CIRRELT

CIRRELT-2023-21

Impact of Distance Data Inaccuracies on Vehicle Routing Algorithms: An Experimental Study

Fernando Obed Guillen Reyes
Jean-Yves Potvin
Michel Gendreau
Thibaut Vidal

June 2023

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Impact of Distance Data Inaccuracies on Vehicle Routing Algorithms: An Experimental Study

Fernando Obed Guillen Reyes[1,2], Jean-Yves Potvin[1,2,*], Michel Gendreau[1,3], Thibaut Vidal[1,3,4]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
2. Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada
3. Département de mathématiques et de génie industriel, Polytechnique Montréal, Canada
4. Chaire de recherche SCALE AI sur les chaînes d'approvisionnement pilotées par les données

Abstract. This paper studies the impact of distance data inaccuracies on different methods for solving the capacitated vehicle routing problem. It is an experimental study examining how simple heuristics, state-of-the-art metaheuristics, and an exact algorithm behave under such inaccuracies. We investigate the two following questions: i) is it worth using sophisticated state-of-the art algorithms when data inaccuracies are greater or similar in their scale to algorithmic error gaps; ii) would the best algorithms be more attracted by inaccurate (e.g., erroneously shorter) distances and be disproportionally affected by inaccurate data? We conduct extensive experiments on the capacitated vehicle routing problem, considering instances ranging from 100 to 1,000 customers with different distance-estimation inaccuracies. Interestingly, we respond to the first question in the affirmative and the second one in the negative: errors from data and algorithms tend to compound, such that state-of-the-art algorithms remain the better choice in all cases. Moreover, the impact of inaccurate data is fairly uniform over the algorithms, except for a simpler construction heuristic (Clarke and Wright algorithm) which seems disproportionally impacted by inaccurate data.

Keywords: Vehicle routing, data inaccuracy, heuristics, exact algorithm.

Acknowledgements. Financial support was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged.

---

* Corresponding author: potvin@iro.umontreal.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2023

1 Introduction

The Capacitated Vehicle Routing Problem (CVRP) seeks to find efficient routes to transport goods from a depot to several customers using a fleet of vehicles limited by their capacity. This problem is of significant interest to researchers because any improvement in the methods developed for it can result in significant savings for transportation companies. Although many complex variants of the CVRP have been studied in the last decades, the canonical CVRP remains challenging for large-scale instances. In this case, one must often resort to heuristics and metaheuristics to produce solutions of good quality in a reasonable amount of computation time.

Formally, the CVRP can be stated as follows. We consider a complete graph $G = (V, A)$, where V is the set of nodes and A is the set of arcs. The set V contains the customers plus the depot. Each customer $i \in V$ has a demand q_i that must be served exactly once by a vehicle. Each arc $(i, j) \in A$ between node i and node j is associated with a distance d_{ij}. A fleet of homogeneous vehicles of capacity Q is located at the depot. Each vehicle performs one closed route that starts at the depot, delivers the demand to a subset of customers, and returns to the depot at the end. The goal is to assign all customers to vehicles and sequence the customers assigned to each vehicle to minimize the total distance (or travel time) while not exceeding the capacity of any vehicle.

In most practical cases, data (e.g., travel times) can be inaccurate due to the devices and methods used to collect them, as well as possible human errors. In view of this, though the development of more and more accurate heuristics is interesting from a methodological standpoint, many practitioners would argue that sophisticated heuristics capable of approximating the exact solutions within a fraction of percent errors do not pay off, given that the magnitude of the errors in the data is likely larger than that of nowadays heuristics. Going even further, in a similar fashion as learning algorithms, some could suspect that state-of-the-art optimization algorithms applied to inaccurate data may "overfit" and deliberately use arcs whose distances are most underestimated.

In this paper, we conduct extensive experimental analyses to examine these common beliefs. To this end, we design a controlled experiment consid-

ering four CVRP heuristics and an exact method tested on instances ranging from 100 to 1,000 customers with different distance-estimation inaccuracies. We show that the two aforementioned beliefs are incorrect and that state-of-the-art methods consistently produce solutions of better quality even when considering the largest level of inaccuracy in the data. Moreover, the impact of inaccurate data is fairly consistent over the algorithms, except for the Clarke-and-Wright construction algorithm, which seems disproportionally impacted by inaccurate data.

The rest of this paper is organized as follows. Section 2 reviews related works, mostly from the machine learning literature, in which the impact of inaccurate data on algorithmic performance is studied. Section 3 presents our experimental setting, the heuristics considered, and the generation of test instances with inaccurate distances from the datasets presented in [1] and the 2021 DIMACS implementation challenge on vehicle routing problems [2]. Section 4 discusses our computational results and Section 5 concludes.

2 Related works

Extensive research has been conducted on the CVRP, but no paper has, to our knowledge, analyzed the impact of data inaccuracies on the results of classical heuristics, metaheuristics, and exact methods. The closest related analyses come from the machine learning and data mining literature where the impact of noise has been studied. Given that noise is unavoidable in real-world problems, several methods in machine learning have been aimed at identifying different types of noise and ways to handle them. For example, the work reported in [3] reviews uncertainty and big data analytics and discusses the impact of different sources of errors in the data and their consequences. Methods to deal with noise and to measure it are also reported. The authors in [4] additionally survey studies published between 1993 and 2018 about noise identification and mitigation.

Outside of the optimization and vehiclle routing domain, many papers in machine learning study the impact of inaccurate data on solutions produced by different algorithms. The following papers share a common approach by comparing the sensitivity of different algorithms to different types and levels of noise in the data. In [5, 6, 7, 8], simple learning algorithms like decision

trees, naive Bayesian classifiers, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and logistic regression are considered. In [8], a new metric called Equalized Loss of Accuracy (ELA) is proposed to evaluate different algorithms based on their overall performance and their robustness to noise. Also, the work in [9] reports an extensive comparison of different Bayesian network structure learning algorithms under noise.

Multiple classifier systems (MCSs) that combine different learning algorithms have also attracted the attention of researchers. MCSs are obtained by combining different learning algorithms. Although MCSs are totally different from the algorithms used to solve the CVRP, it is also quite common to combine different heuristics in problem-solving methods for the CVRP. In [10], the authors study the behavior of MCSs under noise by combining the decision tree algorithm C4.5, SVM and k-NN. Similar studies are reported in [11, 12, 13] using MCSs based on AdaBoost, boosting, randomization and bagging techniques.

In the data mining field, statistical analyses on the impact of noise (and unbalanced datasets) on the performance of 11 learning algorithms have been reported in [14]. In this work, the authors added noise to seven datasets while controlling the overall amount of corrupted data. Through an analysis of variance (ANOVA) they determine which factors, like the level of noise or the percentage of noisy data in a given class, impact the most the considered learning algorithms.

Overall, those studies indicate that noise is almost always present in real-world applications and has a considerable impact on the performance of learning algorithms. Also, it is observed that different types of noise might impact a particular algorithm differently. Thus, accuracy is not the only desirable feature of an algorithm, but also its robustness to noise.

3 Experimental setting

In this section, we describe the experimental setting designed to evaluate the impact of noise on different problem-solving methods for the CVRP. First, the tested heuristics and metaheuristics are introduced. Then, the benchmark instances are described. Finally, we explain how perturbations

to these instances were generated to include different levels of noise in the distances.

3.1 Solution algorithms

A large variety of heuristics have been proposed to solve the CVRP, from simple heuristics to sophisticated metaheuristics. To analyze the impact of noise on different types of methods, we considered the following approaches:

(a) The *Clarke & Wright savings heuristic* (CW) [15] starts with an initial solution in which an individual route serves each customer. Then, it evaluates for each existing route the "distance savings" due to merging each route pair. These savings are then sorted from largest to smallest. Starting at the top of the list of savings, the two routes associated with the current saving are merged to form a single route. This process is then repeated until no pair of routes can be merged without violating the capacity constraints.

(b) The open source software *Google OR-Tools* (GORT) [16] is a generic tool designed to solve various operations research problems, including VRPs. Here, the user must define the data (instance) and write a high-level algorithm that calls the required predefined functions to solve a problem. For example, the user defines the method to compute the initial solution, the local search method, the stopping criterion, etc. In our case, we chose PATH_CHEAPEST_ARC to generate the initial solution and GUIDED_LOCAL_SEARCH to perform the local search, as recommended in the user manual for VRPs.

(c) The *Slack Induction by String Removal* (SISR) [17] is a recent metaheuristic based on the ruin-and-recreate paradigm where, at each iteration, a part of the current solution is partially destroyed by removing several customers from the routes and by creating a new solution through a reinsertion procedure. A simulated annealing criterion is used to determine whether the new solution should be accepted as the current solution. In the computational results, the parameter settings suggested in [17] were used.

(d) The *Hybrid Genetic Search* (HGS) [18] is the current state-of-the-art metaheuristic for the CVRP. It is a hybrid genetic algorithm that combines crossover operations for solution generation with a neighborhood search for solution improvement. Additionally, the population is maintained to ensure a diversified search, and infeasible solutions are allowed with adaptive penalties. For this algorithm, we rely on parameter settings suggested in [18].

Regarding the stopping criterion, it should be noted that CW naturally stops when it is impossible to merge two routes without violating the capacity constraints. For GORT, SISR and HGS, the stopping criterion is set to a time limit that depends linearly on the instance size, as in [18]. More precisely, we use:

$$T(n) = 2.4 \cdot n \text{ seconds} \tag{1}$$

per instance, where n is the number of customers. With this, the time limit ranges from four minutes on an instance with 100 customers to 40 minutes for 1,000 customers.

As previously mentioned, we also include experiments with an exact branch-cut-and-price algorithm [19]. However, since this algorithm is limited in the size of problems that it can solve, our experiments with this approach are applied only to a subset of smaller instances. This is covered in a dedicated section (Section 4.2).

3.2 Test instances

We conducted our experiments on the 100 classical synthetic CVRP instances from Uchoa et al. [1] with Euclidean distances, as well as 12 real-world instances provided by the companies Loggi and ORTEC for the 2021 DIMACS implementation challenge.

The six instances from Loggi contain 400 to 1,000 customers and come from urban delivery services in some of the largest cities in Brazil. The distances provided in these instances were calculated from the urban networks. The six instances from ORTEC contain between 241 and 700 customers and were derived from the activities of a US-based grocery delivery service. In

this latter case, the distances correspond to real driving times.

The Euclidean instances from Uchoa et al. [1] (usually called "Set X") have a size ranging from 100 to 1,000 customers. These instances were generated to reflect a variety of possible factors and situations. Their generation has been driven by six main factors:

- n: number of customers (i.e., instance size)
- *Dep*: location of the depot, either C (center of the grid), E (corner of the grid), or R (random);
- *Cust*: location of the customers, either R (random), C (clustered), or RC(k) (mixed) with $k \in \{3, \ldots, 8\}$ clusters
- *Dem*: customer demand, which can be:

  U: all demands equal to 1
  1-10: demand from UD[1-10] (where UD[a,b] denotes a uniform discrete distribution over interval [a, b])
  5-10: demand from UD[5-10]
  1-100: demand from UD[1-100]
  50-100: demand from UD[50-100]
  Q: demand from UD[1-50] or UD[51-100] depending if the customer is located in an even or odd quadrant of the grid, respectively
  SL: many Small values, few Large values, that is, 70% to 95% of demands are from UD[1-10], while the remaining demands are from UD[50-100]

- r: approximation of the average number of customers per route (see [1]).

3.3 Distance inaccuracies

It is common to observe data inaccuracies due, for example, to the use of geolocalization devices to collect data. To account for these inaccuracies, we introduce perturbations to the distances (or travel times) between each pair of nodes. For a given distance d_{ij}, a value ϵ_{ij} is generated by truncating a normal distribution of mean 1 and standard deviation σ. Thus, the distribution is

defined within the interval $[1-\sigma, 1+\sigma]$. With this, a "noisy" distance d^*_{ij} is obtained by multiplying d_{ij} by ϵ_{ij}, that is:

$$d^*_{ij} = \epsilon_{ij} \cdot d_{ij}. \tag{2}$$

Larger standard deviations σ lead to larger perturbations to the original distances. To analyze the impact of noise, six levels were considered in our computational study: $\sigma = 0.00$ (no noise), and then $\sigma = 0.01, 0.02, 0.05, 0.10,$ and 0.15.

Let us denote the 100 instances of Set X as P_1 to P_{100}, where P_1 is the smallest instance and P_{100} is the largest one. For each original instance P_i with $i \in \{1, \ldots, 100\}$ and noise level $\sigma \in \{0.01, 0.02, 0.05, 0.10, 0.15\}$, we generated ten noisy instances $j \in \{1, \ldots, 10\}$. In the following, each noisy instance is denoted $P^\sigma_{i,j}$. This leads to $100 \times 5 \times 10 = 5000$ noisy instances. Finally, in the case of $\sigma = 0.00$ (no noise), ten runs of SISR and HGS were performed on each instance with different random seeds to obtain averages, as in the other cases. For GORT, instead of random seeds, different orders of customers were used because this algorithm is sensitive to the order of the data.

Let s^σ_{ij} be the solution obtained with a given method on a noisy instance $P^\sigma_{i,j}$. We can evaluate the quality of this solution using the true distances (ground truth) in the original instance P_i to obtain the true total distance (cost) traveled by the vehicles, denoted as $c(s^\sigma_{ij})$. Then, considering s_i the best-known solution of the original instance P_i and the corresponding cost $c(s_i)$ (see the Appendix), the percentage gap between the true cost of solution s^σ_{ij} and the cost of the best-known solution s_i can be computed as:

$$Gap^\sigma_{ij} = 100 \left(\frac{c(s^\sigma_{ij}) - c(s_i)}{c(s_i)} \right). \tag{3}$$

For an original instance P_i and a given noise level σ, the average percentage gap over the ten corresponding noisy instances is calculated as:

$$Avg^\sigma_i = \frac{1}{10} \left(\sum_{j=1}^{10} Gap^\sigma_{ij} \right). \tag{4}$$

Finally, for a given noise level σ, the global average percentage gap over the 100 instances is calculated as:

$$Avg^{\sigma} = \frac{1}{1000}\left(\sum_{i=1}^{100}\sum_{j=1}^{10} Gap_{ij}^{\sigma}\right). \tag{5}$$

The noisy instances based on the 12 real-world instances of Loggi and ORTEC were generated in the same way, leading to $12 \times 5 \times 10 = 600$ noisy instances overall.

4 Experimental study

Our experiments are divided into three parts. First, Section 4.1 examines the impact of noise on the average gap of the four heuristics presented before, using the set of 5,000 noisy test instances derived from Set X. Next, Section 4.2 conducts the same experiment for the exact method, using a subset of small instances from Set X, and compares the results obtained with our heuristic methods. Finally, Section 4.3 examines the impact of noise on real-world instances using a set of 600 noisy instances derived from Loggi's and ORTEC's applications. All algorithms have been run on an Intel Gold 6148 Skylake 2.4 GHz processor with 40 GB of RAM under CentOS 7.8.2003 (one thread).

4.1 Performance of the heuristics on set X instances with inaccurate distances

We start our analysis by examining the average gaps Avg_i^{σ} of each heuristic on the ten noisy instances associated with each original instance P_i, $i = 1$, ..., 100, and each noise level σ, as defined in Equation (4). These results are reported in Figure 1.

Clearly, the simplest constructive heuristic (CW) has the worst performance. The solution quality achieved by GORT is better than that of CW but still far below the solution quality of the two other metaheuristics SISR and HGS. The performance of GORT seems to degrade as the instance size increases, as indicated by the upward trend from left to right in the figure. SISR and HGS are the best heuristics and produce similar results. Also, the

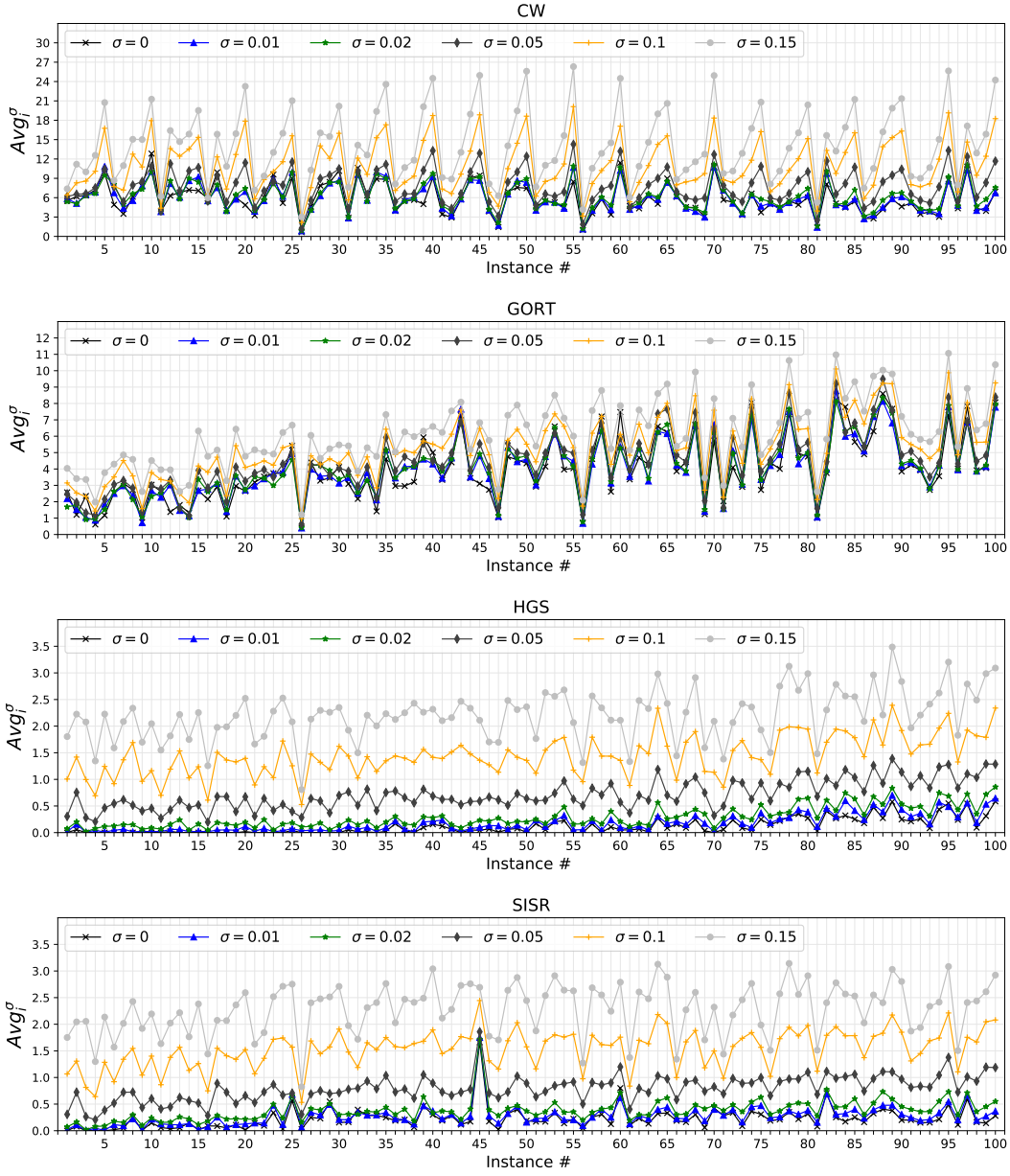

Figure 1: Average gaps obtained on each original instance for each noise level

average gaps increase with the level of noise σ, which was expected, given that the noisy instances differ more significantly from the original instances

when there is more noise.

The figure also shows regular patterns that are a consequence of the design of the Set-X instances in [1]. More precisely, the benchmark is organized into groups of five consecutive instances, such that the average number of customers per route in a solution, as represented by attribute r, increases from the first instance in a group to the fifth instance. Figure 1 thus indicates that the average gap with the best-known solution increases with the average number of customers per route in a solution, with a peak for instances P_5, P_{10}, P_{15}, etc. This behavior is more apparent for CW. We made an exhaustive examination to determine possible relations between the gaps shown in Figure 1 and the other Set-X instances attributes (n, Dep, $Cust$ and Dem), however, we did not find a strong relation.

An insightful view on the impact of noise is obtained when we consider Avg^{σ}, as defined in Equation (5), which is the global average gap with the best-known solution over the 1,000 noisy instances, generated from the 100 original instances, for any given noise level. This is shown in Figure 2, where each dot in the figure corresponds to the global average gap Avg^{σ} for a given heuristic and noise level σ. In this figure, we can observe that, on average, the noise level has a greater impact on CW, compared with the three other heuristics. To distinguish between HGS and SISR, a different scale is used for the graph on the right. This graph indicates that HGS is slightly more robust to noise than SISR.

The poor performance of CW is not much of a surprise. This is a pure construction heuristic whose behavior is totally dependent on the ordering of the savings (from largest to smallest) to merge routes. With inaccurate distances, the ordering of the savings is very likely to change, thus leading to bad decisions. And, as opposed to the three competing metaheuristics, this construction heuristic has no way to recover later when bad decisions are taken.

While Figure 2 shows the global average gaps Avg^{σ} for each heuristic and noise level σ, Figure 3 shows the standard deviations Std^{σ} of the corresponding global gaps which, obviously, increase with the noise level σ. As in Figure 2, another scale is used in the graph shown on the right to distinguish between SISR and HGS. We can see that the standard deviations of

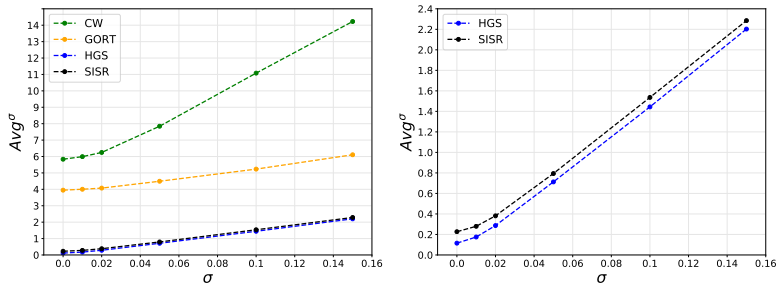

Figure 2: Global average gaps with increasing values of σ for each heuristic.

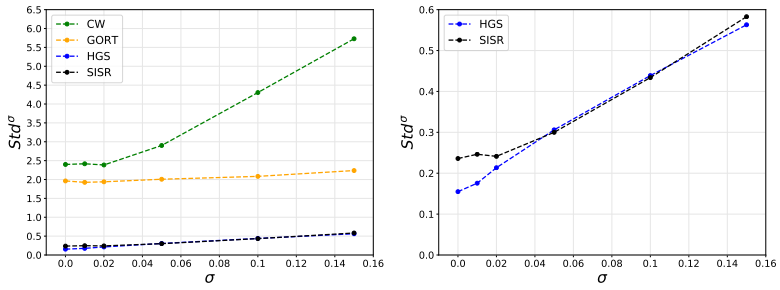

Figure 3: Standard deviation of global average gaps (large dots) with increasing values of σ for each heuristic.

SISR are substantially larger than HGS on the original instances (no noise). It means that the solutions produced via multiple executions of SISR (with different seeds) on the same instance vary more widely from one execution to the next. This difference gradually decreases as the level of noise increases and both SISR and HGS meet at $\sigma = 0.05$, where the noise due to distance inaccuracies start to dominate.

Table 1 shows the explicit global average gap values Avg^σ for each heuristic and noise level, which correspond to the dots in Figure 2, as well as the corresponding standard deviation values Std^σ, which correspond to the dots in Figure 3.

It is worth noting that a Wilcoxon signed-rank test, applied to every pair of methods, showed a statistically significant difference in solution quality in every case with a level of confidence of 95%.

	CW		**GORT**		**HGS**		**SISR**	
	Avg^σ	Std^σ	Avg^σ	Std^σ	Avg^σ	Std^σ	Avg^σ	Std^σ
$\sigma = 0.00$	5.83	2.39	3.94	1.96	0.11	0.15	0.22	0.23
$\sigma = 0.01$	5.99	2.41	4.00	1.92	0.17	0.17	0.27	0.24
$\sigma = 0.02$	6.24	2.38	4.07	1.93	0.28	0.21	0.38	0.24
$\sigma = 0.05$	7.84	2.90	4.49	2.00	0.71	0.30	0.79	0.29
$\sigma = 0.10$	11.08	4.30	5.23	2.08	1.44	0.43	1.53	0.43
$\sigma = 0.15$	14.22	5.72	6.10	2.23	2.20	0.56	2.28	0.58

Table 1: Global average gaps and standard deviations for each method and each noise level

4.2 Performance of an exact algorithm on set X with inaccurate distances

In this section, we additionally consider the results of an exact branch-cut-and-price (BCP) algorithm introduced in [19]. We analyze its sensitivity to noise compared to that of the heuristics (except CW which was largely outperformed by the other methods). For this analysis, we focus on the 13 instances for which it was possible to obtain optimal solutions within 12 hours of computation time for all seeds and noise levels (instances with index $i =$ 1, 2, 3, 4, 6, 7, 12, 13, 16, 17, 18, 21, and 22). This subset of instances is denoted $\hat{P}$ in the following. We also consider two additional noise levels (σ = 0.20, 0.25) to have a better sense of the behavior of SISR and HGS in the presence of very large noise.

Figure 4 shows the global average gaps with the optimum over the 13×10 = 130 instances for each level of noise σ and each method, including BCP. The methods SISR, HGS and BCP are so close on these relatively small instances, with at most 200 customers, that they are difficult to distinguish even by looking at the graph on the right, which uses a different scale. Accordingly, Table 2 reports explicit global average gap values, which correspond to the dots in Figure 4. In addition, the standard deviations of the corresponding global gaps are shown. Obviously, the gap of BCP is null for $\sigma = 0.00$ (no noise) since the original instances with true distances are solved optimally.

We observe that the difference between SISR and HGS stays approximately the same with increasing values of σ. Considering Table 2, the difference between the global average gap values Avg^σ of SISR and HGS remains

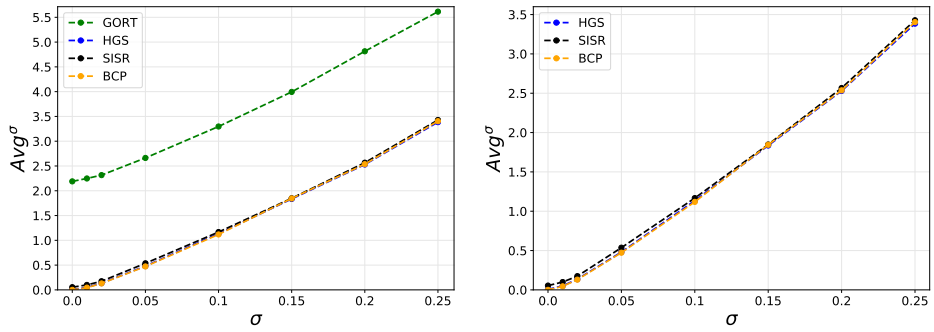

Figure 4: Global average gaps with the optimum for each method with increasing values of σ. On the left, GORT, HGS and SISR are compared with BCP; on the right, using a different scale, only HGS and SISR are compared with BCP.

in the range of 0.03% to 0.05%, except for $\sigma = 0.15$ where it is equal to 1.84% - 1.83% = 0.01%. While the difference between the global average gaps of SISR and HGS stays about the same, both methods get closer to BCP with increasing values of σ, to the point where HGS becomes in fact better than BCP on average for $\sigma = 0.15, 0.20, 0.25$.

To evaluate the statistical significance of these results, we performed multiple pairwise Wilcoxon signed-rank tests, where the null hypothesis states that solution quality is the same for the two methods involved in the test. The p-values obtained are shown in Table 3. Basically, when a p-value is less than or equal to 0.05, the null hypothesis is rejected and a statistically significant difference is observed with a 95% confidence level. Although SISR, HGS, and BCP are very close, statistically significant differences are observed with a low level of noise (see the gray cells, where HGS outperforms SISR and BCP outperforms both SISR and HGS). This is not the case for a high level of noise. Even if the difference between the global average gaps Avg^σ of HGS and SISR stays about the same over the different σ values, no statistically significant difference is observed between the two methods for σ values greater than or equal to 0.10, according to the Wilcoxon test. This is the same when SISR and HGS are compared with BCP. Thus, when two methods are very close with regard to solution quality, the superiority of one method over the other gets blurred when a sufficiently high level of noise is present (i.e., $\sigma = 0.10$ or more in our experiments).

	BCP		**GORT**		**HGS**		**SISR**	
	Avg^σ	Std^σ	Avg^σ	Std^σ	Avg^σ	Std^σ	Avg^σ	Std^σ
$\sigma = 0.00$	0.00	0.00	2.18	0.87	0.01	0.01	0.05	0.06
$\sigma = 0.01$	0.04	0.03	2.24	1.00	0.05	0.04	0.09	0.11
$\sigma = 0.02$	0.13	0.08	2.31	1.04	0.13	0.09	0.17	0.11
$\sigma = 0.05$	0.47	0.22	2.66	1.08	0.48	0.23	0.53	0.26
$\sigma = 0.10$	1.11	0.38	3.29	1.15	1.13	0.40	1.16	0.41
$\sigma = 0.15$	1.84	0.48	3.99	1.24	1.83	0.47	1.84	0.46
$\sigma = 0.20$	2.53	0.67	4.81	1.21	2.52	0.70	2.56	0.68
$\sigma = 0.25$	3.40	0.88	5.61	1.29	3.38	0.90	3.42	0.91

Table 2: Global average gaps and corresponding standard deviations for each method and each noise level

	SISR vs HGS	**SISR vs BCP**	**HGS vs BCP**
$\sigma = 0.00$	6.03×10^{-16}	7.76×10^{-16}	7.41×10^{-5}
$\sigma = 0.01$	2.48×10^{-8}	4.21×10^{-11}	0.03
$\sigma = 0.02$	1.60×10^{-5}	2.76×10^{-7}	0.02
$\sigma = 0.05$	1.90×10^{-5}	3.03×10^{-6}	0.21
$\sigma = 0.10$	0.05	0.01	0.24
$\sigma = 0.15$	0.60	0.96	0.23
$\sigma = 0.20$	0.33	0.37	0.69
$\sigma = 0.25$	0.13	0.27	0.35

Table 3: p-values of Wilcoxon signed-rank test

4.3 Performance of heuristics on real-world instances with inaccurate distances

Here, we examine the performance of our four heuristics over real data. For this purpose, we tested these heuristics on the real-world instances of Loggi and ORTEC. The best-known solutions for these instances can be found in the Appendix. Noise was introduced into the distance data in the same way as in the instances of Set X. Thus, the analysis is similar to that of Section 4.1.

Figure 5 shows the global average gaps Avg^σ for each heuristic and noise level σ over the 120 noisy instances, generated from the 12 real-world instances. As in the previous experiments, CW is much more affected by noise

than the other heuristics. Also, GORT remains far from SISR and HGS, whatever the level of noise. SISR and HGS are again quite close and by changing the scale (see the graph on the right), we observe that SISR is more affected by noise than HGS. Table 4 explicitly reports the global average gaps Avg^{σ}, in addition to the corresponding standard deviations Std^{σ}, for each heuristic and each noise level. We observe that the difference between the global average gaps of the two methods increases from 0.29% with $\sigma = 0.01$ to 0.56% with $\sigma = 0.15$ (no such increase was observed on Set-X instances, see Table 1). In fact, HGS outperforms SISR by a wider margin on the original (no noise) real-world instances when compared to the original Set-X instances. That is, the differences between the global average gaps of HGS and SISR are equal tof 0.28% and 0.11%, respectively.

As in Section 4.1, a Wilcoxon signed-rank test, applied to every pair of methods, showed a statistically significant difference in solution quality in every case with a level of confidence of 95%.

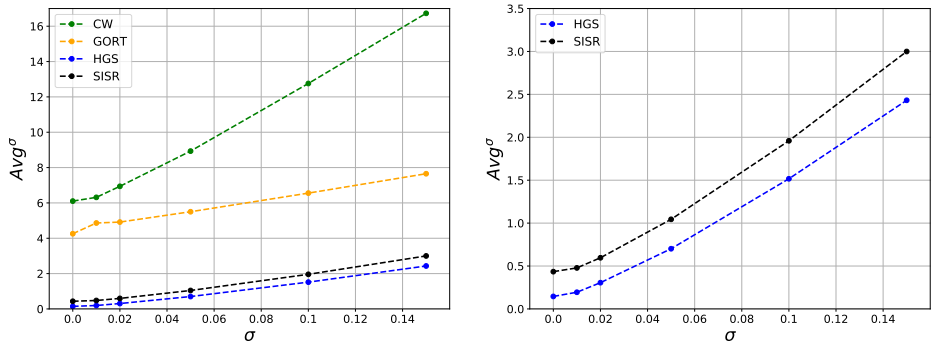

Figure 5: Global average gaps obtained on real instances with increasing values of σ for each heuristic.

5 Conclusions

This work has studied the performance of four heuristics and one exact method for the CVRP over two sets of instances using different levels of noise. The study first shows that not all methods are similarly impacted by noisy distances. In particular, the CW heuristic is much more sensitive to noise than the three other metaheuristics. Second, our results indicate

	CW		**GORT**		**HGS**		**SISR**	
	Avg^{σ}	Std^{σ}	Avg^{σ}	Std^{σ}	Avg^{σ}	Std^{σ}	Avg^{σ}	Std^{σ}
$\sigma = 0.00$	6.10	1.65	4.25	1.71	0.14	0.17	0.43	0.34
$\sigma = 0.01$	6.31	1.89	4.86	1.92	0.19	0.18	0.47	0.31
$\sigma = 0.02$	6.93	1.77	4.91	1.93	0.30	0.22	0.59	0.37
$\sigma = 0.05$	8.93	1.94	5.50	2.17	0.70	0.27	1.04	0.39
$\sigma = 0.10$	12.75	3.04	6.55	2.30	1.51	0.36	1.95	0.59
$\sigma = 0.15$	16.72	3.71	7.65	2.16	2.43	0.47	2.99	0.65

Table 4: Global average gaps and corresponding standard deviations on real instances for each heuristic and noise level

that better methods generally lead to better solutions, even in the presence of noise, which is true for both the synthetic and real-world instances in our test set. Accordingly, sophisticated state-of-the-art methods are still preferable when data are inaccurate. We also observed that when two methods exhibit a sufficiently small, although statistically significant, difference in performance on true data, this statistically significant difference vanishes with high levels of noise. This situation occurred when we compared the two state-of-the-art metaheuristics SISR and HGS with an exact method on a subset of small instances from Set X with at most 200 customers. For future work, it would be interesting to conduct similar studies for other, more complex, variants of the CVRP, for example the VRP with time windows for which many standard benchmark instances are available in the literature.

References

[1] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, A. Subramanian, New benchmark instances for the capacitated vehicle routing problem, European Journal of Operational Research 257 (3) (2017) 845 – 858.

[2] DIMACS, 12th implementation challenge: Vehicle routing, `http://dimacs.rutgers.edu/programs/challenge/vrp/` (April 2022).

[3] R. H. Hariri, E. M. Fredericks, K. M. Bowers, Uncertainty in big data analytics: Survey, opportunities, and challenges, Journal of Big Data 6 (2019) 44.

[4] S. Gupta, A. Gupta, Dealing with noise problem in machine learning data-sets: A systematic review, Procedia Computer Science 161 (2019) 466–474.

[5] A. Atla, R. Tada, V. Sheng, N. Singireddy, Sensitivity of different machine learning algorithms to noise, Journal of Computing Sciences in Colleges 26 (5) (2011) 96103.

[6] Z. Nazari, M. Nazari, M. S. S. Danish, D. Kang, Evaluation of class noise impact on performance of machine learning algorithms, International Journal of Computer Science and Network Security 18 (8) (2018) 148–153.

[7] N. Sushma Rani, Anurag, P. Srinivasa Rao, Study and analysis of noise effect on big data analytics, International Journal of Management, Technology and Enginnering 8 (12) (2019) 5841–5850.

[8] J. A. Sez, J. Luengo, F. Herrera, Evaluating the classifier behavior with noisy data considering performance and robustness: The equalized loss of accuracy measure, Neurocomputing 176 (2016) 26–35.

[9] A. C. Constantinou, Y. Liu, K. Chobtham, Z. Guo, N. K. Kitson, Large-scale empirical validation of Bayesian network structure learning algorithms with noisy data, International Journal of Approximate Reasoning 131 (2021) 151–188.

[10] J. A. Sez, M. Galar, J. Luengo, F. Herrera, Tackling the problem of classification with noisy data using multiple classifier systems: Analysis of the performance and robustness, Information Sciences 247 (2013) 1–20.

[11] T. G. Dietterich, An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization, Machine Learning 40 (2000) 139–157.

[12] P. Melville, N. Shah, L. Mihalkova, R. Mooney, Experiments on ensembles with missing and noisy data, Lecture Notes in Computer Science 3077 (2004) 293–302.

[13] T. M. Khoshgoftaar, J. Van Hulse, A. Napolitano, Comparing boosting and bagging techniques with noisy and imbalanced data, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 41 (3) (2011) 552–568.

[14] J. Van Hulse, T. Khoshgoftaar, Knowledge discovery from imbalanced and noisy data, Data & Knowledge Engineering 68 (12) (2009) 1513–1542.

[15] G. Clarke, J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12 (4) (1964) 568581.

[16] L. Perron, V. Furnon, Or-tools, Google, France (2019).
URL `https://developers.google.com/optimization/`

[17] J. Christiaens, G. Van den Berghe, Slack induction by string removals for vehicle routiing problems, Transportation Science 54 (2) (2020) 417–433.

[18] T. Vidal, Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood, Computers & Operations Research 140 (2022) 105643.

[19] A. Pessoa, R. Sadykov, E. Uchoa, F. Vanderbeck, A generic exact solver for vehicle routing and related problems, Mathematical Programming 183 (2020) 483523.

Appendix

Table 5 shows the number of customers n and best known solution values $c(s_i)$, $i = 1, ..., 100$, for the Set-X instances from [1]. Table 6 shows the number of customers n and best-known solution values $c(s_i)$, $i = 1, ..., 12$, for the real-world instances provided by Loggi and ORTEC for the 2021 DIMACS implementation challenge.

Instance	i	n	$c(s_i)$	**Instance**	i	n	$c(s_i)$	**Instance**	i	n	$c(s_i)$
X-n101-k25	1	101	27591	X-n261-k13	35	261	26558	X-n502-k39	69	502	69226
X-n106-k14	2	106	26362	X-n266-k58	36	266	75478	X-n513-k21	70	513	24201
X-n110-k13	3	110	14971	X-n270-k35	37	270	35291	X-n524-k137	71	524	154593
X-n115-k10	4	115	12747	X-n275-k28	38	275	21245	X-n536-k96	72	536	94868
X-n120-k6	5	120	13332	X-n280-k17	39	280	33503	X-n548-k50	73	548	86700
X-n125-k30	6	125	55539	X-n284-k15	40	284	20215	X-n561-k42	74	561	42717
X-n129-k18	7	129	28940	X-n289-k60	41	289	95151	X-n573-k30	75	573	50673
X-n134-k13	8	134	10916	X-n294-k50	42	294	47161	X-n586-k159	76	586	190316
X-n139-k10	9	139	13590	X-n298-k31	43	298	34231	X-n599-k92	77	599	108451
X-n143-k7	10	143	15700	X-n303-k21	44	303	21736	X-n613-k62	78	613	59535
X-n148-k46	11	148	43448	X-n308-k13	45	308	25859	X-n627-k43	79	627	62164
X-n153-k22	12	153	21220	X-n313-k71	46	313	94043	X-n641-k35	80	641	63694
X-n157-k13	13	157	16876	X-n317-k53	47	317	78355	X-n655-k131	81	655	106780
X-n162-k11	14	162	14138	X-n322-k28	48	322	29834	X-n670-k126	82	670	146332
X-n167-k10	15	167	20557	X-n327-k20	49	327	27532	X-n685-k75	83	685	68205
X-n172-k51	16	172	45607	X-n331-k15	50	331	31102	X-n701-k44	84	701	81923
X-n176-k26	17	176	47812	X-n336-k84	51	336	139111	X-n716-k35	85	716	43387
X-n181-k23	18	181	25569	X-n344-k43	52	344	42050	X-n733-k159	86	733	136190
X-n186-k15	19	186	24145	X-n351-k40	53	351	25896	X-n749-k98	87	749	77314
X-n190-k8	20	190	16980	X-n359-k29	54	359	51505	X-n766-k71	88	766	114454
X-n195-k51	21	195	44225	X-n367-k17	55	367	22814	X-n783-k48	89	783	72394
X-n200-k36	22	200	58578	X-n376-k94	56	376	147713	X-n801-k40	90	801	73305
X-n204-k19	23	204	19565	X-n384-k52	57	384	65940	X-n819-k171	91	819	158121
X-n209-k16	24	209	30656	X-n393-k38	58	393	38260	X-n837-k142	92	837	193737
X-n214-k11	25	214	10856	X-n401-k29	59	401	66163	X-n856-k95	93	856	88965
X-n219-k73	26	219	117595	X-n411-k19	60	411	19712	X-n876-k59	94	876	99299
X-n223-k34	27	223	40437	X-n420-k130	61	420	107798	X-n895-k37	95	895	53860
X-n228-k23	28	228	25742	X-n429-k61	62	429	65449	X-n916-k207	96	916	329179
X-n233-k16	29	233	19230	X-n439-k37	63	439	36391	X-n936-k151	97	936	132725
X-n237-k14	30	237	27042	X-n449-k29	64	449	55233	X-n957-k87	98	957	85465
X-n242-k48	31	242	82751	X-n459-k26	65	459	24139	X-n979-k58	99	979	118987
X-n247-k47	32	247	37274	X-n469-k138	66	469	221824	X-n1001-k43	100	1001	72359
X-n251-k28	33	251	38684	X-n480-k70	67	480	89449				
X-n256-k16	34	256	18839	X-n491-k59	68	491	66487				

Table 5: Best-known solutions on Set-X instances

Loggi				ORTEC			
Instance	i	n	$c(s_i)$	**Instance**	i	n	$c(s_i)$
Loggi-n401-k23	1	400	336903	ORTEC-n242-k12	7	241	123750
Loggi-n501-k24	2	500	177176	ORTEC-n323-k21	8	322	214071
Loggi-n601-k19	3	600	113155	ORTEC-n405-k18	9	404	200986
Loggi-n601-k42	4	600	347059	ORTEC-n455-k41	10	454	292485
Loggi-n901-k42	5	900	246301	ORTEC-n510-k23	11	509	184529
Loggi-n1001-k31	6	1000	284356	ORTEC-n701-k64	12	700	445543

Table 6: Best-known solutions on real-world instances