Computers & Operations Research 194 (2026) 107557

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

The non-disjoint Clustered Traveling Salesperson Problem

Felix Weidinger [a,*], Kris Braekers [b]

[a] Technical University of Darmstadt, Chair for Management Science/Operations Research, Hochschulstr. 1, 64289 Darmstadt, Germany
[b] Hasselt University, Research Group Logistics, Martelarenlaan 42, 3500 Hasselt, Belgium

ARTICLE INFO

Dataset link: https://doi.org/10.5281/zenodo.20489019

Keywords:
Scheduling
Traveling salesman problem
Combinatorial optimization

ABSTRACT

The traveling salesperson problem (TSP) has a long tradition in Operations Research and, thereby, is still relevant as the core and therefore important subproblem of decision problems in today's research. However, the TSP in its general form often lacks specific characteristics of the higher-level planning problems. For that reason, many extensions of the TSP have been developed over recent decades, allowing to consider additional constraints. The paper introduces the non-disjoint Clustered Traveling Salesperson Problem (ndCTSP), for which vertices are a member of at least one cluster. Having multiple non-disjoint clusters, a shortest tour is searched for, which enters each cluster at most once and only visits vertices in direct succession if they belong to the same (entered) cluster. Two variants of the problem are introduced, the unlimited and the limited version, for which the number of entered clusters is either unrestricted or bounded by a problem parameter, respectively. Theoretical insights, three different (mixed) integer linear programming formulations, and complexity results are provided. Further, the broad applicability and relevance of the novel problem are indicated by demonstrating how it generalizes three exemplarily selected planning problems from different domains. Computational tests on the solution performance of the proposed modeling approaches are provided for randomized instances to provide an unbiased evaluation. Finally, a first benchmark on structured instances of one of the generalized problems showcases that solving the ndCTSP outperforms specialized mixed integer models described in literature.

1. Introduction

The traveling salesperson problem (TSP) is a classic problem in Operations Research. It asks for a shortest Hamiltonian circle in an edge-weighted graph, i.e., the sum of the edge-weights involved in the circle is minimal. According to Flood (1956), the problem was introduced in 1934 by H. Whitney. Despite its long history, it still attracts interest, as it is often defining the core of larger and more complex optimization problems. For example, for vehicle routing, several vehicle tours need to be planned simultaneously, but once the locations to be visited by a vehicle are fixed, the remaining core problem constitutes a TSP (see, e.g., Vidal et al., 2020). Similar relations can be observed for many different problem domains if a sequencing component is part of the optimization problem, e.g., in warehousing, when orders need to be picked in a sequence (see, e.g., Füßler and Boysen, 2019), or in exam scheduling, when a group of students has to be evaluated in their individual subjects based on a fair exam schedule (see, e.g., Balakrishnan, 1991). However, the classic TSP is not always resembling all the constraints of the original (sub-)problem. As a result, many variants of the TSP have been studied over time, like, for example, the Bottleneck TSP minimizing the maximum cost over all selected edges while searching for a round tour (see, e.g., Garfinkel and Gilbert, 1978), the Clustered TSP (CTSP), for which each vertex is a member of a cluster and all vertices of the same cluster must be visited in direct succession (see, e.g., Chisman, 1975), or the Generalized TSP (GTSP), where again vertices are part of a cluster and only one vertex per cluster must be visited (see, e.g., Pop et al., 2024). A variant, naturally, introduces further constraints to the classic TSP relevant to different (higher-level) problems. In general, special cases of the respective variant resemble the classic TSP itself, and in some cases, different variants are hierarchically connected, i.e., one variant is a special case of another variant (see, e.g., Noon and Bean, 1993, for a transformation from the GTSP to the CTSP).

This paper provides a new relevant variant of the TSP, the non-disjoint Clustered Traveling Salesperson Problem (ndCTSP), discovered in the context of warehousing but applicable to many fields, like exam scheduling. For the ndCTSP, each vertex of an edge-weighted graph is a member of at least one cluster. The aim of the problem is to find a shortest tour visiting all vertices, while each cluster is entered at most once, and only vertices belonging to the currently active cluster can be visited in direct succession. Thereby, a cluster is active if it has been

* Corresponding author.
E-mail address: felix.weidinger@tu-darmstadt.de (F. Weidinger).

https://doi.org/10.1016/j.cor.2026.107557
Received 2 July 2025; Received in revised form 22 May 2026; Accepted 24 May 2026
Available online 8 June 2026
0305-0548/© 2026 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

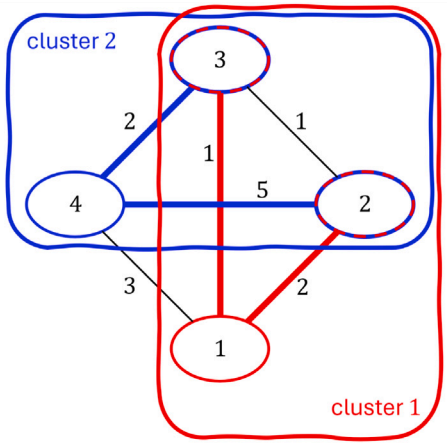


Fig. 1. Example of a minimal ndCTSP instance.

entered last. Entering a cluster is part of the decision problem, limiting the edges selectable to the ones embedded into the currently active cluster.

An instance of the problem is depicted in Fig. 1. Four vertices are given in a fully connected edge-weighted graph, which are members of two non-disjoint clusters. The first (second) cluster consists of vertices 1, 2, and 3 (2, 3, and 4). A feasible solution to the instance is given by tour $3-4-2-1-3$, starting in cluster 2 (see blue edges), which is entered in vertex 3, and leaving it in vertex 2, where cluster 1 is entered, embedding the remaining part of the tour (see red edges). The tour has a length of $2+5+2+1=10$. Please note that tour $3-4-1-2-3$ offers a shorter distance but is infeasible for the ndCTSP as edge $\{4,1\}$ is not embedded in a cluster, i.e., there is no cluster that contains vertices 1 and 4.

The described characteristic of non-disjoint clusters of the ndCTSP is the key to a broad applicability and, therefore, relevance. Having a problem at hand, able to generalize multiple (sub-)problems from a high number of domains, enables researchers to work on increased problem understanding of and efficient solution procedures for a core problem, which then can be applied to a broad field of decision-making. In the paper at hand, the first steps towards a deeper understanding are made by offering the following contributions:

- The work at hand is the first to describe the ndCTSP. Two variants of the problem, i.e., the unlimited and the limited ndCTSP, are presented. While for the unlimited ndCTSP, all clusters can be employed in a solution, the limited ndCTSP comes with a maximum number of clusters allowed to be entered. Complexity results are provided for both variants.
- Three (mixed) integer linear programs ((M)ILPs) are provided for the unlimited ndCTSP and extended to the limited ndCTSP case. The models are benchmarked in a broad computational study, for which a first and extensive dataset for the ndCTSP is generated and made publicly available.
- It is shown how the ndCTSP generalizes other relevant planning problems based on three examples, i.e., the CTSP, the order and bin sequencing problem, and exam scheduling. In a first computational test, solving ndCTSP instances derived from original order and bin sequencing problem instances shows to outperform solving the instances using the originally proposed models. The transformation schemes and computational tests demonstrate the relevance of the novel TSP variant in the context of different domains and its potential to be applied to many different planning problems.

The remainder of the paper is structured as follows: Section 2 provides a literature review of closely related work before the problem is defined in detail, analyzed, and modeled in Section 3. Section 4 describes the computational study and thereby provides first performance insights when solving the ndCTSP. Finally, Section 5 concludes the paper.

2. Literature review

Since the first works on the TSP (see, e.g., Dantzig et al., 1954; Morton and Land, 1955; Flood, 1956), a vast body of literature has accumulated over the last decades. Searching for the TSP in its most popular spellings via Web of Science, more than 2500 Operations Research related journal articles and many books suitable for readers of varying skill levels are listed. In consequence, we focus on basic TSP literature, such as literature reviews, as well as work closely related to ours, like TSP variants based on a segmentation of vertices, i.e., the GTSP and the CTSP.

An overview of the TSP, different formulations, and most basic extensions are provided by Gutin and Punnen (2006). Later, Öncan et al. (2009) compare several problem formulations for the asymmetric TSP, regarding, for example, the strength of LP relaxations. They also derive relationships between the 24 surveyed formulations and identify future research opportunities in the field. An overview of heuristic solution approaches for the TSP is provided by Rego et al. (2011), identifying the classes of Lin–Kernighan (see, e.g., Helsgaun, 2000) and especially stem-and-cycle methods (see, e.g., Gamboa et al., 2006) as the most promising heuristic approaches. Regarding the practical relevance of the TSP and its variants, Nagy and Salhi (2007) and, more recently, Drexl and Schneider (2015) give a general overview of location routing problems, of which the TSP is a special case. Bock et al. (2025) provide an overview of the TSP and its variants as appearing in the warehousing context, while Bagchi et al. (2006) survey TSP-based approaches in flow shop production systems.

According to Pop et al. (2024), the GTSP was introduced independently by Henry-Labordere (1969), Srivastava et al. (1969) and Saskena (1970). The aim of the problem is to find a shortest tour through a set of vertices. Other than for the TSP, however, each vertex is a member of exactly one cluster, and the tour must visit exactly one vertex per cluster instead of each vertex as for the classic TSP. In consequence, the TSP is a special case of the GTSP where each vertex is a member of its own dedicated cluster. The recent study of Pop et al. (2024) provides an overview of the literature on the GTSP. The authors review different mathematical formulations, like the ones from Fischetti et al. (1997), Pop (2007), and Kara et al. (2012), exact and heuristic state-of-the-art solution approaches, and problem variants, like the L-GTSP, for which at least one vertex per cluster must be visited (see also Fischetti et al., 1995). Further, fields of application, like in network design, material flow system design, and stochastic vehicle routing (see, e.g., Laporte et al., 1996), as well as established datasets to be found in the literature, are discussed. Since its publication, to the best of the authors' knowledge, only Deckerová et al. (2024) contributed to the field by suggesting novel combinatorial lower bounds for the GTSP with neighborhoods, where the clusters of the GTSP must be based on neighborhoods, i.e., spatial clusters.

Another famous problem variant of the TSP where the vertices are members of clusters, the CTSP, has been introduced by Chisman (1975). Other than for the GTSP, not only one but all vertices of a cluster must be visited, additionally restricted by the constraint that all vertices of a cluster must be visited in direct succession, i.e., the vertices within each cluster are sequenced as well as the clusters of the higher level tour. Again, the TSP is a special case of the CTSP where all vertices are a member of their own dedicated cluster. Other than for the GTSP, no recent literature review exists for the CTSP. Therefore, we detail some key literature in the following. Laporte and Palekar (2002) provide fields of application of the CTSP, as, for example, in vehicle routing, manufacturing scheduling, or storage defragmentation. Mestria et al.

(2013) propose performant heuristic solution procedures for the symmetric Euclidean CTSP tested successfully against a two-level genetic algorithm. A d-relaxed priority rule for the CTSP is suggested by dos Santos Teixeira and de Araujo (2024) resulting in valid inequalities able to improve the mixed integer model of Hà et al. (2022). Problem variations of the CTSP are tackled by Anily et al. (1999), i.e., the ordered CTSP, for which the sequence of clusters is already given, as well as Angelelli et al. (2014), i.e., the clustered orienteering problem, in which profits associated with visiting a cluster must be maximized while a traveling capacity must be considered.

Zhang et al. (2018) and Baniasadi et al. (2020) consider crossovers of CTSP and GTSP by combining different aspects of the problems, leading to the tabu CTSP and clustered GTSP, respectively.

It can be summarized that the ndCTSP considered in the paper at hand has, to the best of the authors' knowledge, not been considered before in literature.

3. The non-disjoint Clustered Traveling Salesperson Problem

3.1. Problem definition

The **unlimited ndCTSP** is a variant of the TSP, where all vertices V of an edge-weighted graph $G = (V, E, c)$ are elements of at least one cluster. As for the CTSP, all vertices need to be visited exactly once, and a cluster can be entered only once. However, as vertices can be elements of multiple clusters, not all vertices of a cluster must be visited consecutively. Still, only vertices of the currently active cluster can be visited in direct succession. Entering and thereby activating a cluster is part of the optimization decision. However, clusters can be switched only via a vertex that is an element of both clusters, the one to be left, i.e., deactivated, and the one to be newly entered, i.e., activated.

A problem instance of the unlimited ndCTSP is defined by a graph $G = (V, E, c)$ consisting of a set of vertices V, a set of edges E, and a cost function c, assigning a cost value to each edge in E, such that $c : E \mapsto \mathbb{R}$, as well as a set of clusters C, which consists of subsets of V, i.e., $\gamma \subseteq V \ \forall \gamma \in C$. A feasible solution to the problem instance is then provided by a tuple (π, ϖ), i.e., tour $\pi = (\pi_1, \dots, \pi_{|V|})$ and the respective active clusters $\varpi = (\varpi_1, \dots, \varpi_{|V|})$. Without loss of generality, we assume that the sequence starts with a vertex newly entering a cluster if more than one cluster is active in total, i.e., $\varpi_1 \neq \varpi_{|V|}$ if $|\{\gamma \in C | \exists k \in \{1, \dots, |V|\} : \varpi_k = \gamma\}| > 1$. A feasible solution to the ndCTSP needs to fulfill the following requirements associated with the classic TSP:

1. Tour π, with $|\pi| = |V|$, visits all vertices exactly once, i.e., $\exists k \in \{1, \dots, |V|\} : \pi_k = v \ \forall v \in V$, with π_k providing the vertex visited at the kth position of tour π.
2. The tour is closed, i.e., after reaching vertex $\pi_{|V|}$ within cluster $\varpi_{|V|}$, the tour returns to π_1 within cluster ϖ_1, with ϖ_k providing the cluster active at the kth position of tour π.
3. The tour only utilizes existing edges, i.e., $\{\pi_{k-1}, \pi_k\} \in E \ \forall k \in \{2, \dots, |V|\}$ and $\{\pi_{|V|}, \pi_1\} \in E$.

Additionally, following ndCTSP-specific conditions need to be considered:

4. A vertex can only be visited if it is an element of the currently active cluster, i.e., $\pi_k \in \varpi_k \ \forall k \in \{1, \dots, |V|\}$.
5. Clusters can be switched only at vertices that are element of the cluster to be entered and the cluster to be left, i.e., $\pi_{(k+1)} \in \varpi_k \ \forall k \in \{1, \dots, |V| - 1\}$ and $\pi_1 \in \varpi_{|V|}$.
6. Each cluster is entered at most once, i.e., $\varpi_{k_1} = \varpi_{k_3} \rightarrow \varpi_{k_2} = \varpi_{k_1} \ \forall k_1, k_2, k_3 \in \{1, 2, \dots, |V|\} : k_1 < k_2 < k_3$.

Among all feasible solutions, one with the minimal associated costs $Z(\pi, \varpi) = c_{\pi_{|V|}, \pi_1} + \sum_{k=2}^{|V|} c_{\pi_{(k-1)}, \pi_k}$ is searched for the ndCTSP.

Table 1
Notation for the EdgeDFJ-ILP.

Symbol	Description
V	Set of vertices (index: i, j, k; $V = \{1, 2, \dots, \|V\|\}$)
E	Set of edges (index: e)
C	Set of clusters (index: γ, with $\gamma \subseteq V \ \forall \gamma \in C$)
$\bar{C}_e$	Set of feasible clusters for edge $e \in E$
$\delta(S)$	Set of edges crossing border from vertex subset $S \subseteq V$
$E(S)$	Set of edges within the borders of vertex subset $S \subseteq V$
c_e	Cost associated with edge $e \in E$
x_e	Binary variable: 1, iff edge $e \in E$ is included into the tour
$y_{e,\gamma}$	Binary variable: 1, iff edge $e \in E$ is active in cluster $\gamma \in C$

Based on the unlimited version of ndCTSP, the **limited ndCTSP** is a variation, restricting the number of clusters to be entered to Γ, i.e., additional to the above-mentioned requirements, a solution is only feasible if $\exists C' \subseteq C : |C'| \leq \Gamma$ and $\varpi_k \in C' \ \forall k \in \{1, \dots, |V|\}$.

Example: Fig. 2 depicts an instance of the ndCTSP. On the left, seven vertices $V = \{1, 2, 3, 4, 5, 6, 7\}$, seven clusters $C = \{\{1, 2, 3, 7\}, \{1, 2, 6\}, \{1, 4, 5\}, \{2, 6\}, \{3, 4, 6, 7\}, \{4, 7\}, \{5, 6\}\}$, and the edge related costs of the graph are provided. Please note that only edges between vertices that share at least one cluster are introduced. Even if existing, the remaining edges could not be employed in a feasible solution due to the constraints for the ndCTSP. If two vertices share more than one cluster, multiple color-coded edges are introduced into the graph, providing an overview of the alternatives. If the problem instance is considered a classic TSP, clusters are disregarded, and a shortest tour $\pi_{TSP} = (1, 2, 3, 4, 5, 6, 7)$ with length 1246 can be found. If, however, for the ndCTSP, clusters need to be considered, the tour is not feasible, as cluster γ_5 would be entered twice, once for traveling from vertex 3 to 4 and once for traveling from vertex 6 to 7. Respecting clusters, an optimal tour is given by $\pi_{ndCTSP} = (1, 7, 2, 3, 4, 5, 6)$ and $\varpi_{ndCTSP} = (\gamma_1, \gamma_1, \gamma_1, \gamma_5, \gamma_3, \gamma_7, \gamma_2)$ with a length of 1334. For this solution, clusters γ_1, γ_2, γ_3, γ_5, and γ_7 are entered. If the number of entered clusters is limited to three, i.e., the limited ndCTSP with $\Gamma = 3$ is applied, an optimal tour is given by $\pi^{\Gamma=3}_{ndCTSP} = (1, 7, 2, 3, 6, 4, 5)$ and $\varpi^{\Gamma=3}_{ndCTSP} = (\gamma_1, \gamma_1, \gamma_1, \gamma_5, \gamma_5, \gamma_3, \gamma_3)$, resulting in a tour of length 1379 and entered clusters γ_1, γ_3, and γ_5.

Complexity: Clearly, the ndCTSP is a generalization of the NP-hard TSP (see, e.g., Garey and Johnson, 1979, for an NP-hardness proof of the TSP). Each TSP instance can be transformed in polynomial time into an ndCTSP instance by introducing one cluster hosting all vertices, hence establishing the status of NP-hardness of the ndCTSP in the limited and unlimited version.

3.2. (Mixed) integer linear programming formulations for the ndCTSP

For introducing an integer linear program (ILP) formulation of the ndCTSP on a graph $G = (V, E, c)$, we employ an exponentially sized edge-based formulation based on the formulation of Dantzig et al. (1954). Based on the notation used by other authors (see, e.g., Fischetti et al., 1994), function $E(S)$ with $S \subseteq V$ returns all the edges connecting vertices within S, i.e., $E(S) = \{e = \{i, j\} \in E : i, j \in S\}$, and function $\delta(S)$ returns all edges that cross borders to vertex set S, i.e., $\delta(S) = \{e = \{i, j\} \in E : i \in S \text{ and } j \notin S\}$. In line with previous work and for the sake of simplicity, we write $\delta(j)$ instead of $\delta(\{j\})$ if the considered subset involves only one vertex. Additionally, we define $\bar{C}_e$ as the set of clusters edge e can be active in, i.e., $\bar{C}_e = \{\gamma \in C : e = \{i, j\} \text{ and } i, j \in \gamma\}$. Therefore, the sets describe the clusters via which the related edge can be traversed. Making use of this notation and the additional symbols defined in Table 1, formulas (1) to (8) define an exponentially sized edge-based ILP, dubbed EdgeDFJ-ILP, for the unlimited ndCTSP.

EdgeDFJ-ILP:

$$\text{Min: } Z = \sum_{e \in E} c_e x_e \tag{1}$$

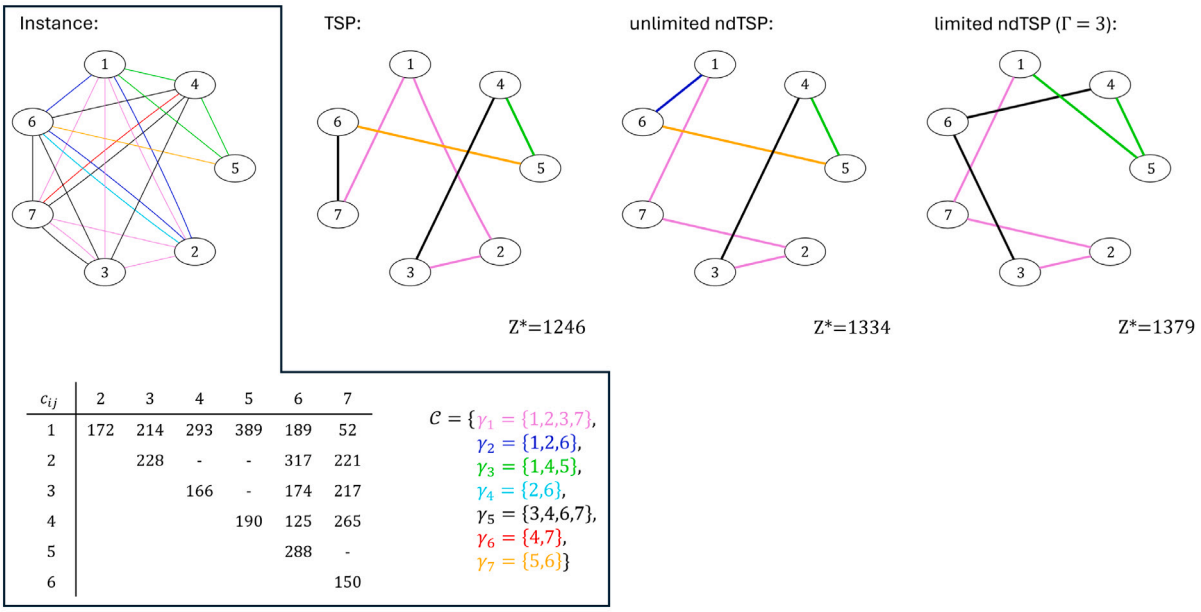


Fig. 2. Example of an ndCTSP instance with an optimal TSP, an optimal unlimited ndCTSP, and an optimal limited ndCTSP ($\Gamma = 3$) solution.

$$\sum_{e \in \delta(j)} x_e = 2 \qquad \forall j \in V \tag{2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad \forall S \subset V : 2 \leq |S| \leq \lfloor |V|/2 \rfloor \tag{3}$$

$$x_e = \sum_{\gamma \in \bar{C}_e} y_{e,\gamma} \qquad \forall e \in E \tag{4}$$

$$y_{e_1,\gamma} + y_{e_2,\gamma} \leq \sum_{e \in \delta(S)} y_{e,\gamma} + 1 \qquad \forall \gamma \in C; S \subset \gamma : |S| \geq 2, |\gamma \setminus S| \geq 2; \; e_1 \in E(S); e_2 \in E(\gamma \setminus S) \tag{5}$$

$$x_e \in \{0,1\} \qquad \forall e \in E \tag{6}$$

$$y_{e,\gamma} \in \{0,1\} \qquad \forall e \in E; \gamma \in \bar{C}_e \tag{7}$$

$$y_{e,\gamma} = 0 \qquad \forall e \in E; \gamma \in C \setminus \bar{C}_e \tag{8}$$

Objective function (1) minimizes the costs, summed up over all selected edges. Hereby, constraints (2) ensure that each vertex is adjacent to two selected edges, i.e., can be entered and left during the tour, while constraints (3) use classic exponentially sized subtour elimination techniques as proposed by Dantzig et al. (1954) to ensure that the tour consists of only one component. Within the tour, constraints (4) make sure that each selected edge is running through an active and feasible cluster. Constraints (5) then define that clusters cannot be entered multiple times during the tour, adapting the formulation of Dantzig et al. (1954). For every combination of two disjoint subsets of vertices belonging to a cluster: if at least one edge within each of the disjoint subsets is active in the considered cluster, the edges must be connected, and, therefore, at least one edge active in the same cluster must cross the border between the subsets, i.e., at least one of the edges of $\delta(S)$ must be active in γ as well. Please note that for only three vertices in a cluster but more than three in the whole graph, at most two edges can be embedded into the cluster according to the subtour elimination constraints. Those need to share one vertex and are, therefore, connected. In accordance, constraints (5) are only applied to clusters containing at least four vertices. Finally, constraints (6), (7), and (8) define the domains of the decision variables.

Given an additional binary variable $\bar{y}_\gamma$, which is assigned one if cluster γ is entered, and the limiting parameter Γ, the limited ndCTSP can be modeled by formulas (1) to (8) as well as the following formulas (9) to (11).

$$y_{e,\gamma} \leq \bar{y}_\gamma \qquad \forall e \in E; \gamma \in \bar{C}_e \tag{9}$$

Table 2
Notation for the ArcDFJ-MILP.

Symbol	Description
A	Set of (directed) arcs
$\dot{C}_i$	Set of clusters containing vertex $i \in V$, i.e., $\dot{C}_i = \{\gamma \in C : i \in \gamma\}$
$c_{i,j}$	Costs associated with arc $(i,j) \in A$
$x_{i,j}$	Binary variable: 1, iff vertex $i \in V$ is the direct predecessor of $j \in V$
$z_{i,\gamma}$	Binary variable: 1, iff cluster $\gamma \in C$ is entered in vertex $i \in V$
$f_{i,j,\gamma}$	Continuous variable: flow from vertex $i \in V$ to vertex $j \in V$ in cluster $\gamma \in C$

$$\sum_{\gamma \in C} \bar{y}_\gamma \leq \Gamma \tag{10}$$

$$\bar{y}_\gamma \in \{0,1\} \qquad \forall \gamma \in C \tag{11}$$

Constraints (9) make sure that each entered cluster is counted, while constraint (10) limits the number of entered clusters to parameter Γ. The domain of the additional variable $\bar{y}_\gamma$ is defined by constraints (11).

While the aforementioned formulation is short in definition, it is limited to symmetric cost matrices. However, as for other variations of the TSP, some applications of the ndCTSP might require asymmetric costs. An arc-based mixed integer linear programming (MILP) formulation is, therefore, presented in the following formulas (12) to (24), able to represent both symmetric and asymmetric versions of the ndCTSP. Again, subtour elimination is based on the exponentially sized formulation of Dantzig et al. (1954), for which reason we refer to the formulation as ArcDFJ-MILP in the following. The coherence of clusters is ensured using a compact and flow-based formulation. Flow-based models are successfully applied in manifold domains (see, e.g., Dewil et al., 2015), and the approach showed its potential in preliminary tests again, outperforming alternative MILP formulations. The formulation is based on the notation presented in Tables 1 and 2.

ArcDFJ-MILP:

$$\text{Min: } Z = \sum_{(i,j) \in A} c_{i,j} x_{i,j} \tag{12}$$

$$\sum_{\substack{(i,j) \in A: \\ j = j'}} x_{i,j} = 1 \qquad \forall j' \in V \tag{13}$$

$$\sum_{\substack{(i,j)\in A:\\ i=i'}} x_{i,j} = 1 \qquad \forall i' \in V \tag{14}$$

$$\sum_{\substack{(i,j)\in A:\\ i,j\in S}} x_{i,j} \leq |S| - 1 \qquad \forall S \subset V : 2 \leq |S| \leq \lfloor |V|/2 \rfloor \tag{15}$$

$$\sum_{i\in\gamma} z_{i,\gamma} \leq 1 \qquad \forall \gamma \in C \tag{16}$$

$$\sum_{\gamma\in\dot{C}_i} z_{i,\gamma} \leq 1 \qquad \forall i \in V \tag{17}$$

$$\sum_{i\in V}\sum_{\gamma\in\dot{C}_i} z_{i,\gamma} \geq 1 \tag{18}$$

$$z_{i,\gamma} \leq \sum_{\substack{j\in\gamma\setminus\{i\}:\\ (i,j)\in A}} f_{i,j,\gamma} \qquad \forall i \in V, \gamma \in \dot{C}_i \tag{19}$$

$$\sum_{\gamma\in\dot{C}_i\cap\dot{C}_j} f_{i,j,\gamma} = x_{i,j} \qquad \forall (i,j) \in A \tag{20}$$

$$\sum_{\substack{j\in\gamma\setminus\{i\}:\\ (j,i)\in A}} f_{j,i,\gamma} \leq \sum_{\substack{j\in\gamma\setminus\{i\}:\\ (i,j)\in A}} f_{i,j,\gamma} + \sum_{\gamma'\in\dot{C}_i\setminus\{\gamma\}} z_{i,\gamma'} \qquad \forall i \in V, \gamma \in \dot{C}_i \tag{21}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A \tag{22}$$

$$z_{i,\gamma} \in \{0,1\} \qquad \forall i \in V; \gamma \in \dot{C}_i \tag{23}$$

$$f_{i,j,\gamma} \geq 0 \qquad \forall (i,j) \in A; \gamma \in \dot{C}_i \cap \dot{C}_j \tag{24}$$

Again, objective function (12) minimizes the total costs of the tour. Constraints (13) and (14) ensure that each vertex is visited and left, respectively, while constraints (15) apply the exponentially sized subtour elimination constraints of Dantzig et al. (1954) in their asymmetric version. Constraints (16), (17), and (18) ensure that each cluster is entered at most once, in each vertex at most one cluster is entered, and at least one cluster is entered in a solution, respectively. In constraints (19), a flow is induced once a cluster is entered, i.e., if vertex i enters cluster γ, there must be an outgoing flow towards any other vertex within cluster γ. Thereby, constraints (20) guarantee that flows can only be led via selected arcs. The flow conservation constraints in (21), combined with previous constraints, ensure that the incoming flow at a vertex results in either an outgoing flow within the same cluster or a new cluster being entered at that vertex. In the latter case, the vertex acts as a sink of the incoming flow, and a new flow in another cluster is induced (see constraints (19)). Finally, the domains of the decision variables are defined in constraints (22), (23), and (24).

For the limited ndCTSP, additional constraints (25) need to be applied, limiting the number of entered clusters to the parameter Γ.

$$\sum_{i\in V}\sum_{\gamma\in\dot{C}_i} z_{i,\gamma} \leq \Gamma \tag{25}$$

The two models presented up to now are both exponential in size due to constraints (3), (5), and (15). The, except for the subtour elimination constraints, compact formulation of the ArcDFJ-MILP, however, offers the opportunity to further present a compact arc-based model. In preliminary tests, several compact subtour elimination techniques have been benchmarked, i.e., the Miller–Tucker–Zemlin formulation by Miller et al. (1960), the ATSP6 formulation by Sherali et al. (2006), and the CLAUS formulation by Claus (1984) in the more compact version of Gouveia and Pires (1999). In these tests, the flow-based CLAUS formulation showed the best results, such that the following compact formulation of ndCTSP is based on the work of Claus (1984) and Gouveia and Pires (1999). We refer to the model as ArcClaus-MILP and define it in the following, based on the notation defined in Tables 1, 2, and 3.

ArcClaus-MILP:

Min: Objective (12)

considering constraints (13), (14), (16) to (24) and

Table 3
Notation for the ArcClaus-MILP.

Symbol	Description
$g_{k,i,j}$	Continuous variable: 1, iff arc $(i,j) \in A$ is on the path from vertex $k \in V$ to vertex 1

$$\sum_{\substack{(j,i)\in A:\\ i=i'}} g_{k,j,i} = \sum_{\substack{(i,j)\in A:\\ i=i'}} g_{k,i,j} \qquad \forall i', k \in V \setminus \{1\} : i' \neq k \tag{26}$$

$$\sum_{\substack{(j,i)\in A:\\ j=1}} g_{k,1,i} - \sum_{\substack{(i,j)\in A:\\ j=1}} g_{k,i,1} = -1 \qquad \forall k \in V \setminus \{1\} \tag{27}$$

$$\sum_{\substack{(i,i')\in A:\\ i=i'}} g_{i',i,j} - \sum_{\substack{(j,i')\in A:\\ i=i'}} g_{i',j,i} = 1 \qquad \forall i' \in V \setminus \{1\} \tag{28}$$

$$x_{i,j} \geq g_{k,i,j} \geq 0 \qquad \forall (i,j) \in A, k \in V \setminus \{1\} \tag{29}$$

In the first novel constraints (26), flow conservation is managed. If there is an incoming flow into a vertex, different to 1 and starting from another vertex (also different to 1), there also must be an equivalent outgoing flow. Constraints (27), then, ensure that for each vertex k different to 1 there is a final sink of the flow, which is the arc entering vertex 1. The arc starting in vertex 1, however, is not inducing a flow. For all vertices different to 1, constraints (28) induce a new flow. Finally, constraints (29) make sure that only selected arcs can host a flow while also ensuring non-negativity of the flow variables g. While this model describes the unlimited version of the ndCTSP, adding constraints (25) additionally defines the limited version of the problem.

In the computational study, see Section 4, all three models are benchmarked on a broad set of problem instances.

3.3. Relation to other planning problems

In this subsection and Appendix A, we show that the ndCTSP is a generalization of several diverse, existing, and relevant planning problems. Researching the ndCTSP to better understand its structure and provide efficient solution procedures would, therefore, allow solving each of these planning problems more efficiently, i.e., providing effective tools for a broad domain of problems while concentrating on one main problem. In the following, the relation of ndCTSP to other planning problems is discussed exemplarily based on the order and bin sequencing problem (Füßler and Boysen, 2019; Ouzidan et al., 2022). In Appendix A, further relationships to the CTSP and the exam scheduling problem are presented. Please note that the selection of problems is naturally subjective and not complete.

Order and bin sequencing problem: This problem occurs in warehouses where a predefined number of orders can be picked in parallel and products, demanded by an order, arrive in bins of sufficiently high capacity at the picking stations. If multiple orders being processed in parallel or in direct succession demand the same product, all demand can be served by a single bin delivery with respect to this product. Exploiting such synergy effects, the number of bin visits is to be minimized, reducing utilization of warehousing hardware regarding the optimized picking station. The problem has been originally described by Füßler and Boysen (2019) in the context of high performance order picking systems and has further been investigated by Ouzidan et al. (2022). It is also relevant as a subproblem in the context of robotic mobile fulfillment systems, having gained high popularity in e-commerce warehousing over recent years (see, e.g., Boysen et al., 2019; Boysen and De Koster, 2025). Given a set of bins $\mathcal{B}'$, a set of orders $O' = \{o'_1, o'_2, \ldots, o'_{k'} : o'_{l'} \subseteq \mathcal{B}' \ \forall l' \in \{1,2,\ldots,k'\}\}$ consisting of sets of the bins demanded by the respective order, and a maximum number of orders processed in parallel c', an instance of the order and bin sequencing problem is defined. The optimization problem can be

Table 4
Notation for the order and bin sequencing MILP.

Symbol	Description
$\mathcal{B}'$	Set of bins (index: b')
O'	Set of orders, with $o' \subseteq \mathcal{B}'\ \forall o' \in O'$
$\mathcal{T}'$	Set of time slots, with $\mathcal{T}' = \{1, 2, \dots, \sum_{o' \in O'} \lvert o' \rvert\}$ (index: t')
c'	Maximum number of orders processed in parallel
$x'_{b',t'}$	Binary variable: 1, iff bin b' is available at picking station in time t'
$z'_{b',o',t'}$	Binary variable: 1, iff bin b' is picked for order o' in time t'
$y'_{o',t'}$	Continuous variable: 1, iff order o' is processed in time t'
$\delta'_{t'}$	Continuous variable: 1, iff the bin at the picking station is switched in time t'

described via the time slot based mixed integer linear program given by formulas (30) to (39) and the notation provided in Table 4, as given in slight variation by Füßler and Boysen (2019). Please note that a bin visit can hereby consist of multiple time slots and time slots resemble a theoretical construct to model bin and/or order swaps. In the model, time slots do not have a real world equivalent and are not of identical size time-wise.

$$\text{Min: } Z = \sum_{t'=2}^{\mathcal{T}'} \delta'_{t'} \tag{30}$$

$$\sum_{b' \in \mathcal{B}'} x'_{b',t'} \leq 1 \qquad \forall t' \in \mathcal{T}' \tag{31}$$

$$\sum_{o' \in O'} y'_{o',t'} \leq c' \qquad \forall t' \in \mathcal{T}' \tag{32}$$

$$y_{o',t'_1} + y_{o',t'_3} \leq 1 + y_{o',t'_2} \qquad \forall o' \in O'; t'_1, t'_2, t'_3 \in \mathcal{T}' : t'_1 < t'_2 < t'_3 \tag{33}$$

$$\sum_{t'=1}^{\mathcal{T}'} z'_{b',o',t'} \geq 1 \qquad \forall o' \in O'; b' \in o' \tag{34}$$

$$2z_{b',o',t'} \leq y'_{o',t'} + x'_{b',t'} \qquad \forall o' \in O'; b' \in o'; t' \in \mathcal{T}' \tag{35}$$

$$\delta'_{t'} \geq x'_{b',t'} - x'_{b',t'-1} \qquad \forall b' \in \mathcal{B}'; t' \in \mathcal{T}' \setminus \{1\} \tag{36}$$

$$x'_{b',t'}, z'_{b',o',t'} \in \{0,1\} \qquad \forall b' \in \mathcal{B}'; o' \in O'; t' \in \mathcal{T}' \tag{37}$$

$$0 \leq y'_{o',t'} \leq 1 \qquad \forall o' \in O'; t' \in \mathcal{T}' \tag{38}$$

$$\delta'_t \geq 0 \qquad \forall t' \in \mathcal{T}' \setminus \{1\} \tag{39}$$

Objective function (30) minimizes the number of bin switches, while constraints (31) and (32) make sure that the capacity is respected bin- and order-wise. Constraints (33) ensure that orders are processed without preemption. A fully satisfied demand of each order is guaranteed by constraints (34). Constraints (35) make sure that a pick is only planned if the respective order and the respective bin are actually available at the picking station. The bin switches, counted in the objective function, are identified in constraints (36). Finally, the domains of the variables are defined in constraints (37) to (39). For a more detailed description of the model, please refer to Füßler and Boysen (2019).

An instance of the order and bin sequencing problem can be translated into an instance of the limited ndCTSP within the following steps.

- Introduce a vertex $v_{o',b'}$ for each combination of an order $o' \in O'$ and bin demanded by that respective order $b' \in o'$. Additionally, create three virtual start-vertices v_{s1}, v_{s2}, and v_{s3}.
- Include edges between all vertices. If the vertices represent demand for the same bin of different orders or one of the adjacent vertices is a virtual start-vertex, costs of the edge are set to zero. Otherwise, costs are set to one.
- Introduce clusters for all combinations of c' orders, where the respective cluster includes all vertices belonging to the selected orders. Additionally, create two clusters for each order, one additionally including the start-vertex v_{s1} and the other v_{s3}, respectively. Again, the clusters host all vertices of the corresponding order. Finally, create clusters $\{v_{s1}, v_{s2}\}$ and $\{v_{s2}, v_{s3}\}$.
- In the last step, set the limiting parameter Γ to $5 + |O'| - c'$.

The limiting parameter Γ makes sure that after starting to process the initial c' orders, at most $|O'| - c'$ changes of the processed orders are allowed, which is equivalent to constraints (33) forbidding preemption. The three virtual vertices v_{s1}, v_{s2}, and v_{s3}, of which one, i.e., v_{s2}, can only be reached via the other two virtual vertices, ensure a well-defined beginning and end of the processing sequence, not representing any order. Thereby, two cluster visits are necessary, i.e., $\{v_{s1}, v_{s2}\}$ and $\{v_{s2}, v_{s3}\}$. Additionally, from the start-point, the tour reaches the first vertex representing an order, again making use of one additional cluster visit. The same is true for the last processed order, from which the tour must be led to the virtual endpoint of the sequence. In between, $|O'| - c' + 1$ cluster visits are allowed, such that before switching the cluster, all vertices corresponding to at least one order must be visited, i.e., each order must be processed without preemption while the whole demand needs to be satisfied. As an outcome of this transformation scheme, the ndCTSP is a generalization of the order and bin sequencing problem. Please note that the transformation, as described, works only if at least two orders are allowed to be processed in parallel ($c' \geq 2$). If only one order can be processed at once ($c' = 1$), a CTSP is at hand, and the transformation scheme described in Appendix A can be applied.

Example: Fig. 3 on the left provides an instance of the order and bin sequencing problem, with three bins, i.e., $\mathcal{B}' = \{1, 2, 3\}$, four orders, i.e., $o'_1 = \{1\}$, $o'_2 = \{3\}$, $o'_3 = \{2\}$, and $o'_4 = \{1, 2\}$, and the capacity to process three orders at once, i.e., $c' = 3$. The instance translates into an ndCTSP instance with eight vertices, i.e., three virtual start vertices and one vertex for each demanded bin of each order (see Fig. 3). The vertices referring to a demand, but for different bins, are connected via edges of cost one, while all remaining vertices are connected cost neutrally. Clusters are introduced for all combinations of three orders, since $c' = 3$, resulting in clusters γ_1 to γ_4. Additionally, for each order, two clusters are created involving all vertices of the respective order as well as virtual vertex v_{s1}, i.e., γ_5 to γ_8, or v_{s3}, i.e., γ_9 to γ_{12}, respectively. Finally, two clusters, connecting virtual vertex v_{s2} to v_{s1} and v_{s3}, respectively, are created, i.e., γ_{13} and γ_{14}. By setting $\Gamma = 5 + |O'| - c' = 6$, the transformation is completed. Solving the ndCTSP instances results in tour $\pi = (v_{s1}, v_{s2}, v_{s3}, v_{o'_4,1}, v_{o'_1,1}, v_{o'_4,2}, v_{o'_3,2}, v_{o'_2,3})$, and respective bin sequence $1, 2, 3$, both associated with an objective value of two.

The three examples provided above and in Appendix A show the flexibility of the ndCTSP, incorporating problem characteristics from a broad domain of scheduling problems. Researching the ndCTSP can help to better understand characteristics of manifold problems, and providing performant solution procedures can benefit multiple problem domains in parallel. The ndCTSP, generalizing heterogeneous optimization problems, therefore, can be positioned as a relevant variation of the TSP, worth further research. First solution procedures, namely the provided (M)ILP formulations, are benchmarked in the following.

4. Computational performance

This section is dedicated to computational tests regarding the ndCTSP. Section 4.1 summarizes the generation methodology of the novel set of generic ndCTSP instances provided with the paper. Further, Section 4.2 provides performance benchmarks of the three presented (M)ILP models based on the generated dataset for the unlimited and the limited ndCTSP. In Section 4.3, two models known from the literature solving the order and bin sequencing problem (see Füßler and

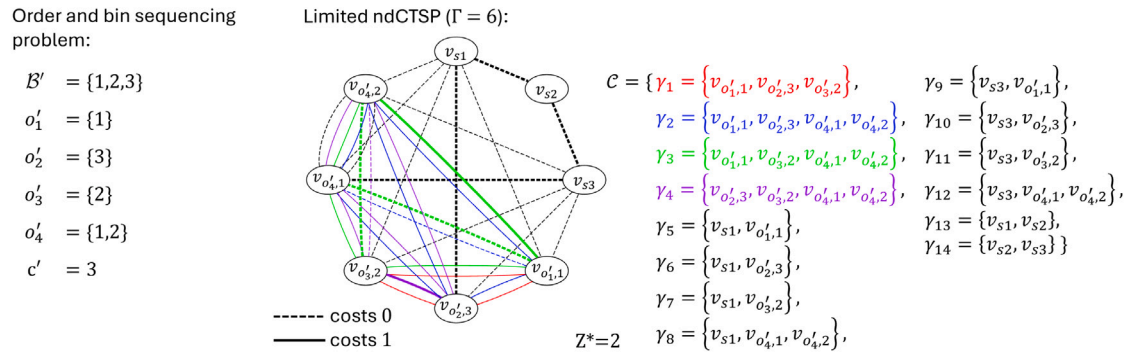


Fig. 3. Example of the transformation from an instance of the order and bin sequencing problem to an ndCTSP instance.

Boysen, 2019; Ouzidan et al., 2022) are benchmarked against ArcClaus-MILP, solving ndCTSP instances transformed from respective order and bin sequencing instances. These instances are, consequently, highly structured and showcase how the ndCTSP can be employed to solve problems from different domains.

All computations have been performed on a PC with an Intel i9-12900KF CPU (16 cores with up to 5.2 GHz), 128 GB of RAM, and Windows 11 Pro as an operating system. Software has been implemented in C# (.NET 8.0 framework), and Gurobi (version 12.0.1) has been applied with a time limit of 10 min per instance for solving mathematical optimization models using all available cores. While Gurobi is presented the complete compact ArcClaus-MILP initially, the exponentially sized formulations, i.e., EdgeDFJ-ILP and ArcDFJ-MILP, are implemented via call-back functions and lazy constraints. Specifically, the (mixed) integer model is presented initially without exponential constraints (3), (5), and (15), avoiding subtours or entering a cluster twice. Every time the solver attains an optimal integer solution to the current version of the model, violated constraints (3) and (5) or (15) are separated. For the classical DFJ subtour elimination constraints, i.e., constraints (3) and (15), the subtours can be determined via binary variables x in polynomial time, and novel cuts can be introduced accordingly (see, e.g., Pferschy and Staněk, 2017). Next, violations of constraints (5) are separated based on the known (sub)tour(s). Following the tour(s), one can efficiently check for clusters entered more than once. Having identified such clusters, constraints (5) are introduced for each subset of vertices S that represents a connected partial tour (i.e., a path) embedded into that very cluster. More specifically, a constraint of type (5) is generated for each pair of edges in the cluster, with one edge embedded into S and one that is not. All violated constraints are added to the model, which then is solved again. Please note that tests on separating cuts at LP solutions have been performed as well. However, the version of the problem formulation for which Gurobi ensures integrality of variables itself and cuts are only applied to integral optimal solutions significantly outperformed separating cuts at LP solutions with the classic approach of Fischetti et al. (1997). Finally, Orange3 (version 3.38.1) has been used to train the random forest model (see Appendix C) employed to analyze the impact of instance generation parameters on model performance.

4.1. Instance generation

Introducing the ndCTSP, this paper also introduces a first generic dataset of ndCTSP instances. Given that different problem transformations (see Section 3.3 and Appendix A) lead to different structural patterns, the goal of the generic dataset introduced is to provide random, unstructured instances to enable an unbiased performance evaluation of differing approaches. Please note that structured instances are additionally introduced in Section 4.3, based on the order and bin sequencing problem (see Section 3.3).

Table 5
Generation parameters of symmetric and asymmetric test instances.

Parameter	Symbol	Values
Number of vertices	$\|V\|$	{5,10,15,20,25,30,40}
Dimensions	d	{1,2,3}
Average clusters per vertex	κ	{3,4,5,6}
Cluster size relative to $\|V\|$	$[\check{\gamma}_{rel}, \hat{\gamma}_{rel}]$	{[0.2,0.4],[0.25,0.45],[0.3,0.5]}

In order to generate generic symmetric and asymmetric ndCTSP instances, generation parameters as presented in Table 5 need to be specified. As a first step of the generation process, coordinates of the $|V| \in \{5, 10, 15, 20, 25, 30, 40\}$ vertices in a d-dimensional space are uniformly drawn from interval $[1, 500]$ for each dimension. Hereby, $d \in \{1, 2, 3\}$, i.e., the vertices lie on a line, a plane, or in a three-dimensional space. The distances between the vertices within the symmetric instances are then determined as the Euclidean distances between the coordinates of the respective vertices rounded to the next integer. When generating asymmetric instances of the ndCTSP, however, distances are systematically distorted. To do so, random noise Δ_η is drawn uniformly distributed from the interval $[0.0, 0.5]$ for each dimension $\eta \in \{1, \dots, d\}$ independently. Given the vector of coordinates X^i of a vertex i, asymmetric distances are then computed as $c_{i,j} = \text{rnd}\left(\sqrt{\sum_{\eta=1}^{d}(X^i_\eta - X^j_\eta)^2} + \sum_{\eta=1}^{d} \Delta_\eta (X^i_\eta - X^j_\eta)\right)$, where rnd() represents rounding a floating-point number to the closest integer value. The noise values Δ_η can be interpreted as the slope of the dimension η.

For each of the determined coordinates and distances, instances with different average clusters per vertex and average cluster sizes are generated. Thereby, a feasible interval of cluster sizes is determined based on input parameters $[\check{\gamma}_{rel}, \hat{\gamma}_{rel}]$. The absolute values are computed as $\check{\gamma}_{abs} = \max\{2, \text{rnd}(\check{\gamma}_{rel}|V|)\}$ and $\hat{\gamma}_{abs} = \min\{|V|, \text{rnd}(\hat{\gamma}_{rel}|V|)\}$, respectively. Again, function rnd() rounds to the nearest integer value. Based on these absolute values, the number of clusters of the currently generated instance is determined by $|C| = \text{rnd}((2|V|\kappa)/(\check{\gamma}_{abs} + \hat{\gamma}_{abs}))$. For each of the $|C|$ clusters, a cluster size is uniformly drawn from the interval $[\check{\gamma}_{abs}, \hat{\gamma}_{abs}]$. Then, each vertex is randomly becoming a member of one cluster that has not reached its predefined size already, making sure that each vertex is assigned at least one cluster. Afterward, vertices are added to the clusters randomly until they attain their predefined size.

The described procedure is performed in a full-factorial setting, resulting in a test bed of 7 (number of vertices) × 3 (dimensions) × 4 (average clusters per vertex) × 3 (relative cluster sizes) × 2 (symmetric or asymmetric) = 504 instances for the unlimited ndCTSP. Please note, however, that not all instances generated in this manner

guarantee a feasible solution, since clusters are not generated in a structured approach. However, as discussed above, we actively decided for a less structured approach in order to avoid systematic biases, as this paper is the first to investigate ndCTSP. Of the generated instances, 477 are feasible and therefore included in the following analysis, while the remaining 27 have been flagged as infeasible by Gurobi during the computational tests. A detailed overview of the percentages of feasible instances generated per parameter setting is given in Appendix B.

The generated test instances can be employed for the unlimited as well as the limited ndCTSP. For the limited version of the problem, reasonable Γ values can be set in the interval $\Gamma \in \{1, 2, \dots, \min\{|V| - 1, |C| - 1\}\}$. Before starting the performance analysis, the smallest Γ value per instance, still allowing for a feasible solution, is determined experimentally, employing Gurobi to solve the models presented in Section 3.2. Thereby, all models are run with a time limit of 10 min for each considered Γ value, starting with the highest possible value. Γ is decreased iteratively every time none of the models flag the limited instance as infeasible within the time limit. Otherwise, the search is terminated. For our test bed of 477 feasible (unlimited) instances, 1747 (limited) instances with different Γ-values have been deduced.

4.2. Results of performance tests

The first test of the computational study tackles the solution performance of the three proposed models when solving the symmetric unlimited ndCTSP. Table 6 presents the average computational time needed to solve an instance in seconds (sec), the average Gurobi internal gap to the lower bound relative to the bound value (%gap), and the percentage of proven optimal solutions attained (%opt) for all approaches. The results are discriminated by the number of vertices ($|V|$) and the average clusters per vertex (κ), being the two instance generation parameters leading to the highest variance in performance of the three models (see Appendix C). If the models differ in performance for an indicator and instance parameter configuration, the best value is highlighted in bold.

As can be seen, the edge-based approach, with its efficient separation techniques, shows superiority for the whole set of instances, providing proven optimal solutions for all instances up to 30 vertices. Up to 25 vertices, EdgeDFJ-ILP provides these solutions in less than a second on average. Starting from 40 vertices, instances become significantly harder to solve for Gurobi solving the EdgeDFJ-ILP, with computation times increasing and the optimality rate decreasing for more restrictive instances with fewer clusters per vertex, i.e., lower κ-values. For Gurobi solving the ArcDFJ-MILP, the performance decrease is more drastic for these instances. While the approach is mostly able to keep up performance-wise up to 30 vertices, all three performance measures fall significantly behind those of EdgeDFJ-ILP for instances with 40 vertices. The third approach, ArcClaus-MILP, performs worst in our tests. While it is able to keep performance up to about 25 vertices, requiring significantly more computation time, though, it starts to struggle at instance sizes of 30 vertices and is not able to find any proven optimal solution for the more restrictive instances with 40 vertices. In conclusion, regarding the average values of all three performance indicators considered, EdgeDFJ-MILP provides the best results.

While EdgeDFJ-ILP shows superior for solving symmetric instances, it cannot be applied to asymmetric instances. Consequently, only ArcDFJ-MILP and ArcClaus-MILP can be benchmarked for these instances. Table 7 provides results in the same way as is known from Table 6. The results are in line with those attained for the symmetric instances. The solution approaches show a significant increase in computation times starting from 30 to 40 vertices per instance. For all instance parameters, ArcDFJ-MILP outperforms ArcClaus-MILP, which, however, still provides robust results. It can further be observed that both models show slightly better performance for asymmetric instances. Seemingly, Gurobi is able to exploit the reduced symmetry of the instances introduced by asymmetric distance values.

The performance of the proposed approaches for symmetric instances of the limited ndCTSP is summarized in Table 8. Thereby, the performance indicators of previous tests are identically reported and, additionally, the %feas-values provide the percentage of instances a feasible solution could be found for, as, different from previous tests, a feasible solution could not be attained for all instances. Γ-values are provided as a normalized measure, i.e., $\Gamma_{rel}^{norm} = \frac{\Gamma}{\min\{|V|-1,|C|-1\}}$. This allows better comparison of instances of different sizes. Moreover, to better accumulate results, parameter Γ_{rel}^{norm} is discretized in three intervals of size 1/3: low ($\Gamma_{rel}^{norm} \leq 1/3$), medium ($1/3 < \Gamma_{rel}^{norm} \leq 2/3$), and high ($\Gamma_{rel}^{norm} > 2/3$). Besides the parameter related to the limitation, results are further discriminated by the number of vertices $|V|$, being the feature with the highest impact on the solution quality of the approaches (see Appendix C).

In line with previous tests, Table 8 shows that all models' performance highly depends on the number of vertices of an instance. Again, larger instances, naturally, become harder to solve. Further, all models in general benefit from less restrictive limitations, i.e., larger Γ_{rel}^{norm} values. Only for Edge-DFJ-ILP and the largest instances considered in the test, i.e., those with 30 and 40 vertices, Γ_{rel}^{norm} = medium consistently showed to be most challenging. Interestingly, the EdgeDFJ-ILP could strengthen its profile in these tests. It outperforms competing models in all performance measures. Starting from 20 vertices and medium or low Γ_{rel}^{norm}-values, it is the only model that reliably provides feasible solutions. While the other models already struggle with feasibility, EdgeDFJ-ILP provides proven optimal solutions up to 30 vertices and high or low Γ_{rel}^{norm}-values. For instances with 40 vertices, the approach is still able to provide feasible solutions for all instances, but the internal gap starts to increase and the optimality rate shrinks. The remaining models can be ranked as in previous tests, with ArcDFJ-MILP providing better results than ArcClaus-MILP. Please note that %gap values are difficult to compare once the approaches do not find feasible solutions to all instances anymore. For 40 vertices and Γ_{rel}^{norm} = medium, ArcDFJ-MILP scores the lowest average gap; however, it is computed on the easiest 28% of the instances only. The results for all models indicate that the limited ndCTSP is considerably harder to solve than the unlimited version, especially for very restrictive Γ values.

The previous test on the unlimited symmetric ndCTSP (see Table 6) showed that proven optimal solutions can be found for instance sizes of up to 30 vertices by the current models within a time limit of 10 min. However, sometimes, larger instances or faster computation times are needed. In order to provide a first test on heuristic results, we test the solution quality of Gurobi solving the ArcClaus-MILP after significantly shorter computation time. The advantage of the ArcClaus-MILP over its two competitors is that it comes from feasible solutions by nature. While cut-based models, e.g., using the DFJ subtour elimination technique, start from a lower bound and attain feasibility via adding respective cuts, already the first solution found by the ArcClaus-MILP is feasible and acts as an upper bound based on which the solver is trying to find improvements. Having restrictive time limits for the procedure, this model results in a higher probability of providing a feasible solution, such that we base our search of heuristic solutions on ArcClaus-MILP. In order to support Gurobi with the aim of finding good solutions instead of proving optimality, Gurobi has been executed with parameters `Presolve=0` and `MIPFocus=1`, as this showed to be the most performant parameter combination in preliminary tests.

The tests are conducted with time limits of 5, 10, 15, 30, 60, 90, 120, 180, 240, and 300 s for the 72 symmetric unlimited instances with 30 and 40 vertices, i.e., for those symmetric instances for which not all optimal solutions could be found within 10 min by the ArcClaus-MILP anymore. During the tests, the percentage of instances for which a feasible (%feas) and optimal solution (%opt) could be found within the respective time limit, as well as the average percentage

Table 6
Average computational time per instance in seconds (s), average Gurobi internal gap to lower bound (%gap), and percentage of achieved proven optimal solutions (%opt) by ArcDFJ-MILP, ArcClaus-MILP, and EdgeDFJ-ILP, discriminated by the number of vertices ($|V|$) and average clusters per vertex (κ) for the symmetric dataset.

$\|V\|$	κ	ArcDFJ-MILP			ArcClaus-MILP			EdgeDFJ-ILP		
		sec	%gap	%opt	sec	%gap	%opt	sec	%gap	%opt
5	3	0.002	0	100	0.001	0	100	**<0.001**	0	100
	4	0.005	0	100	0.003	0	100	**0.002**	0	100
	5	0.007	0	100	0.004	0	100	**<0.001**	0	100
	6	0.003	0	100	0.009	0	100	**<0.001**	0	100
10	3	0.018	0	100	0.022	0	100	**0.001**	0	100
	4	0.012	0	100	0.020	0	100	**0.004**	0	100
	5	0.012	0	100	0.019	0	100	**<0.001**	0	100
	6	0.013	0	100	0.020	0	100	**<0.001**	0	100
15	3	0.079	0	100	0.183	0	100	**0.018**	0	100
	4	0.049	0	100	0.097	0	100	**0.006**	0	100
	5	0.047	0	100	0.096	0	100	**0.006**	0	100
	6	0.064	0	100	0.147	0	100	**0.002**	0	100
20	3	0.288	0	100	1.574	0	100	**0.073**	0	100
	4	0.322	0	100	1.445	0	100	**0.027**	0	100
	5	0.474	0	100	1.868	0	100	**0.033**	0	100
	6	0.206	0	100	0.827	0	100	**0.018**	0	100
25	3	8.696	0	100	75.191	0	100	**0.257**	0	100
	4	3.170	0	100	23.064	0	100	**0.164**	0	100
	5	2.830	0	100	7.594	0	100	**0.104**	0	100
	6	2.796	0	100	11.191	0	100	**0.105**	0	100
30	3	16.369	**0**	**100**	301.814	0.76	77.8	**3.701**	**0**	**100**
	4	43.609	**0**	**100**	312.381	1.09	66.7	**4.132**	**0**	**100**
	5	32.830	**0**	**100**	211.100	0.82	77.8	**2.201**	**0**	**100**
	6	8.061	0	100	35.864	0	100	**0.860**	0	100
40	3	476.752	5.96	22.2	600.452	18.98	0	**244.596**	**1.45**	**66.7**
	4	534.351	9.20	22.2	600.559	27.33	0	**226.534**	**1.09**	**66.7**
	5	540.215	8.12	33.3	600.572	16.93	0	**145.173**	**0.32**	**88.9**
	6	452.722	7.00	33.3	485.025	6.78	33.3	**38.861**	**0**	**100**
Total avg		75.857	1.15	89.0	116.826	2.76	83.1	**23.817**	**0.11**	**97.0**

Table 7
Average computational time per instance in seconds (sec), average Gurobi internal gap to lower bound (%gap), and percentage of achieved proven optimal solutions (%opt) by ArcDFJ-MILP and ArcClaus-MILP, discriminated by the number of vertices ($|V|$) and average clusters per vertex (κ) for the asymmetric dataset.

$\|V\|$	κ	ArcDFJ-MILP			ArcClaus-MILP		
		sec	%gap	%opt	sec	%gap	%opt
5	3	0.003	0	100	**0.002**	0	100
	4	**0.002**	0	100	0.003	0	100
	5	**0.002**	0	100	0.007	0	100
	6	**0.002**	0	100	0.003	0	100
10	3	0.019	0	100	0.019	0	100
	4	**0.017**	0	100	0.033	0	100
	5	**0.012**	0	100	0.023	0	100
	6	**0.015**	0	100	0.027	0	100
15	3	**0.065**	0	100	0.193	0	100
	4	**0.076**	0	100	0.131	0	100
	5	**0.081**	0	100	0.180	0	100
	6	**0.069**	0	100	0.147	0	100
20	3	**0.180**	0	100	1.322	0	100
	4	**0.394**	0	100	1.841	0	100
	5	**0.227**	0	100	1.075	0	100
	6	**0.160**	0	100	0.675	0	100
25	3	**0.826**	0	100	11.439	0	100
	4	**2.386**	0	100	23.263	0	100
	5	**1.695**	0	100	9.952	0	100
	6	**0.426**	0	100	1.635	0	100
30	3	**32.386**	**0**	**100**	308.658	1.30	77.8
	4	**20.030**	**0**	**100**	200.844	0.18	88.9
	5	**85.650**	**0.58**	88.9	149.476	1.61	88.9
	6	**7.619**	0	100	23.990	0	100
40	3	**382.990**	**4.67**	**44.4**	600.308	12.72	0
	4	**475.902**	**5.18**	**33.3**	562.437	13.97	11.1
	5	**445.805**	**4.63**	**33.3**	600.327	6.52	0
	6	**374.464**	**3.74**	**44.4**	477.890	4.34	33.3
Total avg		**65.411**	**0.71**	**90.4**	106.282	1.52	85.0

Table 8
Average computational time per instance in seconds (sec), average Gurobi internal gap to lower bound (%gap), percentage of achieved proven optimal solutions (%opt), and percentage of instances for which feasible solutions could be found (%feas) by ArcDFJ-MILP, ArcClaus-MILP, and EdgeDFJ-ILP, discriminated by the number of vertices ($|V|$) and normalized Γ-values (Γ_{rel}^{norm}) for the symmetric dataset.

$\|V\|$	Γ_{rel}^{norm}	ArcDFJ-MILP				ArcClaus-MILP				EdgeDFJ-ILP			
		sec	%gap	%opt	%feas	sec	%gap	%opt	%feas	sec	%gap	%opt	%feas
10	High	0.020	0	100	100	0.031	0	100	100	**0.003**	0	100	100
	Medium	0.059	0	100	100	0.115	0	100	100	**0.013**	0	100	100
	Low	0.268	0	100	100	0.641	0	100	100	**0.027**	0	100	100
15	High	0.135	0	100	100	0.226	0	100	100	**0.011**	0	100	100
	Medium	1.365	0	100	100	4.547	0	100	100	**0.045**	0	100	100
	Low	38.750	**0**	**100**	100	144.054	0.95	92.9	100	**0.091**	**0**	**100**	100
20	High	1.005	0	100	100	3.083	0	100	100	**0.052**	0	100	100
	Medium	29.273	**0**	99.2	99.2	96.518	0.61	95.3	99.2	**0.165**	**0**	**100**	**100**
	Low	423.444	9.97	41.9	96.8	557.494	15.13	12.9	93.5	**0.336**	**0**	**100**	**100**
25	High	19.050	0.07	98.8	100	83.179	0.37	95.2	100	**0.392**	**0**	**100**	100
	Medium	225.586	2.39	74.8	96.4	376.050	6.32	48.2	94.2	**1.391**	**0**	**100**	**100**
	Low	568.429	12.88	15.6	43.8	600.228	16.55	0	40.6	**1.487**	**0**	**100**	**100**
30	High	133.425	0.96	89.7	**100**	357.146	3.44	58.1	99.4	**13.206**	**0**	**100**	**100**
	Medium	507.885	9.81	21.6	88.8	558.290	15.87	11.2	80.2	**58.263**	**0**	**96.6**	**100**
	Low	600.194	6.69	0	23.8	600.410	3.24	0	9.5	**23.598**	**0**	**100**	**100**
40	High	529.006	14.50	20.1	96.2	584.724	14.77	4.4	81.8	**238.593**	**1.65**	**71.7**	**100**
	Medium	596.957	**6.16**	0.9	28.3	600.400	7.30	0	18.6	**353.663**	8.22	**52.2**	**100**
	Low	600.258	–	–	–	600.397	–	–	–	**214.710**	**2.66**	**83.3**	**100**
Total avg		184.299	3.15	73.6	91.2	240.815	4.38	64.9	88.2	**51.594**	**0.71**	**94.0**	**100**

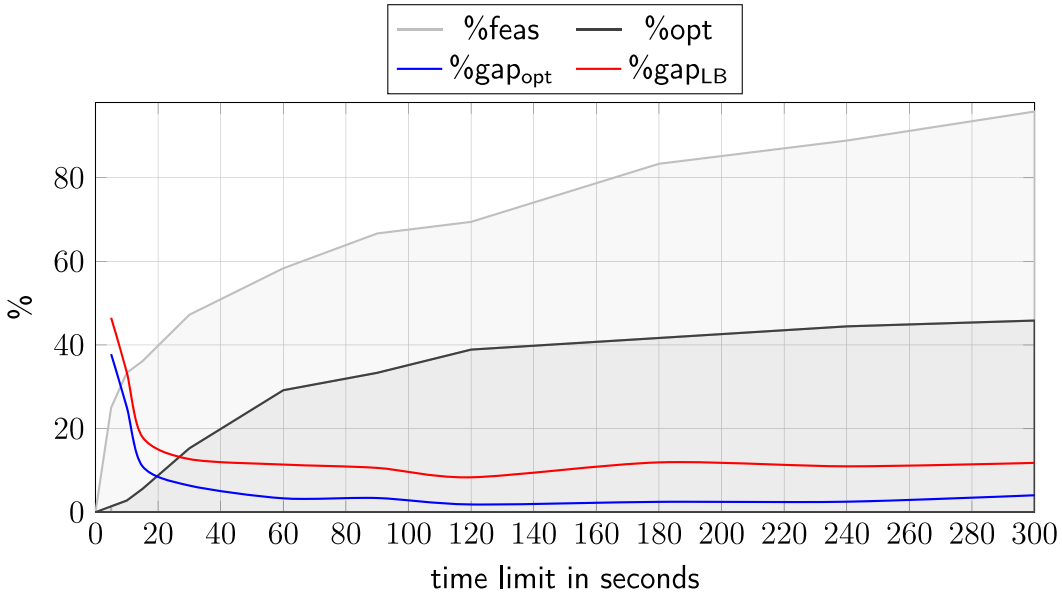


Fig. 4. Percentage of instances solved to feasibility (%feas) and optimality (%opt), as well as average gap to Gurobi internal lower bound ($\%gap_{LB}$) and, if available, known optimal solution ($\%gap_{opt}$).

gap to the Gurobi internal lower bound ($\%gap_{LB}$) and to the optimal solution ($\%gap_{opt}$), if known by previous computations, are tracked. All instances without known optima are thereby excluded from the computation of $\%gap_{opt}$. The performance indices are provided in Fig. 4.

As can be seen, feasible solutions of good quality can be attained in significantly less than 10 min. As expected, the number of feasible and optimal solutions found increases fast at the beginning, but the marginal utility of additional computation time gets less pronounced for longer time limits. Also, in general, solution quality, regarding measures $\%gap_{opt}$ and $\%gap_{LB}$, increases with longer computation times, naturally. The slight decrease in quality, to be observed at several positions within Fig. 4, might seem counterintuitive at first impression but stems from more feasible solutions found, where the newly found solutions are of low quality but can be improved when investing additional computation time. After only three minutes, 83% of the instances are solved to feasibility, 42% even to optimality. While the internal gap to the lower bound still amounts to 11.9% after three minutes, the gap to the real (known) optima is only 2.5% on average. So, it can be concluded that, if faster computation times are needed and heuristic solutions are acceptable, shorter time limits of the ArcClaus-MILP are a suitable first approach. Still, specialized heuristics are for sure a relevant field of future research, especially for larger instance sizes.

4.3. Computational tests on instances of the order and bin sequencing problem

To provide insights into the practical application of the ndCTSP, this section is dedicated to a benchmark on instances of the order and bin sequencing problem based on the paper of Füßler and Boysen (2019) and shown to be a special case of the ndCTSP in Section 3.3.

Table 9
Average computational time per instance in seconds (sec), average Gurobi internal gap to lower bound (%gap), and percentage of achieved proven optimal solutions (%opt) by the original OBSP-MILP by Füßler and Boysen (2019), the ILP proposed by Ouzidan et al. (2022), and ArcClaus-MILP solving the corresponding ndCTSP instance discriminated by the order capacity (c') and the number of orders ($|O'|$).

c'	$\|O'\|$	Füßler and Boysen (2019)			Ouzidan et al. (2022)			ArcClaus-MILP		
		sec	%gap	%opt	sec	%gap	%opt	sec	%gap	%opt
2	4	0.725	0	100	0.717	0	100	**0.079**	0	100
2	6	186.157	5.71	80	130.758	2.86	80	**0.529**	**0**	**100**
2	8	413.186	34.76	40	396.276	20.16	40	**46.638**	**0**	**100**
2	10	600.151	74.47	0	600.058	47.79	0	**285.458**	**7.33**	**60**
3	4	0.472	0	100	0.624	0	100	**0.075**	0	100
3	6	165.416	5.71	80	133.430	2.86	80	**0.246**	**0**	**100**
3	8	449.444	34.76	40	481.020	24.13	20	**2.224**	**0**	**100**
3	10	600.277	72.84	0	600.087	48.29	0	**28.205**	**0**	**100**
Total avg		301.979	28.53	55.0	292.871	18.26	52.5	**45.432**	**0.92**	**95.0**

The instances have been newly generated following the scheme presented in the original paper. Generation parameters have been chosen such that proven optimal solutions could be found for most instances. Firstly, the number of orders is varied between 4, 6, 8, or 10 orders. For each generated order, then, a number of demanded products is drawn uniformly distributed from the integer interval $[1, 3]$. The specific products demanded are, again uniformly distributed, drawn from ten different products. For each number of orders, five different instances are generated, each solved with an order capacity c' of two and three, resulting in a total testbed of forty instances.

Each of the forty instances is then solved by the original model proposed by Füßler and Boysen (2019) and cited in Section 3.3, as well as the model proposed by Ouzidan et al. (2022). After transforming each order bin sequencing problem instance into an ndCTSP instance (see transformation scheme in Section 3.3), it is again solved by the ArcClaus-MILP. Thereby, the selection of the ndCTSP model is based on employing it to find good-quality heuristic solutions in previous tests. As in the main study, a time limit of 10 min has been applied for solving the models via Gurobi.

The results of the computational tests are aggregated in Table 9, discriminated by the number of orders maximally to be processed in parallel c' and the number of orders in total $|O'|$. As can be seen, the original order and bin sequencing problem model by Füßler and Boysen (2019) is able to find all proven optimal solutions for four orders but starts failing to do so within the time limit starting from six orders for both capacity values c'. For the largest examined instances, with ten orders only, the model is not able to find any proven optimal solutions, and the internal gap of Gurobi amounts to more than 70% on average. The model of Ouzidan et al. (2022) yields a slightly decreased number of proven optimal solutions. However, it outperforms the original model of Füßler and Boysen (2019) in terms of the computation time required and the internal gap, which is in line with the computational results presented in Ouzidan et al. (2022). The ArcClaus-MILP, in comparison, can attain proven optimal solutions for up to eight orders given an order capacity of two and for all instances if the order capacity is three. The average Gurobi internal gap amounts to less than 1% with an average computation time of only 45 s. Following previous results, instances become easier to solve for the ArcClaus-MILP with increasing order batch size c', resulting in a higher average number of clusters per vertex (compare κ-values in Tables 6 and 7). Concludingly, the novel proposed ArcClaus-MILP is able to significantly outperform both specialized models presented for the order and bin sequencing problem for the surveyed instance set, showcasing the potential of the new modeling approach and tailor-made procedures to solve the ndCTSP.

5. Conclusion

The paper at hand introduces the non-disjoint Clustered TSP (ndCTSP). A newly discovered TSP variant, for which vertices can be a member of several clusters and a shortest tour is searched, for which all vertices are visited, but the tour is always embedded into a cluster, each of which is entered at most once. For the novel problem, two variants, the limited and the unlimited ndCTSP, a formal definition, and an NP-hardness proof are provided, as well as three (mixed) integer linear programs. These are compared in a broad computational study, for which also, to the best of the authors' knowledge, the first structured instance set for the ndCTSP with more than 2000 instances is generated. For different instances and purposes, the different models are superior. Symmetric instances can be solved best by EdgeDFJ-ILP, while for asymmetric instances ArcDFJ-MILP prevails. If heuristic solutions are acceptable, ArcClaus-MILP is a suitable option. Finally, relations to other fields of research are shown for three exemplary cases, i.e., the CTSP, the order and bin sequencing problem in automated warehouses, and the exam scheduling problem in educational settings. In a computational test, the superiority of solving ndCTSP over the originally proposed models (see Füßler and Boysen, 2019; Ouzidan et al., 2022) is showcased, pronouncing the practical relevance of ndCTSP.

Future research should identify even more relations to other optimization domains. In order to exploit the ndCTSP as a core problem in many areas, however, research on high-performing solution procedures is needed. Especially the field of tailor-made heuristics seems to be a promising research area in this context. Developing exact procedures, new cuts could be identified and tested, leading to higher performance of the (M)ILPs or related branch-and-cut procedures, for example. But also, more specialized procedures tailored to specific ndCTSP variants could be a worthwhile endeavor.

CRediT authorship contribution statement

Felix Weidinger: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Kris Braekers:** Writing – review & editing, Validation, Supervision, Methodology, Formal analysis.

Appendix A. Further relations to other planning problems

This appendix, supplementing Section 3.3, highlights further relationships between the ndCTSP and other scheduling problems, namely, the CTSP and the exam scheduling problem.

CTSP: Given the CTSP (see Section 2 and Chisman, 1975), all vertices are a member of exactly one cluster. As a consequence, the clusters are disjoint, and the tour, therefore, must, other than for the ndCTSP, leave the currently active cluster before a new one is entered. Given a graph $G' = (V', E', c')$ and a set of clusters $\mathcal{C}'$, defining an instance of CTSP, the respective data can be transformed into an instance of the limited ndCTSP, i.e., graph $G = (V, E, c)$, clusters $\mathcal{C}$, and parameter Γ, as follows:

- First, the set of vertices is copied from V' to V. Edges in E' and their respective cost values c' are identically added to E and c of the ndCTSP instance, respectively.
- Clusters C' are transferred slightly manipulated to the ndCTSP instance. All clusters $\gamma' \in C' : |\gamma'| \neq 2$ are just copied to the set of clusters of the ndCTSP instance, i.e., C. For each cluster $\gamma' \in C' : |\gamma'| = 2$, an additional virtual vertex $v^+_{\gamma'}$ is added to V and to the corresponding cluster, such that $\gamma = \{i, j, v^+_{\gamma'}\}$, and connected by edges $\{i, v^+_{\gamma'}\}$ of costs $c_{\{i,v^+_{\gamma'}\}} = c_{\{i,j\}}$ and $\{v^+_{\gamma'}, j\}$ of costs zero to the other two vertices within the same cluster. Then, it is added in its manipulated form γ to C. Finally, the set of clusters is extended by all possible clusters hosting exactly two vertices from different clusters in C', i.e., clusters $\{i, j\} : i, j \in V'$ and $\exists \gamma' \in C' : i \in \gamma'$ and $j \notin \gamma'$ are introduced. We refer to them as transfer clusters in the following.
- In a last step, parameter Γ is set to two times the number of clusters of the CTSP instance $|C'|$ reduced by the number of clusters of size one in C', i.e., $\Gamma = 2|C'| - |\{\gamma' \in C' : |\gamma'| = 1\}|$.

The transformation scheme ensures that each cluster of the original CTSP with a cardinality higher than one, i.e., each cluster in $\{\gamma' \in C' : |\gamma'| > 1\}$, is entered, as the tour can only be conducted between vertices of different (original) clusters C' via the newly added transfer clusters. As a result, $|C'|$ transfer clusters need to be visited in order to switch between the (original CTSP) clusters, allowing only $|C'| - |\{\gamma' \in C' : |\gamma'| = 1\}|$ additional cluster visits. Vertices assigned to an original CTSP cluster of cardinality one can be visited and left via transfer clusters. For that reason, they do not induce an increase in the Γ-value during the transformation. To be able to visit the remaining vertices, the tour can then only be embedded into clusters corresponding with the original CTSP clusters, all with a cardinality larger than two. Therefore, feasible (optimal) solutions to the resulting ndCTSP instance also represent feasible (optimal) solutions to the original CTSP instance, establishing ndCTSP as a generalization of the CTSP.

Example: Fig. 5 on the left depicts a CTSP instance with five vertices, three clusters, and the respective cost values on the right. Vertices, cost values, and clusters with a cardinality different from two, i.e., γ'_1, are transferred to the ndCTSP instance. For clusters with two vertices involved, an additional vertex is introduced to the cluster, i.e., vertex 6 is introduced to cluster γ_2 and vertex 7 to cluster γ_3. The new vertices are connected to the remaining vertices of the cluster, once with cost values identical to the costs of the original single inner cluster edge, and once with cost value 0. Therefore, costs for inner cluster edges of cluster γ_2 (γ_3) are set to $c_{\{2,6\}} = c_{\{2,5\}} = 4$ ($c_{\{3,7\}} = c_{\{3,4\}} = 1$) and $c_{\{5,6\}} = 0$ ($c_{\{4,7\}} = 0$). Additionally, novel clusters $\{i, j\}$ $\forall \gamma' \in C', i \in \gamma', j \in V' \setminus \gamma'$ are introduced to C of the ndCTSP instance (see γ_4 to γ_{11}). Finally, the limiting parameter is set to $\Gamma = 2|C'| - 1 = 5$. Given these instance parameters, both, solving the CTSP and solving the ndCTSP, result in identical optimal solutions with objective value 11 and tour $\pi = (1, 4, 7, 3, 2, 6, 5)$ with clusters $\varpi = (\gamma_6, \gamma_3, \gamma_3, \gamma_8, \gamma_2, \gamma_2, \gamma_7)$. Please note that the example also shows why additional vertices must be added to original clusters of cardinality two. Without vertices 6 and 7, an optimal solution of the ndCTSP instance would be tour $\pi = (1, 2, 3, 4, 5)$, with an objective value of 5, visiting original cluster γ'_2 twice, however, such that it cannot be transferred into a CTSP solution.

Exam scheduling: Exam scheduling problems occur in numerous educational settings. In those problems, exams V' must be sequenced, where $\bar{V}'_{k'} \subset V'$ is the subset of exams to be written by student $k' \in K'$. Hereby, special requirements need to be met, either motivated by legal reasons or fairness. In the considered problem variant, based on Balakrishnan (1991), so-called back-to-back conflicts are forbidden, i.e., for all students $k' \in K'$ exams $i', j' \in \bar{V}'_{k'}$ both written by the considered student are not allowed to be assigned to successive time slots. Further, specific exams V'_e (V'_l, V'_{el}) among all exams V' need to be written early (late, early or late) in the examination period, e.g., because of part-time students participating who are limited in their availability. For example, early exams (V'_e) have to be scheduled before any other exam not to be scheduled early ($V' \setminus (V'_e \cup V'_{el})$). A similar reasoning applies for late exams (V'_l), while early or late exams (V'_{el}) have to be scheduled before or after all exams without such constraints ($V' \setminus (V'_e \cup V'_l \cup V'_{el})$). Given these constraints, a feasible schedule must be found. Again, an integer linear program of the problem is given, described by the symbols defined in Table 10 and formulas (40) to (46).

Table 10
Notation for the exam scheduling ILP.

Symbol	Description
V'	Set of exams (indices: i', j')
$V'_e / V'_l / V'_{el}$	Set of early/late/early or late exams
K'	Set of students (index: k')
$\bar{V}'_{k'}$	Set of exams written by student $k' \in K'$
τ'	Number of time slots, with $\tau' = \lvert V' \rvert$ (index: t')
$\tau'_{e/l}$	Number of time slots reserved for early and late exams, with $\tau'_{e/l} = \lvert V'_e \rvert + \lvert V'_{el} \rvert + \lvert V'_l \rvert$
$y'_{i',t'}$	Binary variable: 1, iff exam $i' \in V'$ is scheduled in time t'
$e'_{t'}$	Binary variable: 1, iff time t' is the last with an early exam

$$\sum_{t'=1}^{\tau'} y'_{i',t'} = 1 \qquad \forall i' \in V' \tag{40}$$

$$\sum_{i' \in V'} y'_{i',t'} = 1 \qquad \forall t' \in \{1, \dots, \tau'\} \tag{41}$$

$$\sum_{i' \in \bar{V}'_{k'}} (y'_{i',t'-1} + y'_{i',t'}) \leq 1 \qquad \forall k' \in K'; t' \in \{2, \dots, \tau'\} \tag{42}$$

$$\sum_{t'_2=1}^{t'_1} \sum_{i' \in V'_e \cup V'_{el}} y_{i',t'_2} \geq t'_1 e'_{t'_1} \qquad \forall t'_1 \in \{|V'_e|, \dots, |V'_e| + |V'_{el}|\} \tag{43}$$

$$\sum_{t'_2=\tau'-\tau'_{e/l}+t'_1+1}^{\tau'} \sum_{i' \in V'_{el} \cup V'_l} y_{i',t'_2} \geq (\tau'_{e/l} - t'_1) e'_{t'_1} \qquad \forall t'_1 \in \{|V'_e|, \dots, |V'_e| + |V'_{el}|\} \tag{44}$$

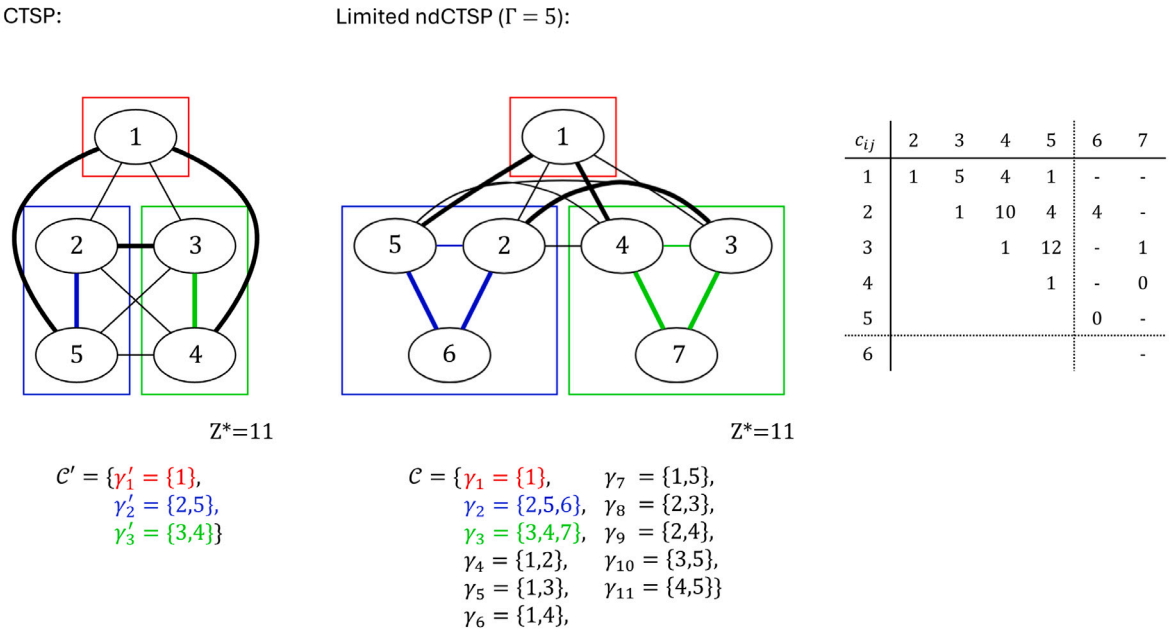
$$\sum_{t'=|V'_e|}^{|V'_e|+|V'_{el}|} e'_{t'} = 1 \tag{45}$$

$$y_{i',t'}, e_{t'} \in \{0, 1\} \qquad \forall i' \in V', t' \in \{1, \dots, \tau'\} \tag{46}$$

Constraints (40) and (41) guarantee that each exam is scheduled in exactly one time slot and vice versa. In constraints (42), back-to-back conflicts are ruled out. Scheduling the early and late exams at respective time slots is ensured by constraints (43) and (44). Hereby, variable $e'_{t'}$ indicates at which specific time slot the early exam period ends, which can only occur once, as ensured by constraints (45). Finally, constraints (46) define the domain of the variables.

The problem can be translated into the unlimited ndCTSP by the following steps:

- The exams V' of the original problem are represented by vertices V of the ndCTSP. Additionally, five virtual vertices are added to V, i.e., one following the last of the early exams v_e, one preceding the first of the late exams v_l, and three starting vertices v_{s1}, v_{s2}, and v_{s3}, as known from the order and bin sequencing problem.
- A first cluster is introduced, including the virtual start vertex v_{s3}, vertex v_e succeeding the vertex related to the last early exam, as well as all vertices representing exams that should or could be scheduled early, i.e., vertices related to V'_e and V'_{el}. The same is repeated for the start vertex v_{s1}, vertex v_l preceding the vertex related to the earliest late exam, and all vertices related to exams to be (possibly) scheduled late, i.e., V'_{el} and V'_l. The remaining exams are added to clusters as follows: a cluster is introduced



c_{ij}	2	3	4	5	6	7
1	1	5	4	1	-	-
2		1	10	4	4	-
3			1	12	-	1
4				1	-	0
5					0	-
6						-

Fig. 5. Example of the transformation from an instance of the CTSP to an ndCTSP instance.

for each exam being part of a back-to-back conflict, i.e., for all $v'_1 \in \bar{V}'_{k'}$ $\forall k' \in K'$ a cluster $\{v'_1, v_e, v_l, v'_2 : v'_2 \in V' \setminus (V'_e \cup V'_{el} \cup V'_l \bigcup_{k' \in K'} \bar{V}'_{k'})\}$ is introduced. Further, to allow scheduling two exams being potentially part of a back-to-back conflict, but belonging to different students $k'_1 \neq k'_2$, in direct succession, those are connected by pairwise clusters, i.e., clusters $\{v'_1, v'_2\} : \nexists k' \in K', v'_1, v'_2 \in \bar{V}'_{k'}$ are generated. Finally, clusters $\{v_{s1}, v_{s2}\}$ and $\{v_{s2}, v_{s3}\}$ are introduced to enforce a well-defined beginning and ending of the sequence.

- In a last step, all possible edges within each cluster are introduced, if not already existing, such that $E = \{\{i, j\} : \forall \gamma \in C \text{ and } i, j \in \gamma \text{ and } i \neq j\}$.

Finding a feasible solution to the resulting unlimited ndCTSP instance is equivalent to finding a feasible solution to the described exam scheduling problem instance, where the sequence of vertices corresponds with a feasible sequence of exams, as only exams that can be scheduled successively share a cluster. Please note that the described approach can be extended to also consider back-to-back conflicts of early and late exams. We, however, assume that such situations do not exist in practice, as part-time students are not enrolled in a large number of exams. For the sake of simplicity, we therefore provide a more handy transformation, illustrating the transformation potential.

Example: An instance of the exam scheduling problem is provided by Fig. 6 on the left. Six exams must be scheduled, whereby exam 1 needs to be scheduled early, exam 2 needs to be scheduled early or late, and exam 6 has to be scheduled late. Additionally, at least one student attends exams 3 and 4 such that those are not allowed to be scheduled in direct succession. The instance can be translated into an ndCTSP instance by introducing eleven vertices, one for each exam, three virtual start vertices, and vertices marking the early and late exam phases, respectively. Six clusters are introduced, one including the virtual start vertex v_{s3}, all vertices related to exams that need to or can be scheduled early, i.e., 1 and 2, and the vertex marking the end of early exams v_e, i.e., γ_1. The next two clusters include one of the vertices representing exams potentially involved in a back-to-back conflict and all remaining exams that are not to be scheduled early or late, as well as the virtual marking vertices, resulting in clusters γ_2 and γ_3. The cluster γ_4 hosts the vertex marking the beginning of the late exam period, i.e., v_l, virtual start vertex v_{s1}, and all vertices related to exams that have to or can be scheduled late. Finally, clusters γ_5 and γ_6 ensure a well-defined sequence of exams, similar to the transformation from the order and bin scheduling problem. A feasible tour of the ndCTSP instance is $\pi = (v_{s1}, v_{s2}, v_{s3}, 1, v_e, 4, 5, 3, v_l, 2, 6)$, representing a feasible exam schedule $(1, 4, 5, 3, 2, 6)$.

Appendix B. Feasibility during instance generation

Table 11 provides detailed data on the feasibility of the generated (unlimited) instances (see Section 4.1). As described, the generation methodology does not ensure feasibility. Table 11 illustrates that it is likely that instances with a restricted number of vertices and clusters are infeasible. The larger the instances get, the higher the chance to generate a feasible instance. In general, only about 5.4% of the generated unlimited instances are infeasible, leading to a test bed of 477 feasible instances.

Appendix C. Feature importance of the three considered (M)ILPs for the unlimited symmetric ndCTSP

In order to better understand the factors influencing Gurobi's performance in solving the models, a random forest regression model (see, e.g., Breiman, 2001) has been trained on the data. Instance generation parameters, i.e., the number of vertices $|V|$, the number of dimensions d, the average number of clusters per vertex κ, and the relative average cluster size $\hat{\gamma}/\check{\gamma}_{rel}$, have been provided as features. The computation time of the thereby defined instance acted as the target variable. The random forest has been trained with 20 trees, three randomly selected features per tree, and a minimum leaf size of five data points. In a second step, the feature importance of the trained model has been evaluated to better understand which instance characteristic has the highest impact on the solution performance. Feature importance is measured by permuting the values of the respective feature five times while measuring the performance decrease based on the coefficient of determination, also known as the R^2-value, which is an established measure in regression analysis. It provides the percentage of variance of the target variable that is described via the respective feature, given

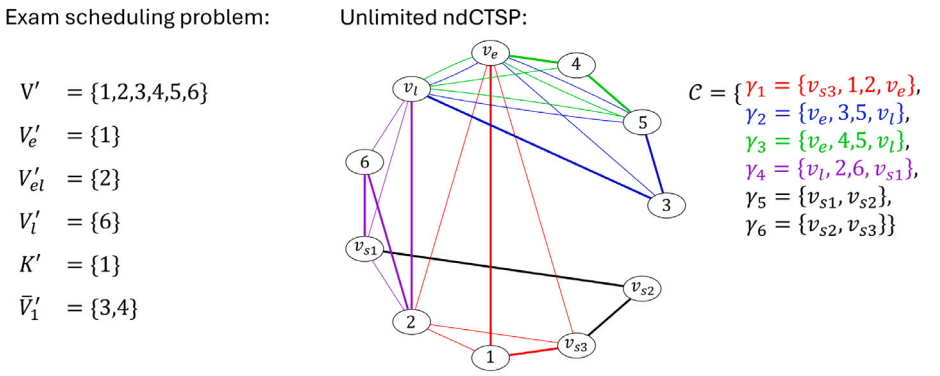


Fig. 6. Example of the transformation from an instance of the exam scheduling problem to an ndCTSP instance.

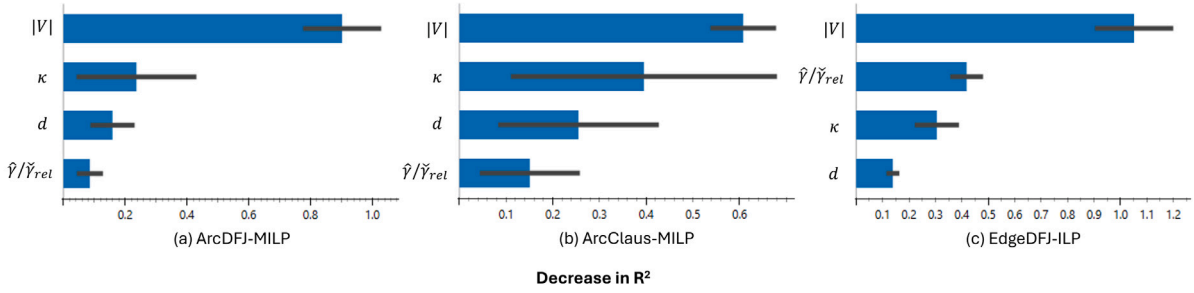


Fig. 7. Feature importance of the three considered models for solving the unlimited symmetric ndCTSP.

Table 11
Number of feasible instances per parameter combination (maximum of 2).

		κ	3			4			5			6		
		$\check{\gamma}_{rel}$	0.2	0.25	0.3	0.2	0.25	0.3	0.2	0.25	0.3	0.2	0.25	0.3
$\|V\|$	d													
5	1		1	0	0	1	1	0	2	2	2	2	2	2
	2		1	2	1	2	2	1	0	0	2	2	2	1
	3		2	0	0	2	1	2	2	2	2	2	2	2
10	1		1	2	2	2	2	2	2	2	2	2	2	2
	2		1	2	2	2	2	2	2	2	2	2	2	2
	3		1	0	2	2	2	2	2	2	2	2	2	2
15	1		2	2	2	2	2	2	2	2	2	2	2	2
	2		2	2	2	2	2	2	2	2	2	2	2	2
	3		2	2	2	2	2	2	2	2	2	2	2	2
20	1		2	2	2	2	2	2	2	2	2	2	2	2
	2		2	2	2	2	2	2	2	2	2	2	2	2
	3		2	2	2	2	2	2	2	2	2	2	2	2
25	1		2	2	2	2	2	2	2	2	2	2	2	2
	2		2	2	2	2	2	2	2	2	2	2	2	2
	3		2	2	2	2	2	2	2	2	2	2	2	2
30	1		2	2	2	2	2	2	2	2	2	2	2	2
	2		2	2	2	2	2	2	2	2	2	2	2	2
	3		2	2	2	2	2	2	2	2	2	2	2	2
40	1		2	2	2	2	2	2	2	2	2	2	2	2
	2		2	2	2	2	2	2	2	2	2	2	2	2
	3		2	2	2	2	2	2	2	2	2	2	2	2

a fitted model, i.e., a high decrease in R^2 in the case of permutation of the feature values (see plots in Fig. 7) indicates that the respective feature is important to the outcome, i.e., the average computational time to solve an instance. As can be seen, the instance feature with the by far highest impact on the solution performance of all models is the number of vertices $|V|$, with more vertices and, therefore, larger instances, leading to longer computation times (see Table 6). For the two arc-based models, then, the second most important feature is the average number of clusters per vertex κ. For the EdgeDFJ-ILP, the relative cluster size $\hat{\gamma}/\check{\gamma}_{rel}$ appears to be slightly more influential than the average number of clusters per vertex κ. However, the margin is thin. In conclusion, the number of vertices $|V|$ and the average number of clusters a vertex is a member of κ seem to influence the performance of Gurobi the most when averaging over the presented models.

Data availability

The instance data required to reproduce the above findings is available to download from https://doi.org/10.5281/zenodo.20489019.

References

Angelelli, E., Archetti, C., Vindigni, M., 2014. The clustered orienteering problem. European J. Oper. Res. 238 (2), 404–414.

Anily, S., Bramel, J., Hertz, A., 1999. A 53-approximation algorithm for the clustered traveling salesman tour and path problems. Oper. Res. Lett. 24 (1–2), 29–35.

Bagchi, T.P., Gupta, J.N., Sriskandarajah, C., 2006. A review of TSP based approaches for flowshop scheduling. European J. Oper. Res. 169 (3), 816–854.

Balakrishnan, N., 1991. Examination scheduling: A computerized application. Omega 19 (1), 37–41.

Baniasadi, P., Foumani, M., Smith-Miles, K., Ejov, V., 2020. A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. European J. Oper. Res. 285 (2), 444–457.

Bock, S., Bomsdorf, S., Boysen, N., Schneider, M., 2025. A survey on the traveling salesman problem and its variants in a warehousing context. European J. Oper. Res. 322 (1), 1–14.

Boysen, N., De Koster, R., 2025. 50 years of warehousing research - an operations research perspective. European J. Oper. Res. 320 (3), 449–464.

Boysen, N., De Koster, R., Weidinger, F., 2019. Warehousing in the e-commerce era: A survey. European J. Oper. Res. 277 (2), 396–411.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.

Chisman, J.A., 1975. The clustered traveling salesman problem. Comput. Oper. Res. 2 (2), 115–119.

Claus, A., 1984. A new formulation for the travelling salesman problem. SIAM J. Algebr. Discret. Methods 5 (1), 21–25.

Dantzig, G., Fulkerson, R., Johnson, S., 1954. Solution of a large-scale traveling-salesman problem. J. Oper. Res. Soc. Am. 2 (4), 393–410.

Deckerová, J., Váňa, P., Faigl, J., 2024. Combinatorial lower bounds for the generalized traveling salesman problem with neighborhoods. Expert Syst. Appl. 258, 125185.

Dewil, R., Vansteenwegen, P., Cattrysse, D., Van Oudheusden, D., 2015. A minimum cost network flow model for the maximum covering and patrol routing problem. European J. Oper. Res. 247 (1), 27–36.

dos Santos Teixeira, E., de Araujo, S.A., 2024. Formulations for the clustered traveling salesman problem with d-relaxed priority rule. Comput. Oper. Res. 161, 106402.

Drexl, M., Schneider, M., 2015. A survey of variants and extensions of the location-routing problem. European J. Oper. Res. 241 (2), 283–308.

Fischetti, M., González, J.J.S., Toth, P., 1995. The symmetric generalized traveling salesman polytope. Networks 26 (2), 113–123.

Fischetti, M., Salazar González, J.J., Toth, P., 1997. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. Oper. Res. 45 (3), 378–394.

Fischetti, M., Toth, P., Vigo, D., 1994. A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. Oper. Res. 42 (5), 846–859.

Flood, M.M., 1956. The traveling-salesman problem. Oper. Res. 4 (1), 61–75.

Füßler, D., Boysen, N., 2019. High-performance order processing in picking workstations. EURO J. Transp. Logist. 8 (1), 65–90.

Gamboa, D., Rego, C., Glover, F., 2006. Implementation analysis of efficient heuristic algorithms for the traveling salesman problem. Comput. Oper. Res. 33 (4), 1154–1172.

Garey, M., Johnson, D., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co.

Garfinkel, R.S., Gilbert, K.C., 1978. The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. J. ACM 25 (3), 435–448.

Gouveia, L., Pires, J.M., 1999. The asymmetric travelling salesman problem and a reformulation of the Miller–Tucker–Zemlin constraints. European J. Oper. Res. 112 (1), 134–146.

Gutin, G., Punnen, A.P., 2006. The Traveling Salesman Problem and Its Variations, vol. 12, Springer Science & Business Media.

Hà, M.H., Nguyen Phuong, H., Tran Ngoc Nhat, H., Langevin, A., Trépanier, M., 2022. Solving the clustered traveling salesman problem with-relaxed priority rule. Int. Trans. Oper. Res. 29 (2), 837–853.

Helsgaun, K., 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. European J. Oper. Res. 126 (1), 106–130.

Henry-Labordere, A., 1969. The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem. RAIRO 43–49.

Kara, I., Guden, H., Koc, O.N., 2012. New formulations for the generalized traveling salesman problem. In: Proceedings of the 6th International Conference on Applied Mathematics, Simulation, Modelling, ASM. vol. 12, pp. 60–65.

Laporte, G., Asef-Vaziri, A., Sriskandarajah, C., 1996. Some applications of the generalized travelling salesman problem. J. Oper. Res. Soc. 47 (12), 1461–1467.

Laporte, G., Palekar, U., 2002. Some applications of the clustered travelling salesman problem. J. Oper. Res. Soc. 53 (9), 972–976.

Mestria, M., Ochi, L.S., de Lima Martins, S., 2013. GRASP with path relinking for the symmetric euclidean clustered traveling salesman problem. Comput. Oper. Res. 40 (12), 3218–3229.

Miller, C.E., Tucker, A.W., Zemlin, R.A., 1960. Integer programming formulation of traveling salesman problems. J. ACM 7 (4), 326–329.

Morton, G., Land, A., 1955. A contribution to the 'travelling-salesman' problem. J. R. Stat. Soc. Ser. B Stat. Methodol. 17 (2), 185–194.

Nagy, G., Salhi, S., 2007. Location-routing: Issues, models and methods. European J. Oper. Res. 177 (2), 649–672.

Noon, C.E., Bean, J.C., 1993. An efficient transformation of the generalized traveling salesman problem. INFOR Inf. Syst. Oper. Res. 31 (1), 39–44.

Öncan, T., Altınel, I.K., Laporte, G., 2009. A comparative analysis of several asymmetric traveling salesman problem formulations. Comput. Oper. Res. 36 (3), 637–654.

Ouzidan, A., Sevaux, M., Olteanu, A.-L., Pardo, E.G., Duarte, A., 2022. On solving the order processing in picking workstations. Optim. Lett. 16 (1), 5–35.

Pferschy, U., Staněk, R., 2017. Generating subtour elimination constraints for the TSP from pure integer solutions. Central Eur. J. Oper. Res. 25 (1), 231–260.

Pop, P.C., 2007. New integer programming formulations of the generalized traveling salesman problem. Am. J. Appl. Sci. 4 (11), 932–937.

Pop, P.C., Cosma, O., Sabo, C., Sitar, C.P., 2024. A comprehensive survey on the generalized traveling salesman problem. European J. Oper. Res. 314 (3), 819–835.

Rego, C., Gamboa, D., Glover, F., Osterman, C., 2011. Traveling salesman problem heuristics: Leading methods, implementations and latest advances. European J. Oper. Res. 211 (3), 427–441.

Saskena, J., 1970. Mathematical model of scheduling clients through welfare agencies. J. Can. Oper. Res. Soc. 8, 185–200.

Sherali, H.D., Sarin, S.C., Tsai, P.-F., 2006. A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. Discrete Optim. 3 (1), 20–32.

Srivastava, S., Kumar, S., Garg, R., Sen, P., 1969. Generalized traveling salesman problem through n sets of nodes. J. Can. Oper. Res. Soc. 7 (2), 97–101.

Vidal, T., Laporte, G., Matl, P., 2020. A concise guide to existing and emerging vehicle routing problem variants. European J. Oper. Res. 286 (2), 401–416.

Zhang, T., Ke, L., Li, J., Li, J., Huang, J., Li, Z., 2018. Metaheuristics for the tabu clustered traveling salesman problem. Comput. Oper. Res. 89, 1–12.