fagerholt, K., & Meisel, F. (2022). A time-dependent vessel routing problem with speed optimization. *European Journal of Operational Research, 303*(2), 891-907.

Voigt, S. (2025). A review and ranking of operators in adaptive large neighborhood search for vehicle routing problems. *European Journal of Operational Research, 322*(2), 357-375.

Xiong, H., Xu, Y., Yan, H., Guo, H., & Zhang, C. (2024). Optimizing electric vehicle routing under traffic congestion: A comprehensive energy consumption model considering drivetrain losses. *Computers & Operations Research*, *168*, 106710.

Xu, Z., & Cai, Y. (2018). Variable neighborhood search for consistent vehicle routing problem. *Expert Systems with Applications*, *113*, 66-76.

Yu, X.-P., Hu, Y.-S., & Wu, P. (2024). The consistent vehicle routing problem considering driver equity and flexible route consistency. *Computers & Industrial Engineering*, *187*, 109803.

Zhang, R., Guo, J., & Wang, J. (2020). A time-dependent electric vehicle routing problem with congestion tolls. *IEEE Transactions on Engineering Management*, *69*(4), 861-873.

Zhang, T., Chaovalitwongse, W. A., & Zhang, Y. (2014). Integrated ant colony and tabu search approach for time dependent vehicle routing problems with simultaneous pickup and delivery. *Journal of Combinatorial Optimization*, *28*, 288-309.

Zhao, J., Poon, M., Tan, V. Y., & Zhang, Z. (2024). A hybrid genetic search and dynamic programming-based split algorithm for the multi-trip time-dependent vehicle routing problem. *European Journal of Operational Research*, *317*(3), 921-935.

Zhen, L., Lv, W., Wang, K., Ma, C., & Xu, Z. (2020). Consistent vehicle routing problem with simultaneous distribution and collection. *Journal of the Operational Research Society*, *71*(5), 813-830.

Zhou, G., Li, D., Bian, J., & Zhang, Y. (2024). Two-echelon time-dependent vehicle routing problem with simultaneous pickup and delivery and satellite synchronization. *Computers & Operations Research*, *167*, 106600.

Appendix A.

Algorithm A.1. Travel time calculation procedure (Ichoua, Gendreau, & Potvin, 2003).

```
$t = t_0$
$d = d_{ij}$
$t' = t + (\frac{d}{v_{cT_k}})$
$while\ t' > \bar{t_k}$
    $d = d - v_{cT_k} * (\bar{t_k} - t)$
    $t = \bar{t_k}$
    $t' = t + (\frac{d}{v_{cT_{k+1}}})$
    $k = k + 1$
return $t' - t_0$
```

Algorithm A.2. Adjustment of vehicle departure times (Kovacs, Golden, Hartl, & Parragh, 2015).

```
$while\ \max PF > \epsilon\ or\ \max PB > \epsilon$
    $i^{max} = the\ customer\ with\ the\ current\ maximum\ ATD$
    $BC = \{i^{max}\}\ \#blocking\ customers\ set$
    $\max PF = \max ATD(s)$
    $for\ all\ j \in BC$
        $\text{Max}\, PF = \min\{\max PF, getMaxPF(j, BC, s)\}$
    $end\ for$
    $if\ \max PF > \epsilon\ then$
        $apply\ PF(maxPF, BC, s)$
    $else\ \#pushing\ forward\ is\ not\ possible\ \Rightarrow\ try\ to\ pull\ backwards$
        $BC = \{i^{max}\}$
        $\max PB = \max ATD(s)$
        $for\ all\ j \in BC\ do$
            $\max PB = \min\{\max PB, getMaxPB(j, BC, s)\}$
        $end\ for$
        $if\ \max PB > \epsilon\ then$
            $apply\ PB(maxPB, BC, s)$
        $end\ if$
    $end\ if$
$end\ while$
```

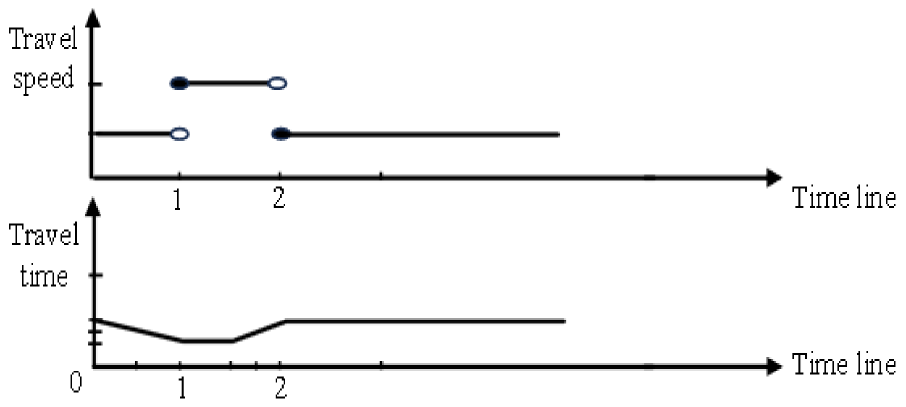


Fig A.1. An example of travel speed and travel time functions (Ichoua, Gendreau, & Potvin, 2003).

Appendix B.

- **Remove Operators**
  - Random Removal

The random removal operator randomly chooses customers and removes them from the template routes. This process repeats until q customers have been removed.

  - Worst Removal

The worst removal operator iteratively identifies customers contributing most to total travel time, and removes them to enable cheaper reinsertion positions. First, all existing customers of the template are sorted in decreasing order in list C based on the saving obtained by temporarily removing them from the template. In every iteration, customer $i = C\,[y\char`^p * |C|]$ is removed. Where y is a random number ∈ [0,1) and p controls the impact of randomization. The saving values are updated and one customer is removed in every iteration until q customers have been removed (Kovacs, Parragh, & Hartl, 2014).

  - Related Removal

The related removal operator is based on the fact that it is easier to interchange customers within a solution when they are somehow related (Kovacs et al., 2014). The relatedness R(i, j) between two customers i and j, combines distance and demand relatedness measures. Distance relatedness measures the Euclidean distance of two customers and demand relatedness is the absolute difference between maximum demands of two given customers among all days. Smaller R(i, j) values indicate higher relatedness. The procedure is initialized by removing a randomly chosen customer from the template and inserting it into the set of removed customers D. In each iteration, one customer is chosen randomly from D to calculate the R(i, j) values. Similar to worst removal, this operator also incorporates randomization to obtain a certain degree of diversification. Therefore, all R(i, j) values are sorted in list C in increasing order and customer $i = C\,[y\char`^p * |C|]$ is removed from the template and added to D. The process continues until q customers are removed.

- **Repair Operators**
  - Greedy Repair

Similar to initial template generation, the greedy repair operator consecutively inserts customers using a similar approach. For each unassigned customer, each feasible insertion position is checked and the customer with lowest insertion cost is assigned to their cheapest feasible position. Feasibility is checked only for capacity constraints based on an artificial capacity limit, selected each LNS iteration.

  - Regret Repairs

Similar to the greedy approach, the regret heuristic inserts customers one after another by checking every feasible insertion position but it includes a look ahead component denoted as regret. This value quantifies potential opportunity costs from delayed insertion (Kovacs et al., 2014). In the basic variant of the regret repair, the customer with the largest difference between inserting into their best position at best route and inserting into the best position at second-best route is inserted in every iteration. This approach extends to multiple routes (q > 2), enabling earlier identification of insertion difficulties. Let Δf_i^q denotes the travel time change for inserting customer i at his cheapest position in his q-cheapest route. If it is not possible to insert a customer into a route, Δf_i^q is set to infinity. In every iteration, the customer i to be inserted is given by eq. B.1:

$$i = argmax_{i\in N'} \left\{ \sum_{h=2}^{\min(q,m)} \left(\Delta f_i^h - \Delta f_i^1\right) \right\} \quad (B.1)$$

Parameter q defines the number of routes considered in the current version of regret and m denotes the number of currently available routes. As in the greedy heuristic, an empty route is added whenever it is not possible to insert further customers in existing routes. We implement four regret heuristics, each with a different setting for q with $q \in (2, 3, 4, m)$.

Table B.1
Characteristics of large ConVRP instances

ID	Instance name	Number of customer nodes (n)	Capacity (Q)	Tour-length limit (T)	l-max limit (L)	Number of available vehicles (k)	Number of periods (d)
Ch-1	Christofides_1_5_0.7	50	160	unlimited	24.38	Unlimited (= n)	5
Ch-2	Christofides_2_5_0.7	75	140	unlimited	34.26	Unlimited (= n)	5
Ch-3	Christofides_3_5_0.7	100	200	unlimited	22.87	Unlimited (= n)	5
Ch-4	Christofides_4_5_0.7	150	200	unlimited	27.53	Unlimited (= n)	5
Ch-5	Christofides_5_5_0.7	199	200	unlimited	26.93	Unlimited (= n)	5
Ch-6	Christofides_6_5_0.7	50	160	200	63.47	Unlimited (= n)	5
Ch-7	Christofides_7_5_0.7	75	140	160	83.96	Unlimited (= n)	5
Ch-8	Christofides_8_5_0.7	100	200	230	73.04	Unlimited (= n)	5
Ch-9	Christofides_9_5_0.7	150	200	200	106.43	Unlimited (= n)	5
Ch-10	Christofides_10_5_0.7	199	200	200	60.17	Unlimited (= n)	5
Ch-11	Christofides_11_5_0.7	120	200	unlimited	16.1	Unlimited (= n)	5
Ch-12	Christofides_12_5_0.7	100	200	unlimited	17.58	Unlimited (= n)	5

Table B.2
Experimental results from the L-9 Taguchi array for ch-3 with profile 2

l-max-constant	l-max-penalty	ol-penalty	ot-penalty	Response
1.75	50	2	2	6347.84
1.75	75	5	5	4906.32
1.75	100	10	10	5055.29
2	50	5	10	6227.94
2	75	10	2	5017.70
2	100	2	5	6492.72
2.25	50	10	5	6305.13
2.25	75	2	10	5278.98
2.25	100	5	2	6181.74

Table B.3
Results of solving instances with proposed THLNS assuming constant travel times

ID	Constant Travel Times				
	Avg. TT	Avg l-max	Min TT	Min l-max	Avg CPU time
Ch-1	2495.59	26.58	2315.68	20.08	276.82
Ch-2	4310.05	34.14	4080.82	34.03	557.46
Ch-3	4247.77	22.67	4048.58	22.36	1632.53
Ch-4	5951.96	26.49	5796.74	25.30	6423.04
Ch-5	7925.52	21.84	7678.22	19.82	15699.15
Ch-6	4568.71	61.94	4437.97	61.11	76.93
Ch-7	7657.18	79.05	7596.72	79.05	206.35
Ch-8	8043.09	71.97	7984.51	70.69	734.92
Ch-9	12041.16	68.67	11857.75	57.11	2357.66
Ch-10	15670.42	59.78	15271.56	59.53	14876.48
Ch-11	5816.02	15.45	5574.57	15	4440.34
Ch-12	4433.47	17.13	4261.69	16.93	1748.34
Average	6,930.078	42.1425	6,742.0675	40.084	4,085.835

Table B.4
Characteristics of small ConVRP instances (equivalent to medium ConTDVRP instances)

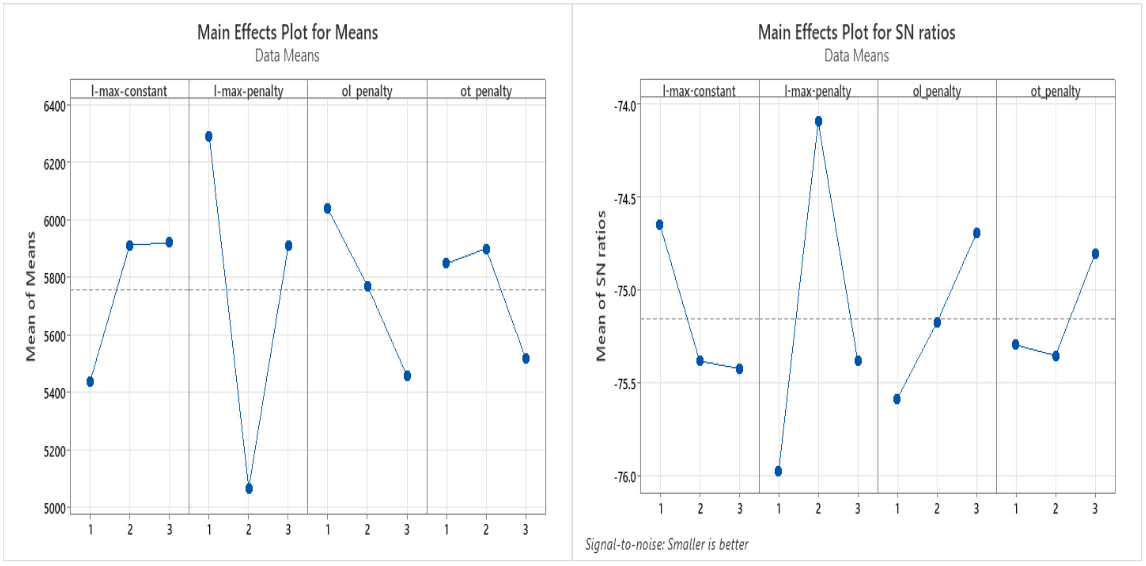
ID	Number of customer nodes (n)	Capacity (Q)	Tour-length limit (T)	l-max limit (L)	Number of available vehicles (k)	Number of periods (d)
10-1	10	15	35	5	Unlimited (= n)	3
10-2	10	15	35	5	Unlimited (= n)	3
10-3	10	15	35	5	Unlimited (= n)	3
10-4	10	15	35	5	Unlimited (= n)	3
10-5	10	15	35	5	Unlimited (= n)	3
12-1	12	15	35	5	Unlimited (= n)	3
12-2	12	15	35	5	Unlimited (= n)	3
12-3	12	15	35	5	Unlimited (= n)	3
12-4	12	15	35	5	Unlimited (= n)	3
12-5	12	15	35	5	Unlimited (= n)	3

Table B.5
Comparison of THLNS (without departure-time adjustment) with three literature approaches for large instances (Profile 1)

ID	THLNS		TALNS		LNS		VNS		l-max Limit (L)	CPU Runtime Limit (sec)
	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max		
Ch-1	2560.33	**30.19**	3109.53	74.14	**2418.66**	31.04	3769.12	109.68	24.38	256.07
Ch-2	**4231.41**	**29.45**	5536.10	79.71	5736.34	196.45	6720.81	79.21	34.26	529.61
Ch-3	**4180.08**	**22.50**	5219.33	175.48	3507.22	89.98	6603.23	58.39	22.87	2198.47
Ch-4	5934.86	**29.59**	8001.31	73.20	**5560.97**	68.76	10254.21	92.14	27.53	8789.10
Ch-5	**8710.42**	**25.71**	10094.21	61.18	11261.98	258.62	13394.86	87.37	26.93	16020.71
Ch-6	4564.68	60.48	**4449.57**	**52.01**	4467.21	143.83	5844.57	74.24	63.47	90.17
Ch-7	**7621.47**	**77.43**	7992.55	68.59	8691.78	123.75	10834.17	114.93	83.96	291.50
Ch-8	8409.34	69.96	**8176.93**	**62.43**	7870.28	152.87	12881.56	97.88	73.04	931.68
Ch-9	11937.39	74.29	**11560.77**	**80.11**	12763.96	166.59	19596.85	74.43	106.43	2562.50
Ch-10	**14924.51**	**78.52**	16860.92	104.45	16158.36	174.93	23643.36	126.56	60.17	18868.65
Ch-11	**5965.49**	**15.36**	7336.60	90.46	7349.04	326.99	8692.46	82.10	16.1	4614.21
Ch-12	**4704.39**	**18.41**	6641.10	162.38	6673.24	265.75	8430.13	120.72	17.58	2708.59
Average	6978.698	44.32417	7914.91	90.345	7704.92	166.63	10888.78	93.1375	46.39	4,821.77

Table B. 6
Comparison of THLNS (without departure-time adjustment) with three literature approaches for large instances (Profile 2)

ID	THLNS		TALNS		LNS		VNS		l-max Limit (L)	CPU Runtime Limit (sec)
	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max	Avg TT (min)	Avg l-max		
Ch-1	2587.35	**29.01**	3087.40	118.06	**2328.61**	99.62	3690.64	101.07	24.38	275.16
Ch-2	**4277.20**	**31.92**	5492.65	122.69	5738.27	196.75	6783.17	115.89	34.26	549.12
Ch-3	4074.68	**30.95**	5757.98	69.50	**3732.82**	69.94	6487.98	98.40	22.87	3198.40
Ch-4	**6134.98**	**26.92**	8075.09	77.68	5006.74	69.12	9729.58	62.78	27.53	7793.79
Ch-5	**8036.07**	**25.04**	11105.01	140.06	11087.99	253.73	13303.76	75.54	26.93	19765.97
Ch-6	4799.10	55.71	**4330.64**	**52.00**	5664.84	169.70	6349.13	61.66	63.47	117.95
Ch-7	7872.98	77.77	**7569.44**	**81.19**	8601.56	126.42	11138.10	92.12	83.96	322.75
Ch-8	**7979.71**	**72.19**	8141.54	65.97	8235.21	172.28	12551.51	164.43	73.04	1103.91
Ch-9	11947.33	81.90	**11685.37**	**104.03**	13491.57	168.99	18869.96	115.73	106.43	2212.37
Ch-10	15070.18	85.02	**20001.64**	**59.93**	16269.80	171.27	24351.97	147.53	60.17	16994.68
Ch-11	5871.00	**17.50**	7465.18	173.06	**5762.62**	177.96	9432.26	85.69	16.1	5585.79
Ch-12	**4515.39**	**26.62**	6860.35	116.41	6676.90	256.04	8422.92	105.81	17.58	2129.74
Average	6930.498	46.7125	8297.691	98.38167	7716.411	160.985	10925.92	102.2208	46.39	5004.14



a b

Fig B.1. Main Effects Plots for the Means and Signal-to-Noise (SN) ratios derived from Table B.2