Pamukkale Univ Muh Bilim Derg, 25(9), 1041-1049, 2019
(LMSCM'2018-16. Uluslararası Lojistik ve Tedarik Zinciri Kongresi Özel Sayısı)

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

A constraint programming approach for the pickup and delivery problem with time windows

Zaman pencereli toplama ve dağıtım problemi için kısıt programlama yaklaşımı

Mustafa KÜÇÜK[1], *Şeyda TOPALOĞLU YILDIZ*[2*]

[1,2]Department of Industrial Engineering, Engineering Faculty, Dokuz Eylul University, İzmir, Turkey.
mustafa.kucuk218@ogr.edu.edu.tr, seyda.topaloglu@deu.edu.tr

Received/Geliş Tarihi: 15.06.2019, Accepted/Kabul Tarihi: 28.11.2019 doi: 10.5505/pajes.2019.56804
* Corresponding author/Yazışılan Yazar Special Issue Article / Özel Sayı Makalesi

Abstract

The pickup and delivery problem with time windows (PDPTW) is studied in this paper. It is referred to as the single-commodity capacitated vehicle routing problem with pickups and deliveries, in which a fleet of vehicles meet a group of customer demands. Each customer demand includes the usage of only one vehicle for both loading a specified quantity of one type of commodity at one place and delivering them to another place. All the demands of customers must be satisfied without exceeding the vehicle capacity and the pickup or delivery places time windows specified for each place. In this study, we introduce a novel Constraint Programming (CP) model for the PDPTW. CP is an exact solution approach that is popular for its performance to state complicated relationships and to achieve high-quality solutions within acceptable computational times for combinatorial optimization problems with complicated constraints such as the PDPTW. The performance of the proposed CP model is tested with well-known benchmark instances from literature. The results of computational analysis indicate that our CP model is very effective in finding high-quality solutions for even large size problems.

Keywords: Constraint programming, Pickup and delivery problem, Time windows

Öz

Bu makale zaman pencereli toplama ve dağıtım problemini (ZPTDP) ele almaktadır. Problem, müşteri taleplerinin bir araç filosu tarafından karşılandığı, tek ürünlü, toplama ve dağıtımlı araç rotalama problemi olarak adlandırılmaktadır. Her müşteri talebi belli miktardaki tek tip ürünün bir lokasyondan yüklenmesini ve başka bir lokasyona teslim edilmesini içermektedir. Müşteri talepleri araçların kapasitesi ve her bir lokasyon için belirlenmiş toplama ve dağıtım zaman pencereleri ihlal edilmeden karşılanmalıdır. Bu çalışmada, ZPTDP için yeni bir kısıt programlama (KP) modeli sunmaktayız. KP, ZPTDP gibi zor kısıtlı kombinatorik optimizasyon problemlerinin karmaşık ilişkilerinin tanımlanmasında ve kabul edilebilir hesaplama süresi içinde yüksek kaliteli çözümler bulmada yeterliliği iyi bilinen, kesin bir çözüm yaklaşımıdır. Önerilen KP modelini literatürde sıkça kullanılan karşılaştırma örneklerine uyguladık. Aldığımız sonuçlar KP modelimizin büyük boyutlu problemlerde bile yüksek kaliteli sonuçlar verebilecek kadar etkili olduğunu göstermiştir.

Anahtar kelimeler: Kısıt programlama, Toplama ve dağıtım problemi, Zaman pencereleri

1 Introduction

Vehicle routing problem (VRP) is a very popular research area owing to its benefits for transportation and logistics applications. In recent times, numerous new constraints have been added to this problem to meet real-life demands. Some of these constraints are related with capacity, service times, time windows, loading, transshipment etc. One of the useful extensions of the VRP is the pickup and delivery problem which includes capacity and time windows constraints. The pickup and delivery problem occurs for a single vehicle or a group of vehicles to meet a group of customer demands. Each customer demand enforces the usage of a single vehicle both to load a quantity of goods at one place and to deliver them to another place. All customer demands are met within the customer time window restriction specified for each place and considering the vehicle capacity limitations. The objective of the classic pickup and delivery problems with time windows (PDPTW) is to minimize the number of vehicles employed or to minimize the sum of travel distances of vehicles.

Different variants of the PDPTW are studied in literature and classified according to their configurations, the pickup and delivery activities in the nodes, commodity types and the number of employed vehicles. For example, in one-to-one PDPTW, each vehicle has pairs of one pickup point and one delivery point. The vehicles visit any place as an origin or as a destination in the many-to-many type [1]. Solving the PDPTW contributes to various logistics applications such as forward or reverse logistics, shipping cars, home delivery, collection and distribution of empty cans and bottles, bike repositioning activities, school bus routing, airline scheduling etc. [2]

In this study, we introduce a novel constraint programming (CP) model for the PDPTW. The performance of the proposed CP model is evaluated for small size problems against the mixed integer programming (MIP) formulation over randomly generated problem instances. The CP model is also implemented to solve the well-known benchmark instances from the related literature. The results of computations show that the suggested CP model is very effective in finding good quality feasible solutions.

The remainder of this paper is arranged as following: in Section 2 literature review is given. Problem description and formulation of model are provided in Section 3. In Section 4, the developed CP model for the PDPTW is presented. In Section 5, the computational results are illustrated. Eventually, in Section 6, conclusion and future research considerations are given.

2 Literature review

The pickup and delivery problem with time windows (PDPTW) was for the first time defined by Lenstra et al. [3]. It has arisen from the vehicle routing problems with time windows (VRPTW), in which a quantity of vehicles with specified capacities located at a depot give service to dispersed customers within specified time windows. After this study, numerous studies have been issued in literature using various techniques such as exact methods, heuristics and meta-heuristics, with focus on solving benchmark instances of the PDPTW.

In most of the studies, the objective functions are either to minimize the cumulated cost of the vehicles or to minimize the cumulated distance traveled by all the vehicles. The most frequently used PDPTW constraints can be categorized as vehicle capacity constraints (identical or non-identical vehicle fleets), coupling constraints (the usage of the same vehicle for visiting the pickup and delivery places), precedence constraints (pickup place scheduled before the respective delivery place), depot constraints (start and end of travel), resource constraints (availability of drivers and vehicles), visiting constraints (exactly one visit to each pickup and delivery places), and time window constraints [4].

Solomon and Desrosiers [5] issued a significant review of the VRPTW and related problems such as traveling salesman problems (TSPTW) and PDPTW. Savelsbergh and Sol [6] discussed various characteristics of the pickup and delivery problems (PDP) and gave an overview of the solution approaches about the PDPs. They also classified the PDP into cases, with vehicle numbers and time windows. In addition, Toth and Vigo [7] introduced a very successful survey of the vehicle routing problems that included methods for solving the PDPTW.

Li and Lim [8] introduced a tabu-embedded simulated annealing based solution approach and asserted that their approach is the first efficient one to solve large size PDPTW problem instances. Bent and Van Hentenryck [9] introduced a two-stage hybrid algorithm for the PDPTW, where in the first stage a simulated annealing algorithm is used to minimize the number of vehicles, and in the second stage the travel cost is decreased by a large neighborhood search. Ropke and Pisinger [10] developed an extension to the large neighborhood search and the ruin-and-recreated heuristic and named it as the adaptive large neighborhood search heuristic (ALNS). Until 2008, for a detailed survey on the pickup and delivery problems and their variants, see Parragh et al. [11]. Nagata and Kobayashi [12] presented a guided ejection search algorithm to the PDPTW. Nalepa and Blocho [13] studied on a parallel guided ejection search algorithm to minimize the number of vehicles in the PDPTW. In their study, parallel processes cooperate sporadically to increase the quality of results and to decrease the convergence time of computations.

Furthermore, many variants of the PDPTW stimulated by real-life problems are studied in literature. Xu et al. [14] proposed a column generation-based solution approach for the PDPTW with many real-life complications, in which customer orders and vehicle types must satisfy the compatibility constraints, and an order cannot be unloaded until all the previously loaded orders into the vehicle are unloaded. Qu and Bard [15] introduced an extension to the PDP with non-identical fleet in which the capacity of each vehicle can be changed to satisfy different types of customer demands. Bettinelli et al. [16] proposed a branch and price algorithm to tackle PDPs with soft time windows. In their study, a penalty is attributed when a time window is violated. Mannel and Borthfeld [17] extended the classical PDP to an integrated routing and three-dimensional loading problem. They developed a hybrid algorithm that combined the large neighborhood and the tree search heuristic. They also tested their algorithm on PDPTW benchmark instances. Tchoupo et al. [18] evolved a Bender's decomposition algorithm for the PDPTW with non-identical fleet to decrease the cumulated routing cost. For the identical case, their introduced approach was able to solve the large size problems optimally in acceptable computational time. Györgyi and Kis [19] studied a dynamic and stochastic PDP via an effective solution method based on an attentive analysis of the transfer probability between the customers. Lu and Yang [20] proposed a hybrid approach, called iterative logistics solution planner.

In this paper, we introduce a novel solution method for the PDPTW. We develop a formulation in which all the constraints and the objective function are combined within the framework of CP, and we call this model as CP-PDPTW. We use IBM ILOG CPLEX Optimizer for implementing CP-PDPTW.

3 Problem statement and formulation

The PDPTW can be described on graph theory terms as following: Let $G(N, A)$ be a graph having the point set N and the arc set A. For i and $j \in N$, we indicate the arc from i to j as $(i, j) \in A$. We use the term "network" to mean a graph having additional data on its nodes and arcs. $N = \{0,1,2, \ldots, n+1\}$ is the set of places (nodes), in which 0 represents the initial depot and $n+1$ represents the final depot. Each customer $\{1,2, \ldots, n\}$ requires a non-negative delivery quantity d_i and a non-negative pickup quantity p_i. P and D represent the set of pickup and delivery places (nodes), respectively. Let r denote the number of customer demands (requests) to satisfy. Each customer request $r \in R$ includes a pickup node $P(r)$ and a delivery node $D(r)$. Additionally each node $i \in N$ has a load q_i and a positive service duration d_i. The loads and durations of the depots are equal to 0 ($q_0 = q_{n+1} = 0$, $d_0 = d_{n+1} = 0$). A limitless fleet of distinct or identical vehicles with capacity C_k is ready to meet the customer demands. Every arc $(i, j) \in A$ has a travel cost c_{ij} connected to travel time t_{ij}. Moreover every node $i \in P \cup D$ has also a time window$[e_i, l_i]$, where e_i and l_i indicate the possible first and last service time, respectively. The pickup or delivery service may start between these times at node i. The depot usually has time windows represented by $[e_0, l_0]$ and $[e_{n+1}, l_{n+1}]$. The objective function of this problem is to find out a set of vehicle routes that makes the total cost minimum, while satisfying all customer demands.

The presented mixed integer programming model for the PDPTW (MIP-PDPTW) is adapted from the related models in Ropke et al. [21], Parragh et al. [11] and Rais et al. [22]. The triangle inequality is valid for travel costs and times. The notation for MIP-PDPTW is given as follows:

Sets:

N set of all nodes, $\{0,1,2, \ldots, n+1\}$

V set of all vehicles, $\{1,2, \ldots, m\}$

P set of all pickup nodes

D set of all delivery nodes

R set of customer requests (demands)

Parameters:

C^k capacity of vehicle $k \in V$

c_{ij} traveling cost between nodes i and $j, i \in N, j \in N$

t_{ij} traveling time between nodes i and $j, i \in N, j \in N$

d_i service duration of customer $i \in N$

q_i load amount of customer $i \in N$

M_{ij}^k big number for beginning time of arc $i, j \in N$ and vehicle $k \in V$

W_{ij}^k big number for load of arc $i, j \in N$ and vehicle $k \in V$

Decision variables:

x_{ij}^k 1 if arc(i, j) is traveled by vehicle $k \in V$, 0 otherwise

B_i^k beginning service time of vehicle $k \in V$ at node $i \in N$

Q_i^k load of vehicle $k \in V$ when leaving node $i \in N$

According to the above notation, the MIP-PDPTW formulation can be given as follows:

$$Min \sum_{k\in V}\sum_{i\in N}\sum_{j\in N} c_{ij}\, x_{ij}^k \quad (1)$$

Subject to:

$$\sum_{k\in V}\sum_{j\in N} x_{ij}^k = 1 \qquad \forall i \in N \quad (2)$$

$$\sum_{j\in N} x_{0j}^k = 1 \qquad \forall k \in V \quad (3)$$

$$\sum_{i\in N} x_{i(n+1)}^k = 1 \qquad \forall k \in V \quad (4)$$

$$\sum_{i\in N} x_{ij}^k - \sum_{i\in N} x_{ji}^k = 0 \qquad \forall j \in N, \forall k \in V \quad (5)$$

$$\sum_{j\in N} x_{P(r)j}^k - \sum_{j\in N} x_{D(r)j}^k = 0 \quad \forall r \in R, \forall k \in V \quad (6)$$

$$B_j^k \geq B_i^k + d_i + t_{ij} - M_{ijk}(1 - x_{ijk}) \qquad \forall i, j \in N, \quad \forall k \in V \quad (7)$$

$$B_{P(r)}^k \leq B_{D(r)}^k \qquad \forall r \in R, \forall k \in V \quad (8)$$

$$e_i \leq B_i^k \leq l_i \qquad \forall i \in N, \forall k \in V \quad (9)$$

$$Q_j^k \geq Q_i^k + q_i - W_{ij}^k(1 - x_{ijk}) \qquad \forall i, j \in N, \forall k \in V \quad (10)$$

$$max(0, q_i) \leq Q_i^k \leq min(C^k, C^k + q_i) \quad \forall i \in N, \forall k \in V \quad (11)$$

$$x_{ij}^k \in \{0,1\} \quad \forall i, j \in N, \forall k \in V \quad (12)$$

The objective function (1) minimizes total traveling cost. Constraints (2) state that each node has to be served only once. Constraints (3) and (4) ensure that each vehicle starts at the initial depot, and after visiting a set of nodes in its tour, it returns to the final depot, but these constraints do not enforce that each vehicle has to be used. If a vehicle uses only arc $(0, n+1)$, it means that the vehicle is not used in the pickup and delivery activities. Constraints (5) ensure flow conservation. Constraints (6) enforce that the same vehicle must serve both origin and destination of a demand. Constraint (7) eliminates subtours by using time variables, given that $(t_{ij} + d_i) > 0$ for all $\forall i, j \in A$. According to constraints (8), delivery can only occur after pickup. Constraints (9) prohibit violation of time windows. Constraints (10) and (11) ensure that a vehicle's capacity must not be exceeded during its tour. The validity of Constraints (7) and (10) is ensured by setting $M_{ij}^k \geq max\{0, l_i + d_i + t_{ij} - e_j\}$ and $W_{ijk} \geq min\{Q_{ik}, Q_{ik} + q_i\}$ [23].

4 Proposed constraint programming model (CP-PDPTW)

Constraint programming (CP) is considered as a robust technique for solving combinatorial search problems, and it is based on various techniques employed from artificial intelligence, operations research and graph theory [24]. In this paper, we use CP as a solution method to solve the PDPTW. CP formulates this problem as a constraint satisfaction problem which is to assign a convenient value to every variable in order to satisfy all constraints. In usual way of solving a CP model, the user states the real-life problem like this represented in terms of decision variables and constraints, and a CP solver is used to solve them.

In literature, there are various studies related to CP. Some of these have emerged very recently; for instance, Gedik et al. [25] utilized CP to solve the team orienteering problem. Rahimianet et al. [26] proposed a hybrid algorithm which integrates integer programming and CP to efficiently solve the nurse scheduling problem with a large number of constraints. Hojabri et al. [27] developed a CP-based adaptive large neighborhood search (ALNS) for solving the VRP with time windows. They proved that CP was very useful for a number of vehicle routing problem variants and the presence of synchronization constraints made this problem more convenient for a CP-based approach. Laborie et al. [28] indicated that the CP optimizer is continuously improving and that they will continue to increase the performance of the search strategies.

Considering the effectiveness of CP, we developed a CP model for the PDPTW in this part of the study (CP-PDPTW). While developing this model, we used interval variables that represent an interval of time between the start and end of an activity and its position in timeline is an unknown part of the problem [29]. The interval variable is featured by a start time and an end time value, a length and a size. There are several advantages of interval variables [30]. In cases where the activities are represented by those interval variables with the optionality feature defined explicitly, no additional constraint is needed to ensure the binary correlation for the absence or presence of an activity in the schedule. Subsequently, when some optional interval variables are not observed in the last solution, it can be inferred that their related domains are

empty, and they do not exist in the final schedule. In the CP formulation, final status of interval variable can be queried by using the $presenceOf(Interval\,Variable)$ function. The domains of the interval variables in the proposed CP model are the possible time intervals that represent the start and end times of a place visit. The duration of the visit that corresponds to the size of interval variable is equal to the difference between the end and start times of the visit [25].

Interval variables' start time and end time remark service beginning and finishing time of a visit, respectively. It does not mean that start time is exactly the vehicle's arrival time to pickup or delivery point because the vehicle needs to wait until the time window for customer service initiates. A visit is performed by only one vehicle, that is why the utilization rate is equal to "one" and this visit is represented by an interval variable that also represents the service start and end time as seen in Figure 1.

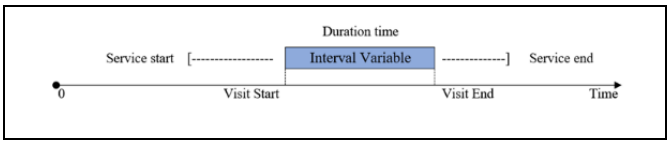

Figure 1: Representing interval variable.

The x_{ik} is an interval variable representing visit of node $i \in N$, using vehicle $k \in V$, and duration time of this visit is d_i. Because all of the customers' demands must be satisfied, interval variables for all customers must exist in the solution. The start time and the end time are specified according to time windows constraints.

$Q_k = \{x_{0k}, x_{1k}, x_{2k}, \dots, x_{ik}, \dots, x_{(n+1)k}\}$ is a collection of optional interval variables for each vehicle k that represents visits to each node $i \in N$. This collection of interval variables is called as an interval sequence variable for each vehicle k and it examines the feasibility of a visits' sequence for each vehicle [25].

In this study, we used several global constraints such as $Alternative, NoOverlap, Pulse$ etc. They are generated by IBM's CP Optimizer. The $Alternative\ \{y_i, \{x_{i1}, \dots, x_{im}\}$ global constraint chooses a single alternative between interval variables $\{x_{i1}, \dots, x_{im}\}$[31]. This constraint enforces that only one of the decision variables from the collection of $\{x_{i1}, \dots, x_{im}\}$ is obtained if the decision variable y_i is obtained. Furthermore, y_i starts and ends together with the chosen from the collection of $\{x_{i1}, \dots, x_{im}\}$. This constraint is used to assign a visit to each node by only one vehicle. $TransitionDistance(t_{ij})$ is a function that creates a two-dimensional array which stores the distances between all pickup and delivery points. This function is used with another global constraint that is called $NoOverlap\ \{x_{i1}, \dots x_{im}\}$. The $NoOverlap$ constraint featured with $TransitionDistance\ (t_{ij})$ function obtains a sequence of interval decision variables that do not overlap and has minimum time or distance between each consecutive variable in the collection of $\{x_{i1}, \dots x_{im}\}$. Thus, the $NoOverlap$ constraint featured with $TransitionDistance(\ t_{ij})$ matrix ensures that a vehicle does not visit more than one node at any certain time and also ensures that if a vehicle visits a node with fixed velocity, the time to visit the next node has to be obtained after the computed travel time between the nodes elapses. Furthermore, $Cumulative$ is another global constraint that tracks the cumulated utilization of the resources by the visits as a function of time. A visit to the pickup node naturally boosts the cumulated resource utilization by one unit, whereas a visit to a delivery node by the same vehicle decreases it by one unit.

For analyzing the impact of each interval variable during the activity time, several elementary cumulative functions can be used. For example, $Step$ is used to obtain the rate of resource level. In this study, the $Cumulative$ constraint with the $Step$ function is applied to limit the capacity of vehicles during the travel.

Finally, let c denote the total cost of each vehicle k which starts to visit nodes starting from the initial depot and ends at the final depot. This decision variable may depend on total time or distance. Considering the global constraints and interval decision variables given before, CP-PDPTW model can be introduced as follows:

$$Min\ c \tag{13}$$

Subject to:

$$Alternative\ (y_i, \{x_{i1}, x_{i2} \dots x_{im}\}) \qquad \forall i \in N, i \neq 0,\ i \neq n+1 \tag{14}$$

$$Cumulative\ (Step(x_{ik}, q_i | i \in N))\ \ \forall k \in V \tag{15}$$

$$NoOverlap\ \left(Q_k, Transitiondistance\left(t_{ij} | i, j \in N\right)\right) \qquad \forall k \in V \tag{16}$$

$$Q_k.First(y_0) \qquad \forall k \in V \tag{17}$$

$$Q_k.Last(y_{n+1}) \qquad \forall k \in V \tag{18}$$

$$StartOf\left(x_{D(r)k}\right) \geq StartOf\left(x_{P(r)k}\right) \qquad \forall k \in V \tag{19}$$

The objective function of CP-PDPTW minimizes the sum of traveling cost of each vehicle (13). Constraints (14) assure that each place, except the initial and final depots, is visited by exactly one vehicle. The $Alternative$ global constraints make it possible by ensuring that y_i must be in the solution if only one of the x_{ik} interval variables is in the solution. The $Cumulative$ global constraints (15) provide a consistent control of the utilization of each vehicle k during the visits. They guarantee that the total quantity loaded on each vehicle during the tour cannot be more than the capacity of vehicle k. The vehicle usage during the activity time is computed using the sub-function $Step(x_{ik})$. It changes the total utilization of each vehicle by loading and unloading at the pickup and delivery nodes of interval decision variable y_j, respectively.

The specified time windows for each node are given in the related interval variables as start and end times. The $NoOverlap$ constraints (16) ensure that the interval decision variables in sequence Q_k indicate the possible visits of vehicle k without overlapping each other. Besides, with the help of $TransitionDistance(t_{ij})$, the $NoOverlap$ global constraints obtain a minimum travel time (t_{ij}) between the end and start times of interval decision variables x_{ik} and x_{jk}, respectively which represent the consecutive visits to node i. Constraints (17) and (18) make the initial depot the first (y_o) and the final depot (y_{n+1}) the last node to be visited by each vehicle k. Finally, constraints (19) ensure that both origin and destination of a demand is served by the same vehicle and delivery action can only perform after pickup.

$$c = \sum_{k \in V} endOf\left(x_{(n+1)k}\right) - \sum_{i \in N} dur_i \quad (20)$$

$$c = \sum_{k \in V} \sum_{i \in N} t_{i(typeOfNext(Q_k, x_{ik}, i, i)} \quad (21)$$

$$c = \sum_{k \in V} endOf\left(x'_{(n+1)k}\right) \quad (22)$$

In the proposed CP formulation, for solving the PDPTW, we have tried three types of expressions for c; in the first type, same as in Laborie et al. [28], we employed the $endOf(\,)$ expression (20) that represents the end time of interval variable to find the total time for vehicles' usage, but if we need to minimize the total distance, this expression is ineffective because of time windows. So in the second type, we utilized the $typeOfNext(\,)$ function (21) that returns the type of the next interval in a given sequence to find out consecutive nodes in a tour. The sum of the distances between these nodes give us the total distance that is performed by all vehicles. Note that, if the presence of x_{ik} is "0", that is, vehicle k does not visit node i, then the function gives t_{ii} as equal to "0". In the last type of c expression (22), we utilized additional interval variable x'_{ik} $(i \in N, k \in V)$ which has no size (duration) and no time windows, and also we utilized additional sequence interval variable Q'_k that indicates possible permutations of x'_{ik}. Thereafter, we linked the two sequence interval variables Q_k and Q'_k with the $sameSequence(\,)$ constraint. In this way, c calculates the sum of times that elapses only between the visited nodes by vehicle k using the $endOf\left(x'_{(n+1)k}\right)$ function to derive the last node's visit time [24]. In this type c expression, we need to multiply c by vehicles' velocity in order to get the total distance traveled.

For tuning the performance, there are some useful ways to make the CP Optimizer application more efficient such as using multiple data sets, examining the tolerance of the objective function, using expertise from the problem, optimizing propagation and search and finally, considering whether removing symmetry [29].

Mostly, a direct model of the real-life problems has symmetries. For example, these symmetries can be attributed to a fleet of identical vehicles, a set of identical containers and engineers with identical skills [29]. Given a fleet of identical vehicles or containers, it is normal to describe an arbitrary sequence among them by numbering them. After that, they are not technically identical anymore and symmetries rising from their original interchangeability are thus eliminated. In our model there are also symmetric solutions and for eliminating these solutions, we have added symmetry breaking constraints (23) that ensure sorting the identical vehicles by the number of visits. Equivalent feasible solutions owing to arbitrary repeats, as seen in Figure 2, are eliminated by symmetry breaking constraints which do not have any impact on the objective function.

$$\sum_{i \in N} presenceOf(x_{ik}) \geq \sum_{i \in N} presenceOf\left(x_{i(k+1)}\right) \quad \forall k \in V \quad (23)$$

There are various search algorithms in the IBM's CP Optimizer for rapid convergence towards good solutions while solving the CP models. In some cases, it can be obtained better performance by tuning the search that includes setting parameters, selecting search types, specifying search phases and so on. A search phase defines an instantiation strategy to improve the search process [29]. Mainly there are two ways in determining search phases. First of these ways is designed upon the interval variables, which works either on a unique interval variable or on an array of interval variables. This search phase fixes the values of interval variables in the one-way style, starting to fix from the initial intervals to the last one. The search phase on sequence variables is the second way which fixes the value of the specified unique sequence variable or array of sequence variables. During this phase each sequence variable is assigned an entirely ordered sequence of executed interval variables. It must be noted that this search is useful only if interval variables have limited values especially for start and end times. Consequently, the decision interval variables and their potential values can be sorted so that the optimizer can fix the key decision interval variables in the beginning of the process.

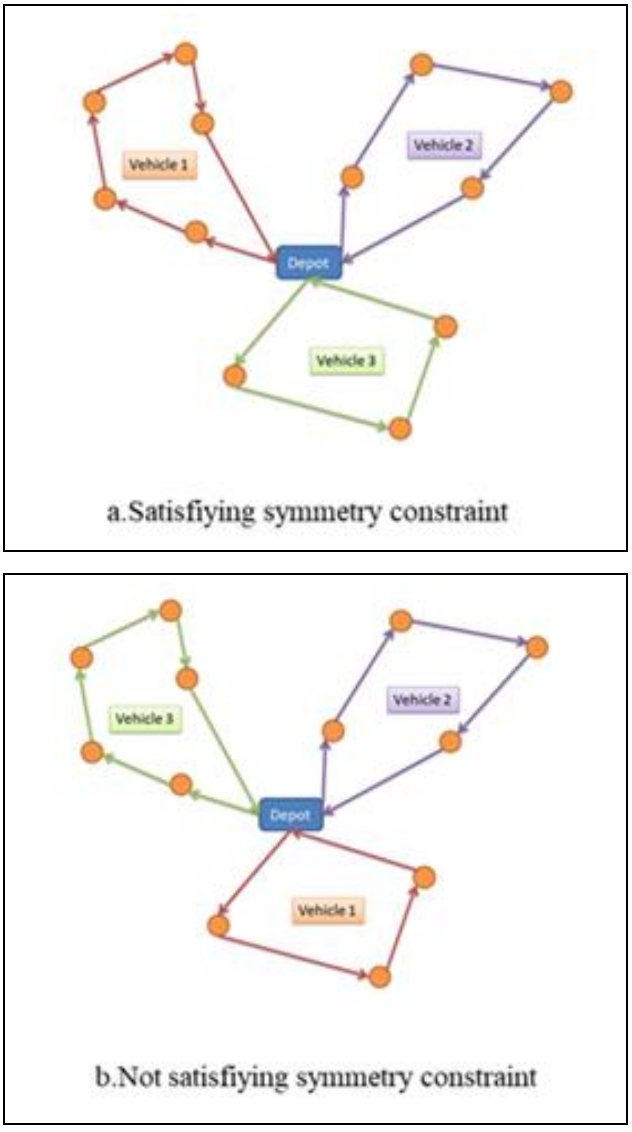

Figure 2: Illustration of symmetric solutions.

5 Computational results

Initially, we tested the performance of CP-PDPTW using small size problem instances in relevance to MIP-PDPTW. From the datasets of Li and Lim [8] including 100 nodes and 50 demands, we generated problems by randomly picking those data sets including between 16 to 30 nodes, and then pairing these nodes in one-to-one correspondence to form pickup and delivery demands of customers.

The number of nodes is assumed to be even when excluding the initial and final depots. For computational analysis, the models were implemented in C#, and executed though IBM ILOG CPLEX Optimization Studio V12.9.0, using the MIP optimizer for MIP-PDPTW and CP Optimizer for CP-PDPTW. All problem instances were solved on an Intel Core i5-7400 processor at 3.0 GHz using 12 GB of RAM. The execution of the modelsended after the optimal solution was found. All computational results reported in the comparison of CP-PDPTW and MIP-PDPTW were obtained by default settings except the settings on CP parameters which are *searchtype* and *logperiod*. Additionally, we identified search phases to specify the sequence of the search movements and the values to be tried during the computation. We obtained the best solutions from $searchtype = restart$, $logperiod = 10000$ and $searchphase(sequence\ variable)$. We show the optimal solution values and the required solution times for each problem in Table 1. The last column in the table gives the solution time gap between the models. Accordingly, CP-PDPTW requires in average 29.15% less time compared to MIP-PDPTW to find the optimum solution.

Following, the performance of CP-PDPTW has been also compared in regard to related solution methods in literature with the well-known problem instances of Li and Lim [8] which are derived from the instances of Solomon [32]. The travel time is assumed to be as same as the distance in these instances and the instances represent different classes of problems. In the first class of instances, LR, the nodes are randomly located; in the second class of instances, LC, the nodes are clustered, and finally in the third class of instances, LRC, the nodes are both randomly located and clustered. Furthermore, there are two types for these problem classes which are named as Type 1 and Type 2, respectively. The instances of Type 1 have small service time windows, whereas those of Type 2 have large service time windows. Since Type 2 problems are less constrained, and they increase the computational complexity of CP, we retained our computational analysis to Type 1 instances. The datasets and best-known solutions (BKNs) are available at http://www.sintef.no/Projectweb/TOP/PDPTW/. We compared our CP-PDPTW with the best known methods in related literature which are respectively ALNS of Ropke and Pisinger [10] and hybrid algorithm of Mannel and Borthfeld [17]. We adjusted *timelimit* parameter of CP-PDPTW for three hours. As shown in Table 2, there are 36 instances for comparison and the problem characteristics are given in the first three columns. In the fourth column, the BKNs are given, whereas the fifth, sixth, and seventh columns show the solution values found by the referenced works and CP-PDPTW, respectively. In the eighth and last columns, the computational time required by CP-PDPTW and the gap values between the BKNs and the solutions of our model are given, respectively.

To deal with infinite domain ranges, the fractional distance values between the nodes in the problem instances have been multiplied by 100. Also, this multiplier has been used for time window start and end times and service time. For adjustment, after computation the obtained solution values have been divided by 100.

Additionally, in our experiments we have observed that CP-PDPTW converges to the best solutions in relatively short computation times, as seen in Figure 3. According to the results, CP-PDPTW model has found 22 BKNs out of 36 instances, and it has improved one BKN. It is of notice that the average solution gap of CP-PDPTW is only 1.32% compared to the BKNs. On the other hand, CP-PDPTW has found better solution values for the LC instances in which the nodes are clustered, compared to the LR and LRC instances in which the nodes are randomly located, and both randomly located and clustered, respectively. The difficulty of CP in solving LR and LRC instances lies behind the fact that searching in randomly placed nodes set, compared to the clustered nodes set, is like searching in a wide area which makes constraint propagation less effective, and the cost of getting a better solution may be relatively great.

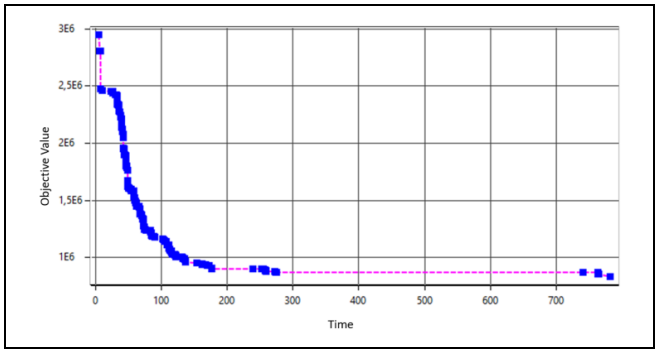

Figure 3: The convergence graph of CP for the objective value.

6 Conclusion

In this paper, we developed a new CP model for the PDPTW and compared our model with a MIP model adapted from the existing literature. The novel model CP-PDPTW is shown to be very competitive with MIP-PDPTW and can be a reference model for solving the variations of vehicle routing problems. CP-PDPTW was also compared with the proposed algorithms in related literature on the well-known existing problem instances. The results of computation show that CP-PDPTW performs very well to find out satisfying solutions. It is optimistic to say that CP as an exact algorithm competes with the metaheuristics for solution times; however, it can obtain high-quality solutions in reasonable computation times using a rather simple model formulation when compared to the complex design of metaheuristic approaches.

Future work on this topic can be related to the formulation of effective and efficient CP models for different types of the pickup and delivery problems by adding new redundant, symmetric and global constraints, and devising problem specific search strategies.

Table 1: Comparison of CP and MIP model solutions for the PDPTW.

Instances	Number Of Customers	Number of Demands	MIP-CPDTW		CP-PDPTW		Solution Time Gap (%)
			Solution Values	CPU Times (s)	Solution Values	CPU Times (s)	
LC01	16	8	197	1.36	197	0.45	-66.91
LC02	18	9	193	1.62	193	0.55	-66.05
LC03	18	9	101	0.48	101	0.29	-39.58
LC04	18	9	173	3.28	173	0.43	-86.89
LC05	18	9	226	21.11	226	0.36	-98.29
LC06	20	10	160	2.14	160	0.25	-88.32
LC07	20	10	184	24.74	184	2.31	-90.66
LC08	20	10	303	16.24	303	10.8	-33.50
LC09	20	10	232	0.65	232	0.89	36.92
LC10	20	10	194	2.14	194	0.52	-75.70
LC11	20	10	223	0.74	223	0.55	-25.68
LC12	20	10	198	5.34	198	0.37	-93.07
LC13	20	10	253	24.09	253	1.9	-92.11
LC14	20	10	195	25.09	195	0.83	-96.69
LC15	20	10	217	23.26	217	2.08	-91.06
LC16	20	10	204	6.53	204	0.51	-92.19
LC17	22	11	279	60.22	279	2.08	-96.55
LC18	22	11	158	5.96	158	2.08	-65.10
LC19	22	11	264	71.21	264	276	287.59
LC20	22	11	105	0.78	105	0.41	-47.44
LC21	22	11	256	0.73	256	2.66	264.38
LC22	22	11	195	6.64	195	0.39	-94.13
LC23	22	11	140	6.03	140	1.75	-70.98
LC24	22	11	108	6.14	108	2.26	-63.19
LC25	22	11	184	1636.43	184	10.71	-99.35
LC26	24	12	317	1.06	317	1.12	5.66
LC27	24	12	284	5.17	284	1.75	-66.15
LC28	24	12	182	9.25	182	11.32	22.38
LC29	24	12	111	0.79	111	0.49	-37.97
LC30	24	12	249	0.98	249	2.95	201.02
LC31	26	13	220	1.25	220	1.87	49.60
LC32	26	13	324	24.09	324	0.41	-98.30
LC33	26	13	217	1.28	217	3.95	208.59
LC34	28	14	165	29.95	165	2.01	-93.29
LC35	28	14	193	34.9	193	9.14	-73.81
LC36	30	15	160	24.97	160	4.31	-82.74
AVG	-	-	-	57.96	-	10.02	-29.15

Table 2: Comparison of CP-PDPTW model and related algorithms.

Instances	Number of Customers	Number of Demands	#BKNs	Männel and Bortfeldt [17]	Ropke and Pisinger [10]	CP-PDPTW	Time (s)	Solution Values Gap (%)
LC101	106	53	828.94	828.94	828.94	828,94	96	0.00
LC102	106	53	828.94	828.94	828.94	828.94	2080	0.00
LC103	104	52	1035.35	1035.35	1035.35	827.87	2337	-20.04[a]
LC104	106	53	860.01	860.01	860.01	818.60	7450	-4.82[a]
LC105	106	53	828.94	828.94	828.94	828.94	152	0.00
LC106	106	53	828.94	828.94	828.94	828.94	737	0.00
LC107	106	53	828.94	828.94	828.94	828.94	487	0.00
LC108	106	53	826.44	826.44	826.44	826.44	75	0.00
LC109	106	53	1000.60	1000.60	1000.60	827.82	1749	-17.27[a]
LC201	102	51	591.56	591.56	591.56	591.56	20	0.00
LC202	102	51	591.56	591.56	591.56	591.56	296	0.00
LC203	102	51	591.17	585.56	591.17	591.17	1512	0.00
LC204	102	51	590.60	590.60	590.60	590.60	551	0.00
LC205	102	51	588.88	588.88	588.88	588.88	215	0.00
LC206	102	51	588.49	588.49	588.49	588.49	67	0.00
LC207	102	51	588.29	588.29	588.29	588.29	141	0.00
LC208	102	51	588.32	588.32	588.32	588.32	54	0.00
LR101	106	53	1650.80	1650.80	1650.80	1650.80	206	0.00
LR102	110	55	1487.57	1487.57	1487.57	1487.57	9580	0.00
LR103	104	52	1292.68	1292.68	1292.68	1343.89	108000	3.96
LR104	104	52	1013.39	1013.39	1013.39	1013.39	439	0.00
LR105	106	53	1377.11	1377.11	1377.11	1384.60	10800	0.54
LR106	104	52	1252.62	1252.62	1252.62	1252.61	651	0.00
LR107	104	52	1111.31	1111.31	1111.31	1159.62	10800	4.35
LR108	100	50	968.97	968.97	968.97	968.96	701	0.00
LR109	106	53	1208.96	1208.97	1208.97	1237.71	10800	2.38
LR110	104	52	1159.35	1159.35	1159.35	1213.66	10800	4.68
LR111	108	54	1108.90	1108.90	1108.90	1161.53	10800	4.75
LR112	106	53	1003.77	1003.77	1003.77	1106.39	10800	10.22
LRC101	106	53	1708.80	1708.80	1708.80	1703.21	282	-0.33
LRC102	106	53	1558.07	1558.07	1558.07	1558.07	367	0.00
LRC103	106	53	1258.74	1258.74	1258.74	1272.97	10800	1.13
LRC104	108	54	1128.40	1128.40	1128.40	1128.40	406	0.00
LRC105	108	54	1637.62	1637.62	1637.62	1698.44	10800	3.71
LRC106	106	53	1424.73	1424.73	1424.73	1536.63	10800	7.85
LRC107	106	53	1230.14	1230.15	1230.15	1230.15	1334	0.00
AVG	-	-	1032.44	1032.29	1032.44	1035.36	1639.80	1.32[b]

[a] Solution values are given when the number of cars used is 10, instead of 9; [b]The average gap for the solution values is calculated excluding instances LC103, LC104 and LC109.

7 References

[1] Hernández-Pérez H, Salazar-González JJ. "The multi-commodity pickup-and-delivery traveling salesman problem". *Networks*, 63(1), 46-59, 2014.

[2] Ho SC, Szeto WY. "GRASP with path relinking for the selective pickup and delivery problem". *Expert Systems with Applications*, 51, 14-25, 2016.

[3] Lenstra JK, Desroches M, Savelbergh MWP, Soumis F. "Vehicle routing with time windows: optimization and approximation". Vehicle routing: Methods and studies, CWI Report, 65-84, 1988.

[4] Desrosiers J, Dumas Y, Solomon MM, Soumis F. "Time constrained routing and scheduling". *Handbooks in operations research and management science*, 8, 35-139, 1995.

[5] Solomon MM, Desrosiers J. "Survey paper: time window constrained routing and scheduling problems". *Transportation Science*, 22(1), 1-13, 1988.
[6] Savelsbergh MW, Sol M. "The general pickup and delivery problem". *Transportation Science*, 29(1), 17-29, 1995.
[7] Toth P, Vigo D. *The vehicle routing problem*. Philadelphia, USA, SIAM, 2002.
[8] Li H, Lim A. "A metaheuristic for the pickup and delivery problem with time windows". *International Journal on Artificial Intelligence Tools*, 12(2), 173-186, 2003.
[9] Bent R, Van Hentenryck P. "A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows". *Computers and Operations Research,* 33(4), 875-893, 2006.
[10] Ropke S, Pisinger D. "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows". *Transportation Science*, 40(4), 455-472, (2006)..
[11] Parragh SN, Doerner KF, Hartl RF. "A survey on pickup and delivery models part ii: Transportation between pickup and delivery places". *Journal für Betriebswirtschaft*, 58(2), 81-117, 2008.
[12] Nagata Y, Kobayashi S. "Guided ejection search for the pickup and delivery problem with time windows". *Evolutionary Computation in Combinatorial Optimization,* Istanbul, Turkey, 7-9 April 2010.
[13] Nalepa J, Blocho M. "Enhanced guided ejection search for the pickup and delivery problem with time windows". *Intelligent Information and Database Systems,* Da Nang, Vietnam, 14–16 March 2016.
[14] Xu H, Chen ZL, Rajagopal S, Arunapuram S. "Solving a practical pickup and delivery problem". *Transportation Science,* 37(3), 347-364, 2003.
[15] Qu Y, Bard JF. "The heterogeneous pickup and delivery problem with configurable vehicle capacity". *Transportation Research Part C: Emerging Technologies*, 32, 1-20, 2013.
[16] Bettinelli A, Ceselli A, Righini G. "A branch-and-price algorithm for the multi-depot heterogeneous-fleet pickup and delivery problem with soft time windows". *Mathematical Programming Computation,* 6(2), 171-197, 2014.
[17] Männel D, Bortfeldt A. "A hybrid algorithm for the vehicle routing problem with pickup and delivery and three-dimensional loading constraints". *European Journal of Operational Research,* 254(3), 840-858, 2016.
[18] Tchoupo MN, Yalaoui A, Amodeo L, Yalaoui F, Flori P, Lutz F. "An efficient column-generation algorithm for a new fleet size and mix pickup and delivery problem with Time windows". *IFAC-PapersOnLine,* 51(9), 440-445, 2018.
[19] Györgyi P, Kis T. "A probabilistic approach to pickup and delivery problems with time window uncertainty". *European Journal of Operational Research*, 274(3), 909-923, 2019.
[20] Lu EHC, Yang YW. "A hybrid route planning approach for logistics with pickup and delivery". *Expert Systems with Applications*, 118, 482-492, 2019.
[21] Ropke S, Cordeau JF, Laporte G. "Models and branch-and-cut algorithms for pickup and delivery problems with time windows". *Networks: An International Journal,* 49(4), 258-272, 2007.
[22] Rais A, Alvelos F, Carvalho MS. "New mixed integer-programming model for the pickup-and-delivery problem with transshipment". *European Journal of Operational Research,* 235(3), 530-539, 2014.
[23] Cordeau JF. "A branch-and-cut algorithm for the dial-a-ride problem". *Operations Research*, 54(3), 573-586, 2006.
[24] Rossi F, Van Beek P, Walsh T. "Constraint programming". *Foundations of Artificial Intelligence*, 3, 181-211, 2008.
[25] Gedik R, Kirac E, Milburn AB, Rainwater C. "A constraint programming approach for the team orienteering problem with time windows". *Computers and Industrial Engineering,* 107, 178-195, 2017.
[26] Rahimian E, Akartunalı K, Levine J. "A hybrid integer and constraint programming approach to solve nurse rostering problems". *Computers and Operations Research*, 82, 83-94, 2017.
[27] Hojabri H, Gendreau M, Potvin JY, Rousseau LM. "Large neighborhood search with constraint programming for a vehicle routing problem with synchronization constraints". *Computers and Operations Research*, 92, 87-97, 2018.
[28] Laborie P, Rogerie J, Shaw P, Vilím P. "IBM ILOG CP optimizer for scheduling". *Constraints*, 23(2), 210-250, 2018.
[29] IBM Knowledge Center. "IBM ILOG CPLEX Optimization Studio V12.8.0 documentation". https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.8.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html (09.06.2019).
[30] Laborie P, Rogerie J. "Reasoning with conditional time-intervals". *Twenty-First International FLAIRS Conference,* Florida, USA, 15–17 May 2008.
[31] Laborie P, Rogerie J, Shaw P, Vilím P. "Reasoning with conditional time intervals. Part II: An algebraical model for resources". *Twenty-Second International FLAIRS Conference,* Florida, USA, 15–17 May 2009.
[32] Solomon MM. "Algorithms for the vehicle routing and scheduling problems with time window constraints". *Operations Research,* 35(2), 254–265, 1987.