Journal of Artificial Intelligence Research 82 (2025) 2167-2188 Submitted 02/2025; published 04/2025

On the Phase Transition of the Euclidean Travelling Salesman Problem with Time Windows

Omar Rifki OMAR.RIFKI@USMBA.AC.MA
Faculty of Legal, Economic, and Social Sciences, University Sidi Mohamed Ben Abdellah, Fez, Morocco.
LISIC, University Littoral Côte d'Opale, Calais, France.

Christine Solnon CHRISTINE.SOLNON@INSA-LYON.FR
(Corresponding author)
INSA Lyon, Inria, CITI, UR3720, 69621 Villeurbanne, France.

Abstract

Algorithms are often evaluated on randomly generated instances to study scale-up properties with respect to features such as the size, for example. Also, machine learning based approaches often train models on randomly generated instances as they need large sets of training instances. In this paper, we consider the Euclidean Travelling Salesman Problem with Time Windows (TSPTW), and we study the impact of parameters used to randomly generate TSPTW instances on hardness and feasibility. We first consider the decision version of the problem, where feasibility depends on start and end times of time windows. We introduce two parameters, α and β, for controlling the tightness of the time horizon and the time windows. We show that instance hardness is related to a phase transition phenomenon: as we increase α and β, we pass from an unfeasible region (where almost all generated instances have no solution) to a feasible region (where almost all generated instances have solutions), and the hardest instances are located within the transition zone. We formally relate this transition zone with respect to α and β, thus allowing us to control hardness and feasibility when randomly generating instances. Then, we study the optimization problem, the goal of which is to find the smallest tour that satisfies all time windows. We show that the empirical hardness is still related to the phase transition: hardness increases when moving from the infeasible region to the transition zone, as in the decision problem. However, unlike the decision problem, some hard instances are also located in the feasible region where instances are very loosely constrained.

1. Introduction

The Travelling Salesman Problem (TSP) aims at finding a shortest tour that visits every vertex in a complete weighted graph. This problem occurs in many real world applications such as vehicle routing or task scheduling problems, for example. Even though the TSP is NP-hard, finding a feasible solution is straightforward as any permutation of the vertices is a feasible solution when the graph is complete. However, in many applications constraints are added on earliest and latest visit times, and finding a feasible tour that satisfies all these Time-Window (TW) constraints is NP-complete (Savelsbergh, 1985). In practice,

we observe that feasible solutions are very easily found for some instances (typically, when TWs are very wide), whereas this is much more challenging for some other instances.

Many public benchmarks for the TSP with TWs (TSPTW) contain Euclidean instances (such that vertices are associated with 2D coordinates and edge costs correspond to Euclidean distances between vertex coordinates) which are randomly generated: parameters are used to control features of the generated instances (such as the number of vertices, the distribution of vertex coordinates, or the TW tightness, for example), thus allowing one to evaluate scale-up properties of algorithms with respect to these features. This is the case, for example, of the benchmarks introduced by Langevin et al. (1993), by Dumas et al. (1995), by Gendreau et al. (1998), or by Ohlmann and Thomas (2007).

Selection approaches, such as SATzilla (Xu et al., 2007), Llama (Kotthoff, 2013), or Instance Space Analysis (ISA) (Smith-Miles & Muñoz, 2023), study the impact of instance features on the performance of a portfolio of algorithms in order to select the expected best solver for each instance to solve. ISA may also be used to gain insights into how features explain algorithm performance, and to generate new instances that are expected to perform well (or, conversely, bad) on some given algorithm. In particular, Smith-Miles and van Hemert (2011) use ISA to generate Euclidean TSP instances that are hard for one algorithm and easy for another algorithm, using an evolutionary process.

In this paper, we study the impact of instance features on feasibility and on empirical hardness of randomly generated Euclidean TSPTW instances, thus allowing us to control hardness when generating instances. Contrary to selection approaches such as ISA or Llama, we relate instance features to hardness independently from solving algorithms.

Overview and contributions In Section 2, we define the TSPTW, we describe the solvers considered in this study, and we review related work on the phase transition phenomenon which relates instance hardness to a transition from a feasible region to an infeasible region.

In Section 3, we consider the decision problem that aims at deciding whether there exists a tour satisfying all the TWs. We first introduce two parameters, α and β, for controlling the tightness of the time horizon and the TWs. We show that the hardness of the generated instances is related to a phase transition phenomenon: as we increase α and β, we pass from an unfeasible region (where almost all generated instances have no solutions) to a feasible region (where almost all generated instances have solutions), and the transition between these two regions is abrupt in the sense that a small variation in the values of α or β takes us from one region to the other. We locate this transition zone empirically, by measuring the percentage of feasible instances. We also measure run times of three different solvers: an exact solver based on an anytime extension of the A* search algorithm (Fontaine et al., 2023), and two heuristic approaches based on variable neighbourhood search (Da Silva & Urrutia, 2010) and BeamACO (López-Ibáñez et al., 2013). We show that the hardest instances are positioned within the transition zone for the three solvers. We also locate the phase transition region from a theoretical point of view, by evaluating the probability that an instance is feasible, and show that this theoretical analysis matches experimental results. We introduce a third parameter γ, which is a function of α and β, and we show that the phase transition occurs when $\gamma = 1$. Finally, we extend our study to the case where the width of each TW is randomly generated and to the case where it is necessary to wait for

a service time at each vertex. In both cases, we show how to adapt the definition of γ so that the phase transition still occurs when $\gamma = 1$.

In Section 4, we consider the optimization problem which aims at finding the shortest feasible tour. We show that the empirical hardness is still related to the phase transition: hardness increases when moving from the infeasible region to the transition zone, as in the decision problem. However, unlike the decision problem, some hard instances are also located in the feasible region, where instances are very loosely constrained. For these loosely constrained instances, we study the evolution of hardness during the solving process of anytime solvers that compute a sequence of tours of decreasing length and tighten the time horizon after each computed tour in order to search for shorter tours (or prove optimality whenever the last computed tour is optimal).

In Section 5, we conclude by summarizing our main contributions and discussing further work.

2. Context

2.1 Notations and Definition of the TSPTW

$\mathcal{P} = \{0, \ldots, n\}$ denotes a set of points such that 0 is the starting point and n the ending point (in practice, these two points often refer to a same location which corresponds to the depot). For each point $i \in \mathcal{P}$, e_i and l_i denote the earliest arrival time and the latest arrival time, respectively. We assume that the tour starts at time 0 and that $e_0 = l_0 = e_n = 0$. l_n corresponds to the time horizon. For each couple of points $i, j \in \mathcal{P}$, d_{ij} denotes the time needed to travel from i to j. We assume that all times are integer values greater than or equal to 0.

The decision problem associated with the TSPTW aims at finding a path that starts from 0 at time 0, visits each point $i \in \mathcal{P}$ within its TW $[e_i, l_i]$, and finally ends on n. The path may arrive at some point i before e_i but, in this case, it is necessary to wait until e_i before starting serving i. More formally, we introduce two decision variables for each point $i \in \mathcal{P}$: the arrival time on i, A_i, and the next point visited after i, N_i. The constraints to satisfy are:

$$A_i \in [e_i, l_i], \qquad \forall i \in \mathcal{P} \tag{1}$$

$$N_i \in \mathcal{P} \setminus \{0, i\}, \qquad \forall i \in \mathcal{P} \setminus \{n\} \tag{2}$$

$$N_i \neq N_j, \qquad \forall \{i, j\} \subseteq \mathcal{P} \setminus \{n\} \tag{3}$$

$$A_i + d_{iN_i} \leq A_{N_i}, \qquad \forall i \in \mathcal{P} \setminus \{n\} \tag{4}$$

Constraint (1) ensures that every point is visited during its TW. Constraint (2) ensures that the successor of any point different from the ending point is another point different from the starting point. Constraint (3) ensures that a same point is not visited twice. Constraint (4) ensures that we have enough time to travel from i to the next point after i before arriving on this next point (it also ensures that there is no subtour as we cannot have a cycle with increasing arrival times).

Constraint (4) is not an equality because when $A_i + d_{iN_i} < e_{N_i}$ we have to wait for the beginning of the TW of N_i before starting serving N_i. These waiting times imply that the

arrival time on n may be larger than the sum of all travel times. Hence, we may consider two different objective functions to minimize:

- the makespan A_n, that includes waiting times;
- the travel time $TT = \sum_{i \in \mathcal{P} \setminus \{n\}} d_{iN_i}$, that excludes waiting times.

In this paper, we consider the makespan objective when we solve the optimization problem (in Section 5).

Euclidean TSPTW Euclidean TSPTW instances are such that edge costs correspond to Euclidean distances. More precisely, each point $i \in \mathcal{P}$ is associated with 2D coordinates $(x_i, y_i) \in \mathbb{R}^2$, and d_{ij} is the Euclidean distance between the coordinates associated with i and j, *i.e.*,

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Many public benchmarks for the TSP and the TSPTW contain Euclidean instances (*e.g.*, those introduced by Langevin et al. (1993), by Dumas et al. (1995), by Gendreau et al. (1998), and by Ohlmann and Thomas (2007)). In our study we only consider Euclidean instances.

2.2 Solving Approaches Considered in Our Study

There exist many approaches for solving the TSPTW. Furthermore, the TSPTW is equivalent to the single-machine scheduling problem with task-dependent setup times, release dates and due dates (denoted $1|r_j\overline{d_j}|C_{max}$ in the notation of Graham et al. (1979)), for which there also exist many solving approaches. For our study, we have selected three open source solvers, denoted DP, VNS, and BeamACO, and described below. These solvers are based on very different approaches in order to investigate whether hardness is related to the kind of solving approach considered.

DP DP is an exact approach based on Dynamic Programming and introduced by Fontaine et al. (2023). It is able to find optimal solutions and prove optimality, provided that it has enough time and memory. It combines an anytime variant of A* (to explore the state space) with bounding and TW constraint propagation (to prune states) and local search (to find better solutions sooner). It has been designed for solving Time-Dependent TSPTW instances (for which the cost of an edge (i, j) depends on the starting time from i), but it may be used to solve TSPTW instances with constant costs as well and it obtains state-of-the-art results.

VNS VNS is a heuristic approach based on Variable Neighborhood Search and introduced by Da Silva and Urrutia (2010). It is a two phase heuristic approach: the first phase aims at constructing a feasible solution (by minimizing the sum, for each point i, of the positive difference between the arrival time on i and l_i), whereas the second phase aims at minimizing the cost of the path (this cost is the travel time in the original code and we have adapted it to minimize the makespan). Both phases are based on Variable Neighborhood Search.

BeamACO BeamACO is a heuristic approach based on Ant Colony Optimization and Beam search and introduced by López-Ibáñez et al. (2013). It has been initially designed for the travel time objective by López-Ibáñez and Blum (2010), and López-Ibáñez et al. (2013) show how to adapt it to the makespan objective.

Both VNS and BeamACO are open source and widely used. They are able to quickly find good solutions for most feasible instances. Some more recent meta-heuristic approaches may outperform them. However, our goal is not to compare the best performing approaches but to evaluate the impact of the parameters used to randomly generate instances on their hardness. Note that neither VNS nor BeamACO can prove inconsistency of infeasible instances.

2.3 Phase Transition

Cheeseman et al. (1991) show that randomly generated instances of NP-complete problems can usually be summarized by control parameters which separate the instance space in two regions: when varying the values of these control parameters, we move from a feasible region (where almost all generated instances have solutions) to an infeasible region (where almost all generated instances have no solutions). Empirical studies on three decision problems (Hamiltonian Circuits, k-Colouring, and SAT) show us that the transition between the two regions is abrupt and corresponds to the hardest instances. This phenomenon has been studied for various NP-complete problems such as Constraint Satisfaction (Smith & Dyer, 1996) or Subgraph Isomorphism (McCreesh et al., 2018), for example. These studies allow us to better understand empirical hardness of NP-complete problems and to control hardness of randomly generated instances (Leyton-Brown et al., 2014).

Cheeseman et al. (1991) also study the empirical hardness of the TSP, for the optimization problem that aims at finding a minimal-cost tour. In this study, edge costs are randomly generated according to a log-normal distribution such that the average cost is equal to ten and the standard deviation is controlled by a parameter σ. Edge costs are then rounded to the closest integer value. For this model, instances are easy when σ is either very small or very large, and the hardest instances occur when σ has intermediate values. However, as pointed out by Sleegers et al. (2020), this phenomenon is no longer observed when costs are not rounded to integer values. Indeed when σ is large, there are many cost values which have a value smaller than 0.5 so that they are rounded to zero. In this case, the probability that there exists a tour of total cost equal to zero quickly increases when σ increases and proving the optimality of a tour of cost zero is very easy. This is no longer the case when cost values are not rounded.

Zhang (2004) studies phase transitions for the asymmetric TSP when distances are randomly generated in $[0, R]$ (where R is actually defined by the number of digits used to encode distances) according to a uniform distribution. Phase transition is not studied with respect to feasibility, as all instances are feasible, but with respect to different features such as, for example, (i) the fraction of distinct distance values, (ii) the fraction of backbone edges (*i.e.*, edges that belong to all optimal solutions), or (iii) the gap between the assignment problem relaxation and the optimal solution: Zhang shows that when increasing R these features increase from 0% to 100% with a same transition pattern. This explains why the asymmetric TSP becomes harder as R increases.

Beardwood et al. (1959) study the asymptotic length of the shortest tour, denoted l_{opt}, in randomly generated Euclidean TSP instances. They show that when point coordinates are randomly and independently chosen in a square of width a according to a uniform distribution, we have

$$\lim_{n\to\infty} l_{opt} = a \cdot k \cdot \sqrt{n} \tag{5}$$

where k is a constant which is estimated to 0.765 by Stein (1978). Gent and Walsh (1996) study phase transition for the decision problem associated with the Euclidean TSP, *i.e.*, when the goal is to decide whether there exists a tour of length smaller than a given bound l. They show that a phase transition occurs when

$$\frac{a \cdot k \cdot \sqrt{n}}{l} = 1. \tag{6}$$

Rifki et al. (2021) extend this study to the Euclidean TSP with uniform non-overlapping TWs: in this case, the time horizon $[0, l_n]$ is partitioned into m TWs of equal width $w = \frac{l_n+1}{m}$ such that, for each $i \in [1, m]$, the i^{th} TW starts at time $(i-1)\cdot w$ and ends at time $i\cdot w$. The n points are uniformly distributed over these m TWs. As TWs are non-overlapping, we know that all points assigned to the i^{th} TW must be visited before the points assigned to the j^{th} TW whenever $j > i$. This property is exploited to give asymptotic approximations of the length, the total waiting time, and the arrival time of the shortest tour for this particular case of TSPTW. Also, the phase transition from feasible to infeasible instances is shown to occur when

$$\frac{a \cdot k \cdot \sqrt{n}}{\sqrt{w \cdot l_n}} = 1. \tag{7}$$

In Section 3, we extend this work to the more general case of TWs that may overlap, to the case where service times are added at each visited point, and to the case where the width of each TW is randomly chosen. We further extend our study to the optimisation problem in Section 4.

2.4 Experimental Setup

All experiments are performed on an *Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz* processor with *32 GB* RAM memory machine, and a SWAP memory partition of *200 GB*.

The code for generating instances and analyzing results is available at `https://gitlab.com/orifki/phase-transition-tsptw`.

3. Phase Transition for the Decision Problem

In this section, we characterize the phase transition that appears when solving the decision problem associated with the Euclidean TSPTW when TWs may overlap. We first consider the case where all TWs have the same width and where each point is left as soon as it is reached. Then, we extend our study to TWs with variable widths in Section 3.6 and to the case where a service time must be added for each visited point in Section 3.7.

3.1 Random Model ET(n, h, w) for Euclidean TSPTW Instances

For all experiments reported in this section, we have randomly generated Euclidean instances according to a model denoted ET(n, h, w). The three input parameters of this model are the number n of points to visit ($n-1$ customer points and a depot on which the tour starts and ends), the time horizon h, and the TW width $0 \leq w \leq h$. Given the values of these parameters, a Euclidean TSPTW instance is randomly generated as follows:

- For each point $i \in \mathcal{P} \setminus \{n\}$, x_i and y_i are randomly chosen in $[0, a-1]$ with $a = 50$ according to a uniform distribution. We set $(x_n, y_n) = (x_0, y_0)$ to ensure that the last visited point has the same coordinates as the first visited point, corresponding to the depot.

- For each couple of points $i, j \in \mathcal{P}$, the distance d_{ij} is set to the Euclidean distance between (x_i, y_i) and (x_j, y_j) rounded to the closest integer value.

- For each point $i \in \mathcal{P} \setminus \{0, n\}$, e_i is randomly chosen in $[0, h-w]$ according to a uniform distribution, and we set $l_i = e_i + w$ so that all TWs have a same width equal to w. We set $l_n = h$. When $w = h$, we have $e_i = 0$ and $l_i = h$ for every point $i \in \mathcal{P} \setminus \{0, n\}$ so that the problem is equivalent to a Euclidean TSP without TWs but with an upper bound h on the tour length.

3.2 First Empirical Evidence

Let us start studying the impact of n, h, and w on feasibility and on run times of DP by considering three values for h (*i.e.*, 250, 300, and 350) and two values for n (*i.e.*, 21 and 31). For each combination of h and n, we plot the evolution of feasibility and run times when increasing w from 0 to h (by steps of 0.01) in Figure 1. In all cases, we observe a similar behavior: when w is close to 0, all generated instances are infeasible whereas when w is close to h, all generated instances are feasible. The hardest instances always occur in the transition between these two regions, and this transition is rather sharp. The infeasible region increases when increasing n as there are more points to visit within a same time horizon. For example, when $h = 300$, the transition zone is shifted from $w = 116$ when $n = 21$ to $w \in [168, 198]$ when $n = 31$. Also, when h increases, the feasible region increases (as it becomes easier to find tours that fit within the time horizon), and both feasible and infeasible instances become easier.

3.3 Normalized Control Parameters α and β

In order to normalize the time horizon and the time window width with respect to the number of points to visit, we introduce two parameters α and β:

- α measures the tightness of the time horizon with respect to the number of points to visit and it is obtained by dividing h by the expected length of the optimal tour l_{opt} as defined in Eq. (5) (see Section 2.3), *i.e.*, $\alpha = \frac{h}{ak\sqrt{n}}$ where a is the square width of point coordinates (set to 50 in all our experiments) and k is the constant introduced by Beardwood et al. (1959) (set to 0.765 in all our experiments).

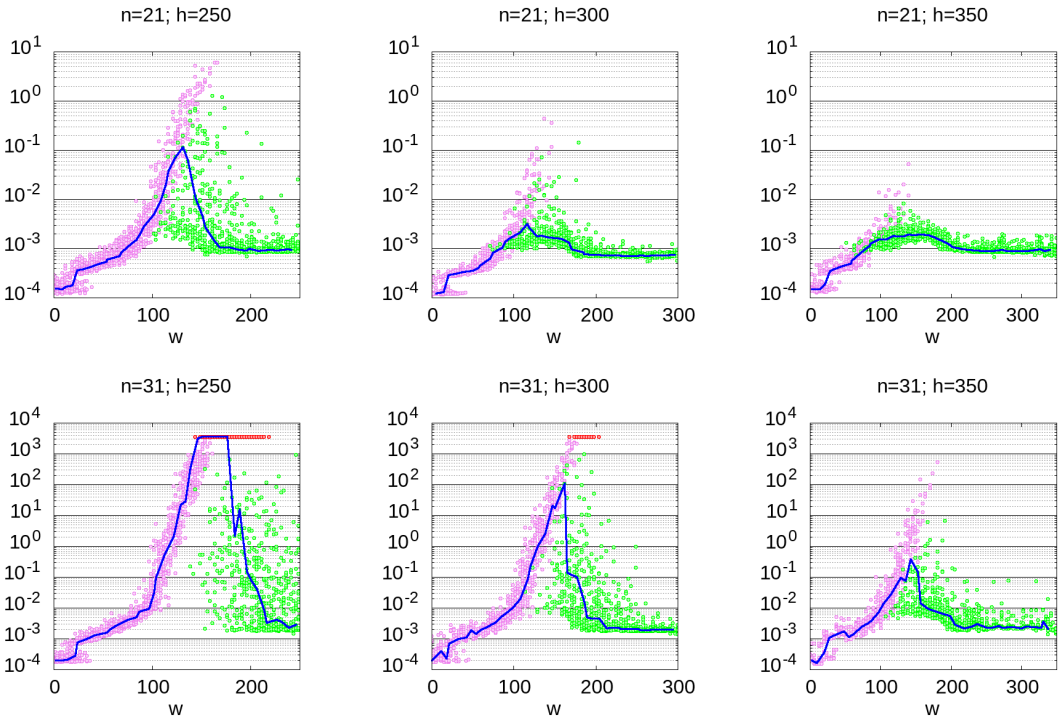

Figure 1: Runtime and feasibility of instances of ET(n, h, w) when $n \in \{21, 31\}$ (from top to bottom) and $h \in \{250, 300, 350\}$ (from left to right). For each $w \in [0, h]$, we have generated 20 instances. For each instance, we plot a point (x, y) where $x = w$ and y is the run time of DP in seconds (with a logscale). The point is pink if the instance is infeasible, green if it is feasible, and red if DP has not solved the instance within a time limit of one hour. The blue curve plots the evolution of the median run time with respect to w.

- β measures the tightness of the TWs with respect to the time horizon and it is obtained by dividing w by h, *i.e.*, $\beta = \frac{w}{h}$. When $\beta = 0$ all TWs are empty and all instances are trivially infeasible; when $\beta = 1$ all TWs are equal to the time horizon and deciding whether the instance is feasible or not is equivalent to deciding whether there exists a tour of length bounded by h or not.

By abuse of language, we denote by ET(n, α, β) the random model equivalent to ET(n, h, w) with $\alpha = \frac{h}{ak\sqrt{n}}$ and $\beta = \frac{w}{h}$.

The top left plot of Figure 2 displays the percentage of feasible instances when instances are randomly generated with ET(n, α, β) when $n = 21$ and $(\alpha, \beta) \in [1, 3] \times [0, 1]$. The infeasible region (pink points corresponding to 0% of feasible instances) is located on the bottom left corner, whereas the feasible region (green points corresponding to 100% of feasible instances) is located on the top right corner.

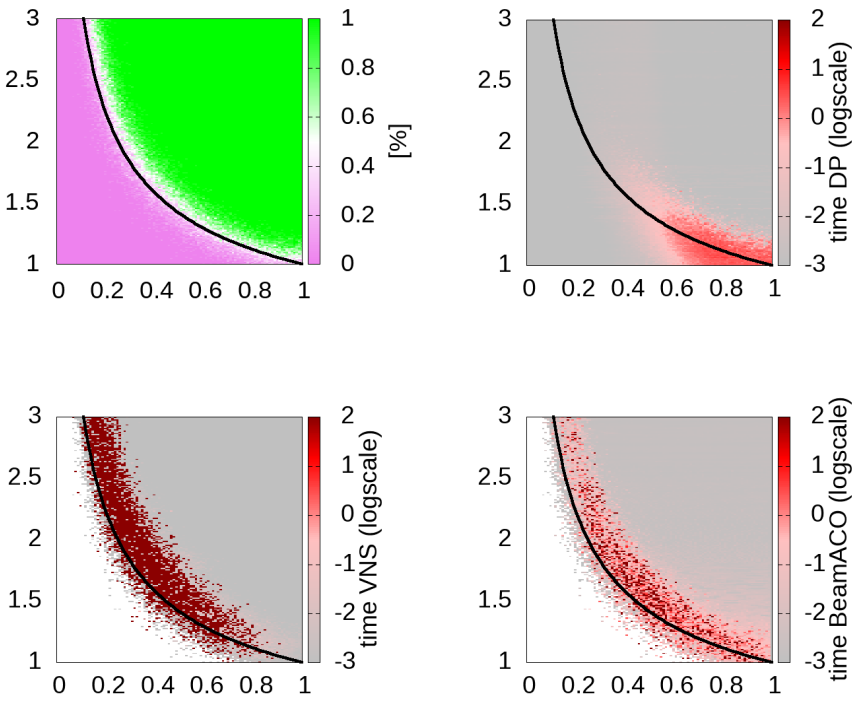

Figure 2: Evolution of feasibility and run times with respect to α (y-axis) and β (x-axis) when $n = 21$. For each $(\alpha, \beta) \in [1, 3] \times [0, 1]$ by steps of 0.01, 20 instances have been randomly generated according to $\mathrm{ET}(21, \alpha, \beta)$. We report the percentage of feasible instances (top left), and average run times in seconds (with a logscale) of DP (top right), VNS (bottom left), and BeamACO (bottom right). For VNS and BeamACO, we report average run times for feasible instances only and draw a white point whenever there is no feasible instance. The black curve is the function $\alpha = \frac{1}{\sqrt{\beta}}$.

- When $\beta = 0$, all TWs have null width and all instances are trivially infeasible, independently from α.
- When $\beta = 1$, all TWs have the same width as the time horizon, *i.e.*, the problem is equivalent to a classical TSP without TWs and the goal is to decide whether there exists a tour no longer than the time horizon h. In this case, feasibility depends on α: when $\alpha = 1$, $h = l_{opt}$ and around 50% of the generated instances are feasible; when increasing α all generated instances become feasible, which is consistent with experiments reported by Gent and Walsh (1996).
- When $0 < \beta < 1$, the width w of the TWs is non null and smaller than the time horizon. In this case, feasibility depends on α: when $\alpha = 1$, all instances are infeasible

as the expected length of the optimal tour that satisfies all TW constraints is larger than l_{opt}; when $\alpha = 3$, all instances are feasible (except for very tight TWs, when β is very close to 0) as the time horizon is three times as large as l_{opt}. The phase transition between the infeasible region and the feasible region is very sharp and corresponds to the white points in the top left plot.

We also display average run times of DP, VNS and BeamACO in Figure 2 (for VNS and BeamACO, these run times are computed on feasible instances only as they are not able to prove infeasibility). For the three considered approaches, instances that are far from the transition zone are easy to solve, and hard instances are always located in the transition zone. For DP, instances in the transition zone are not equally hard: instances become harder when β is greater than 0.4, *i.e.*, when TWs are larger than 40% of the time horizon. This comes from the fact that DP efficiently propagates tight TWs to filter the search space. For VNS and BeamACO, instances in the transition zone are much harder than those that are far from it and, even if these approaches cannot prove infeasibility, we observe that they quickly solve the few instances that are feasible while being on the bottom left of the black curve and far enough from the transition region.

3.4 Scale-up Properties with Respect to n

In Figure 3, we plot the evolution of feasibility and run times when increasing n from 21 to 31, 41, and 51. As these experiments are rather time consuming, we consider a tighter interval of values for β, *i.e.*, $\beta \in [0.1, 0.5]$. Also, we consider a coarser grain (steps of 0.05 instead of 0.01) for evolving α and β.

This figure shows us that the phase transition location does not depend on n, when considering the two normalized parameters α and β: the percentage of feasible instances with respect to α and β is very similar for all values of n.

Like when $n = 21$, the hardest instances are located in the transition zone. If DP is able to solve instances in the transition zone when n is small enough ($n \leq 31$) or when α is large enough ($\alpha \geq 2$), neither VNS nor BeamACO are able to solve instances in the transition zone within one hour of CPU time.

3.5 Locating the Phase Transition Region

As the TWs are randomly and independently chosen according to a uniform distribution, for each time $t \in [0, h - w]$, the expected number of points i such that $l_i \in [t, t + w]$ is equal to $\frac{n \cdot w}{h}$. In other words, the expected number of points that must be visited in w units of time is equal to $\frac{n \cdot w}{h}$. Therefore, the probability that a randomly generated instance is feasible is equal to 0.5 when the expected length of a shortest path that contains $\frac{n \cdot w}{h}$ points is equal to w. When using Eq. (5) to estimate the length of this path, we expect the phase transition to occur when

$$w = \frac{a \cdot k \cdot \sqrt{n \cdot w}}{\sqrt{h}}. \tag{8}$$

This is equivalent to Eq. (7), which has been introduced for the case of non-overlapping TWs in (Rifki et al., 2021): the fact that TWs are overlapping or not does not change the position of the phase transition.

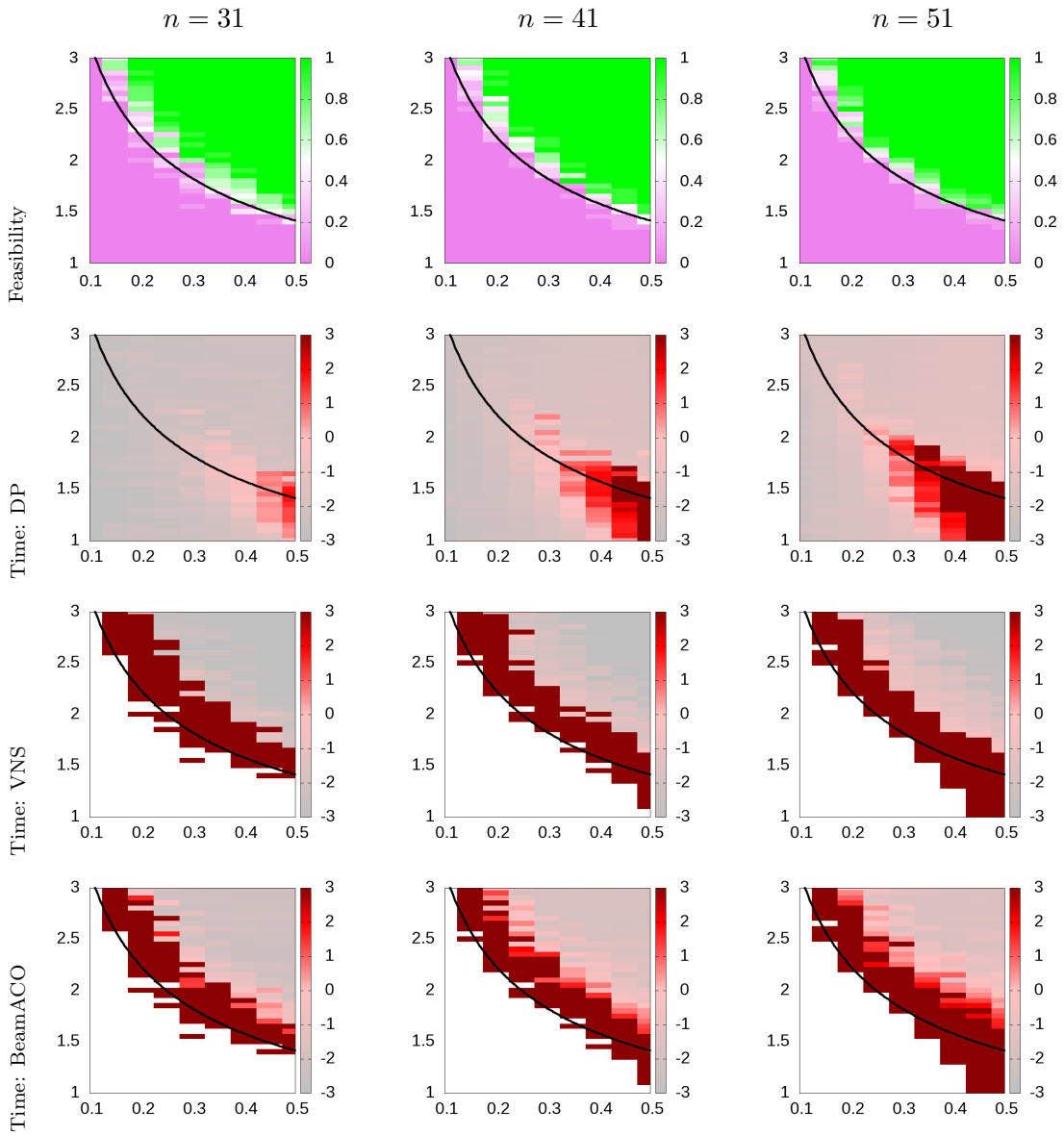

Figure 3: Evolution of feasibility and runtime with respect to α (y-axis) and β (x-axis) when $n = 31$ (left), 41 (middle), and 51 (right). For each $(\alpha, \beta) \in [1, 3] \times [0.1, 0.5]$, 10 instances have been randomly generated with ET(n, α, β). Top row: Percentage of feasible instances. Row 2 (resp. 3 and 4): Average run times in seconds (with a logscale) of DP (resp. VNS and BeamACO). For VNS and BeamACO, we report times on feasible instances only and draw a white point whenever there is no feasible instance. If time exceeds 1000s for one instance, the average time is set to 1000s. The black curve is the function $\alpha = \frac{1}{\sqrt{\beta}}$.

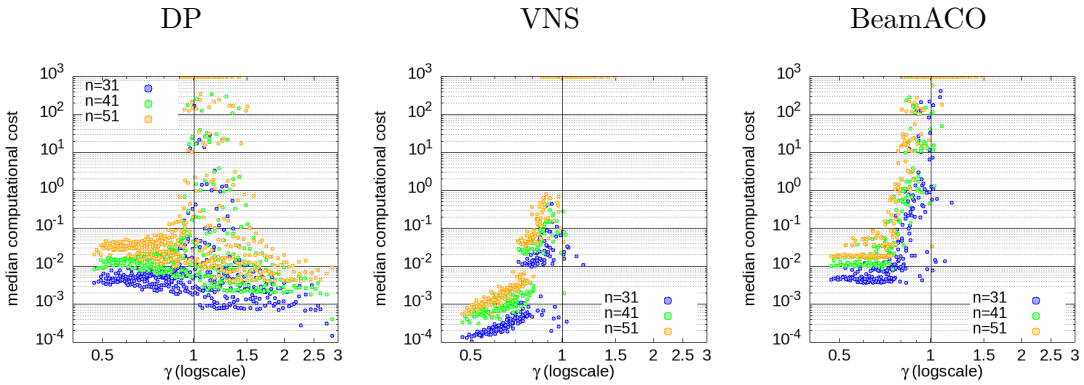

Figure 4: Evolution of the run time (y-axis) with respect to γ (x-axis) when $n \in \{31, 41, 51\}$. Each point (γ, t) corresponds to 10 instances with a same value for $(\alpha, \beta) \in [1, 3] \times [0.1, 0.5]$. γ is computed from α and β as defined in Eq. (9), and t is the median run time of DP (left), VNS (middle) and BeamACO (right) for these 10 instances.

As $\alpha = \frac{h}{ak\sqrt{n}}$ and $\beta = \frac{w}{h}$, Eq. (8) is equivalent to $\alpha = \frac{1}{\sqrt{\beta}}$, which corresponds to the black curves displayed in Fig. 2 and 3. We observe that these curves fit well to the empirical location of the phase transition region, that separates pink and green points.

Finally, let us introduce a single parameter, denoted γ, such that the phase transition occurs when $\gamma = 1$. By dividing Eq. (8) by w, we obtain:

$$\gamma = \frac{a \cdot k \cdot \sqrt{n}}{\sqrt{w \cdot h}} = \frac{1}{\alpha\sqrt{\beta}} \tag{9}$$

The feasible region corresponds to $\gamma < 1$ whereas the infeasible region corresponds to $\gamma > 1$.

We display in Fig. 4 the evolution of the run time of DP, VNS and BeamACO with respect to γ. We observe that the hardest instances occur when $0.9 \leq \gamma \leq 1.5$ for DP. As VNS and BeamACO are not able to prove infeasibility, we display run times for feasible instances only for these solvers. We observe that instances become much harder when γ gets closer to 1.

3.6 Extension to TWs with Non-fixed Lengths

Let us now extend our model to the case where TWs have variable widths, *i.e.*, the width of each TW is randomly generated. We consider two different models for generating TW widths:

- $U(\frac{w}{2}, \frac{3w}{2})$ is a uniform distribution within the range $[\frac{w}{2}, \frac{3w}{2}]$;
- $\mathcal{N}(w, \frac{w^2}{8})$ is a normal distribution with mean $\mu = w$ and variance $\sigma^2 = \frac{w^2}{8}$ trimmed to $[0, 2w]$

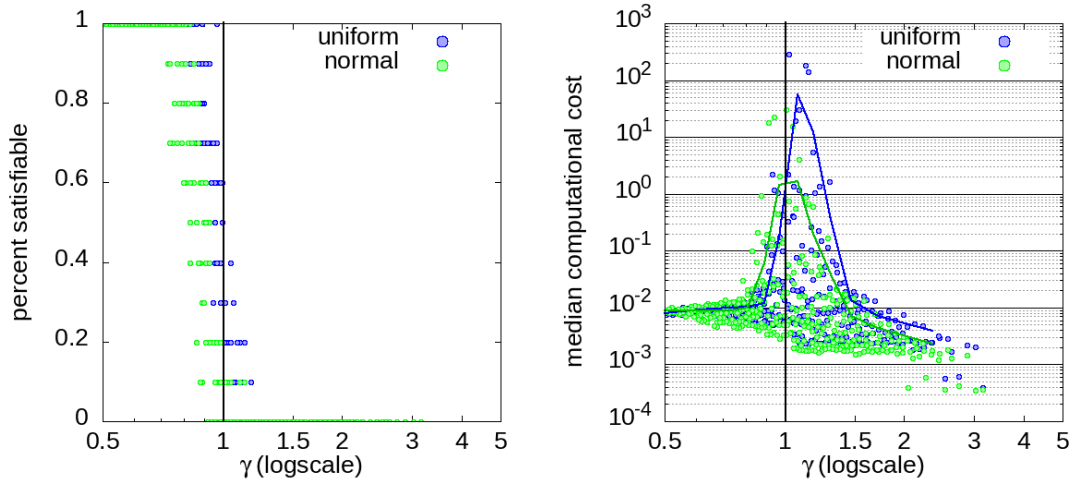

Figure 5: Evolution of feasibility (left) and run times (right) with respect to γ (x-axis) when $n = 31$ and the width of each TW is randomly generated according to $U(\frac{w}{2}, \frac{3w}{2})$ (blue points) or $\mathcal{N}(w, \frac{w^2}{4})$ (green points). Left (resp. Right): Each point (γ, f) (resp. (γ, t)) corresponds to 10 instances with a same value for $(\alpha, \beta) \in [1, 3] \times [0.1, 0.5]$. γ is computed from α and β as defined in Eq. (9) and $w = \beta\alpha ak\sqrt{n}$. f is the ratio of feasible instances (resp. t is the median run time of DP) for these 10 instances. The blue (resp. green) curve is the evolution of the average run time with respect to γ.

where w is computed from α and β, *i.e.*, $w = \beta\alpha ak\sqrt{n}$. In both models, the expected width of a TW is equal to w. Hence, the phase transition is expected to occur at the same position as when all TWs have a width equal to w, as defined in Eq. 9.

In Fig. 5, we display the evolution of feasibility with respect to γ for these two distributions when $(\alpha, \beta) \in [1, 3] \times [0.1, 0.5]$ and $n = 31$. We observe that the transition from the feasible region to the infeasible region actually occurs when γ is close to 1. We also observe that the hardest instances for DP occur within this phase transition, and that instances generated using the uniform distribution are harder in average than those generated with the normal distribution.

3.7 Impact of Service Times on the Phase Transition Region

In many practical applications of the TSPTW, we cannot leave a point at the same time as we arrive on it: we must wait for a given service time s. In this case, Eq. (4) must be replaced with:

$$A_i + s + d_{iN_i} \leq A_{N_i}, \forall i \in \mathcal{P} \setminus \{n\}.$$

To adapt Eq (8) to the TSPTW with service times, we add the service times of the $\frac{n \cdot w}{h}$ points that are expected to be visited in w units of time to the expected length of the

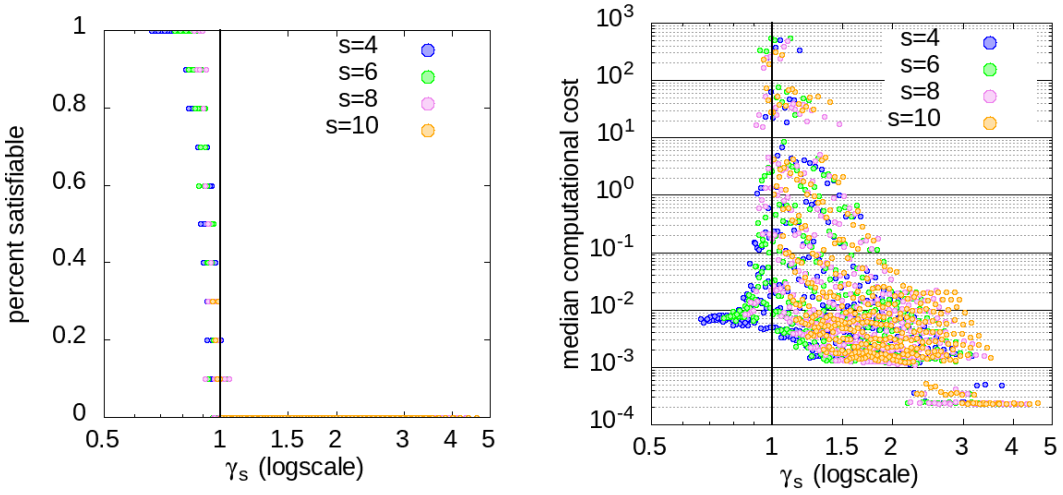

Figure 6: Evolution of feasibility (left) and run times (right) with respect to γ_s (x-axis) when the service time $s \in \{4, 6, 8, 10\}$ and when $n = 31$. Left (resp. Right): Each point (γ_s, f) (resp. (γ_s, t)) corresponds to 10 instances with a same value for $(\alpha, \beta) \in [1, 3] \times [0.1, 0.5]$. γ_s is computed from α and β as defined in Eq. (11). f is the ratio of feasible instances (resp. t is the median run time of DP) for these 10 instances.

shortest path that contains these points, *i.e.*, we expect the phase transition to occur when

$$w = \frac{a \cdot k \cdot \sqrt{n \cdot w}}{\sqrt{h}} + \frac{n \cdot w \cdot s}{h}. \tag{10}$$

We introduce the parameter γ_s to locate the phase transition:

$$\gamma_s = \frac{a \cdot k \cdot \sqrt{n}}{\sqrt{w \cdot h}} + \frac{n \cdot s}{h}. \tag{11}$$

We display in Fig. 6 the evolution of feasibility for different service times s ranging from 4 to 10 when $n = 31$. It shows us that the percentage of feasible instances drops from 100% to 0% when γ_s evolves from 0.75 to 1.2, and that the solving time of DP strongly increases in the transition zone.

4. Empirical Hardness of the Optimization Problem

In practice, we usually not only search for a feasible tour which satisfies all TW constraints, but also search for the shortest feasible tour. In Figure 7, we display the average runtime of DP to find the optimal solution of ten instances randomly generated according to $\mathrm{ET}(n, \alpha, \beta)$ when $n \in \{20, 30, 40\}$, $\alpha \in [1, 3]$, and $\beta \in [0, 0.5]$. (We limit the number of instances to three for $\alpha \geq 2.70$ (resp. 1.7, and 1.2) when $\beta = 0.4$ (resp. 0.45, and 0.5) and $n = 41$, as these

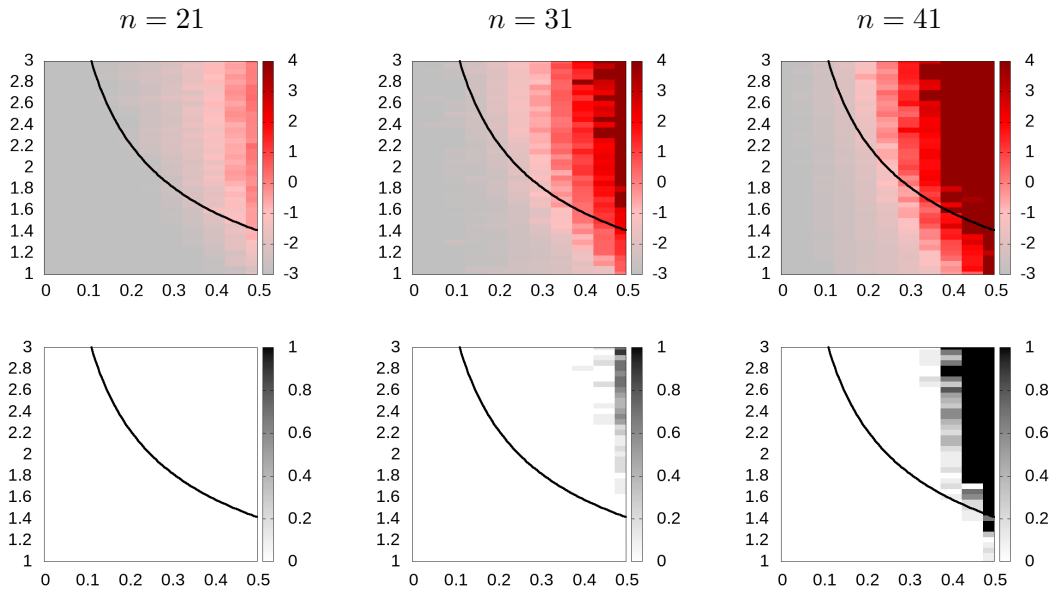

Figure 7: Top: Average runtime of DP to compute optimal solutions of 10 instances of $\mathrm{ET}(n, \alpha, \beta)$ when $n \in \{21, 31, 41\}$ (from left to right), $\alpha \in [1, 3]$ (y-axis), and $\beta \in [0, 0.5]$ (x-axis). Bottom: Percentage of instances which are not solved by DP within 2 hours. The black curve is the function $\alpha = \frac{1}{\sqrt{\beta}}$

instances need more time to be solved.) We consider that an instance is solved when the optimal solution has been found and has been proven optimal. As the optimization problem is harder to solve than the decision problem, we extend the time limit to two hours. If all instances with $n = 21$ are solved within this limit, 2.6% of the instances with $n = 31$ and 16.8% of the instances with $n = 41$ are not solved.

This figure shows us that hardness more strongly depends on β than on α, *i.e.*, on TW tightness than on the horizon tightness. The hardest instances occur when $\beta \geq 0.4$, *i.e.*, when the width of the TWs is larger than 40% of the time horizon h. When increasing β, we increase the number of feasible solutions, and DP needs more time to find the shortest one and prove its optimality.

Let us now consider the time needed to find the reference solution for the feasible instances, thus allowing us to report experimental results of non-exact solvers too. This reference solution is the best solution found by DP within two hours. In Figure 8, we display the average time needed by DP, VNS, and BeamACO to find the reference solution (considering the same instances as in Figure 7). Runtimes have been limited to 100 seconds, and we display a black point whenever the reference solution has not been found within this limit for at least one instance. For both VNS and BeamACO, hard instances are located in two different regions: in the phase transition zone, where finding a feasible instance is difficult for these two heuristic approaches, and in the region with very large TWs when β gets closer to 0.5.

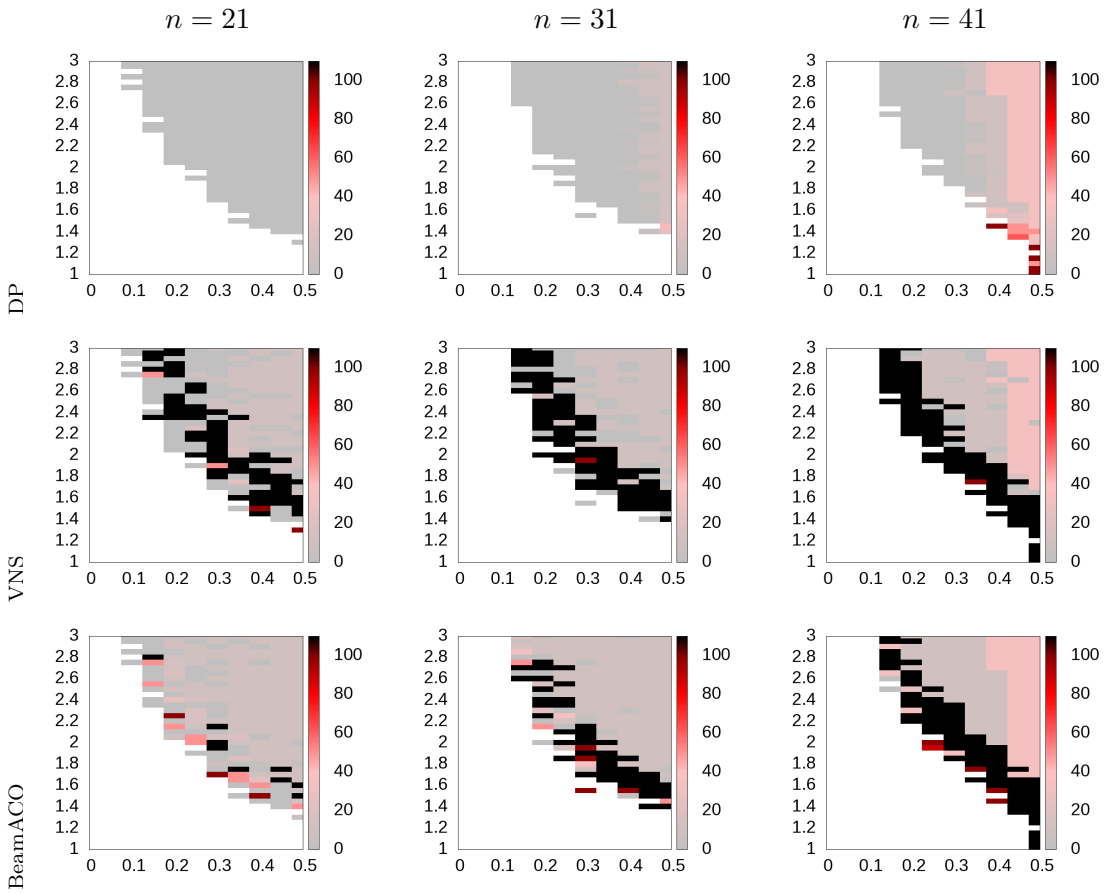

Figure 8: Time of DP, VNS, and BeamACO to find the reference solution (on average for ten instances) when $n \in \{21, 31, 41\}$ (from left to right), $\alpha \in [1, 3]$ (y-axis), and $\beta \in [0, 0.5]$ (x-axis). We display a black point when the reference solution has not been found within 100s for at least one of the ten instances.

As the reference solution is not found within 100 seconds for many instances, we also display in Fig. 9 the average gap to the reference solution (considering the same instances as in Figure 7): given the length l of the shortest tour computed within 100 seconds, and the length $l*$ of the reference solution, the gap is defined by $\frac{l-l*}{l*}$. This gap is nearly always very close to 0 for DP. However, both VNS and BeamACO struggle to find feasible solutions for some instances within the phase transition zone. BeamACO and VNS exhibit rather complementary performance. Indeed, BeamACO is able to find more feasible solutions than VNS for instances in the transition zone: when $n = 21$ (resp. $n = 31$ and $n = 41$), BeamACO cannot find feasible solutions for 9 (resp. 61 and 136) instances whereas VNS cannot find solutions for 85 (resp. 142 and 172) instances. As a counterpart, VNS often computes much better solutions than BeamACO for instances that are far enough from the phase transition zone: when $n = 21$ (resp. $n = 31$ and $n = 41$), the number of instances for

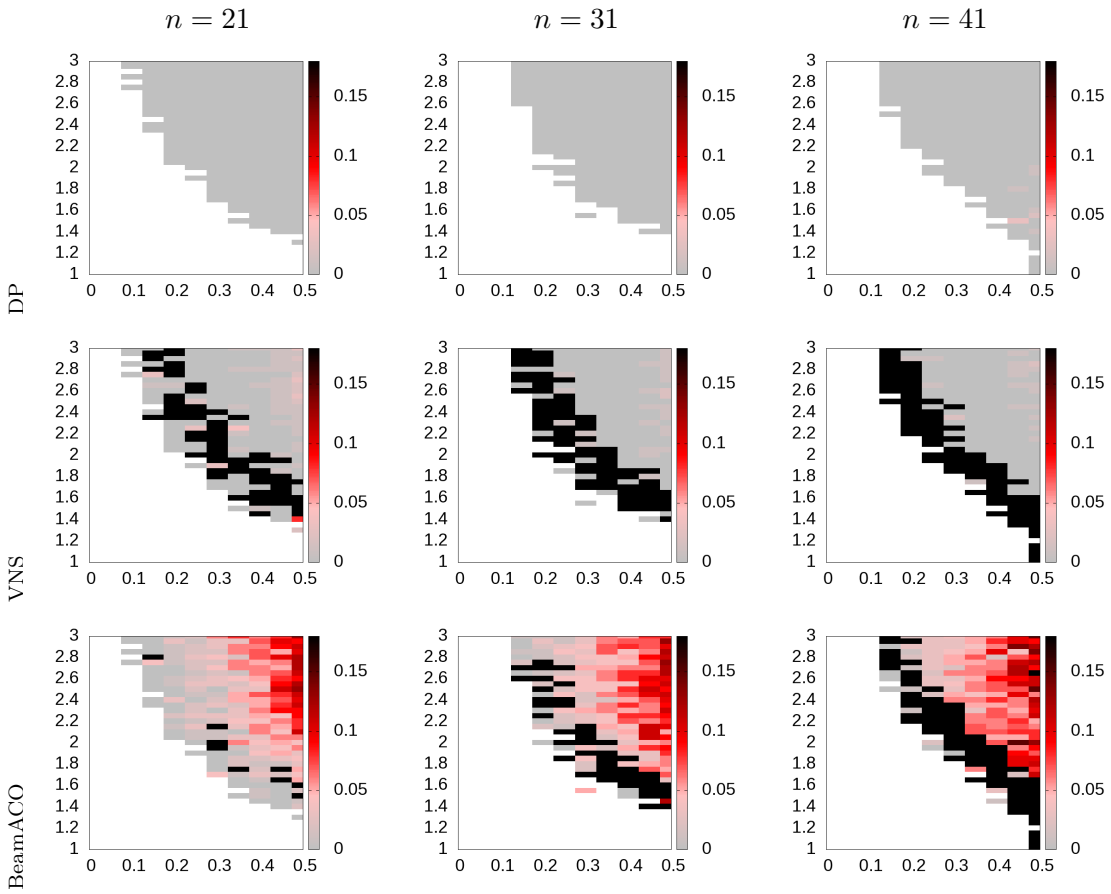

Figure 9: Gap to the reference solution (on average for ten instances) for DP, VNS and BeamACO when runtimes are limited to 100s and when $n \in \{21, 31, 41\}$ (from left to right), $\alpha \in [1, 3]$ (y-axis), and $\beta \in [0, 0.5]$ (x-axis). We display a black point when no feasible solution has been found within 100s for at least one of the ten instances.

which the gap is larger than 10% after 100s is equal to 4 (resp. 4 and 6) for VNS whereas it is equal to 236 (resp. 272 and 140) for BeamACO.

Both DP, VNS, and BeamACO are anytime approaches which output a sequence of feasible tours of decreasing lengths: Each time a new feasible tour is found, the horizon h is upper bounded by its length to search for a shorter feasible tour. Let $s_k, s_{k-1}, \ldots, s_2, s_1$ be the solutions successively found by an anytime solver, where s_k is the first found solution, and s_1 the last one (*i.e.*, s_1 is the shortest tour if the solver is exact). For each solution s_i, let l_i be the corresponding tour length, and $\gamma_i = \frac{a \cdot k \cdot \sqrt{n}}{\sqrt{w \cdot l_i}}$ be the value of γ when replacing h with l_i. For each solution $1 < i \leq k$, we have $l_i > l_{i-1}$ and, therefore, $\gamma_i < \gamma_{i-1}$. For loosely constrained instances, the initial value of γ is expected to be much smaller than 1, so that the first solution s_k is expected to be quickly computed. Then, each time a new solution s_i

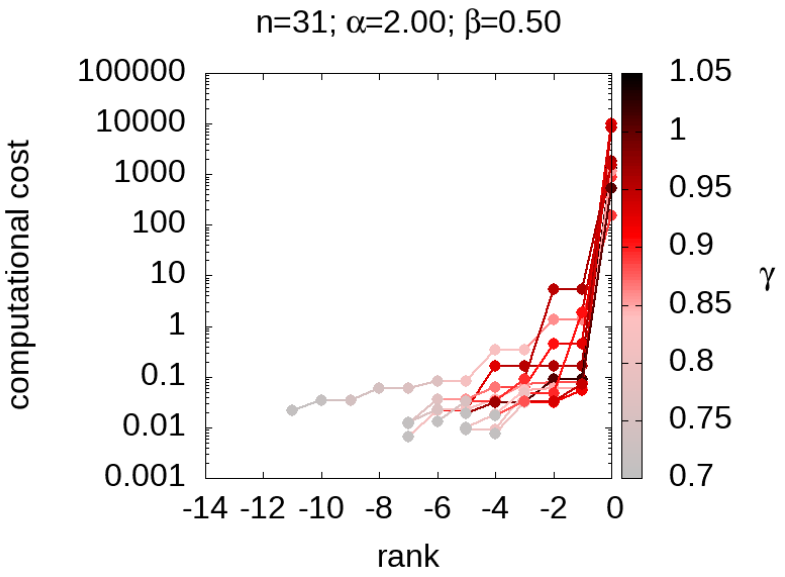

Figure 10: Evolution of γ_i and solution times, for ten instances generated with ET$(31, 2, 0.5)$ and solved with DP. For each instance, if DP has successively computed k solutions $s_k, \ldots, s_1$, we display a curve composed of $k+1$ points: for each solution s_i, we draw the point $(-i, y_i)$ where y_i is the time at which s_i has been found, and finally, we draw the point $(0, y)$ where y is the total solving time (including optimality proof). Point colors correspond to γ_{i+1}.

is found, the value of γ_i increases and gets closer to 1, so that the time needed to compute the next solution is expected to increase.

This is illustrated in Fig. 10, for 10 instances with $n = 31$, $\alpha = 2$ and $\beta = 0.5$ solved with DP. We observe that the initial value of γ never exceeds 0.71 and that the time to compute the first solution never exceeds 0.22 seconds. When γ_i is lower than 0.85, solutions are quickly computed. It starts increasing when γ_i exceeds 0.9. When $i = 1$ (*i.e.*, when computing the optimal solution), $\gamma_i \in [0.85, 1.02]$ (average 0.92), and the solving time is in the interval $[0.05, 5.45]$. Finally, when proving optimality, $\gamma_i \in [0.86, 1.03]$ (average 0.94), and the solving time is in $[155.45, 10080]$.

Note that we do not observe "wiggly" lines as observed by McCreesh et al. (2019) for the maximum clique problem. These wiggly lines are the results of the gaps between the complexity peaks of different decision problems, that involve deciding whether a clique of a given order exists or not, and they are more particularly visible when graphs are not very dense, *i.e.*, when maximum cliques are rather small (*e.g.*, less than 20): in this case, adding one to the objective function lower bound when a new lower bound is found significantly changes the position of an instance with respect to the phase transition. As a comparison, when $n = 31$, optimal tour lengths vary between 250 and 600, depending on α and β: in this case, subtracting one from the objective function upper bound does not really change the

value of γ, as observed in Fig. 10 when comparing γ_1 (associated with the optimal solution s_1) and γ_0 (associated with $s_1 + 1$).

5. Conclusion

We have studied the feasibility and the empirical hardness of randomly generated instances of the TSPTW. We have introduced two parameters: α, that defines the tightness of the time horizon with respect to the number of points to visit, and β, that defines the tightness of the TWs with respect to the time horizon. When both parameters have small (resp. large) values, nearly all generated instances are infeasible (resp. feasible). The transition from the infeasible region to the feasible one is sharp and corresponds to instances for which solvers struggle to find feasible tours (if the instance is feasible) or prove inconsistency (if the instance is infeasible and the solver is exact). We have shown that this transition occurs when $\alpha = \frac{1}{\sqrt{\beta}}$, and we have introduced a third parameter $\gamma = \frac{1}{\alpha\sqrt{\beta}}$) to locate the phase transition zone with a single parameter.

When considering the optimization problem, solvers not only search for a feasible tour, but for the shortest one. Anytime solvers output sequences of feasible tours of decreasing length by tightening the time horizon each time a new feasible tour is found. As a consequence, instances get progressively closer to the transition zone during the solving process. We have experimentally shown that the two heuristic solvers VNS and BeamACO have complementary performance with respect to γ: BeamACO is more successful in the transition zone whereas VNS finds better solutions quicker in the feasible region. Hence, our study could be used to dynamically select the best suited approach: BeamACO when γ is close to 1, and VNS otherwise. Also, γ could be used to dynamically adapt algorithms during the search. For example, DP combines the exploration of a state-transition graph derived from Bellman equations with local search, to find better solutions quicker, and bounding functions, to prune states that cannot lead to shorter tours. When γ is much lower than 1, local search is very useful as it usually strongly improves the computed tours, whereas sophisticated bounding functions (based on the computation of minimal spanning arborescences, for example) are often useless as many states can still lead to shorter tours. On the opposite, when γ gets close to 1, local search becomes useless and could be deactivated, whereas sophisticated bounding functions become useful.

Our study may be used to design new benchmarks for the TSPTW. Existing benchmarks such as those introduced by Langevin et al. (1993), by Dumas et al. (1995), by Gendreau et al. (1998), or by Ohlmann and Thomas (2007) only contain feasible instances: To ensure feasibility, TWs are defined so that they include the visit time of an *a priori* tour which is either randomly or greedily generated. When considering the decision problem, these instances are not really challenging: for example, DP is able to find a feasible solution within a time limit of 1000 seconds for all these instances, including the largest instances that have more than 200 points. Our approach allows one to generate much more challenging instances for the decision problem, e.g., instances with 51 vertices for which none of the three considered solvers are able to find a feasible solution within 1000 seconds. We believe exact solvers should be evaluated on these hard instances too. Our two parameters α and β allow one to evaluate scale-up properties of solvers with respect to TW and horizon tightness.

Our study may also be used to generate more relevant training sets of instances for approaches based on machine learning. Indeed, these approaches need large sets of instances to train models and, in many cases, these training instances are randomly generated. For example, in (Zheng et al., 2023) and in (Cappart et al., 2021), TSPTW instances are randomly generated according to the same model as the one used in (Langevin et al., 1993; Dumas et al., 1995; Gendreau et al., 1998; Ohlmann & Thomas, 2007) in order to ensure feasibility. Our model could be used to generate training sets that contain instances with various levels of hardness, and this should improve the quality of the learned models.

For this first study, we have considered Euclidean instances only. We plan to extend our work to non Euclidean instances, and more particularly to real-world instances. We are more particularly interested in real-world instances that consider time-dependent cost functions in order to take into account traffic conditions, such as the benchmark introduced by Rifki et al. (2020). To do this, a key point is to evaluate the expected length of shortest Hamiltonian tours without TWs. This could be done, for example, by computing shortest Hamiltonian tours for randomly sampled sets of points, and we plan to investigate whether we can extend our model in this case.

We also plan to extend our study to the Vehicle Routing Problem with Time Windows (VRPTW), where the set of points must be visited by a fleet of k vehicles. If $k \geq n-1$, then the decision problem may be solved in polynomial time: if $e_0 + d_{0i} \leq l_i$ and $l_i + d_{i0} \leq h$ for every point $i \in \mathcal{P} \setminus \{0\}$, then the instance is feasible (a feasible solution is obtained by visiting each point with a different vehicle); otherwise it is infeasible. However, if $k < n-1$, then the decision problem becomes $\mathcal{NP}$-complete as it is more general than the TSPTW (which corresponds to the case where $k = 1$). In this case, the phase transition depends not only on the tightness of the horizon and of the TWs, but also on the number k of vehicles, and we need to extend our model to that case.

Acknowledgments

This work was primarily conducted for Omar Rifki as a member of LISIC laboratory.
Christine Solnon acknowledges the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-22-CE22-0016-01 (project MAMUT).

References

Beardwood, J., Halton, J. H., & Hammersley, J. M. (1959). The shortest path through many points. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 55, pp. 299–327. Cambridge University Press.

Cappart, Q., Moisan, T., Rousseau, L., Prémont-Schwarz, I., & Ciré, A. A. (2021). Combining reinforcement learning and constraint programming for combinatorial optimization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pp. 3677–3687. AAAI Press.

Cheeseman, P. C., Kanefsky, B., & Taylor, W. M. (1991). Where the really hard problems are.. In *IJCAI*, Vol. 91, pp. 331–337.

Da Silva, R. F., & Urrutia, S. (2010). A general VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, *7*(4), 203–211.

Dumas, Y., Desrosiers, J., Gelinas, E., & Solomon, M. M. (1995). An optimal algorithm for the traveling salesman problem with time windows. *Operations research*, *43*(2), 367–371.

Fontaine, R., Dibangoye, J., & Solnon, C. (2023). Exact and anytime approach for solving the time dependent traveling salesman problem with time windows. *Eur. J. Oper. Res.*, *311*(3), 833–844.

Gendreau, M., Hertz, A., Laporte, G., & Stan, M. (1998). A generalized insertion heuristic for the traveling salesman problem with time windows. *Operations Research*, *46*(3), 330–335.

Gent, I. P., & Walsh, T. (1996). The TSP phase transition. *Artificial Intelligence*, *88*(1-2), 349–358.

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In Hammer, P., Johnson, E., & Korte, B. (Eds.), *Discrete Optimization II*, Vol. 5 of *Annals of Discrete Mathematics*, pp. 287–326. Elsevier.

Kotthoff, L. (2013). Llama: leveraging learning to automatically manage algorithms. *ArXiv preprint arXiv:1306.1031*.

Langevin, A., Desrochers, M., Desrosiers, J., Gélinas, S., & Soumis, F. (1993). A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. *Networks*, *23*(7), 631–640.

Leyton-Brown, K., Hoos, H. H., Hutter, F., & Xu, L. (2014). Understanding the empirical hardness of NP-complete problems. *Communications of the ACM*, *57*(5), 98–107.

López-Ibáñez, M., & Blum, C. (2010). Beam-ACO for the travelling salesman problem with time windows. *Computers & operations research*, *37*(9), 1570–1583.

López-Ibáñez, M., Blum, C., Ohlmann, J. W., & Thomas, B. W. (2013). The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization. *Appl. Soft Comput.*, *13*, 3806–3815.

McCreesh, C., Pettersson, W., & Prosser, P. (2019). Understanding the empirical hardness of random optimisation problems. In Schiex, T., & de Givry, S. (Eds.), *Principles and Practice of Constraint Programming*, Vol. 11802 of *Lecture Notes in Computer Science*, pp. 333–349. Springer.

McCreesh, C., Prosser, P., Solnon, C., & Trimble, J. (2018). When subgraph isomorphism is really hard, and why this matters for graph databases. *J. Artif. Intell. Res.*, *61*, 723–759.

Ohlmann, J. W., & Thomas, B. W. (2007). A compressed-annealing heuristic for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, *19*(1), 80–90.

Rifki, O., Chiabaut, N., & Solnon, C. (2020). On the impact of spatio-temporal granularity of traffic conditions on the quality of pickup and delivery optimal tours. *Transportation Research Part E: Logistics and Transportation Review*, *142*, 102085.

Rifki, O., Garaix, T., & Solnon, C. (2021). An asymptotic approximation of the traveling salesman problem with uniform non-overlapping time windows. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pp. 983–988. IEEE.

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

Sleegers, J., Olij, R., van Horn, G., & van den Berg, D. (2020). Where the really hard problems aren't. *Operations Research Perspectives*, *7*, 100160.

Smith, B. M., & Dyer, M. E. (1996). Locating the phase transition in binary constraint satisfaction problems. *Artif. Intell.*, *81*(1-2), 155–181.

Smith-Miles, K., & Muñoz, M. A. (2023). Instance space analysis for algorithm testing: Methodology and software tools. *ACM Comput. Surv.*, *55*(12).

Smith-Miles, K., & van Hemert, J. (2011). Discovering the suitability of optimisation algorithms by learning from evolved instances. *Ann Math Artif Intell*, *61*, 87–104.

Stein, D. M. (1978). An asymptotic, probabilistic analysis of a routing problem. *Mathematics of Operations Research*, *3*(2), 89–101.

Xu, L., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2007). SATzilla-07: The design and analysis of an algorithm portfolio for SAT. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 712–727. LNCS 4741, Springer.

Zhang, W. (2004). Phase transitions and backbones of the asymmetric traveling salesman problem. *J. Artif. Intell. Res.*, *21*, 471–497.

Zheng, J., He, K., Zhou, J., Jin, Y., & Li, C.-M. (2023). Reinforced lin–kernighan–helsgaun algorithms for the traveling salesman problems. *Knowledge-Based Systems*, *260*, 110144.