The 35th Canadian Conference on Artificial Intelligence
DOI: 0
Canadian AI 2022
30 May–3 June 2022, Toronto

Complexity Analysis of Green Pickup-and-Delivery Problems on Ring Structures

Xing Tan[†,*], Jimmy Xiangji Huang[†], Kai Huang[◇]
[†] Department of Computer Science, Lakehead University, Canada
[‡] Information Retrieval and Knowledge Management Research Lab, York University, Canada
[◇] DeGroote School of Business, McMaster University, Canada

Abstract

In a Green Pickup-and Delivery problem (GPD), *green vehicles* (e.g., electric or hybrid cars) are used for routing in the problem domain, and these vehicles are particularly constrained with limited fuel capacity, thus shorter traveling range. A refueling infrastructure providing wide-area coverage, meanwhile, has not been fully developed to this end. A recent study reveals that vehicle routing in essence with green constraints only, is at least weakly NP-hard, motivating us to pursue a deeper understanding of the underlying computational challenge associated with this "green" intractability. In this paper we perform complexity analysis on several GPD subproblems (namely, RINGs), that is, the problems whose task-graphs are under a ring structure. Using measures of width/length of rings, we delineate clearly two complexity boundaries: one between weakly and strongly NP-complete, and the other between tractable and intractable in general. Our results bring new insights for the research and development (current and active, but with limited success only to this point) of heuristics or algorithms for solving GPDs.

Keywords: Green Vehicle, Pickup and Delivery Problems, NP-completeness, Intractable Problems, Greedy Algorithm

1. Introduction

Vehicle routing for picking up entities (such as goods, packages or passengers) from origin locations and delivering them to their destination locations (PD), is a classic problem well-studied in Operations Research [1–3], and in several areas of Artificial Intelligence, including planning and scheduling [4], robotics [5], multi-agent systems and path finding [6–8], and intelligent transportation [9], for example. Vehicle routing in Green Pickup-and Delivery problems (GPD), however, are particularly constrained with limited fuel (battery) capacity, thus shorter traveling range, and with limited availability of refueling (battery-charging) infrastructure providing wide-area coverage for these vehicles [10–12].

Various algorithmic techniques for solving GPDs have been developed in the past decade. With an assumption that GPD is computationally intractable, either these algorithms are incomplete ones based on approximation or heuristics (for example, see the evolutionary algorithm proposed in [13], the genetic algorithm proposed in [14], and simulated annealing in [15]), or they can only be applied to some special cases (e.g., [16–18]).

An inherent intractability of these green components in vehicle routing is recently revealed in [19], where several highly-restricted, but non-trivial, subproblems of GPD were analyzed, with a conclusion that all these problems are NP-hard. Nevertheless, when the task graph of a GPD is with the so-called ring structure, GPD is actually only weakly NP-hard, (i.e., it is NP-hard, but a pseudo-polynomial time complete algorithm has also been identified for the problem).

GPD on ring structures (called RING problems in the paper) serves as an important special case of vehicle routing for achieving tasks of entity picking-up and delivery. More specifically, a ring structure exists as a core component in many real-world settings where

- distribution facilities or units are restricted to a few suppliers only (from which to pickup entities), and a few customers only (to which entities are delivered), and

[*] xing.tan@lakeheadu.ca

- a plurality of these facilities are deployed in approximately a ring configuration (or a supply chain, in the context of business operation management)

To better understand the underlying computational challenge associated with GPDs and RINGs, we perform in this paper a comprehensive complexity analysis on several RING variants. We work on two natural directions towards obtaining RING tractability: Control the maximal volume of task flows allowed in the ring structure, i.e., applying restrictions on δ the ring-width; And limit the total number of participating facilities deployed in the ring configuration, i.e., using a bounded λ on the ring-length. From both the δ-direction and the λ-direction, we show how RING can cross over the two complexity boundaries: one between weakly and strongly NP-hard, and the other between tractable and intractable in general (illustrated in Figure 5). A greedy algorithm is proposed to show the polynomial-time solvability of RING problems restricted to two participating facilities only (that is, length λ equals 2, but the width δ is not bounded). Correctness of the greedy algorithm is proved.

The remainder of this paper is organized as follows. Section 2 provides the preliminaries. Complexity and algorithmic analysis for the RING problems are presented next in Section 3. Section 4 summarizes the paper.

2. Preliminaries

In this section, GPD and its RING variants are defined first. Several NP-complete problems (PARTITION(2), PARTITION(3), 3PARTITION, N3DM), which would be used in the proofs of this paper, are then briefly reviewed.

2.1. GPD and Its RING Variants

We define GPD and several variants constrained on ring structures: GPD$^{\kappa}$, and RING(δ, λ), where κ, δ and λ are variables. In a general Pickup-and-Delivery problem (PD), a set of routes are constructed for vehicles to transport goods or passengers from their origin cities to destination cities [1]. In a GPD, PD further is subject to refueling constraints on its vehicles. That is, how far a vehicle in a GPD can travel on road is proportional to its fuel-tank capacity. Hence, visiting fuel stations are not tasks required to be completed, but actions necessarily to be taken before fuel tanks of these vehicles are empty.

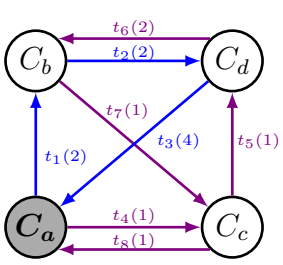
Definition 1. *In a **GPD**[1], the followings are given.*

- *A set $\mathcal{C}$ of cities;*
- *A single vehicle vhl, with a fuel-tank capacity $f^c \in \mathbb{N}$;*
- *A single depot $\in \mathcal{C}$, where vhl starts/finishes (i.e., origin city and destination city are the same), and the city also serves as the only fuel station for vhl;*
- *A set $\mathcal{T}$ of tasks. Given task $T = \langle C_p, C_d, f_{p,d} \rangle \in \mathcal{T}$, specifically a one-unit entity (goods or passenger) needs to be picked up from C_p and delivered to C_d, and fuel consumption of vhl travelling from C_p to C_d is $f_{p,d}$;*

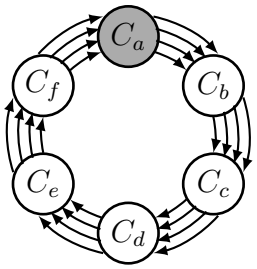
Tasks can be straightforwardly defined on a weighted, directed graph $\mathcal{G} = \langle \mathcal{C}, \mathcal{T} \rangle$, where vertices, edges, and weights in $\mathcal{G}$ correspond to cities, tasks, and actual fuel consumptions to accomplish these tasks, respectively. A walk in the graph leaving and finishing at the depot such that all tasks in $\mathcal{T}$ are traversed corresponds to an Euler Tour[2] in the graph $\mathcal{G}$. An Euler Tour is fuel-feasible if the tour also satisfies the fuel-capacity constraint applied on vhl. Since an Euler Tour might consist of several closed trails explored in a sequence, this fuel-feasibility equals the requirement of

[1] Since this paper elaborates on the line dividing tractable and intractable of GPD, the definition presented here is already a highly restricted variant of GPD (but NP-hard).

[2] We use standard definitions in graph theory: A trail is a walk in a graph with no repeated edge; A trail is closed if its two end-nodes are the same; An Euler Tour of a graph is a closed trail (or a sequence of closed trails) containing all edges in the graph.



(a) A General Task Graph



(b) Ring

Figure 1. Two example GPD task graphs. (a) A general example with a fuel-feasible Euler Tour highlighted. City C_a is the only depot, and the only fuel station, for the only vehicle. The tour consists of a first trail (in blue color, with fuel consumption 8) and a second trail (in violet, fuel consumption 6). (b) An example ring graph, whose width equals 4, and length equals 6. C_a is the only depot and the only fuel-station.

$$f_{TR} \leq f^c, \text{ for each trail } TR \text{ in } ET.$$

INSTANCE: Let $\Theta = \langle \mathcal{G}, depot, f^c \rangle$, *where* $\mathcal{G} = \langle \mathcal{C}, \mathcal{T} \rangle$.
QUESTION: Does there exist a fuel-feasible Euler Tour of $\mathcal{G}$ *for the vehicle* vhl*?*

The GPD example in Figure 1-(a) has four cities, from C_a up to C_d. The grey-color city C_a is the only depot, and the only fuel station. There are eight tasks in T, from t_1 up to t_8. Fuel consumptions traveling between cities to achieve these tasks are indicated in parentheses in the Figure. Fuel capacity of the vehicle is 8. It is straightforward to determine whether a graph has an Euler Tour: just check whether the in-degree and the out-degree for any city in the graph are equal. For a GPD to have a solution, however, it has to have a fuel-feasible Euler Tour. Euler Tour ET_0 :$[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8]$ is a solution (illustrated in Figure (1-a)). It takes 8 fuel units to walk the first trail $[t_1, t_2, t_3]$ (in blue color). After the first trail, the vehicle returns to C_a and gets refueled, ready for its second trail $[t_4, t_5, t_6, t_7, t_8]$ (in violet color), which takes 6. Another Euler Tour ET_1 :$[t_4, t_8, t_1, t_7, t_5, t_6, t_2, t_3]$, however, is not a solution. After the first trail $[t_4, t_8]$, the vehicle returns to C_a, even if it is refueled, it can not complete the second trail $[t_1, t_7, t_5, t_6, t_2, t_3]$, as it takes 12 fuel units, which is greater than the vehicle fuel capacity.

General GPD is intractable [19]. It is natural to consider cases, which are restricted on the topology of the task graphs, leading particularly to the definition of the ring graphs. However, even for these restricted GPD cases, it is known that the problem is at least weakly NP-hard [19].

Definition 2. *GPD*$^\kappa$ *is a restricted GPD, where each city in the task graph of the problem has in-degree=out-degree* $\leq \kappa$.

Definition 3. *A task graph is a* ***ring*** *if all cities/nodes in the graph are singly-connected one after another around a circle; Tasks are in one direction ("counter-clockwise" or "clockwise"). Maximal total number of tasks between two cities of the graph defines ring width* δ. *Total number of cities in the graph defines ring length* λ.

Definition 4. *RING*(δ, λ) *is a GPD whose task graph is a ring with width* δ, *and length* λ.

Theorem 1. *GPD*$^\kappa$ *is strongly NP-complete, for a general* κ *value, and for* $\kappa = 2$. *RING*$(2, \lambda)$ *is weakly NP-complete. (Theorems 2, 3, and 4 in [19]).*

Accordingly, GDP for a general κ value, GDP for $\kappa = 2$, and the problem of RING$(2, \lambda)$, these three problems are represented as white rectangles in Figure 5. It is indeed implied that GPD$^\kappa$ should also be NP-hard, if we already know that GPD^2 is NP-hard. In [Tan and Huang, 2021]), however, GPD$^\kappa$ is alternatively restricted into RING$(2, \lambda)$, which is only weakly NP-complete. In this paper, we investigate more closely regarding this complexity difference.

In the example of Figure 1-(a), κ actually equals 2. Figure 1-(b) is the task graph for RING$(4, 6)$. The width of the ring is 4, and the length is 6. City C_a is again the only depot/station in the example of Figure 1-(b).

2.2. **Relevant NP-complete Partition and Matching Problems**

Several of the well-known NP-complete problems, in number partitioning and matching, are reviewed in this section. Their relationships to each other are also briefly explained. These problems are used in the current paper to perform reductions for proving NP-hardness results.

Definition 5. ***PARTITION(2)*** *Given a multi-set $\mathcal{A}$ of positive integers[3], is there a $\mathcal{A}' \subseteq \mathcal{A}$ such that*

- $\sum_{a\in\mathcal{A}'} a = \sum_{a\in(\mathcal{A}-\mathcal{A}')} a$*?*

Example 1. *Given $\mathcal{A} = \{1, 1, 3, 4, 5\}$, the set can actually be partitioned into $\mathcal{A}_1 = \{1, 1, 5\}$ and $\mathcal{A}_2 = \mathcal{A} - \mathcal{A}_1 = \{3, 4\}$, while $1 + 1 + 5 = 3 + 4 = 7$.*

Definition 6. ***PARTITION(3)*** *Given a multi-set $\mathcal{A}$ of positive integers, are there three disjoint sets $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ such that*

$$(\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3) \equiv \mathcal{A}, \text{ and } \sum_{a\in\mathcal{A}_1} a = \sum_{a\in\mathcal{A}_2} a = \sum_{a\in\mathcal{A}_3} a?$$

Example 2. *Given $\mathcal{A} = \{1, 2, 3, 4, 5\}$, the set can actually be partitioned into three disjoint sets $\mathcal{A}_1 = \{1, 4\}$, $\mathcal{A}_2 = \{2, 3\}$, and $\mathcal{A}_3 = \{5\}$, while $1 + 4 = 2 + 3 = 5$.*

Definition 7. ***3PARTITION*** *Given a multi-set $\mathcal{A}$ of $3m$ positive integers, a bound B such that*

$$B/4 < a < B/2, \text{ for all } a \in \mathcal{A}; \text{ and } \sum_{a\in\mathcal{A}} a = mB.$$

Can the set $\mathcal{A}$ be 3-partitioned into m disjoint sets $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m$ such that, for $1 \le i \le m$, $\sum_{a\in\mathcal{A}_i} a = B$?

Example 3. *Given a set $\mathcal{A} = \{9, 9, 9, 9, 10, 10, 11, 11, 11, 12, 13, 14\}$, an integer $m = 4$, and a bound $B = 32$. It is the case $\sum_{a\in\mathcal{A}} a = 4 \times 32 = 128$, and for any integer $a \in \mathcal{A}$, $8 < a < 16$. The set $\mathcal{A}$ can be 3-partitioned into 4 disjoint sets $\mathcal{A}_1 = \{9, 9, 14\}$, $\mathcal{A}_2 = \{9, 10, 13\}$, $\mathcal{A}_3 = \{9, 11, 12\}$, $\mathcal{A}_4 = \{10, 11, 11\}$, and for $1 \le i \le 4$, $\sum_{a\in\mathcal{A}_i} a = 32 = B$.*

Definition 8. ***Numerical 3Dimensional Matching (N3DM)*** *Three multi-sets $\mathcal{W}$, $\mathcal{X}$, and $\mathcal{Y}$ each containing m non-negative integers, and a positive-integer bound B, are given. Can $\mathcal{W} \cup \mathcal{X} \cup \mathcal{Y}$ be partitioned into m disjoint sets $\mathcal{A}_1$, $\mathcal{A}_2$ $\ldots$ $\mathcal{A}_m$ such that*

- *each $\mathcal{A}_i$ contains exactly one number from each one of these three sets and*
- *for $1 \le i \le m$, the sum of all the elements in $\mathcal{A}_i$ equals B?*

Example 4. *Let $m = 4$, $B = 32$, and we are given three sets: $\mathcal{W} = \{9, 9, 9, 10\}$, $\mathcal{X} = \{9, 10, 11, 11\}$, $\mathcal{Y} = \{11, 12, 13, 14\}$. These three sets can be partitioned into $m = 4$ disjoint sets: $\mathcal{A}_1 = \{9, 9, 14\}$, $\mathcal{A}_2 = \{9, 10, 13\}$, $\mathcal{A}_3 = \{9, 11, 12\}$, $\mathcal{A}_4 = \{10, 11, 11\}$; It is the case for each $1 \le i \le 4$, we have $\sum_{a\in\mathcal{A}_i} a = 32 = B$.*

3PARTITION and N3DM are strongly NP-complete and PARTITION(2) is weakly NP-complete ([SP15], [SP16] and [SP12] in [20]). We also have the following results.

Theorem 2. *The problem PARTITION(3) is at least weakly NP-complete[4].*

Proof. In short, a set $\mathcal{A}$ in a PARTITION(2) problem instance can be transformed into a set $\mathcal{A}'$, which is a PARTITION(3) problem instance, by adding into $\mathcal{A}$ a number whose value equals $(1/2) \times \left(\sum_{a\in\mathcal{A}} a\right)$, assuming w.l.o.g. that $\left(\sum_{a\in\mathcal{A}} a\right)$ is an even number. □

[3] In this paper, integers in sets are integer-objects, thus one same integer value can have multiple distinct objects.

[4] We emphasize here the difference between PARTITION(3), which is weakly NP-complete, and 3PARTITION, which is strongly NP-complete.

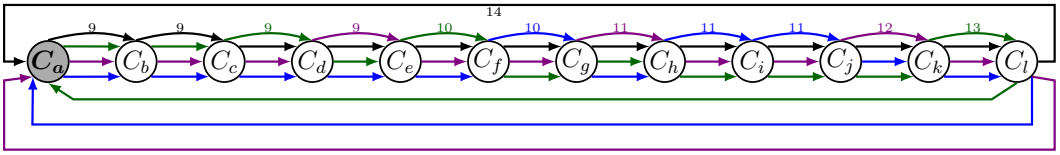


Figure 2. The RING(δ, λ) instance created from transforming a 3PARTITION problem instance (which is strongly NP-hard): $\mathcal{A} = \{9, 9, 9, 9, 10, 10, 11, 11, 11, 12, 13, 14\}$ (Example 3). The city C_a is the only depot city and the only fuel station. The only vehicle in the example has a fuel capacity 32. Fuel consumptions to achieve these tasks are labeled accordingly on top of these tasks Tasks not associated with numbers have zero fuel-consumption. A solution to RING(δ, λ) is highlighted using different colors. The solution Euler Tour contains four trails: The black trail taking 12 tasks $\{9, 9, \ldots, 14\}$ (with nine zero-tasks in between), the green trail $\{0, 0, 9, 0, 10, \ldots, 13, 0\}$, the violet trail consisting 12 tasks in the sequence $\{0, 0, 0, 9, 0, 0, 11, 0, 0, 12, 0, 0\}$, and the blue trail $\{\ldots, 10, 0, 11, 11, \ldots\}$ (skipped in between are zero tasks).

The result can be generalized into k, where k is any positive integer.

Theorem 3. *The PARTITION(k) problem, where k is an integer, is at least weakly NP-complete.*

Proof. Again we use PARTITION(2) to perform the transformation. Given the set $\mathcal{A}$, we now add $(k-2)$ values all equaling $(1/2) \times \left(\sum_{a \in \mathcal{A}} a\right)$ into $\mathcal{A}$ to build up $\mathcal{A}'$. It is the case $\mathcal{A}$ can be partitioned evenly into two subsets iff $\mathcal{A}'$ can be partitioned evenly into k subsets. □

3. Complexity Analysis

In this section, we show RING(δ, λ) and RING$(\delta, 3)$ are strong NP-complete (Theorem 4 and Theorem 5, respectively). However RING$(3, \lambda)$ is only weakly NP-complete (Theorem 6). Meanwhile, a polynomial time greedy algorithm, for RING$(\delta, 2)$, is presented (Theorem 8).

3.1. Intractability of RINGs

Theorem 4. *RING(δ, λ) is strongly NP-complete.*

Proof. [5] RING(δ, λ) is in **NP**. It runs in time polynomial to the problem size, to verify whether a sequence of pickup-and-delivery tasks in a RING(δ, λ) is a solution to a given problem instance.
NP-hardness. We perform a poly-time transformation from 3PARTITION. Given a 3PARTITION instance $\langle m, \mathcal{A}, B \rangle$, we need to create an instance $\Theta = \langle \mathcal{G}, depot, f^c \rangle$ in RING(δ, λ). We use Example 3 to illustrate the transformation. In Θ, the fuel-tank capacity of the only vehicle $v \in \mathcal{V}$ equals B (i.e., the sum of all integers in $\mathcal{A}$, divided by m). Hence, in the resulting Θ, the capacity for the only vehicle is 32. In total $|\mathcal{A}| = 12$ cities are introduced in the set $\mathcal{C}$ of $\mathcal{G}$. Pictorially (shown in Figure 2) assume all these cities are aligned in a row from left to right, there exist exactly $m = 4$ pickup-and-delivery tasks from a given city to its right neighbor. To distinguish these four tasks we call them a "top" task, a "higher" task, a "lower" task, and a "bottom" task. In addition, there are four pickup-and-delivery tasks from the rightmost city (C_l in Figure 2) to the leftmost city (C_a in Figure 2). No other tasks are introduced in Θ. All tasks other than the "top" ones need zero fuel consumption[6]. All the numbers in the set $\mathcal{A}$ are one-on-one and onto mapped to the fuel consumption of "top" tasks. For the sake of convenience, these numbers are arranged into an ascending order in Figure 2. The leftmost city is the only depot city where the vehicle starts/finishes, and the city is the only fuel station in Θ. Given $\mathcal{A}$, the resulting RING(δ, λ) instance is shown in Figure 2. Transformation is complete.

[5] Transformation for proving the NP-hardness is explained by detailing an intuitive example, which saves us from presenting actually a formal construction.

[6] A task with zero fuel consumption might not be a reasonable reflection of real world. However, we can always add a fixed value to all the tasks in the transformed task graph, with certainly the validity of the transformation preserved.

Note that, ignoring the fuel consumption constraint, vehicle vhl can finish an Euler Tour leaving and returning to C_a for four times in Figure 2. The tour consists of four trails (name them respectively, the first, second, third, and the fourth trail). Each trail will achieve exactly one of the four (top, higher, lower or bottom) tasks between two cities. The remaining tasks construct the second, third, and the fourth trails, and etc. As shown in Figure 2, an Euler Tour (with depot/fuel_station C_a) consists of four trails (highlighted in black, green, violet, and blue, respectively).

For the NP-hardness part we prove that there exists a 3PARTITION to the multi-set $\mathcal{A}$ iff a vehicle starting from C_a can accomplish all tasks, returning eventually to C_a with the fuel capacity constraint applied on C_a satisfied.

($\Rightarrow$): If there exists a 3PARTITION to $\mathcal{A}$, each $\mathcal{A}_i$ corresponds to three "top" tasks, and we assign all these three tasks to one single trail. All the other $(m-3)$ tasks in the trail would be the zero fuel-consumption ones. If between two cities, a task to be taken by the task is not a "top" one, any of the (higher, lower, bottom) tasks would work. In doing so, we can construct exactly m trails. Each trail is fuel-feasible, since the capacity of the vehicle is B, which actually equals to the fuel consumption for all the tasks in the trail (including the three "top" ones) added up. Connecting all these trails together, we construct an Euler Tour with the fuel-capacity constraint B applied to all the m trails satisfied. That is, there exists a solution in the transformed RING(δ, λ) instance Θ.

($\Leftarrow$): If there exists a solution for the transformed Θ instance (i.e., an Euler Tour with the fuel consumption constraint satisfied), the Euler Tour solution must contain m trails. Particularly we can prove that

1) Each trail in the Euler Tour contains exactly three "top" tasks. If all "top" tasks are evenly distributed among trails, each trail will take exactly 3 "top" tasks (as we have $3m$ tasks and m trails). If any trail takes less than 3 "top" tasks, it necessarily means at least one of the other $(m-1)$ trail (say TR$_k$) will contain at least four "top" tasks. Back in the 3PARTITION instance, we have that for all $a \in \mathcal{A}$, $B/4 < a < B/2$, hence fuel-consumption for TR$_k$ must be greater than $4 \times (B/4) = B$. TR$_k$ is not fuel-feasible and the corresponding Euler Tour is not a solution, a contradiction.

2) Each trail consumes exactly B units of fuel. Fuel consumption for all trails is added up to $(m \times B)$. If each trail consumes exactly B, then the numbers would add up to mB. If any trail consumes a number that is less than B, it necessarily means that some another trail (say TR$_k$) consumes more than B. But this means that TR$_k$ is infeasible and the Euler Tour is not a solution, a contradiction.

From 1) and 2), we can conclude that a solution to Θ contains m trails, each trail contains three "top" tasks, and the fuel consumption for these three "top" tasks added up equals B. This means that there exists a solution to the original 3PARTITION instance.

NP-completeness in the strong sense. The transformation is bounded by δ, λ. RING(δ, λ) is strongly NP-complete. This is also implied by the fact that the original problem 3PARTITION is strongly NP-hard. □

Theorem 5. *RING$(\delta, 3)$ is strongly NP-complete.*

Proof. RING$(\delta, 3)$ is in **NP**: It runs in time polynomial to the problem size, to verify whether a sequence of pickup-and-delivery tasks in a RING$(\delta, 3)$ is a solution to a given problem instance.

NP-hardness. NP-hardness can be proved through a straightforward transformation from N3DM, which is strongly NP-complete. From an instance $\langle m, \mathcal{W}, \mathcal{X}, \mathcal{Y}, B \rangle$ in N3DM, an instance $\Theta = \langle \mathcal{G}, depot, f^c \rangle$ in RING$(\delta, 3)$ is created. We use Example 4 to illustrate the transformation (the resulting task graph in Θ is shown in Figure 3). In Θ, the fuel-tank capacity of the only vehicle equals $B = 32$ (which is the sum of all integers in all the three sets, divided by $m = 4$). In total three cities C_a, C_b and C_c are introduced and they are aligned in a row from left to right, as shown in Figure 3. There exist exactly $m = 4$ pickup-and-delivery tasks from a given city to its right neighbor: "top", "higher", "lower", and "bottom". In addition, there are four pickup-and-delivery tasks from city (C_c in Figure 3) to city (C_a in Figure 3). No other tasks are introduced in Θ. In the order of $\mathcal{W}$, $\mathcal{X}$, and $\mathcal{Y}$, all the numbers in each one of these three sets are one-to-one and onto mapped into tasks between these three cities. City C_a is the only depot city where the vehicle starts/finishes, and

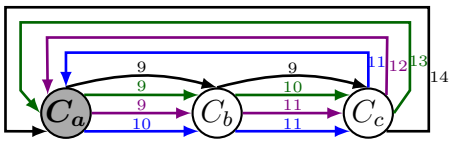


Figure 3. The RING$(\delta, 3)$ instance created from transforming an instance of the N3DM problem (which is strongly NP-hard): $\mathcal{W} = \{9, 9, 9, 10\}$, $\mathcal{X} = \{9, 10, 11, 11\}$, $\mathcal{Y} = \{11, 12, 13, 14\}$ (Example 4). The city C_a is the only depot city and the only fuel station. The only vehicle in the example has a fuel capacity 32. Fuel consumptions to achieve these tasks are labeled accordingly. A solution is highlighted using different colors. It contains four trails: The black trail taking tasks of $\{9, 9, 14\}$, the green trail taking tasks of $\{9, 10, 13\}$, the violet trail $\{9, 11, 12\}$, and the blue trail $\{10, 11, 11\}$.

the only fuel station in Θ. Given the N3DM instance, the resulting RING$(\delta, 3)$ instance is obtained, and shown in Figure 3. Transformation is complete. The solution in the figure is again highlighted in black, green, violet, and blue colors, respectively, referring to four feasible trails in the solution (an Euler Tour).

A N3DM problem instance $\langle m, \mathcal{W}, \mathcal{X}, \mathcal{Y}, B\rangle$ exists a solution iff a vehicle starting from C_a can accomplish all tasks, returning eventually to C_a with the fuel capacity constraint applied on C_a satisfied. The $(\Rightarrow)$ direction is straightforward and is skipped here.

For the $(\Leftarrow)$ direction, we prove that existence of an Euler Tour solution in Θ implies that $\langle m, \mathcal{W}, \mathcal{X}, \mathcal{Y}, B\rangle$ has a solution. For each trail TR in Θ the fuel consumption can not be exceeding B, in order to be feasible. Meanwhile, it can not be less than B. Since the overall requirement on consumption for all the m trails are $m \times B$, if fuel consumption for TR is less than B, it necessarily means that fuel consumption for some another trail in Θ would be greater than B, which is infeasible. Hence, the fuel consumption for each trail in Θ is exactly B, but all these trails define a solution in the original N3DM problem instance.

NP-completeness in the strong sense. Since the transformation remains to be bounded by δ, RING$(\delta, 3)$ remains to be strongly NP-complete. This is also implied by the fact that the original problem N3DM is strongly NP-hard. □

Theorem 6. *RING$(3, \lambda)$ is weakly NP-complete.*

Proof. **NP-hardness, at least weakly.** We use the weakly NP-complete PARTITION(3) (Theorem 2) to perform the transformation[7]. For RING$(3, \lambda)$. Figure 4 illustrates a transformation using Example 2, where the set $\mathcal{A} = \{1, 2, 3, 4, 5\}$ is given, and it can actually be partitioned into three disjoint sets $\mathcal{A}_1 = \{1, 4\}$, $\mathcal{A}_2 = \{2, 3\}$, and $\mathcal{A}_3 = \{5\}$. Accordingly there are three trails in the transformed RING$(3, \lambda)$ instance (Figure 4). It holds in general, $\mathcal{A}$ can be partitioned evenly into three subsets iff the transformed RING$(3, \lambda)$ has a solution.

A psedo-polytime algorithm (hence actually NP-hard only in the weak sense). Additionally we can construct a city-fuel table (similar to the one constructed in [19]) and use a pseudo-polynomial time dynamic-programming algorithm to check for a solution, after the table is filled. □

Since weakly NP-completeness can be generalized, from PARTITION(2,) to PARTITION(3) (Theorem 2), and then to PARTITION(k) (Theorem 3), naturally we have the following result (actual proof skipped).

Corollary 7. *RING(k, λ), where k is any positive integer, is weakly NP-complete.*

Moving towards the direction from $RING(2, \lambda)$ to $RING(3, \lambda)$, and $RING(k, \lambda)$, we are assured that $RING(3, \lambda)$ must be at the least weakly NP-hard. However it is not possible at the point

[7]The proof is similar to Theorem 4 in [19], where RING$(2, \lambda)$ is shown to be weakly NP-complete.

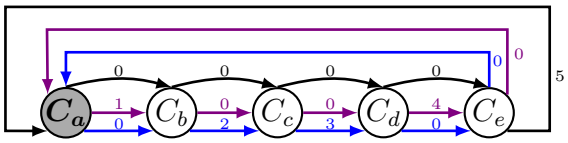


Figure 4. The RING$(3, \lambda)$ instance created from transforming a PARTITION(3) problem instance (which is only weakly NP-hard): $\mathcal{A} = \{1, 2, 3, 4, 5\}$ (Example 2). The city C_a is the only depot city and the only fuel station. The only vehicle in the example has a fuel capacity 5. Fuel consumptions to achieve these tasks are labeled accordingly. A solution is highlighted using different colors. It contains three trails: The black trail taking tasks of $\{0, 0, 0, 0, 5\}$, the violet trail of $\{1, 0, 0, 4, 0\}$, and the blue trail of $\{0, 2, 3, 0, 0\}$.

Algorithm 2 A RING$(\delta, 2)$ Solver

Input: $\Theta = \{C_L, C_R, \mathcal{A}, \mathcal{B}, \text{depot} \equiv C_L, f^c\}$
Output: Boolean variable `result`

1: Set `result` to *true*;
2: Define $\mathcal{A} \equiv \mathcal{A}^0$, and $\mathcal{B} \equiv \mathcal{B}^0$;
3: Sort all tasks in $\mathcal{A}$ in decreasing order $a_1, \ldots, a_\delta$;
4: **for** (each task $a_i \in \mathcal{A}$, from a_1 to a_δ) **do**
5: Search for $\hat{b}$ from $\mathcal{B}^{(i-1)}$ such that
- $(a_i + \hat{b}) \leq f^c$; and
- for any $\bar{b} \in \mathcal{B}^{(i-1)}$: $\Big[\big((a_i + \bar{b}) \leq f^c\big) \implies (\hat{b} \geq \bar{b})\Big]$;

6: **if** ($\hat{b}$ does not exist) **then**
7: Set `result` to *false*;
8: Break the **while** loop;
9: **end if**
10: **end for**
11: Remove task a_i from $\mathcal{A}^{(i-1)}$ to construct $\mathcal{A}^i$;
12: Remove task $\hat{b}$ from $\mathcal{B}^{(i-1)}$ to construct $\mathcal{B}^i$;
13: **Return** `result`

to conclude that the problem is in fact strongly NP-hard. Theorem 6 thus enable us to draw the first dashed-line in Figure 5, which divides the Strongly and the Weakly NP-completeness.

3.2. A Greedy Algorithm

Theorem 8. *RING$(\delta, 2)$ is polynomial time solvable.*

Proof. We present a greedy algorithm which solves RING$(\delta, 2)$ in polynomial time. An input instance to the algorithm has two cities, the *left* city C_L and the *right* city C_R. Set $\mathcal{A}$ contains δ pickup-and-delivery tasks, from C_L to C_R. Set $\mathcal{B}$ contains another set of δ tasks, from C_R back to C_L. City C_L is the station/depot for the only vehicle vhl. Fuel capacity for vhl is f^c. The algorithm returns a Boolean value `result` to indicate whether the input RING$(\delta, 2)$ problem instance has a solution.

Respectively Algorithm 2 first copies $\mathcal{A}$ to $\mathcal{A}^0$, and copies $\mathcal{B}$ to $\mathcal{B}^0$. It then sorts all the tasks in $\mathcal{A}$ in decreasing order, from a_1 to a_δ. The algorithm goes through these tasks one iteration at a time using a for-loop, generating sets $\{\mathcal{A}^1, \mathcal{B}^1\}, \ldots, \{\mathcal{A}^\delta, \mathcal{B}^\delta\}$ one at a time. If the loop finishes with all δ iterations performed completely, variable `result`, which is initialized to "true", remains "true", and Sets $\{\mathcal{A}^\delta, \mathcal{B}^\delta\}$ will be generated and they are empty sets.

At the i^{th} iteration, a_i is removed from $\mathcal{A}^{i-1}$ to create $\mathcal{A}^i$, and $\hat{b}$ is removed from $\mathcal{B}^{i-1}$ to create $\mathcal{B}^i$ (line 10-11). Among all tasks in $\mathcal{B}^{i-1}$ who can be picked up right after task a_i to construct a fuel-feasible trail from C_L to C_R then back to C_L, $\hat{b}$ is the one with maximal fuel consumption value (line 5, the actual greedy part of the algorithm). At any iteration, if $\hat{b}$ does not exist, `result` will be set to false (i.e., the problem instance is not solvable), and the algorithm breaks the loop and terminates right away.

Time complexity. The algorithm runs in $O(\delta \log \delta)$: It contains a $O(\delta \log \delta)$ time sorting step, on δ tasks in $\mathcal{A}$. A loop for maximally δ iterations, and each iteration i involves a $O(\log \delta)$ binary searching for the $\hat{b}$ in the set $\mathcal{B}^i$ (assuming $\mathcal{B}^i$ is also sorted).

Correctness. We prove that RING$(\delta, 2)$ has a solution iff Algorithm 2 returns "true". If Algorithm 2 returns "true", obviously it implies the existence of a solution to the problem. What is not immediately clear is the other direction: If a solution exists, Algorithm 2 will necessarily return "true".

But we reason that, if a solution does exist, after the i^{th} iteration, the algorithm can be expanded to a solution using tasks that have not been considered (in $\mathcal{A}^i$), and the tasks in $\mathcal{B}^i$. Hence when the algorithm terminates, necessarily all the tasks in $\mathcal{A}$ will be one-on-one and onto matched to all the tasks in $\mathcal{B}$. In other words, the algorithm actually finishes with a solution to the problem.

We can formalize the reasoning using mathematical induction over δ w.r.t. a RING$(\delta, 2)$ instance $\Theta = \{C_L, C_R, \mathcal{A}, \mathcal{B}, \text{depot} \equiv C_L, f^c\}$, with the definition of *promising* in the following:

Definition 9. *The pair $\{\mathcal{A}^i, \mathcal{B}^i\}$ is promising after iteration i in Algorithm 2, if using all the tasks in $\mathcal{A}^i$ (in total $(\delta - i)$ of them), and all the tasks in $\mathcal{B}^i$ (also in total $(\delta - i)$ of them), we can construct in total $(\delta - i)$ trails, between C_L and C_R.*

Lemma 9. *Assume that a given RING$(\delta, 2)$ has a solution. Let $\mathcal{A}^i$ and $\mathcal{B}^i$ be the two sets generated after the i^{th} iteration (for $0 \leq i \leq \delta$) in Algorithm 2, assuming that the i^{th} iteration is complete, thus line 10-11 are executed, and $\mathcal{A}^i$ and $\mathcal{B}^i$ are generated. Let $P(i)$ be the statement of "After the i^{th} iteration, the sets $\mathcal{A}^i$ and $\mathcal{B}^i$ can be generated, and $\{\mathcal{A}^i, \mathcal{B}^i\}$ is promising". $P(i)$ holds for every i, where $0 \leq i \leq \delta$.*

Base case. $P(0)$ holds because we start with $\mathcal{A} \equiv \mathcal{A}^0$, and $\mathcal{B} \equiv \mathcal{B}^0$, and it is assumed that the instance has a solution, which is equivalent to the statement that using tasks in the original $\mathcal{A}$ and $\mathcal{B}$, a solution can be constructed.

Induction step. Let $0 \leq i < \delta$ and we show that, if $P(i)$ holds, $P(i+1)$ will also hold. At iteration $(i+1)$ we consider task $a_{(i+1)}$ and remove it from $\mathcal{A}^i$ to create $\mathcal{A}^{(i+1)}$. Regarding $\hat{b}$ in $\mathcal{B}^i$, there are exactly three cases possible:

Case One. There does not exist any task $\hat{b} \in \mathcal{B}^i$ satisfying $(a_{(i+1)} + \hat{b}) \leq f^c$. This is impossible, because we have $P(i)$, hence $\{\mathcal{A}^i, \mathcal{B}^i\}$ is promising in terms of finding eventually a solution including certainly matching a task $\hat{b}$, for $a_{(i+1)}$.

Case Two. There exists a $\hat{b} \in \mathcal{B}^i$ such that $(a_{(i+1)} + \hat{b}) \leq f^c$, and for any $\bar{b} \in \mathcal{B}^i$,

$$\Big[\big((a_{(i+1)} + \bar{b}) \leq f^c\big) \implies (\hat{b} \geq \bar{b})\Big],$$

and $P(i+1)$ holds, but this means the resulting $\{\mathcal{A}^{(i+1)}, \mathcal{B}^{(i+1)}\}$ is still promising.

Case Three. There exists a $\hat{b} \in \mathcal{B}^i$ such that $(a_{(i+1)} + \hat{b}) \leq f^c$, and for any $\bar{b} \in \mathcal{B}^i$,

$$\Big[\big((a_{(i+1)} + \bar{b}) \leq f^c\big) \implies (\hat{b} \geq \bar{b})\Big],$$

but $P(i+1)$ does not hold. In other words, after removing $a_{(i+1)}$ from $\mathcal{A}^i$ to create $\mathcal{A}^{(i+1)}$, and removing $\hat{b}$ from $\mathcal{B}^i$ to create $\mathcal{B}^{(i+1)}$, the resulting $\{\mathcal{A}^{(i+1)}, \mathcal{B}^{(i+1)}\}$ is no longer promising. Nevertheless, given that $\{\mathcal{A}^i, \mathcal{B}^i\}$ is promising, $\{\mathcal{A}^i, \mathcal{B}^i\}$ will lead to a solution to the problem, say $\mathcal{S}$. The following two statements must be valid in $\mathcal{S}$:

- $a_{(i+1)} \in \mathcal{A}^i$ is matched to some $\bar{b} \in \mathcal{B}^i$, where $\bar{b} \leq \hat{b}$ must hold (remember that the algorithm is greedy);
- $\hat{b}$ is matched to some $\bar{a} \in \mathcal{A}^i$, where $\bar{a} \leq a_{(i+1)}$ must hold (remember that all the tasks in $\mathcal{A}$ are sorted in decreasing order).

We accordingly define $\mathcal{S}^{alt}$, which is an alternative to $\mathcal{S}$. $\mathcal{S}^{alt}$ differs from $\mathcal{S}$ only in the following two matchings:

- $a_{(i+1)}$ is instead matched to $\hat{b}$ in $\mathcal{S}^{alt}$;
- $\bar{a}$ is instead matched to $\bar{b}$ in $\mathcal{S}^{alt}$;

Since $\bar{a} \leq a_{(i+1)}$, $\bar{b} \leq \hat{b}$, and the trail of $a_{(i+1)}$ followed by $\hat{b}$ is fuel-feasible, it is the case the trail of $\bar{a}$ followed by $\bar{b}$ must also be fuel-feasible. Hence $\mathcal{S}^{alt}$ must also be a solution to the problem. That is, removing $a_{(i+1)}$ from $\mathcal{A}^i$ to create $\mathcal{A}^{(i+1)}$, and removing $\hat{b}$ from $\mathcal{B}^i$ to create $\mathcal{B}^{(i+1)}$, the resulting $\{\mathcal{A}^{(i+1)}, \mathcal{B}^{(i+1)}\}$ remains to be promising. That is, $P(i+1)$ also holds for Case Three.

We have proved Lemma 9: If a $RING(\delta, 2)$ problem instance has a solution, we will always obtain $P(\delta)$ from running Algorithm 2 on $RING(\delta, 2)$, with two empty sets $\mathcal{A}^\delta$ and $\mathcal{B}^\delta$ generated, which means, if a solution exists to the problem, Algorithm will return "true". □

Example 5. *Given cities C_L (depot) and C_R, three tasks $\mathcal{A} = \{6, 4, 4\}$ from C_L to C_R, three other tasks $\mathcal{B} = \{4, 3, 1\}$ from C_R to C_L. When the fuel capacity f^c is 8, the example has a solution: 1st trail $(6, 1)$, 2nd trail $(4, 4)$, and 3rd trail $(4, 3)$. However if f^c is reduced to 7, the algorithm will return false after the two trails $(6, 1)$ and $(4, 3)$. Meanwhile, the example does not have a solution.*

Theorem 10. *$RING(\Delta, \Lambda)$ is polynomial-time solvable.*

Proof. Trivially in $O(1)$, as both Δ the width, and Λ the length of the problem, are constants. □

4. Conclusion

This paper studies the Green Pickup-and-Delivery problems from complexity-theoretic perspective, with a focus on the GPDs whose task graphs are on ring structures. Through tweaking on the values of the width and the length, of the rings in RINGs, several complexity results, either tractable or intractable, are obtained. Relationships between these results are pictorially summarized in Figure 5, where arrows connect problems to their restricted variants. Two dashed lines in the figure respectively separate among the defined RING problems, strongly and weakly NP-hard, and intractable and polynomial-time solvable.

These new results deepen our understanding of the green intractability in vehicle routing. For example, they confirmed the relevance of restricting the topology of task graphs for reducing GPD computational complexity in vehicle routing for entity picking up and delivery.

Availability of these results particularly enhances our knowledge on computational properties of RING problems. We now have quantitative measures regarding how to achieve RING tractability through applying restrictions on either the width or the length of rings in RING problems. Specifically we observe that

- when λ is a variable, RING remains to be weakly NP-complete, for $\delta = 2, 3, 4, \ldots, k$; However
- when δ is a variable, RING is strongly NP-complete when $\lambda = 3$, but polynomial-time solvable when $\lambda = 2$.

Being able to separate weakly/strongly NP-hard GPDs has valuable implications. For GPD problems which are only weakly NP-hard, we remark that although pseudo-polynomial time algorithms go exponential with the input size of f^c, it is reasonable, however, to assume that the fuel capacity of vehicles are always small in their values, making it practical to use these algorithms to address real-world problems, and to develop applications.

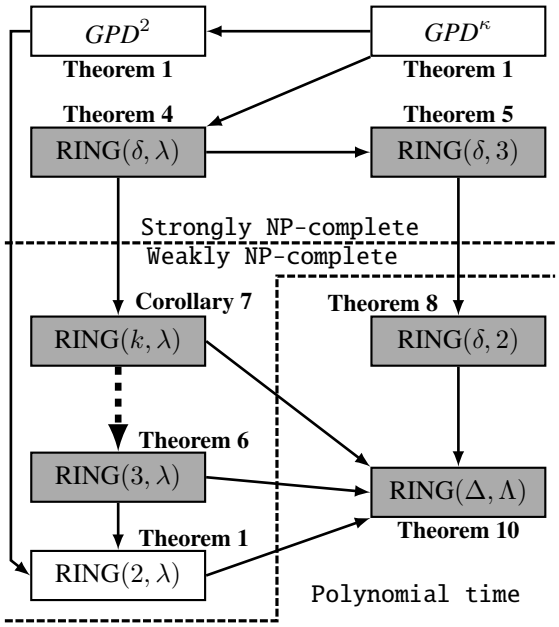


Figure 5. Computational hierarchy of restricted GPD/RING problems. Arrows connect problems to their restricted variants. In GPD$^{\kappa}$, the degree of nodes in its task graph is bounded by κ. In RING(δ, λ), width, and length, of the task ring are δ and λ, respectively. The parameters Δ and Λ in RING(Δ, Λ) are constants. Theorems 4-6, Corollary 7, and Theorem 8, 10 are newly-obtained results (highlighted in grey-color). The dashed line between RING(k, λ) and RING$(3, \lambda)$ highlights the fact that as long as the width is kept as a constant, the problem is only weakly NP-hard.

Modern dispatching systems often face the situation where multiple inventories need to be exchanged or transited between two major clusters or distribution centers (to avoid splitting of one order into multiple shipments, or to meet the safety stock requirements, for example). In correspondence, such a situation naturally can be modelled/approximated as a RING$(\delta, 2)$ problem to solve. Polynomial-time solvability of RING$(\delta, 2)$ as identified in this paper (Theorem 8) provides critical insight in support of the development and evaluation of real-world applications as such. Further, the greedy algorithm itself can be conveniently adjusted to solve an optimization variant to RING$(\delta, 2)$, with numeric minimizaion/maximization objectives in general sense considered (not just referring to the vehicle fuel consumption). Finally, as part of our future work, we will investigate how to use the new insights obtained from this research, for design and development of algorithms efficient in practice for solving GPD/RING problems (e.g., heuristics, meta-heuristics, approximation, etc).

Acknowledgements

We are thankful to anonymous reviewers for their feedback and suggestions. This work was supported by a start-up research fund from the Faculty of Science and Environmental Studies, Lakehead University, a research opportunity grant from Social Sciences and Humanities Research Council of Canada (SSHRC), and a discovery grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] M. Savelsbergh and M. Sol. "The General Pickup and Delivery Problem". In: *Transportation Science* 29.1 (1995), pp. 17–29.

[2] P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2014.

[3] X. Tan and J. Huang. "On Computational Complexity of Pickup-and-Delivery Problems with Precedence Constraints or Time Windows". In: *Proc. of the 28th IJCAI*. 2019, pp. 5635–5643.

[4] B. Coltin and M. Veloso. "Scheduling for Transfers in Pickup and Delivery Problems with Very Large Neighborhood Search". In: *Proc. of the 28th AAAI*. Québec City, Québec, Canada, 2014, pp. 2250–2256.

[5] M. L. Gini. "Multi-Robot Allocation of Tasks with Temporal and Ordering Constraints." In: *Proc. of the 31st AAAI*. 2017, pp. 4863–4869.

[6] C. Zhang and J. A. Shah. "Co-Optimizating Multi-Agent Placement with Task Assignment and Scheduling". In: *Proc. of the 25th IJCAI*. 2016, pp. 3308–3314.

[7] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, R. Barták, and E. Boyarski. "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks". In: *Proc. of the 12th SOCS*. AAAI Press, 2019, pp. 151–159.

[8] O. Salzman and R. Stern. "Research Challenges and Opportunities in Multi-Agent Path Finding and Multi-Agent Pickup and Delivery Problems". In: *Proc. of the 19th AAMAS*. 2020, pp. 1711–1715.

[9] F. Bistaffa, A. Farinelli, and S. D. Ramchurn. "Sharing Rides with Friends: A Coalition Formation Algorithm for Ridesharing". In: *Proc. of the 29th AAAI*. Austin, Texas, 2015, pp. 608–614.

[10] S. Erdoğan and E. Miller-Hooks. "A Green Vehicle Routing Problem". In: *Transportation Research Part E: Logistics and Transportation Review* 48.1 (2012), pp. 100 –114.

[11] E. Demir, T. Bektaş, and G. Laporte. "A review of recent research on green road freight transportation". In: *European Journal of Operational Research* 237.3 (2014), pp. 775 –793.

[12] T. Bektaş, E. Demir, and G. Laporte. "Green Vehicle Routing". In: *Green Transportation Logistics: The Quest for Win-Win Solutions*. Springer International Publishing, 2016, pp. 243–265.

[13] J. Jemai, M. Zekri, and K. Mellouli. "An NSGA-II Algorithm for the Green Vehicle Routing Problem". In: *Evolutionary Computation in Combinatorial Optimization*. Ed. by J.-K. Hao and M. Middendorf. Springer Berlin Heidelberg, 2012, pp. 37–48.

[14] Y. Xiao and A. Konak. "A genetic algorithm with exact dynamic programming for the green vehicle routing & scheduling problem". In: *Journal of Cleaner Production* 167 (2017), pp. 1450 –1463.

[15] Çağrı Koç and I. Karaoglan. "The green vehicle routing problem: A heuristic based exact solution approach". In: *Applied Soft Computing* 39 (2016), pp. 154 –164. ISSN: 1568-4946.

[16] Y. Xiong, J. Gan, B. An, C. Miao, and A. L. C. Bazzan. "Optimal Electric Vehicle Charging Station Placement". In: *Proc. of the 24th IJCAI*. Buenos Aires, Argentina, 2015, pp. 2662–2668.

[17] H. C. Lau, L. Agussurja, S.-F. Cheng, and P. J. Tan. "A Multi-objective Memetic Algorithm for Vehicle Resource Allocation in Sustainable Transportation Planning". In: *Proc. of the 23rd IJCAI*. Beijing, China, 2013, pp. 2833–2839.

[18] J. Eisner, S. Funke, and S. Storandt. "Optimal Route Planning for Electric Vehicles in Large Networks". In: *Proc. of the 25th AAAI*. 2011.

[19] X. Tan and J. Huang. "A Complexity-theoretic Analysis of Green Pickup-and-Delivery Problems". In: *Proc. of the 35th AAAI*. 2021, pp. 11990–11997.

[20] M. Garey and D. Johnson. *Computers and intractability - a guide to NP-completeness*. W.H. Freeman and Company, 1979.