On Computational Complexity of Pickup-and-Delivery Problems with Precedence Constraints or Time Windows

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

Abstract

Pickup-and-Delivery (PD) problems consider routing vehicles to achieve a set of tasks related to "Pickup", and to "Delivery". Meanwhile these tasks might subject to Precedence Constraints (PDPC) or Time Windows (PDTW) constraints. PD is a variant to Vehicle Routing Problems (VRP), which have been extensively studied for decades. In the recent years, PD demonstrates its closer relevance to AI. With an awareness that few work has been dedicated so far in addressing where the tractability boundary line can be drawn for PD problems, we identify in this paper a set of highly restricted PD problems and prove their NP-completeness. Many problems from a multitude of applications and industry domains are general versions of PDPC. Thus this new result of NP-hardness, of PDPC, not only clarifies the computational complexity of these problems, but also sets up a firm base for the requirement on use of approximation or heuristics, as opposed to looking for exact but intractable algorithms for solving them. We move on to perform an empirical study to locate sources of intractability in PD problems. That is, we propose a local-search formalism and algorithm for solving PDPC problems in particular. Experimental results support strongly effectiveness and efficiency of the local-search. Using the local-search as a solver for randomly generated PDPC problem instances, we obtained interesting and potentially useful insights regarding computational hardness of PDPC and PD.

1 Introduction and Background

General Pickup-and-Delivery Problems (PD) consider routing vehicles to pick up passengers/goods from different origins and delivery them to different destinations with various constraints including for example time windows or precedence constraints being satisfied [Parragh *et al.*, 2008; Battarra *et al.*, 2014; Doerner and Salazar-González, 2014; Dumas *et al.*, 1991; Baldacci *et al.*, 2011]. The problems are important variants of Vehicle Routing Problems (VRP) [Toth and Vigo, 2014], which have been well studied as one of the central combinatorial optimization problems for decades in Operations Research. PD problems are of considerable relevance to AI research, in areas including for examples, planning and scheduling [Beck *et al.*, 2003]; automatous robotics [Gini, 2017; Nunes *et al.*, 2017]; multi-agent systems [Vokřínek *et al.*, 2010; Coltin and Veloso, 2014; Maciejewski *et al.*, 2017], and constraints satisfaction [Shaw, 1998; Bent and Van Hentenryck, 2007a; Bent and Van Hentenryck, 2007b]. With the rising popularity in the recent years of ridesharing and drivesharing services, PDs are of increasing importance towards building up intelligent transportation network systems for several AI initiatives such as smart cities [Bistaffa *et al.*, 2015; Yao *et al.*, 2018].

Despite the importance and relevance of VRP/PD problems, very limited work has been dedicated specifically in evaluating computational complexities of restricted classes/variants of these problems. Considering VRP/PDs themselves are generalizations to Traveling Salesman Problems, assumptions on NP-hardness of VRP/PD restricted variants in many cases should hold, that is, usually one does not have to worry about that these problems could actually be polynomial-time solvable.

Expectedly all exact algorithms for solving VPR/PD problems with optimality (which have been developed in the past and are mostly based on traditional Operations Research techniques such as branch-and-cut, Bender's decomposition, or mixed integer linear programming) are limited in their scalability [Cordeau, 2006; Hernández-Pérez and Salazar-González, 2009]. In general, instances with sizes greater than 100 in the number of total customers/cities could not be solved optimally; A phenomenon empirically supports the validity of the assumption on intractability of these problems. For the sake of practicality, a wide variety of heuristics have been proposed and developed to this end, including the ones, for examples, based on meta-heuristics, tabu search, local search, genetic search, and their hybridizations [Gendreau *et al.*, 1994; Potvin and Bengio, 1996; Taillard *et al.*, 1997; Bräysy and Gendreau, 2005; Laporte *et al.*, 2006] It indeed works fairly well that, when all exact algorithms for a given a VRP/PD problem are not scalable, heuristic solutions are to be sought alternatively. Nevertheless, in an order to justify applications of any heuristics, we owe a "yes" answer to the basic question associated with the problem: *Is the problem really NP-hard?*

Complexities of many classical AI problems (such as planning, Bayesian networks and theirs probabilistic inferences) have been extensively studied, with multiple dividing lines on tractability delineated successfully [Bylander, 1994; Dean and Boddy, 1988; Cooper, 1990; Blum and Rivest, 1988]. Availability of a set of boundary intractable cases enables convenient generalizations of complexity results. For example, given a new problem A, if we know A is a general version problem to NP-hard problem B, we know right away that A is also NP-hard. Because of this derived fact of NP-hardness of A, it is justified developing heuristics for A.

We believe complexities of PDs deserve systematic studies in their own rights. In this paper, we define PDPC/PDTW (vehicle routing Pickup-and-Delivery problems with Precedence Constraints, or Time Windows). PDPC (PDTW) consists of a set of tasks which can be represented as a connected graph $\mathcal{G}$, and a set of Precedence Constraints $\mathcal{P}$ (a set of Time Windows $\mathcal{W}$) on these tasks. Starting and finishing at the depot, a vehicle with unit capacity in PDPC (PDTW) needs to figure out a permutation of tasks which satisfies both $\mathcal{G}$ and $\mathcal{P}$ (or $\mathcal{W}$). Key and critical complexity-theoretic results obtained in this research are: We show both PDPC and PDTW are NP-complete and they are boundary cases, meaning a relaxation of either $\mathcal{G}$ or $\mathcal{P}/\mathcal{W}$ leads to tractability of the problem (see Figure 8 in the conclusion section for further reference).

We move on to investigate the source of intractability in PDPCs. Specifically we first propose a local search model, whose effectiveness are empirically verified. Using the model as a solver for sets of randomly generalized PDPC instances, we are able to allocate phase transition from solvability to unsolvability, which in turn enables us to pinpoint sources of computational intractability in both the topology and size of the cities, and in the size of precedence constraints to PDPC problems. We believe the formalism can act as a base for developing a full-fledged practical PDPC solver, and the results we obtained on computational hardness and on the sources contributing to this hardness will shed light to a better understanding of PDPC/PD, both in theory, and in development of practical PDPC/PD solvers.

The remainder of the paper is organized as follows. Section 2 presents definitions and complexity results. Possible source on intractability is investigated and analyzed empirically in Section 3. A local search for PDPC is also proposed in the same section. The paper is concluded in Section 4.

2 NP-Completeness of PDPC and PDTW

In this section, we first define PDPC/PDTW (vehicle routing Pickup-and-Delivery problems with Precedence Constraints, or Time Windows) and their various variants in Section 2.1. NP-Completeness theorems are provided next in Section 2.2.

2.1 Definitions

Definitions in this section are grouped into two categories: on Precedence Constraints, and on Time Windows[1].

[1]For our own research purpose, problems in this section are all defined around the computational tractability boundaries. Nevertheless, general versions of PDs can be easily found in the literature, for example, [Toth and Vigo, 2014], [Savelsbergh and Sol, 1995], and [Parragh *et al.*, 2008].

Definition 1 (PDPC) *A vehicle routing Pickup-and-Delivery problem with Precedence Constraints is defined with respect to a 2-tuple $\Theta = \langle \mathcal{G}, \mathcal{P} \rangle$ where*

- *$\mathcal{G} = (C, T, depot)$ is a directed graph and specifically, C is the set of vertices (corresponding to cities), T is the set of edges (corresponding to tasks). For example, if we have $(c_i, c_j) \in T$, a one-unit entity (passenger, package, commodity, etc.) needs to be picked up from c_i and delivered to c_j, and depot $\in C$, which is the city where the vehicle starts/finishes;*
- *Set $\mathcal{P}$ consists of precedence constraints applied on T. That is, for example, if we have $t_i, t_j \in T$ and $[t_i \prec t_j] \in \mathcal{P}$, it is required that t_i should occur before t_j.*

Given a unit-capacity vehicle, does there exist a single path from and finishing at the depot such that all tasks in T are executed, with all precedence constraints in $\mathcal{P}$ satisfied?

Definition 2 (PDPC-ST) *PDPC-ST is a variant to PDPC, with the depot city in PDPC replaced by c_s, the starting city of the vehicle, and c_t, the finishing city of the vehicle. Given a unit-capacity vehicle, does there exist a single path from c_s to c_t such that all tasks in T are executed, with all precedence constraints in $\mathcal{P}$ satisfied?*

Definition 3 (PDPC-M-ST) *PDPC-M-ST is a variant to PDPC. In the problem, there are m vehicles with m starting cities and m finishing cities. Given m unit-capacity vehicles, do there exist m paths for each of the m vehicle (say i^{th}) to start at c_s^i and to finish at c_t^i? In the end all tasks in T are executed, with all precedence constraints in $\mathcal{P}$ satisfied.*

Definition 4 (PDTW) *A vehicle routing Pickup-and-Delivery problem with Time Windows is defined with respect to a 2-tuple $\langle \mathcal{G}, \mathcal{W} \rangle$ where*

- *$\mathcal{G} = (C, T, depot)$ is defined the same as the $\mathcal{G}$ in PDPC.*
- *In addition $\mathcal{W}$ is a set of time-window constraints applied on T. That is, for example, if we have $t \in T$ and $[t, \tau_i, \tau_j] \in \mathcal{W}$, it is required that task t should occur between the time τ_i and τ_j.*

Definition 5 (PDTW-ST) *Same problem as PDPC-ST, but $\mathcal{P}$ in PDPC-ST is replaced by $\mathcal{W}$ in PDTW-ST.*

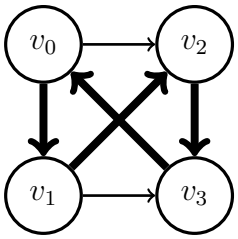
Definition 6 (PDTW-M-ST) *Same problem as PDPC-M-ST, but $\mathcal{P}$ in PDPC-M-ST is replaced by $\mathcal{W}$ in PDTW-M-ST.*

2.2 Complexity Results

Several NP-completeness results are presented in this section. For brevity, proof for Theorem 1 only is elaborated.

Theorem 1 *PDPC is NP-complete.*

Proof: NP-hardness of PDPC is proved by a polytime transformation from the NP-complete Directed Hamiltonian Circuit Problem (DHC) to PDPC. Figure 1 is an example DHC. From an instance digraph G, we need to create Θ, an instance of PDPC. We use the example in Figure 1 as the running example to help explain the transformation.



Note 1): A DHC $\langle v_0, v_1, v_2, v_3, v_0\rangle$ is highlighted with thick lines.

Note 2): DHC is listed as [GT38] in [Garey and Johnson, 1979].

Figure 1: A Directed Hamiltonian Circuit in $G = (V, E)$

For each vertex $v_i \in V$ of G, we create two cities L_i and R_i in the set C of $\mathcal{G}$. Each vertex v thus corresponds to exactly one dotted rectangle in Figure 2. (Rectangle v_0 further includes a depot node, to be explained soon.) Each rectangle boxes two cities, one left and one right. Inside each box, there is a task leaving the left city and entering the right city. If there is an edge leaving v_i in G, there is a corresponding task that leaving the city R_i in $\mathcal{G}$. Similarly, if there is an edge entering v_i in G, there is a corresponding task entering the city L_i in $\mathcal{G}$.

Each edge $e \in E$ of G corresponds to a task t in T_E, and $T_E \subset T$. We need to add additional in-cites and out-cities wherever the in-degree and the out-degree of a city in $\mathcal{G}$ are not equal. In the example, these cities are R_0, R_1, L_2 and L_3. Accordingly, we add city I_0^0 which enters R_0, I_1^0 entering R_1, O_2^0 which leaves L_2, and O_3^0 leaving L_3.

We continue with the step of **creating the depot**: Without loss of generality, we pickup rectangle v_0. We add the depot between L_0 and R_0. As shown in Figure 2, instead of entering city R_0 directly, city L_0 now enters the depot, and then the depot enters R_0. The set T_V consists of all tasks linking left cities to right cities in the rectangles. For rectangle v_0 however, there are two tasks (L_0, depot) and (depot, R_0). In Figure 2 for example, T_V consists of five tasks $\{t_1, t_3, t_5, t_7, t_{19}\}$. T_V is also a subset of T; To make sure the existence of an Euler Tour in $\mathcal{G}$, we need further a step of **balancing the degrees of the in-cities and the out-cities**: We first pick up the left city in the same rectangle of the depot city. In the case of Figure 2, it is L_0. L_0 will enter all the in-cites, and all the out-cites will enter L_0.

Figure 2 corresponds to $\mathcal{G} = (C, T, depot)$, which is created from the input graph $G = (V, E)$. The set C is a union of left cities, right cities, the depot, in-cities, and out-cities. Define T_B as the set of all tasks that need to enter/leave the in/out cities (Subscript "B" in T_B refers to **B**alancing degrees for cities in $\mathcal{G}$). The set T is thus a union of T_V, T_E and T_B, i.e., $T \equiv T_V \cup T_E \cup T_B$. Size of T in Figure 2 is 19, referring to all tasks to be accomplished by the only vehicle. All tasks in Figure 2 are labeled.

We now specify the precedence constraint $\mathcal{P}$. Let the only task leaving depot be t_1 and the only task entering it be $t_{|T|}$ (in Figure 2, $t_{|T|}$ is t_{19}). $T_V^{reduced} \equiv T_v - t_1 - t_{|T|}$, consisting of all left-city-to-right-city tasks. $\mathcal{P}$ asks that t_1 precedes, and $t_{|T|}$ is preceded by $T_V^{reduced}$. Further, it is required that $T_V^{reduced}$ precedes all tasks in T_B, and T_B precedes $t_{|T|}$. Formally $\mathcal{P}$ is defined as[2]

$$t_1 \prec \{\text{all tasks in } T_V^{reduced}\} \prec \{\text{all tasks in } T_B\} \prec t_{|T|}.$$

In the example of Figure 2, we have $\mathcal{P}$ in the form of

$$t_1 \prec \{t_3, t_5, t_7\} \prec \{t_9, t_{10}, t_{12}, t_{13}, t_{14}, t_{15}, t_{17}, t_{18}\} \prec t_{19}.$$

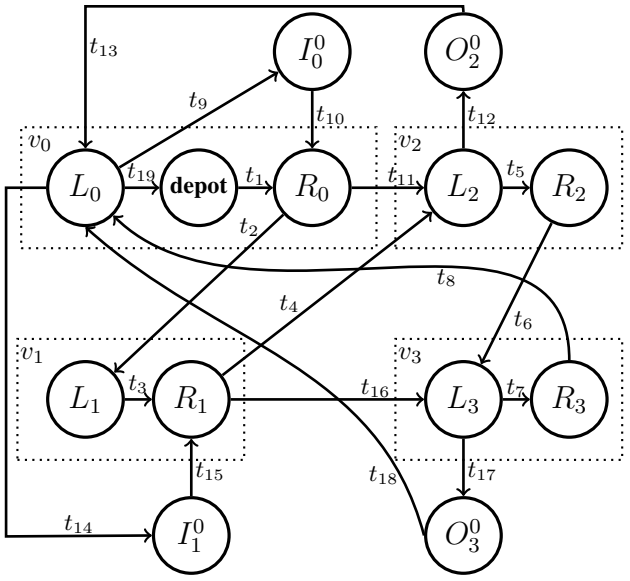


Figure 2: $\mathcal{G}$ of $\Theta(G)$ for proving Theorem 1

Transformation is complete. PDPC is in NP, as it is easy to verify if a given task sequence completes all tasks with precedence constraints being followed. For the NP-hardness part we prove that there exists a DHC in a graph G iff a vehicle starting from the depot can accomplish all tasks in $\mathcal{G}$ and returns to the depot in the end, with all precedence constraints in $\mathcal{P}$ satisfied.

($\Rightarrow$): If there exists a DHC $E' \subseteq E$ where $G = (V, E)$, E' corresponds to a sequence of tasks T_{dhc} starting from the depot to city L_0. The vehicle is one task ($(L_0, \text{depot}) = t_{|T|}$) away from the depot. Task $t_{|T|}$ can not be achieved right away because precedence constraints require that all tasks in T_B need to be achieved before $t_{|T|}$. Note that in general some, but not all, tasks in T_E are included in T_{dhc}. In the example, T_{dhc} is a sequence of $[t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8]$. Tasks $\{t_2, t_4, t_6, t_8\} \subset T_E$ are in T_{dhc}. Tasks $\{t_{11}, t_{16}\} \subset T_E$ however are not included in T_{dhc}. But the vehicle at L_0 can always achieve a task to enter an in-city and then enter a rectangle to achieve one of the so-far unachieved tasks in T_E before it arrives at the corresponding out-city and returns back to L_0. In the example, we achieve task t_{11} with a sequence of $[t_9, t_{10}, t_{11}, t_{12}, t_{13}]$, and we achieve task t_{16} with a sequence of $[t_{14}, t_{15}, t_{16}, t_{17}, t_{18}]$. Finally the vehicle achieves task t_{19}. Note that no precedence constraints are violated with this sequence. Since we have a DHC

[2] Note that the two tasks, t_1 and $t_{|T|}$, do not have to be included in $\mathcal{P}$ for the proof to work. As a result, the hight of $\mathcal{P}$ (i.e., the size of the longest chains in $\mathcal{P}$) is bounded by 2.

$\langle v_0, v_1, v_2, v_3, v_0 \rangle$ in G, we end up obtaining a sequence of tasks $[t_1, t_2, t_3, \ldots, t_{17}, t_{18}, t_{19}]$ in $\Theta(G)$.

($\Leftarrow$): Suppose there exists a sequence of tasks T in $\Theta(G)$ starting and finishing both at the depot. The first task t_1 in T must be (depot, R_0) and the last task $t_{|T|}$ must be (L_0, depot). All tasks in T_V other than $t_{|T|}$ must be achieved before any task in T_B, because T follows the precedence constraints $\mathcal{P}$. Hence in T, when the vehicle first time arrives at L_0, the path it has explored to the point must be a DHC in G. □

Theorem 2 *PDPC-ST is NP-complete.*

Proof Sketch: It is easier to prove NP-hardness of PDPC-ST directly from PDPC. The depot in an instance of PDPC is replaced in transformation by the starting point depot_s and the finishing point depot_t in the resulting instance of PDPD-ST. All tasks that initially leave the depot now leave depot_s. All tasks that initially arrive at the depot now arrive at depot_t. There exists a completion of tasks from the depot to the depot iff there exists a completion from depot_s to depot_t. □

We can obtain further the following (the proof is skipped):

Theorem 3 *PDPC-M-ST is NP-complete.*

The following three complexity results are related to time windows.

Theorem 4 *PDTW is NP-complete.*

Proof Sketch: Transformation in proving NP-completeness of PDPC can be repeated here basically. Precedence constraints set $\mathcal{P}$ defined as

$$t_1 \prec \{\text{all tasks in } T_V^{reduced}\} \prec \{\text{all tasks in } T_B\} \prec t_{|T|}$$

in PDPC however need to be replaced by a $\mathcal{W}$. In this $\mathcal{W}$, we only need to introduce three time points τ_1, τ_2, and τ_3 such that $\tau_1 < \tau_2 < \tau_3$, and we require that t_1 occurs before τ_1; All tasks in $T_V^{reduced}$ occur between τ_1 and τ_2; All tasks in T_B occur between τ_2 and τ_3; and $t_{|T|}$ occurs after τ_3. There exists a DHC in a graph G iff a vehicle starting from the depot accomplishes all tasks in $\mathcal{G}$ of PDTW and returns to the depot in the end, with $\mathcal{W}$ satisfied. □

We can obtain further the following (proofs skipped):

Theorem 5 *PDTW-ST is NP-complete.*

Theorem 6 *PDTW-M-ST is NP-complete.*

3 Analysis on Computational Hardness

In this section, we present the local search approach for solving PDPC. First, we introduce the algorithm in Section 3.1. We then perform empirical analysis to test the effectiveness of this new local search approach in Section 3.2. Finally in Section 3.3, local search is used as a solver to investigate sources contributing to the hardness of PDPC problems.

3.1 A Local Search for PDPC

We use Figure 3 as a PDPC example instance to help explain concepts. The example has four cities from C_a to C_d in C and the depot is city C_a. There are eight tasks in T from t_1 up to t_8. The precedence constraints $\mathcal{P} \equiv \{t_2 \prec t_3; t_2 \prec t_6; t_4 \prec t_8\}$. Obviously permutation $[t_1, t_2, \ldots, t_7, t_8]$ serves as a solution to the problem (also shown in Figure 3).

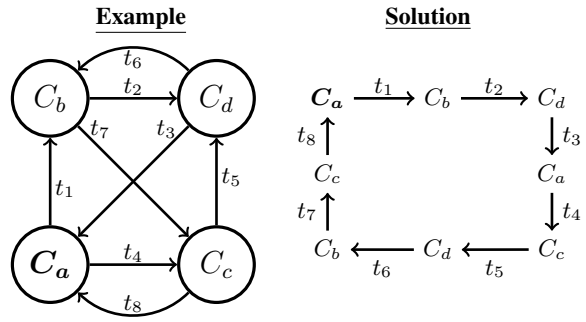


Figure 3: An example PDPC and its solution

Algorithm 1: A Local Search for PDPC

Input: $\langle \mathcal{G}, \mathcal{P} \rangle$, an instance of PDPC, where $\mathcal{G} = (C, T, depot)$.
Output: T_{output}, a permutation of all tasks in T of $\mathcal{G}$; and the cost c associated with T_{output}.

1 Initialization: Set cost c to its maximal possible value; Set threshold value for maximal rounds $maxRound$; And create a random permutation T_{output};
2 **while** *($c > 0$ or maxRound not reached)* **do**
3 Check each task $t \in T$ for every alternative position p in the current permutation T_{output}; If there are positions where c value can be reduced, update T_{output} by moving t to the position that yields greatest c value reduction; Update c accordingly;
4 **end**

A high level algorithmic description of using local search for solutions to PDPC is provided in Algorithm 1. Details more related to implementation are skipped here in the interest of brevity for a focus on key ideas. Necessary explanation to the algorithm however is presented as follows.

Given a PDPC instance, the algorithm starts from a random permutation of tasks T_{output}, which is associated with a cost c. Up to *maxRound* rounds of local search, for a repeatedly revised task sequence T_{output} with gradually reduced c value, are to be performed. Of course after each iteration, if $c = 0$ is achieved, it means that a solution to the problem – an Euler Tour starting from and finishing at the depot with all precedence constraints satisfied – is found. As a result, the algorithm will break the loop and return T_{output}. For each round, each task t is considered once, for its optimal position p with mostly reduced c value. After the round, task t is moved to position p to update both T_{output} and c, which are the inputs for the next loop, if c remains to be greater than 0.

Note that size of the local-search neighbourhood is bounded by the total number of tasks. As indicated in Algorithm 1, the task in investigation tries all candidate positions, if a better one is found, the task is removed from its current position and inserted into the new position.

We now discuss a few important issues must be taken into consideration for Algorithm 1 to actually work. First, how

Task:	t_3	t_5	t_6	t_7	t_2	t_1	t_4	t_8
Position:	1	2	3	4	5	6	7	8

Figure 4: A random permutation of eight tasks

a permutation of tasks is represented? We can simply use a conceptually circular array consisting of $|T|$ elements/tasks, with an awareness that if the sequence is an Euler Tour, any two neighbouring tasks should be connected by a city, and the first and the last task in the array should be connected too, by the depot city. If a permutation of tasks in the array is a Euler Tour to $\mathcal{G}$, and a linear extension to $\mathcal{P}$, the permutation is a solution to the PDPC problem $\Theta = \langle \mathcal{G}, \mathcal{P} \rangle$, subject further to the fact that the first task starts from the depot. For example, the permutation of tasks in Figure 5 offers an alternative solution to the PDPC problem in Figure 3.

But how the cost of a permutation is decided? In our model, the cost is a weighted summation over three parts

$$c = w_1 c_1 + w_2 c_2 + w_3 c_3,$$

where $c_1 = 1$, if the task on position one is not from the depot, $c_1 = 0$ otherwise; c_2 is the total number of non-neighboring tasks in the permutation; And c_3 is the total number of precedence constraints in $\mathcal{P}$ violated. Weights are set to $w_1 = w_2 = w_3 = 1$. Figure 4 is an example random permutation. It's cost is 8 = 1 + 5 + 2. That is, task t_3 is not from the depot, and there are five violations of neighborhood requirements as specified in $\mathcal{G}$ of Figure 3. Regarding precedence constraints, only $(t_4 \prec t_8) \in \mathcal{P}$ holds. To reduce the cost, we need to perform local search on better position for the task being evaluated. If one such position is found, we just remove the task from its current position, and insert it into the new position. Other tasks get affected will only need to move either all left, or all right. Circular movement of tasks are not necessary, and position 1 will be more likely holding a task that is actually from the depot, as local search converges to a solution when they exist. It can be verified that in Figure 4, if we move task t_1 to position 1, tasks t_3, t_5, t_6, t_7, and t_2 need to move one position to their right respectively, and the total cost is reduced from 8 to 6, We then move task t_2 to position 2, the total cost is further reduced from 6 to 2. Task t_3 to position 6, and task t_5 to position 5, resulting in the permutation in Figure 5. Finally, when there are equal choices of tasks movements, we choose a random one. We do not adopt any other trap escaping strategy except for random starts if the algorithm can not get out of a local minimum after a threshold number of non-improvement rounds.

3.2 Empirical Evaluation on Local Search

In this section, we test the effectiveness of the proposed local search using six sets of randomly generated data (corresponding respectively to the six subfigures in Figure 6). Each set is characterized by two measures, $|C|$ the total number of cities in an instance, and $|T|$ the total number of tasks.

Task:	t_1	t_2	t_6	t_7	t_5	t_3	t_4	t_8
Position:	1	2	3	4	5	6	7	8

Figure 5: Another solution to the example PDPC problem

Let $k = |T|$, we know that the range of total possible number of precedence constraints are from 0 to maximal $k(k-1)/2$. For each set, we consider five different cases where a range of 0.0%, 10%, 20%, 40%, 80% of maximal total possible precedence constraints are considered. Hence we have five curves in different colors for these five cases in each one of the sub-figures in Figure 6. For each case we create 500 random examples[3] using 500 different random seeds. Each example allows up to 1000 rounds to solve. After n rounds (where n equals the total number of tasks) no-improvement encountered, local search creates another random Euler Tour, and continue. Observations from these experiments are as follow.

Effectiveness. The local search formalism proposed here, and the cost function designed as the sum of violated number of connectedness and precedence of tasks, can actually be used as heuristics to guide the search, leading to finding of a solution with a reasonable searching efforts. With exception to the red cases (the ones having 10% of precedence constraints), almost all problem instances can be solved in less than 300 rounds. Note that we do not adopt any strategy for escaping local minima, because current research focus on the work has been dedicated to carrying out a qualitative evaluation on the effectiveness and efficiency of the local search, instead of developing an off-the-shelf PDPC solver).

Presence of precedence. When comparing the 5 curves in any sub-figure (i.e., comparing different extents of precedence constraints on task, it is obvious that all green problems can be solved in ten rounds for all six sets. Since they are not constrained on precedence, local search actually deals with Euler Tour finding, for which we know that polynomial algorithms exist. When the problems are constrained to 10% precedence (red curves), local search becomes greatly slower in convergence to all problems, i.e., approaching the ceiling. When the ratios are further increased to 20%, and 40% (blue problems and black problems, respectively), the problems become easier and easier to solve, indicated by the fact that the curves are raised higher earlier, meaning it takes fewer rounds to solve. For almost all the magenta problems, they can be solved within 1-2 rounds. This fast convergence phenomenon is explainable: Considering that magenta problems are highly constrained on precedence, a linear extension is almost there. Given that we have the guarantee on solvability of these problems, it means that the linear extension, whihc is shaped almost completely by these 80% precedence constraints, is actually the solution to the PDPC problems.

Time efficiency. In Figure 6, from left to right, from the first row, to the second row, the problem size is getting larger, with the total rounds kept the same to 1000. For the red problems, this becomes somewhat problematic, as less and less of them can actually be solved while we know beforehand that solutions to the problems exist. This is supposed to be happening when we are using a poly-time complete algorithm to tackle NP-complete problems (assuming P!=NP). Note that

[3] Examples are with guarantee on existence of solutions.

our local search is a complete solver to PDPC in the sense that through random restarts a solution to a given problem instance can always be found by the approach as long as it exists. How long it takes or how many random starts are needed is a separate concern.

3.3 Sources Contributing to PDPC Hardness

From Figure 6, it appears that the real computational hardness for PDPCs occurs more likely around those 10% regions. Accordingly we perform another theme of experiments, which are dedicated to investigating sources that contribute to PDPC hardness. For these experiments, we use the local search model as a complete solver to PDPC problems, which means, if a PDPC instance is solvable, the procedure will find it within the given maximal round. This assumption is reasonable, should we feed local search only problems comparatively smaller in size (e.g., 1000 rounds maximal is apparently enough for those 30 tasks cases (b) and (d) sub-figures in Figure 6); We use two different random permutations to generate Euler Tour and precedence constraints for creating a random PDPC instance[4].

We consider three different experimental groups, where the total number of cities equals to 12, 18, and 20 respectively. For each group, we further consider four cities, which correspond to four different-color curves in each one of the three sub-figures in Figure 7. Again when the size of tasks $|T| = k$ is given, maximal possible precedence constraints can be calculated using $k(k-1)/2$. For $k = 12, 18, 20$, the values are 66, 153, and 190, respectively, corresponding to horizontal axis in Figure 7. Given k, for each precedence constraint value $p \in [0, k(k-1)/2]$, we create 100 random PDPC problem instances for each one of the four different cities. Results obtained from these experiments are pictorially illustrated in Figure 7. Below is our main observations on PDPC hardness.

When PDPC problem instances are very lightly constrained by precedences (when the number of constraints is less than 10 or so), almost all PDPC problems are solvable. In case they are heavily constrained (rule of thumbs, when the number of constraints is greater than 40% or 60%), all PDPC problems are unsolvable; Putting this argument to the extreme: When there is no precedence constraints, PDPC is reduced to finding an Euler Tour, and all the problem instances are solvable, and when there is full precedence constraints, order of tasks are constrained into a specifical linearization, which, almost impossible, coincides with the Euler tour permutation of tasks.

Although transitions between solvability and solvability phases in Figure 7 are not rapid, difficult problem instances however do occur around transition regions, where about 50% of the problem instances are solvable. For example, the most computationally challenging case for sub-figure-a in Figure 6 (10 cities and 20 tasks) is the one with 10% constraints. At the same time, the black curve in sub-figure-c of Figure 7 (corresponding to 10 cities and 20 tasks too) makes phase transition on solvability between 1 to 30 precedence constraints, and $19 = 190 \times 10\%$ falls in this region.

[4]Different from instances generated in Section 3.2, we no longer have the guarantee on the solvability of these problem instances.

Drawing a vertical cut on the four curves with a given number of precedence constraint within the transition region, it is always the case from top to bottom in the color: green, red, blue and black. This necessarily implies that, accordingly to this observation, for two PDPC problems constrained by the same number of precedence constraints, the problem with more cities are more likely to be unsolvable. Hence we conclude that both size and topology of $\mathcal{G}$ affect the solvability of PDPC problems.

Similarly, drawing a horizontal cut on the four curves at the y-axis point 50 (meaning 50% of the problems are solved), the cut will cross the curves in the order of black, blue, red, and green, meaning that for a given number of tasks, for PDPC problems with more cities, fewer precedence constraints are needed to reach the computationally difficult region of PDPC problems.

4 Conclusions and Future Work

This paper contributes to a better understanding of computational properties of the vehicle routing PD problems. It shows several highly restricted variants of PD (including PDPC and PDTW) are NP-complete. To our best knowledge, these complexity results are the first ones of the kind. The results justify applications of heuristic methods for any problem which is a general version of either PDPC or PDTW. After all, it makes less sense, for approaching a problem using approximation or heuristics without knowing that the problem is indeed computationally intractable in the first place.

Complexity results are summarized in Figure 8 pictorially, where an arrow connects a problem to its restricted version. An interesting computational tractability line is delineated in the figure. With Pickup-and-delivery only (i.e., $\mathcal{P}$ or $\mathcal{W}$ is empty) PDPC/PDTW equals to the polytime-solvable problem of finding an Euler Tour in $\mathcal{G}$; With precedence constraints only (i.e., $\mathcal{G}_{PC} \in \emptyset$), PDPC equals to the poly-time problem of topological sorting on tasks. With both $\mathcal{P}$ and $\mathcal{G}_{PC}$, PDPC is NP-complete. However for PDTW, with $\mathcal{W}$ alone (i.e., $\mathcal{G}_{TW} \in \emptyset$), the problem remains NP-hard: It is easy to constrain n tasks with Allen's Interval Algebra (IA) [Allen, 1983] in terms of time windows; And it is well known Satisfiability for Allen's IA is NP-complete [Vilain and Kautz, 1986].

As noted previously in the proof, the transformation for proving Theorem 1 can be used to obtain NP-hardness of PDPC where the height of $\mathcal{P}$ is bounded by 2. Also, it is noted that the Directed Hamiltonian Circuit Problem serves as the core problem in reductions in the NP-hardness proofs for Temporal Projection Problems [Tan and Gruninger, 2009; Tan, 2012] and for the Partial-Order Plan Viability Problems [Tan and Gruninger, 2014]. Further restriction on either $\mathcal{G}$ or $\mathcal{P}$ will eventually land us into tractable zones. For example, we conjecture that, 1) when $\mathcal{G}_{PC}$ is bounded by in-degree=out-degree=2, or 2) when $\mathcal{P}$ is further restricted with topological structures such as N-free or series-parallel [Möhring, 1989], PDPC would become poly-time solvable. Complexity results on Allen's IA are comprehensive in the literature, where all maximal subalgebra for Allen's IA have been identified [Nebel and Bürckert, 1995;

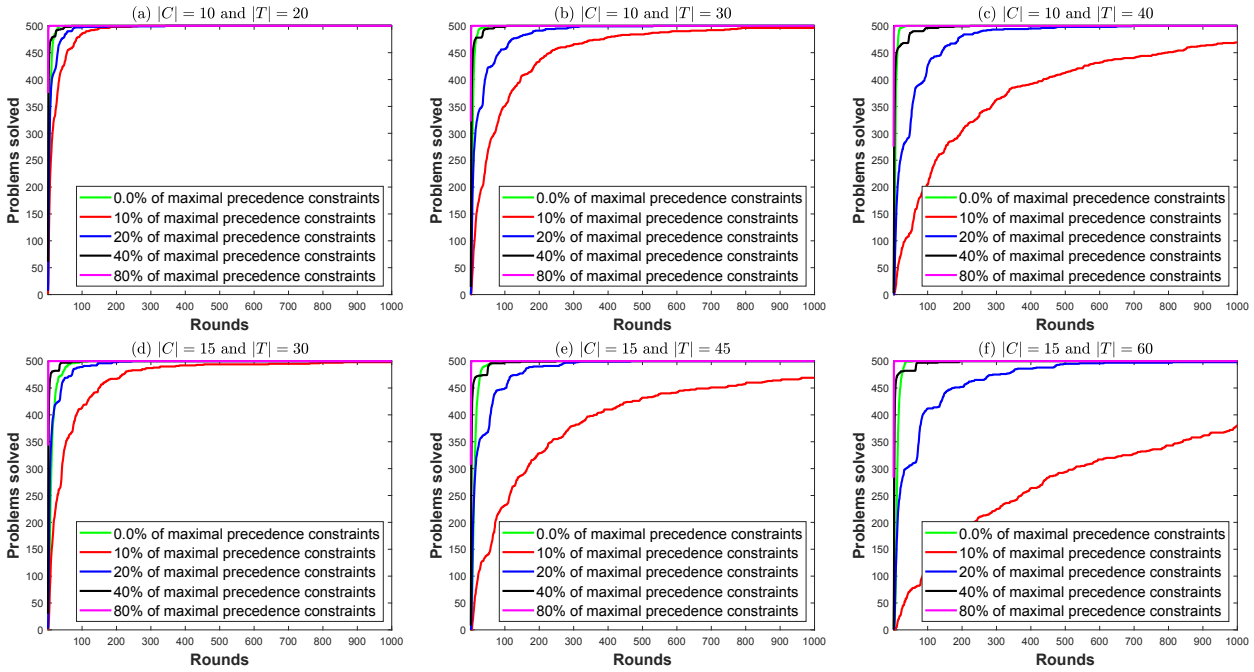


Figure 6: Experiments to test effectiveness of local search for PDPC

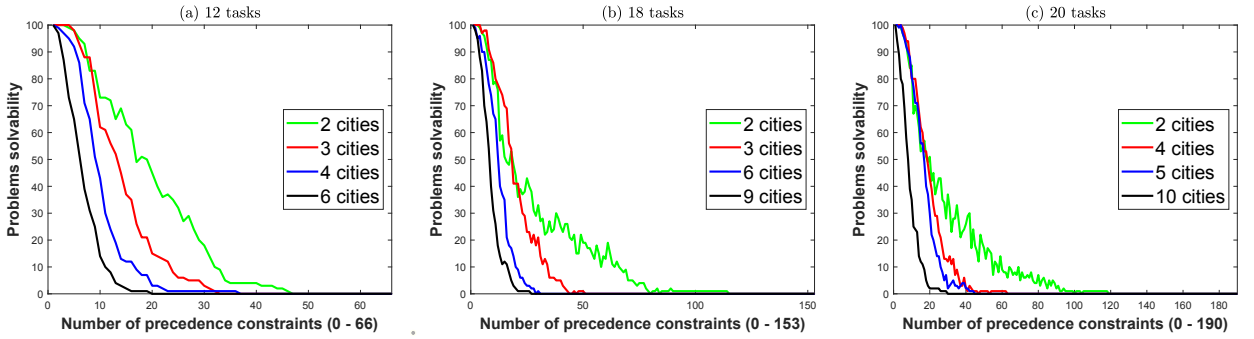


Figure 7: (Investigation on) relationship between the *number of precedence constraints* and *PDPC problem hardness*

Krokhin *et al.*, 2003]. Since $\mathcal{W}$ in the proof for Theorem 4 is obviously a tractable subalgebra of Allen's (it just includes the operator "before"), we have in addition the following research question – a restricted version of PDTW, where time windows for tasks are restricted to certain maximal subalgebra of Allen's IA, is polytime solvable.

A local search-based model has been proposed in this paper to find solutions for PDPC problems. Using it as a complete PDPC solver, we investigated the possible sources leading to the intractability of PDPC, through studying on how phase transition from solvability to unsolvability with increasing precedence constraints. Local search has long been used as a tool in tackling NP-hard problems [Johnson *et al.*, 1988]. Some motivating examples are [Savelsbergh, 1985], [Minton *et al.*, 1992] and [Vaessens *et al.*, 1996]. While we believe local-search encodings other than the one proposed in the paper will also serve our research well, the current approach is straightforward and rather simple – we take the benefit of defining the total number of costs as simply a weighted sum of violated constraints.

A recent research effort looked into the relationships between problem difficulty, phase transition, heuristic search, and cost-based heuristics [Cohen and Beck, 2017b; Cohen and Beck, 2017a]. Particularly in [Cohen and Beck, 2017b], an analytical framework for investigating the phase transition in heuristic search is proposed. Evaluating the applicability of this abstract model to the analysis of the local search for PDPC, would be an interesting line of future work.

Since our main purpose is to make fair comparisons between problem instances on total rounds necessary to find a solution, efforts have yet to be taken in turning up the model for optimized performance. For future work, standard local minimum escaping techniques such as Discrete Lagrangian Method [Shang and Wah, 1998] might be further incorpo-

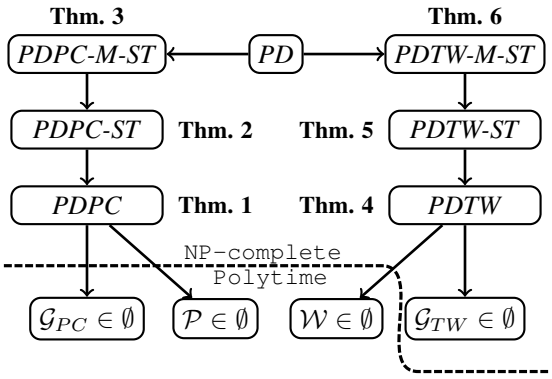


Figure 8: Computational hierarchy of the PD problems

rated into the model as necessary. We are also considering encoding PDPC problems as propositional satisfiability problems [Gomes *et al.*, 2008; Pham *et al.*, 2008]. The objective seems to be within a reasonably quick reach, given the CSP (constraint satisfaction problem) nature of these problems.

Finally, it should be interesting to make use of these new insights on intractability of PDPC to test benchmarks generally used in the literature, and to investigate their applicability on real-world domains such as Uber trips, or robots deployments in Amazon fulfillment and distribution centers.

Acknowledgments

We are grateful to all the anonymous reviewers for their insightful feedback and detailed suggestions. This research was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), an Ontario Research Fund-Research Excellence award in BRAIN Alliance, and the York Research Chairs (YRC) program. The first author is additionally supported by a Research Fellow Opportunity Grant from the Social Sciences and Humanities Research Council of Canada (SSHRC).

References

[Allen, 1983] James Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[Baldacci *et al.*, 2011] Roberto Baldacci, Enrico Bartolini, and Aristide Mingozzi. An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59(2):414–426, 2011.

[Battarra *et al.*, 2014] Maria Battarra, Jean-François Cordeau, and Manuel Iori. Chapter 6: Pickup-and-delivery problems for goods transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 161–191. SIAM, 2014.

[Beck *et al.*, 2003] J Christopher Beck, Patrick Prosser, and Evgeny Selensky. Vehicle routing and job shop scheduling: What's the difference? In *ICAPS*, pages 267–276, 2003.

[Bent and Van Hentenryck, 2007a] Russell Bent and Pascal Van Hentenryck. Randomized adaptive spatial decoupling for large-scale vehicle routing with time windows. In *AAAI*, volume 7, pages 173–178, 2007.

[Bent and Van Hentenryck, 2007b] Russell Bent and Pascal Van Hentenryck. Waiting and relocation strategies in online stochastic vehicle routing. In *IJCAI*, pages 1816–1821, 2007.

[Bistaffa *et al.*, 2015] Filippo Bistaffa, Alessandro Farinelli, and Sarvapali D. Ramchurn. Sharing rides with friends: A coalition formation algorithm for ridesharing. In *AAAI*, pages 608–614, 2015.

[Blum and Rivest, 1988] Avrim Blum and Ronald L Rivest. Training a 3-node neural network is NP-complete. In *NIPS*, pages 494–501, 1988.

[Bräysy and Gendreau, 2005] Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[Cohen and Beck, 2017a] Eldan Cohen and J. Christopher Beck. Cost-based heuristics and node re-expansions across the phase transition. In *SOCS*, pages 11–19, 2017.

[Cohen and Beck, 2017b] Eldan Cohen and J. Christopher Beck. Problem difficulty and the phase transition in heuristic search. In *AAAI*, pages 780–786, 2017.

[Coltin and Veloso, 2014] Brian Coltin and Manuela Veloso. Scheduling for transfers in pickup and delivery problems with very large neighborhood search. In *AAAI*, pages 2250–2256, 2014.

[Cooper, 1990] Gregory F Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[Cordeau, 2006] Jean-François Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54(3):573–586, 2006.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36(3):375–399, 1988.

[Doerner and Salazar-González, 2014] Karl F Doerner and Juan-José Salazar-González. Pickup-and-delivery problems for people transportation. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 193–212. SIAM, 2014.

[Dumas *et al.*, 1991] Yvan Dumas, Jacques Desrosiers, and François Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7 – 22, 1991.

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

[Gendreau *et al.*, 1994] Michel Gendreau, Alain Hertz, and Gilbert Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[Gini, 2017] Maria L Gini. Multi-robot allocation of tasks with temporal and ordering constraints. In *AAAI*, pages 4863–4869, 2017.

[Gomes *et al.*, 2008] Carla P Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. *Foundations of Artificial Intelligence*, 3:89–134, 2008.

[Hernández-Pérez and Salazar-González, 2009] Hipólito Hernández-Pérez and Juan-José Salazar-González. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *European Journal of Operational Research*, 196(3):987–995, 2009.

[Johnson *et al.*, 1988] David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79 – 100, 1988.

[Krokhin *et al.*, 2003] Andrei Krokhin, Peter Jeavons, and Peter Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *J. ACM*, 50(5):591–640, September 2003.

[Laporte *et al.*, 2006] Gilbert Laporte, Michel Gendreau, Jean-Yves Potvin, and Frédéric Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7(4-5):285–300, 2006.

[Maciejewski *et al.*, 2017] Michal Maciejewski, Joschka Bischoff, Sebastian Hörl, and Kai Nagel. Towards a testbed for dynamic vehicle routing algorithms. In *AAMAS*, pages 69–79, 2017.

[Minton *et al.*, 1992] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58(1):161 – 205, 1992.

[Möhring, 1989] Rolf H Möhring. Computationally tractable classes of ordered sets. In *Algorithms and Order*, pages 105–193. Springer, 1989.

[Nebel and Bürckert, 1995] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *J. ACM*, 42(1):43–66, 1995.

[Nunes *et al.*, 2017] Ernesto Nunes, Marie D. Manner, Hakim Mitiche, and Maria L. Gini. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, 90:55–70, 2017.

[Parragh *et al.*, 2008] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[Pham *et al.*, 2008] Duc Nghia Pham, John Thornton, and Abdul Sattar. Modelling and solving temporal reasoning as propositional satisfiability. *Artificial Intelligence*, 172(15):1752 – 1782, 2008.

[Potvin and Bengio, 1996] Jean-Yves Potvin and Samy Bengio. The vehicle routing problem with time windows part II: Genetic search. *INFORMS Journal on Computing*, 8(2):165–172, 1996.

[Savelsbergh and Sol, 1995] Martin Savelsbergh and Marc Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[Savelsbergh, 1985] M. W. P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4(1):285–305, Dec 1985.

[Shang and Wah, 1998] Yi Shang and Benjamin W Wah. A discrete Lagrangian-based global-search method for solving satisfiability problems. *Journal of global optimization*, 12(1):61–99, 1998.

[Shaw, 1998] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *CP*, pages 417–431, 1998.

[Taillard *et al.*, 1997] Éric Taillard, Philippe Badeau, Michel Gendreau, François Guertin, and Jean-Yves Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186, 1997.

[Tan and Gruninger, 2009] Xing Tan and Michael Gruninger. Towards tractable reasoning on temporal projection problems. In *Proceedigns of the IEEE International Conference on Intelligent Computing and Intelligent Systems*, volume 1, pages 723–727, 2009.

[Tan and Gruninger, 2014] Xing Tan and Michael Gruninger. The complexity of partial-order plan viability problems. In *ICAPS*, pages 307–313, 2014.

[Tan, 2012] Xing Tan. *The Application of Ontologies to Reasoning with Process Modeling Formalisms*. PhD thesis, University of Toronto, 2012.

[Toth and Vigo, 2014] Paolo Toth and Daniele Vigo. *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2014.

[Vaessens *et al.*, 1996] R. J. M. Vaessens, E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by local search. *INFORMS Journal on Computing*, 8(3):302–317, 1996.

[Vilain and Kautz, 1986] Marc B. Vilain and Henry A. Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI*, pages 377–382, 1986.

[Vokřínek *et al.*, 2010] Jiří Vokřínek, Antonín Komenda, and Michal Pěchouček. Agents towards vehicle routing problems. In *AAMAS*, pages 773–780, 2010.

[Yao *et al.*, 2018] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI*, pages 2588–2595, 2018.