A Complexity-Theoretic Analysis of Green Pickup-and-Delivery Problems

Xing Tan and **Jimmy Xiangji Huang**
Information Retrieval and Knowledge Management Research Lab
York University, Toronto, Ontario, Canada
{xtan, jhuang}@yorku.ca

Abstract

In a Green Pickup-and-Delivery problem (GPD), vehicles traveling in a transport network achieving pickup-and-delivery tasks are in particular subject to the two *green* constraints: limited vehicle fuel capacity thus short vehicle traveling range, and limited availability of refueling infrastructure for the vehicles. GPD adds additional but probably insignificant computational complexity to the classic and already NP-hard Pickup-and-Delivery problem and Vehicle Routing Problem. Nevertheless, we demonstrate in this paper an inherent intractability of these green components themselves. More precisely, we show that GPD problems whose total constraints are reduced to almost the green ones only, remain to be NP-complete in the strong sense. We figure out a specifically constrained variant of GPD that, however, is weakly NP-complete – a practical pseudo-polynomial time algorithm solving the variant problem is identified. Insight obtained from this complexity-theoretic analysis would shed light for a deeper understanding of GPDs, and on better development of heuristics for solving these problems, leading to promisingly many real-world applications.

Introduction and Motivation

Pickup-and-Delivery problem (PD) is an important variant of the classic Vehicle Routing Problem (VRP) [Toth and Vigo 2014; Savelsbergh and Sol 1995]. PD concerns construction of vehicle routes in a transportation network to satisfy requests in terms of picking up and delivering loads (e.g., goods, packages, or passengers) between origin and destination locations [Savelsbergh and Sol 1995]. The problem is closely related to several areas in AI including, for example, planning and scheduling [Beck, Prosser, and Selensky 2003; Coltin and Veloso 2014], robotics [Gini 2017], multi-agent systems [Vokřínek, Komenda, and Pěchouček 2010; Zhang and Shah 2016], and intelligent transportation [Bistaffa, Farinelli, and Ramchurn 2015].

A lot of recent studies in VRP and PD have focused on routing with vehicles powered by alternative energy sources such as electricity, hydrogen, or natural gas. Road transport with a more prevalent use of these so-called *green* vehicles in partial replacement of fossil-fuel powered conventional vehicles, reduces greenhouse gas emission, and creates positive impacts for environment, economy, and human health. However, due to the limited fuel capabilities of green vehicles, and the lack of the refueling infrastructures for these vehicles, the practice of modern transport planning calls for effective green-specific adaptation and mitigation.

A Green Pickup-and-Delivery problem (GPD) extends PD, addressing in particular the constraints on driving range of vehicles, and on the availability of refueling infrastructure for vehicles [Erdoğan and Miller-Hooks 2012; Demir, Bektaş, and Laporte 2014]. Computationally, GPD adds an additional, green, dimension of complexity to PD/VRP, which are already NP-hard. Reasonably so, most current approaches solving GPD problems work on special application cases [Eisner, Funke, and Storandt 2011; Lau et al. 2013; Xiong et al. 2015], or are based on certain heuristics for approximate solutions [Ángel Felipe et al. 2014]. Examples include using genetic algorithms [Xiao and Konak 2017], evolutionary algorithms [Jemai, Zekri, and Mellouli 2012], simulated annealing [Çağrı Koç and Karaoglan 2016], sampling [Montoya et al. 2016], and abstraction [Schönfelder and Leucker 2015].

Despite the algorithmic progress, we ought to know at the outset *whether the green constraints themselves actually create another dimension of computational intractability*. In this paper, we first cast GPD into its simplest form. To be precise, the pickup-and-delivery part is reduced into a polytime solvable Euler Tour. Accordingly the only possible source of computational intractability comes from the green components, i.e., from the constraint that a vehicle might need to get refilled before finishing its tour for accomplishing tasks. Next we consider applications of further constraints, on the total number of depots and fuel stations available, on bounded value of in-degree and out-degree of cities, and on topological structure of tasks, attempting to construct boundary intractable GPD subproblems. We report that the green part in GPD is inherently intractable – all the nontrivially restricted subproblems considered are, either strongly or weakly, NP-complete. When the topology of the task graph in a GPD is a ring (so-called), a pseudo-polynomial time algorithm solving the problem exists. Insight obtained from this complexity-theoretic analysis would shed light for a deeper understanding of GPD, and for better practice of GPD heuristics development, an active research area with many important applications in real world, potentially creating environmental, economic, and social impacts.

The remainder of this paper is organized as follows. First we present definitions of GPD and several of its constrained variants. Complexity results for these GPD problems are presented and proved in the next section. Finally we summarize the paper.

Definitions

We define in this section the Green Pickup-and-Delivery problem (GPD) and several of the constrained variants of GPD: GPD_{stn}, GPD_{dgr}, GPD_{stn_dgr}, and $GPD^{ring}_{stn\_dgr}$.

Definition 1 (PD)**.** *In a general Pickup-and-Delivery problem, we have a set of cities $\mathcal{C}$. Between two cities $C_p, C_d \in \mathcal{C}$. A transportation task $T = \langle C_p, C_d, n_{p,d}\rangle$ refers to picking up a load of size $n_{p,d}$ from city C_p and delivering it to city C_d. All such tasks define the set $\mathcal{T}$. Vehicles in $\mathcal{V}$ serve tasks in $\mathcal{T}$. Each $vhl \in \mathcal{V}$ has its own origin city and a destination city. These two cities are also in $\mathcal{C}$, and the work-load capacity of vehicle vhl is constrained by $l_{vhl} \in \mathbb{N}$.*

There might be additional specifications or constraints considered, such as precedence requirements on order of tasks to be served, cost, time windows, and etc.

A PD problem asks for finding a feasible plan consisting of routes for vehicles in $\mathcal{V}$ serving all the tasks, with values of specific objective functions minimized (or satisfied). Formal definitions on several variants of PDs can be found in [Savelsbergh and Sol 1995].

Definition 2 (GPD)**.** *A Green Pickup-and-Delivery problem is a PD and in addition*

- *The definition of task $T \in \mathcal{T}$ is extended to include $f_{p,d} \in \mathbb{N}$, a non-negative number reflecting the actual fuel consumption travelling from city C_p to city C_d;*
- *A vehicle $vhl \in \mathcal{V}$ is further constrained with a fuel-tank capacity $f_{vhl} \in \mathbb{N}$; and*
- *A set $\mathcal{S}$ of fuel stations is considered, where vehicle vhl visiting of any $S \in \mathcal{S}$ will refuel the vehicle to its full. Fuel supplies in the stations are unlimited.*

Hence, all vehicles in GPD are subject to the constraints of how far they can travel on road, proportional to their fuel-tank capacities. Visiting fuel stations are not tasks required to be achieved by vehicles, but are necessarily actions to be taken before their fuel tanks running empty.

Definition 3. *GPD_{stn} is a highly-restricted GPD, which defines a 5-tuple $\Theta = \langle \mathcal{C}, \mathcal{T}, \mathcal{V}, \mathcal{S}, depot\rangle$ where*

- *$\mathcal{C}$ is the set of vertices (corresponding to cities);*
- *$\mathcal{T}$ is the set of edges (corresponding to tasks). For any $T = \langle C_i, C_j, n_{i,j}, f_{i,j}\rangle \in \mathcal{T}$, a one-unit entity (passenger, package, commodity, etc.) needs to be picked up from city C_i and delivered to city C_j (i.e.,$n_{i,j} = 1$);*
- *Vehicle $vhl \in \mathcal{V}$ is the only vehicle (i.e., $|\mathcal{V}| = 1$), with a fuel-tank capacity $f_{vhl} \in \mathbb{N}$; Work-load capacity of vhl is also 1 (i.e., $l_{vhl} = 1$);*
- *depot $\in \mathcal{C}$ is the only depot city where the vehicle starts/finishes (i.e., origin city and destination city are the same);*
- *depot $\in \mathcal{S}$, and $|\mathcal{S}| = 1$. That is, the depot city also serves as the only fuel station in the problem.*

Graph $\mathcal{G} = \langle \mathcal{C}, \mathcal{T}\rangle$ defines a weighted, directed graph, where vertices, edges, and weights in $\mathcal{G}$ respectively correspond to cities, tasks, and actual fuel consumptions to accomplish these tasks. A walk in the graph leaving and finishing at the depot such that all tasks in T are traversed corresponds to an Euler Tour[1] in the graph $\mathcal{G}$. An Euler Tour is feasible if the tour also satisfies the fuel capability constraint applied on the vehicle vhl. Also note that an Euler Tour might consist of more than one closed trail in a sequence.

INSTANCE: A GPD_{stn} and its 5-tuple Θ.
QUESTION: Does there exist a feasible Euler Tour of $\mathcal{G}$ for the vehicle vhl?

A GPD_{stn} example is given in Figure 1. The example 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. 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_{stn} to have a solution, however, it has to have a feasible Euler Tour in the graph. Taking the following two Euler Tours for example,

$$ET0: [t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8]; ET1: [t_4, t_8, t_1, t_7, t_5, t_6, t_2, t_3].$$

ET0 (as illustrated in Figure 1) is a solution: It takes 8 fuel units to walk the first trail $[t_1, t_2, t_3]$. After the first trail, the vehicle returns to C_a and gets refueled before walking its second trail $[t_4, t_5, t_6, t_7, t_8]$, which takes 6. ET1 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 2nd 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.

Definition 4. *GPD_{dgr} is another highly-restricted GPD, and a close variant to GPD_{stn}. GPD_{dgr} differs from GPD_{stn} only in that, any city in a GPD_{dgr} problem has in-degree=out-degree $\leq$ 2, but the problem might have more than one depot/fuel_station.*

Definition 5. *GPD_{stn_dgr} is a further restricted variant, to both GPD_{dgr} and GPD_{stn}. That is, the condition of in-degree=out-degree $\leq$ 2 holds and there is just one depot/fuel_station in GPD_{stn_dgr}.*

Definition 6. *$GPD^{ring}_{stn\_dgr}$ applies further restrictions to GPD_{stn_dgr}. That is, the topology of the task graph $\mathcal{G}$ is a ring: For any city A in the task graph, there are exactly two tasks picked up from some city B and delivered to A, and there are exactly two tasks picked up from A and delivered to some another city C, where these three cities are distinct.*

[1] We use standard definitions from 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 containing all edges in the graph.

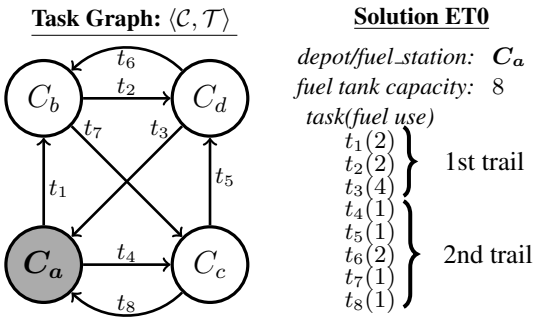

Figure 1: A GPD_{stn} example and its solution ET0. The grey-color city C_a is the only depot, and the only fuel station, for the only vehicle in the example. ET0 is an Euler Tour consisting of the first trail (fuel consumption 8) and the second trail (fuel consumption 6).

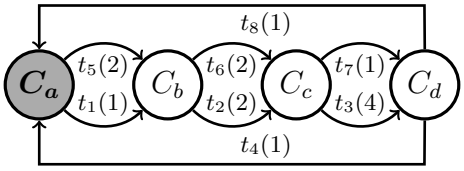

Figure 2: An Example $GPD^{ring}_{stn\_dgr}$. Fuel consumptions of tasks are indicated in the parentheses after their task names.

Figure 2 is an example $GPD^{ring}_{stn\_dgr}$. City C_a is the only depot/fuel_station. Fuel capacity for the vehicle is 8. Fuel consumptions of tasks are indicated in the parentheses after task names. The constraint of in-degree=out-degree $= 2$ holds for all the four cities.

Complexity Results

GPD_{stn}, GPD_{dgr}, and GPD_{stn_dgr}, are shown to be strongly NP-complete in this section. $GPD^{ring}_{stn\_dgr}$ however is weakly NP-complete.

Theorem 1. *GPD_{stn} is strongly NP-complete.*

Proof Sketch[2]**:** GPD_{stn} is in ***NP***. It is easy to verify whether a sequence of pickup-and-delivery tasks is a solution to a given problem instance.

NP-hardness. We perform a polytime transformation from the NP-complete Directed Hamiltonian Circuit Problem (DHC) to GPD_{stn} ([GT38] in [Garey and Johnson 1979]). Figure 3 is an example DHC, which is also used as the running example to illustrate the polytime transformation. From a digraph $G = \langle V, E\rangle$ in a DHC instance, where $|V| = n$, $|E| = m$, we create a corresponding Θ in GPD_{stn}. The task set $\mathcal{T}$ in Θ contains four different kinds of tasks: edge-task, node-task, fixer-task, and returning-task.

For each vertex $v_i \in V$ of G in DHC, we create two cities L_i and R_i in the set $\mathcal{C}$ in Θ in GPD_{stn}. Each vertex v thus corresponds to exactly one dotted rectangle in Figure 4. Each rectangle boxes two cities, one left and one right. Without loss of generality, set L_0 the left-city in v_0 as the depot/fuel_station.

For each edge $(v_i, v_j) \in E$ of G, we create in $\mathcal{T}$ an edge-task picking up a one-unit entity from the city R_i, to be delivered to the city L_j. Inside each dotted rectangle, there is a node-task leaving the left city and entering the right city. We need to ensure that the in-degree of any city equals the out-degree of it. This is to ensure the task graph has Euler Tours in the first place (otherwise the transformed GPD_{stn} problem instance will be simply unsolvable). Note that the in-degree of any right-city equals 1. Hence if the out-degree of a right-city equals $k > 1$, $(k-1)$ fixer-tasks leaving L_0 and entering the right-city will be created. Meanwhile the out-degree of any left-city equals 1. Hence if the in-degree of a left-city equals $k > 1$, $(k-1)$ returning-tasks leaving the left-city and entering L_0 will be created. In Figure 4, thick arrows are fixer-tasks and dashed arrows are returning-tasks. All 14 tasks in the figure are labelled. Fuel consumption for fixer-tasks (t_9 and t_{12} in Figure 4) are set to be $(2n-1)$ (which equals $2 \times 4 - 1 = 7$ for the example in Figure 4). Fuel consumptions for returning-tasks (t_{11} and t_{14} in the example of Figure 4) equals 0. Fuel consumptions for node-tasks and edge-tasks are set to 1. Fuel capacity of the only vhl equals $(2 \times n)$ (which equals 8 for the example).

There exists a DHC in a graph $G = \langle V, E\rangle$ iff there exists a feasible Euler Tour in the transformed Θ for the vehicle.

$(\Rightarrow)$: If G has a DHC, this DHC visits n edges and leaves out $(m-n)$ edges unvisited. The DHC corresponds to a "DHC" trail containing n edge-tasks and n node-tasks (fuel consumptions summed to $2n$ exactly), leaving $(m-n)$ edge-tasks unaccomplished in Θ. The "DHC" trail in Figure 4 is $[t_1, t_2, \ldots, t_7, t_8]$. For each one of these unaccomplished edge-tasks, we use a trail in the following pattern to accomplish it (fuel consumption for the trail is always $2n - 1 + 1 + 0 = 2n$): [fixer-task(2n-1), edge-task(1), returning-task(0)].

Figure 4 has two such trails $[t_9, t_{10}, t_{11}]$, and $[t_{12}, t_{13}, t_{14}]$. Walking in a sequence all these trails, including the "DHC" one, constructs a solution to Θ.

$(\Leftarrow)$: Assume Θ has a solution. A trail in the solution starts with a fixer-task, and then an edge-task, has to return to L_0 using the returning-task (the vehicle runs out of fuel). After all these $(m-n)$ trails involving fixer-tasks, there are exactly n edge-tasks and n node-tasks unaccomplished. One more trail is left, and the trail has to accomplish these $2n$ tasks all, implying that there exists a DHC in G.

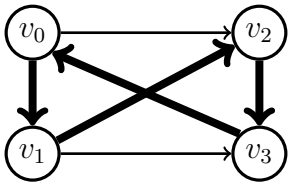

Figure 3: A graph and its Directed Hamiltonian Circuit $\langle v_0, v_1, v_2, v_3, v_0\rangle$, highlighted in thick lines.

[2]How to construct the transformation for proving NP-hardness is sketchily explained by going through an illustrative example.

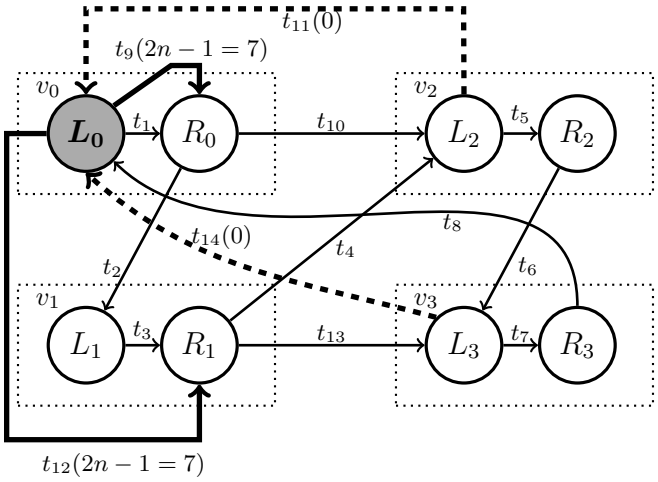

Figure 4: Task Graph $\mathcal{G}$ of $\Theta(G)$ for proving NP-hardness of GPD_{stn} (Theorem 1). Fuel consumptions of tasks by default equal to 1. Other than 1 values are indicated in the parentheses after their task names (t_9, t_{11}, t_{12}, and t_{14}). $G = \langle V, E\rangle$, $|V| = n$, $|E| = m$.

NP-completeness in the strong sense. We see that the transformation is bounded by either $2n$ or m, whichever is greater in value. GPD_{stn} is thus strongly NP-complete. □

Theorem 2. *GPD_{dgr} is strongly NP-complete.*

Proof Sketch: It is noted DHC remains to be NP-complete, even when no node/vertex in the graph is involved with more than 3 edges ((Noted in [GT38] in [Garey and Johnson 1979]). If we use such a graph, denoted as $G_{\leq 3}$, to perform the transformation, the resulting task graph $\bar{\mathcal{G}}$ in Θ (except for the depot/fuel_station node L_0) is bounded by "in-degree = out-degree ≤ 2". But we can add more depot/fuel_stations (i.e., "New" cities) to resolve the issue associated with L_0. Figure 5 illustrates the intuition using the same DHC example in Figure 3 (the example is actually bounded by "in-degree = out-degree ≤ 2"). In the figure, a "New" city is newly introduced and, like L_0, it is also a depot and fuel-station for the only vehicle. To save space, only the component around the dashed rectangle v_0 is shown in Figure 5. Other parts remain to be the same as in Figure 4. All the original 14 tasks in Figure 4 are labelled in the same way in Figure 5. Fuel consumptions for these tasks are also kept unchanged. However task t_9 now starts at the "New" city, and task t_{11} now finishes at the "New" city. The only new task t_{new} starts at the "New" city and finishes at L_0. We see that both the "New" city and L_0 are bounded by the constraint "in-degree = out-degree ≤ 2".

The proof is similar to the one for Theorem 1, and is skipped here to save space. For illustration purpose, a feasible solution for Θ in Figure 5 is given here:

$$[t_9, t_{10}, t_{11}, t_{new}, t_{12}, t_{13}, t_{14}, t_1, t_2, \ldots t_7, t_8].$$

The only vehicle starts at "New". The first trail $[t_9, t_{10}, t_{11}]$ starts and finishes at "New" to achieve t_{10}. The second trail $[t_{12}, t_{13}, t_{14}]$ starts and finishes at L_0 to achieve t_{13}. The two trails are connected by task t_{new}. Tasks t_1 up to t_8 correspond to a "DHC" back in the graph $G_{\leq 3}$. The vehicle returns to "New" after task t_8.

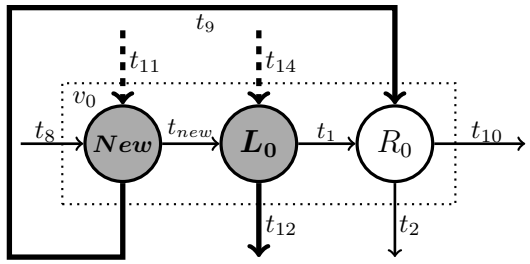

Figure 5: The graph component around the dashed rectangle v_0 in $\mathcal{G}$ of $\Theta(G)$ for proving NP-hardness of GPD_{dgr} (Theorem 2). Fuel consumptions of tasks are not shown in the Figure.

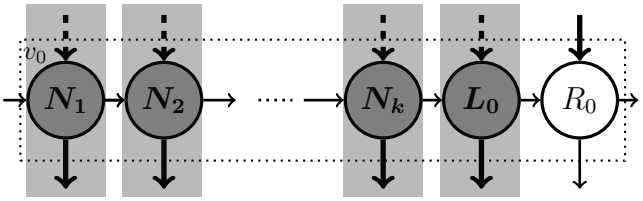

Figure 6: A transformation with k "New" cities involved (from N_1 up to N_k). Each "New" city defines a so-called *lane* (highlighted by a shaded rectangle). The only vehicle is initially at the city N_1. All these fuel stations in addition to L_0 are also fuel stations.

Note that when the transformation is performed on a problem instance large in size, several "New" cities will be introduced into the dashed rectangle v_0. Without affecting the bidirectional argument on the validity of the transformation, it is required some careful consideration on the order these "New" cities to be visited. Specifically the "Stay on one lane, whenever the vehicle can" rule needs to be followed (explained below).

In Figure 6, each shaded-rectangle defines a lane, which contains a "New" city, a returning-task entering the city, and a fixer-task leaving the city. When it visits a "New" city for the first time after fulfilling the returning-task entering the city, the vehicle has to continue in the same lane taking the fixer-task to leave the city. Taking the horizontal task moving to the next "New" city is not allowed. When the vehicle returns to N_1 after visiting several (zero to many) such lanes, it will have to take the horizontal tasks and stop at any "New" city it has never visited before. At this city, the vehicle moves down taking the fixer-task. The procedure continues until the vehicle arrives at L_0, and then R_0 (through the task in between). Again if G has a DHC, Θ has a solution. There exists a "DHC" trail from R_0 to N_1.

NP-completeness in the strong sense. The argument in Theorem 1 regarding NP-completeness in the strong sense remains to be valid here. □

Theorem 3. *GPD_{stn_dgr} is strongly NP-complete.*

Proof Sketch: For GPD_{stn_dgr}, we use again $G_{\leq 3}$. There will be just one fixer-task in the resulting Θ. Returning-tasks now are used to connect unaccomplished edge-tasks before eventually returning to L_0. And this trail is the only one other than the "DHC" one, should a solution exist for the original DHC problem. Figure 7 is created directly from Figure 4, by removing t_{12} and redirecting t_{11} from returning to L_0 to entering R_1 instead.

Fuel capacity remains to be $2n$. Fuel consumptions, for node-tasks and edge-tasks remain to be 1, and for returning-tasks remain to be 0. However fuel consumption for the only fixer-task is set to be $4n-(n+m) = 3n-m$. In the example, fuel consumption for t_9 is thus $4 \times 4 - (4+6) = 6$.

Proof for the "($\Rightarrow$)" direction should be straightforward at this point, hence is skipped to save space. We give some necessary explanation on the "($\Leftarrow$)" direction. Note the fact that fuel consumption for all tasks added up equals to $4n$, and the capacity for the vehicle is $2n$. Meanwhile, if there exists a feasible Euler Tour in the Θ, the tour consists of exactly two trails, each consumes $2n$ before the vehicle returns eventually to L_0. Let TR_0 be the trail that does not start with the only fixer-task, and TR_1 the other trail. If TR_0 repeats node-task/edge-task in turn, eventually there are n node-tasks and n edge-tasks in TR_0, corresponding to a "DHC" back in the graph G. If on the way TR_0 deviates to take a returning-task instead, one node-task would be by-passed. Observe that any edge-task in TR_0 has to be immediately preceded by either a node-task, or a returning-task, it is the case TR_0 can have at most n edge-tasks. A deviation of taking a returning-task will result in the value of total fuel consumption of TR_0 being reduced by 1 (i.e., $2n-1$), leading to a contradiction: Fuel consumption for the other trail TR_1 will have to be $2n+1$, which is greater than the fuel capacity of the vehicle, thus infeasible.

Using Figure 7 for illustration, the two trails are:

$$TR_0 : [t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8]; TR_1 : [t_9, t_{10}, t_{11}, t_{13}, t_{14}].$$

Trail TR_0 corresponds to a "DHC" back in the graph $G_{\leq 3}$. The only other trail TR_1 is dedicated in achieving the left-unachieved tasks (t_{10} and t_{13} in the example). Note that t_{12} as introduced in Theorem 1 and Figure 4 is no longer needed.

Note that between Figure 4 and Figure 7, the returning-task t_{11}, which leaves L_2, is redirected from entering L_0 to entering R_1. After the redirection, there is an edge-task t_4, which leaves R_1 and enters L_2. In general, t_4 can be an unaccomplished edge-task, and if so, the trail (other than the "DHC" one) will get stuck visiting L_2 twice: $[\ldots, L_2, t_{11}, R_1, t_4, L_2]$. When a transformation is performed on a problem large in size, a lot of returning-tasks need to be redirected and a lot of fixer-tasks need to be removed (referring to the procedure of creating Figure 7 from Figure 4). Given the fact that "in-degree=out-degree $\leq$ 2", when there are at least three fixer-tasks to be removed, it is always possible to pick up a returning-task leaving some city L, and to redirect the task to a city R where there is no edge-task leaving R and entering L. When there are only two fixer-tasks left, we can create a "Switch" city for flexibility in determining which one of the two returning-tasks to fix which one of the two unaccomplished tasks. Taking Figure 7 for example, the two remaining returning-tasks t_{11} and t_{14} are redirected to enter "Switch", which in turn enters L_0 and R_1 through $t_{14}^{'}$ and $t_{11}^{'}$ (Figure 8).

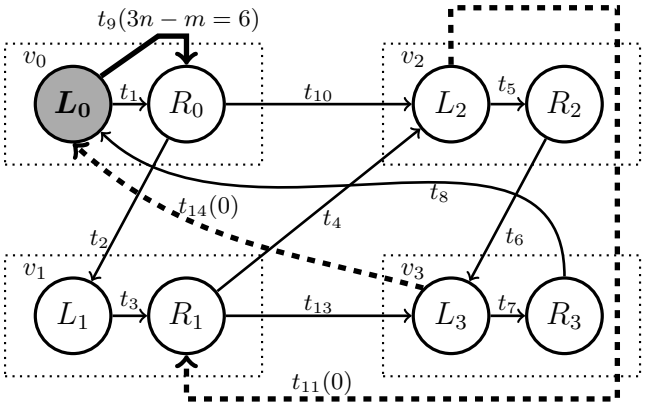

Figure 7: Task graph $\mathcal{G}$ of $\Theta(G)$ for proving NP-hardness of GPD_{stn_dgr} (Theorem 3). Fuel consumptions of tasks by default equal to 1. Other than 1 values are indicated in the parentheses after their task names (t_9, t_{11}, and t_{14}). $G = \langle V, E \rangle$, $|V| = n$, $|E| = m$.

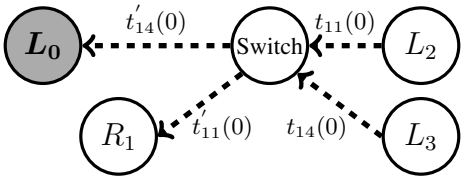

Figure 8: The city "Switch" is added to the task graph $\mathcal{G}$ of $\Theta(G)$ in Figure 7. The two returning tasks (t_{11} and t_{14}) enters "Switch" first, and $t_{11}^{'}$ and $t_{14}^{'}$ leaves "Switch". Now (either L_2 or L_3) can enter (either R_1 or L_0).

NP-completeness in the strong sense. The argument in Theorem 1 regarding NP-completeness in the strong sense remains to be valid here. □

Theorem 4. *$GPD^{ring}_{stn\_dgr}$ is weakly NP-complete.*

Proof Sketch: $GPD^{ring}_{stn\_dgr}$ is again in NP. The proof has two major parts: First we show $GPD^{ring}_{stn\_dgr}$ is at least weakly NP-hard by providing a polytime transformation from PARTITION, which is weakly NP-complete ([SP12] in [Garey and Johnson 1979]); Second we provide a pseudo-polynomial time algorithm to solve $GPD^{ring}_{stn\_dgr}$. Running time of the algorithm is polynomial in terms of the numeric input value, the fuel capacity of the vehicle.

NP-hardness (at least weakly). Given a set of positive integers S, PARTITION tries to divide S into two exclusive sets S_1 and S_2, where the sum of integers in S_1 equals the sum of integers in S_2. For example, given

$S = \{1, 1, 3, 4, 5\}$, the set can actually be partitioned into $S_1 = \{1, 1, 5\}$ and $S_2 = \{3, 4\}$, and $1 + 1 + 5 = 3 + 4 = 7$.

From an instance S in PARTITION, we need to create an instance $\Theta = \langle \mathcal{C}, \mathcal{T}, \mathcal{V}, \mathcal{S}, depot \rangle$ in $GPD^{ring}_{stn\_dgr}$. We use $S = \{1, 1, 3, 4, 5\}$ as the running example to illustrate the transformation. In Θ, the fuel-tank capacity of the only vehicle $v \in \mathcal{V}$ equals half of the sum of all integers in S (i.e., the capacity is 7 in the resulting Θ transformed from S in the example). In addition $|S|$ cities are introduced in $\mathcal{C}$. Pictorially (shown in Figure 9) assume all these cities are aligned in a row from left to right, there exist exactly two pickup-and-delivery tasks from a given city to its right neighbor. To distinguish these two tasks we call one of them a "top" task and the other a "bottom" task. In addition, there are two pickup-and-delivery tasks from the rightmost city (C_5 in Figure 9) to the leftmost city (C_1 in Figure 9). No other tasks are introduced in Θ. All "top" tasks need zero fuel consumption. All the numbers in S are 1-on-1 and onto mapped to the fuel consumption of "bottom" tasks. For the sake of convenience, these numbers are mapped into an ascending order. The leftmost city is the only depot city where the vehicle starts/finishes, and the city is the only fuel station in Θ. Given S, the resulting $GPD^{ring}_{stn\_dgr}$ instance is shown in Figure 9. Transformation is complete.

Notice that, ignoring the fuel consumption constraint, vehicle v can finish an Euler Tour leaving and returning to C_1 twice. The tour consists of two trails (name them respectively the first trail and the second trail). The first trail will achieve exactly one of the (top or bottom) tasks between two cities. The remaining tasks construct the second trail. As shown in Figure 10, an Euler Tour (with depot/fuel_station C_1) consists of the first trail (in thick lines) and the second trail (in dashed lines).

For the NP-hardness part we prove that there exists a partition to S iff a vehicle starting from C_1 can accomplish all tasks, returning eventually to C_1 with the fuel capacity constraint applied on C_1 satisfied.

($\Rightarrow$): If there exists a partition to S, we just assign the tasks accordingly (either a "top" one or a "bottom" one is selected, depending on how S is partitioned) into the first trail and the second trail. The resulting Euler Tour must satisfy the fuel consumption constraints.

($\Leftarrow$): If there exists a solution for the transformed $GPD^{ring}_{stn\_dgr}$ instance (i.e., an Euler Tour with the fuel consumption constraint satisfied), consumption of fuel for two trails amounts to exactly double the vehicle fuel capacity. If one trail consumes less than the fuel capacity, the other trail must consume more than the fuel capacity, which is infeasible. Hence fuel consumptions for the two trails are exactly the same, equaling to the vehicle fuel capacity. We know that fuel consumptions for the bottom tasks in the two trails, respectively, correspond to the sets S_1 and S_2 for S.

A pseudo-polynomial time algorithm (hence actually NP-hard in the weak sense). Let f_{all} be the overall fuel consumption for all tasks in a given $GPD^{ring}_{stn\_dgr}$ instance (assume existence of Euler Tours, otherwise no solution). Solvability of the instance is now related to both f_{all} and f_{vhl} the capacity of the vehicle. If $f_{all} > 2f_{vhl}$, the instance is infeasible – inevitably at least one of the two trails will run out of fuel. If $f_{all} < f_{vhl}$, the instance is always feasible – one full tank of fuel suffices to achieve all the tasks.

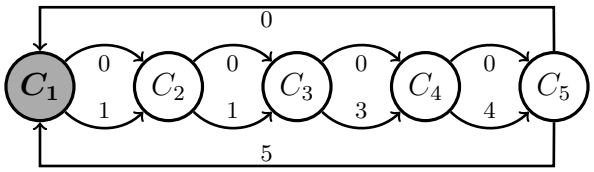

Figure 9: The $GPD^{ring}_{stn\_dgr}$ instance created from transforming the PARTITION problem $S = \{1, 1, 3, 4, 5\}$. The city C_1 is the only depot city and the only fuel station. The only vehicle in the example has a fuel capacity 7, which is determined by $(1 + 1 + 3 + 4 + 5)/2$.

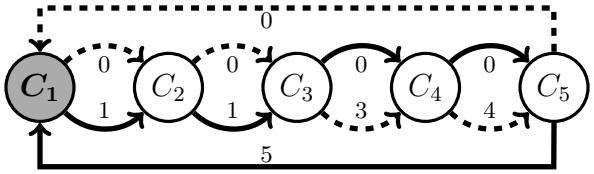

Figure 10: A solution to the $GPD^{ring}_{stn\_dgr}$ example. It consists of two trails in order: the first one is in thick lines, and the second one is in dashed lines.

Conditional solvability happens with the inequality of $f_{vhl} \leq f_{all} \leq 2f_{vhl}$. In this case, $GPD^{ring}_{stn\_dgr}$ is solvable iff there exists a trail TR_0 whose fuel consumption satisfies $(f_{all} - f_{vhl}) \leq f_{TR_0} \leq f_{vhl}$. Taking negation, with f_{all} added to each term, we obtain

$$(-f_{vhl} + f_{all}) \leq (-f_{TR_0} + f_{all}) \leq \big(-(f_{all} - f_{vhl}) + f_{all}\big),$$

which means we also have $(f_{all} - f_{vhl}) \leq f_{TR1} \leq f_{vhl}$, where $f_{all} = f_{TR0} + f_{TR1}$.

Algorithm 1 solves $GPD^{ring}_{stn\_dgr}$. Note that line 1 covers the case where "$f_{all} > 2f_{vhl}$", thus the instance is infeasible. The loop (line 5-9) deals with the case where "$f_{vhl} \leq f_{all} \leq 2f_{vhl}$". It is implemented through using dynamic programming to fill a table. For illustration purpose, we use the example of Figure 2 to explain how the corresponding table (such as Table 1) is going to be filled. In the table, columns are fuel consumption[3] ranging from 0 to f_{vhl}, and rows are the cities from C_a, and back to C_a. Cell (C_k, n) denotes the truth value (**t** or f) of the statement "there is a walk from depot/fuel_station C_a to city C_k with fuel consumption n". We fill the table in the order of top-down, row by row, left to right, column by column. It starts with $(C_a, 0) = \mathbf{t}$, the table is updated with respect to the fuel consumptions of tasks labelled in Figure 2. Since there are two tasks (t_1 and t_5), we update $(C_b, 1) = \mathbf{t}$, and $(C_b, 2) = \mathbf{t}$. It continues until we have filled all cells. The grey cells in the bottom row highlights the range of $(f_{all} - f_{vhl}) \leq f \leq f_{vhl}$. In our example,

[3]Fuel consumptions in the algorithm are with integer values. For real numbers, we only need to group them into ranges of integers closest to them, for the algorithm to work.

city\fuel	0	1	2	3	4	5	6	7	8
$\mathbf{C_a}$	**t**	f	f	f	f	f	f	f	f
C_b	f	**t**	**t**	f	f	f	f	f	f
C_c	f	f	f	**t**	**t**	f	f	f	f
C_d	f	f	f	f	f	**t**	f	**t**	**t**
$\mathbf{C_a}$	f	f	f	f	f	f	**t**	f	**t**

Table 1: A 2-dimensional matrix of *(city, fuel_consumption)* for solving the example of Figure 2. The table is filled through dynamic programming. A "t" value in any of the grey-color cells at the bottom row indicates solvability.

Algorithm 1: A Pseudo-polynomial Time Solver

Input: A $GPD^{ring}_{stn\_dgr}$ instance Θ
Output: a Boolean variable "result"

```
"result" is set to false
if (f_all < f_vhl) then
    "result" is set to true
end
for each f such that (f_all − f_vhl) ≤ f ≤ f_vhl do
    if a trail with fuel consumption f exists then
        "result" is set to true
    end
end
return "result"
```

it is $6 \leq f \leq 8$, as f_{all} for Figure 2 is 14. Any "**t**" value in a greyed-cell indicates existence of solution to the instance. For example, from the fact that $(C_a, 6) = \mathbf{t}$, we can recover a trail of $TR_0 = [t_5, t_6, t_7, t_8]$, with $f_{TR0} = 6$. Hence the other trail $TR_1 = [t_1, t_2, t_3, t_4]$, with $f_{TR1} = 8$. Both trails are feasible. TR_0 and TR_1 corresponds to a solution to the instance. The time it takes to fill the table is bounded by $O(n \times f_{vhl})$, which is a polynomial time bound in terms of the magnitude, but not the actual length, of the input f_{vhl}. Algorithm 1 only runs in pseudo-polynomial time. □

Summary and Future Work

Vehicle routing turning green reflects an important social movement towards significant reduction of overall greenhouse-gas emission. However from complexity-theoretic perspective, an addition of green components poses additional computational challenge to problems of PD and VRP, which are already intractable. In fact as demonstrated in this paper, the green component in GPD is inherently NP-hard to tackle. Several strong constraints, on bounded value of in-degree and out-degree of cities, on total number of fuel stations, and on topological structure of tasks, are explored in searching for non-trivial tractable GPD subproblems.

Complexity results obtained are pictorially summarized in Figure 11, where an arrow connects a problem to its restricted version. The dashed line separates strongly NP-complete problems from the weakly NP-complete ones. Existence of pseudo-polynomial time algorithm for $GPD^{ring}_{stn\_dgr}$ indeed offers very useful insight towards development of more effective heuristics for real-world applications. GPD problems whose task graphs follow a certain topological structure, are easier to solve: When the task graph is with the so-called ring structure, we reduce the problem complexity from strongly NP-complete to weakly NP-complete, with a pseudo-polynomial time algorithm identified. In our future work, we will explore other practical graph structures, where GPD become polytime solvable.

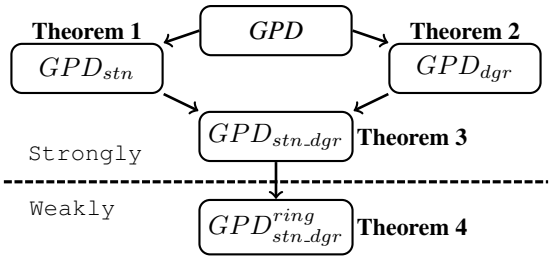

Figure 11: NP-complete GPD problems

Observe again the growth of the pseudo-polynomial time Algorithm 1 for $GPD^{ring}_{stn\_dgr}$ is exponential only with respect to the size of f_{vhl}, the fuel capacity of the vehicle. And practically it is simply reasonable to assume that f_{vhl} will not be a very large number, making the quadratic-time effort in filling the table an affordable approach. Accessible opportunities for real-world cases and applications should exist, where we can first approximate the task graph of a given GPD problem into a ring, and then solve the problem.

Analysis of complexity with an application of precedence constraints, or time windows, to tasks in a backbone PD problem (i.e., an Euler Tour) was examined recently [Tan and Huang 2019]. In this current study, we evaluate the role of the green constraints in contributing to the intractability of GPD. It should be an interesting future work investigating possible existence of phase transition [Gent et al. 1996] in the solution space of GPD, and how problem difficulty of GPD is related to both phase transition and heuristic search, using for example an analytical framework proposed in [Cohen and Beck 2017b,a].

Consider that in the computational sense, phase transition describes in the solution space, the phenomenon of a sharp transition between solvability and unsolvability of problem instances. Using a Constraints Programming model as a solver, we can work particularly on $GPD^{ring}_{stn\_dgr}$, and there are two settings particularly of relevance: 1) With a fixed vehicle fuel capacity and an increasing overall fuel consumption of all tasks, the problem is moving (likely a sharp transition) from always feasible to always infeasible. We know when $f_{all} > 2f_{vhl}$, the instance is infeasible, and if $f_{all} < f_{vhl}$, it is always feasible; and 2) With the value of f_{vhl} fixed, and increase the value of f_{all}, from f_{vhl} to $2f_{vhl}$ (i.e., from feasible to infeasible). Alternatively, we can work on GPD_{stn}, which has just one station. When the degree of the stations is greater than 1, and both task graph and f_{all} are fixed, we can try generating the tasks using normal distribution with increasing value of standard deviation.

Acknowledgments

We are thankful to all the anonymous reviewers for their insightful feedback and suggestions. We are particularly thankful to Reviewer #3 for his/her dedicated efforts looking into technical details of the proofs in the paper. This research was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and the York Research Chairs (YRC) program.

References

Ángel Felipe; Ortuño, M. T.; Righini, G.; and Tirado, G. 2014. A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review* 71: 111 – 128.

Beck, J. C.; Prosser, P.; and Selensky, E. 2003. Vehicle routing and job shop scheduling: What's the difference? In *Proc. of the 13th ICAPS*, 267–276.

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

Çağrı Koç; and Karaoglan, I. 2016. The green vehicle routing problem: A heuristic based exact solution approach. *Applied Soft Computing* 39: 154 – 164. ISSN 1568-4946.

Cohen, E.; and Beck, J. C. 2017a. Cost-Based Heuristics and Node Re-Expansions across the Phase Transition. In *Proc. of the 10th SOCS*, 11–19.

Cohen, E.; and Beck, J. C. 2017b. Problem Difficulty and the Phase Transition in Heuristic Search. In *Proc. of the 31st AAAI*, 780–786.

Coltin, B.; and Veloso, M. 2014. Scheduling for Transfers in Pickup and Delivery Problems with Very Large Neighborhood Search. In *Proc. of the 28th AAAI*, 2250–2256.

Demir, E.; Bektaş, T.; and Laporte, G. 2014. A review of recent research on green road freight transportation. *European Journal of Operational Research* 237(3): 775 – 793.

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

Erdoğan, S.; and Miller-Hooks, E. 2012. A Green Vehicle Routing Problem. *Transportation Research Part E: Logistics and Transportation Review* 48(1): 100 – 114.

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

Gent, I. P.; MacIntyre, E.; Prosser, P.; and Walsh, T. 1996. The Constrainedness of Search. In *Proc. of the 13th AAAI*, AAAI'96, 246–252.

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

Jemai, J.; Zekri, M.; and Mellouli, K. 2012. An NSGA-II Algorithm for the Green Vehicle Routing Problem. In Hao, J.-K.; and Middendorf, M., eds., *Evolutionary Computation in Combinatorial Optimization*, 37–48. Springer Berlin Heidelberg.

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

Montoya, A.; Guéret, C.; Mendoza, J. E.; and Villegas, J. G. 2016. A multi-space sampling heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies* 70: 113 – 128.

Savelsbergh, M.; and Sol, M. 1995. The General Pickup and Delivery Problem. *Transportation Science* 29(1): 17–29.

Schönfelder, R.; and Leucker, M. 2015. Abstract Routing Models and Abstractions in the Context of Vehicle Routing. In *Proc. of the 24th IJCAI*, 2639–2645.

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

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

Vokřínek, J.; Komenda, A.; and Pěchouček, M. 2010. Agents towards vehicle routing problems. In *Proc. of the 9th AAMAS*, 773–780.

Xiao, Y.; and Konak, A. 2017. A genetic algorithm with exact dynamic programming for the green vehicle routing & scheduling problem. *Journal of Cleaner Production* 167: 1450 – 1463.

Xiong, Y.; Gan, J.; An, B.; Miao, C.; and Bazzan, A. L. C. 2015. Optimal Electric Vehicle Charging Station Placement. In *Proc. of the 24th IJCAI*, 2662–2668.

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