ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Integrating Machine Learning into state-of-the-art Vehicle Routing Heuristics

Daniele Vigo

DEI and CIRI-ICT, University of Bologna, Italy

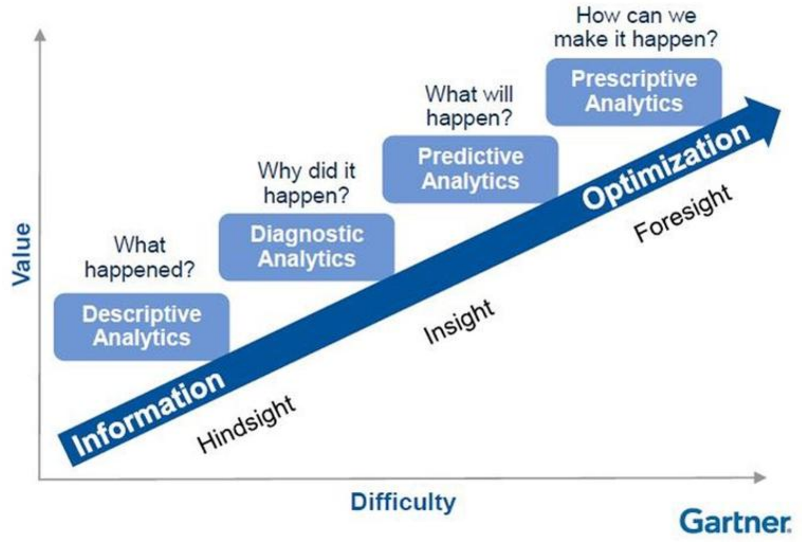
based on joint research work with Luca Accorsi, DEI

Agenda

- Introduction: OR and ML a bright future ?
- our work arena: the CVRP
- Attempt 1: Neighborhood ranking in VNS
- Intermezzo: Aggregate bounding
- Attempt 2: Characterization of solution quality
- Some preliminary conclusions

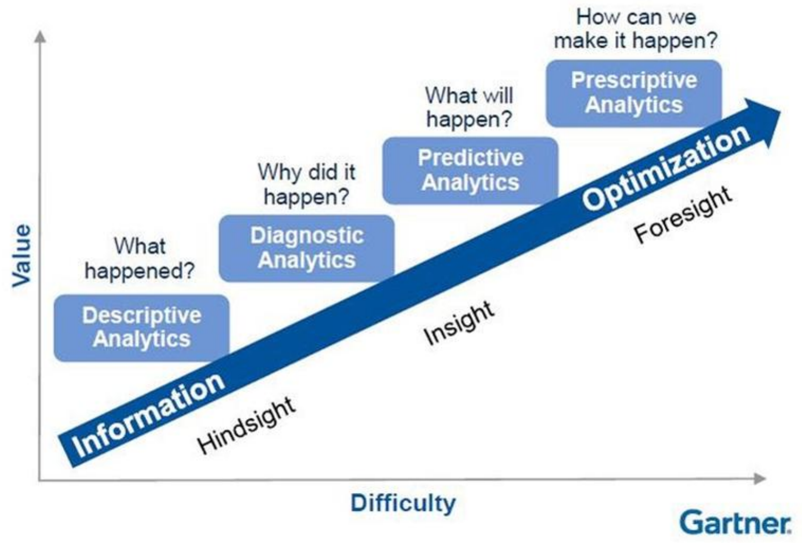
Operations Research and Machine Learning: a bright future

- Operations Research (a.k.a. Prescriptive Analytics) has an established and industry-recognized capability in supporting decision making by solving (combinatorial) optimization problems
    - OR methods work very well with accurate (deterministic) data and for structured cases 😀
    - unfortunately most problems are large, badly defined and with non-deterministic data 😢



Operations Research and Machine Learning: a bright future

- Machine Learning has an established capability of extracting from large quantities of data models that reproduce their behavior
  - ML may be used effectively to predict the outcome associated with data patterns,
    - to provide a fast approximation of a system
    - to identify behaviors from the observation of data 😀
  - ML methods outcomes provide valuable insights but are not generally sufficient for decision making 😢



What ML can do for OR methods

- Fast approximation of input/output patterns provided by ML algorithms may be used to replace computationally heavy tasks within OR algorithms (e.g. constraint or objective function evaluation)
- Learning mechanisms can be incorporated to guide the algorithm or to select the most effective components depending on
  - preliminary training of the algorithm
  - runtime observation of the algorithm's behavior
- By combining ML&OR shortcomings of either method may be mitigated

- Research in this field is still at a very early stage but activity is large and some promising results are coming
  - see Benjo,Lodi,Prouvost https://arxiv.org/pdf/1811.06128.pdf

Goal of this lecture

- Illustrate some (preliminary) attempts towards the integration of ML techniques for the benefit of heuristic methods
- We will focus on a well-known combinatorial optimization problem: the Capacitated Vehicle Routing Problem (CVRP)
  - Attempt 1: Neighborhood ranking in VNS
  - Intermezzo: Aggregate bounding
  - Attempt 2: Characterization of solution quality
  - Some preliminary conclusions

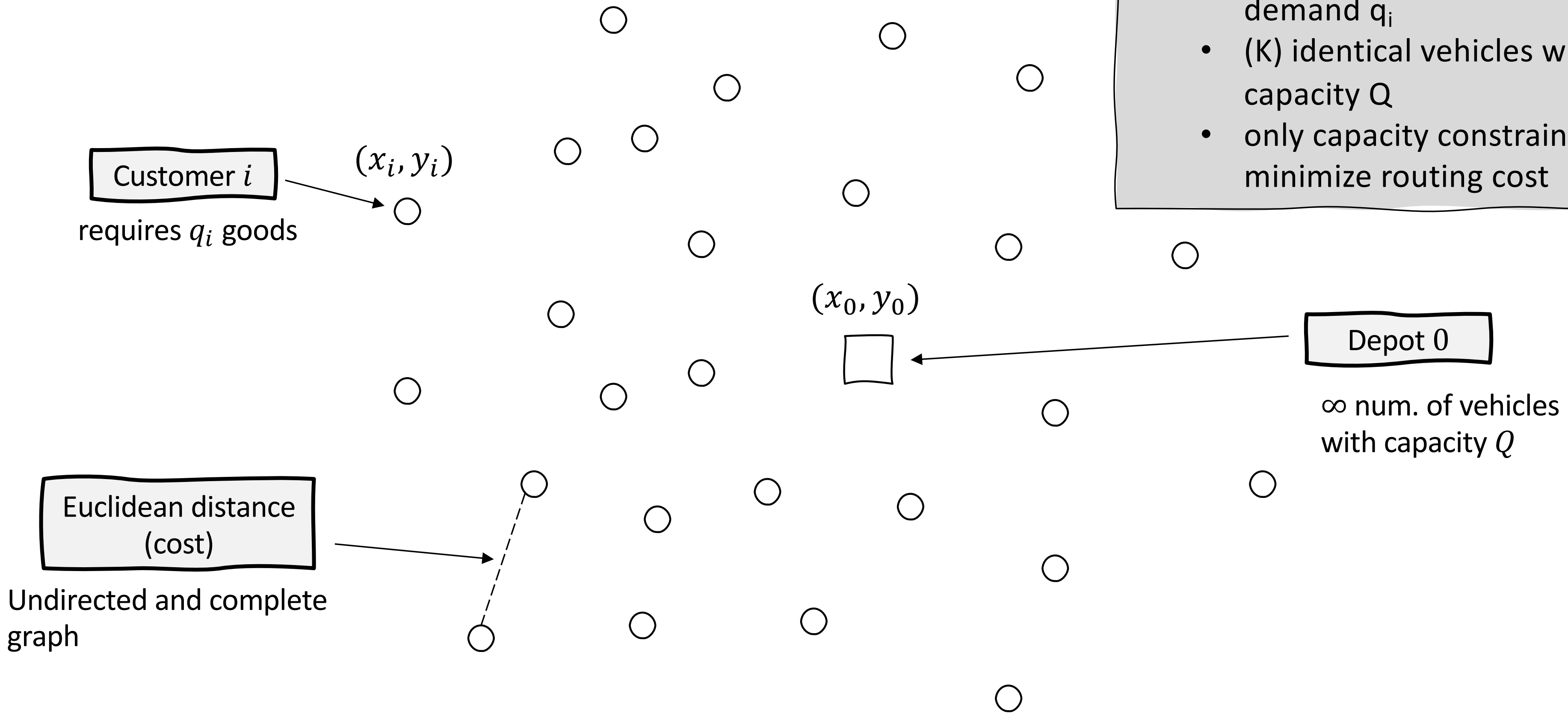
Our Work Arena

The Capacitated Vehicle Routing Problem
(CVRP, the godfather of all CO problems)

A CVRP Instance

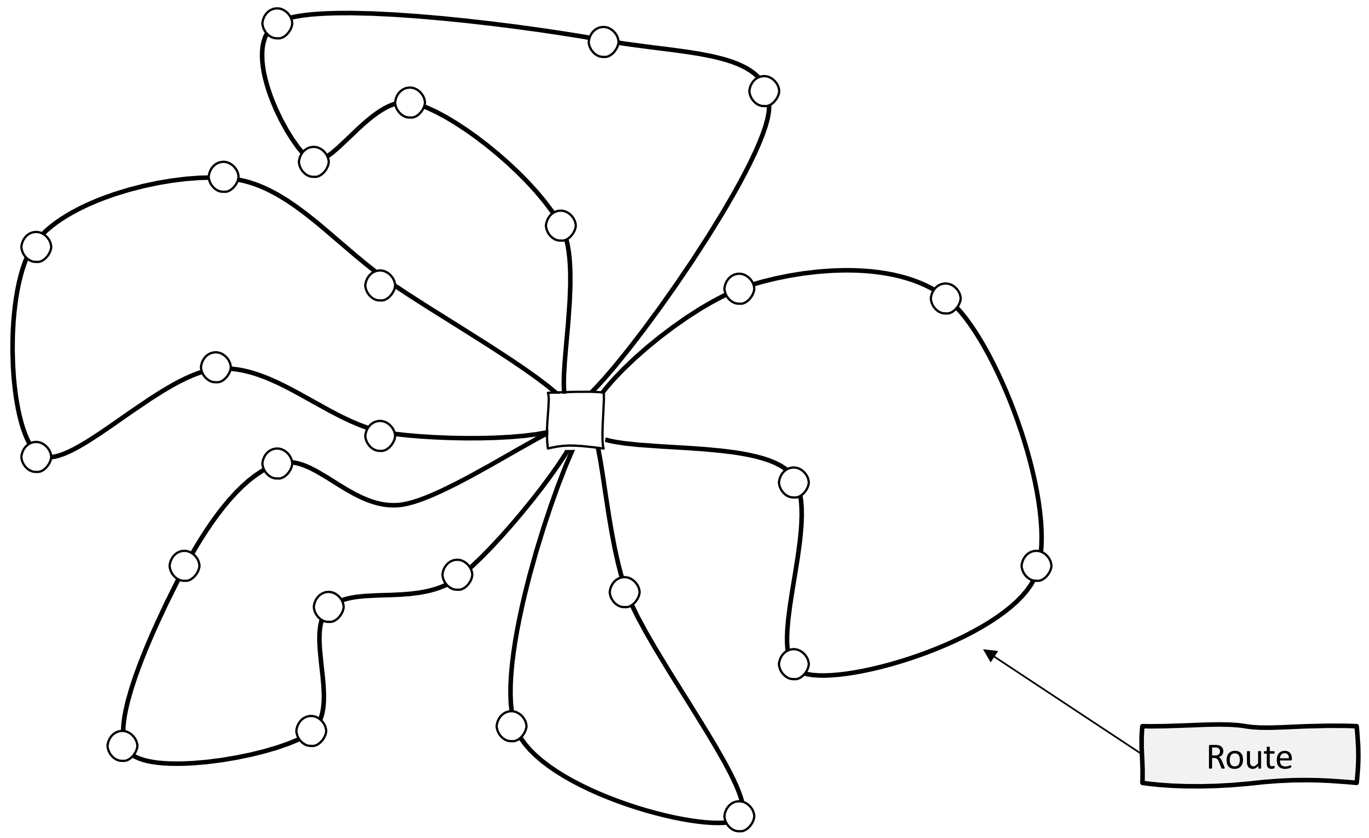
Capacitated Vehicle Routing Problem (CVRP)

- 1 depot, n customers with demand q_i
- (K) identical vehicles with capacity Q
- only capacity constraints and minimize routing cost



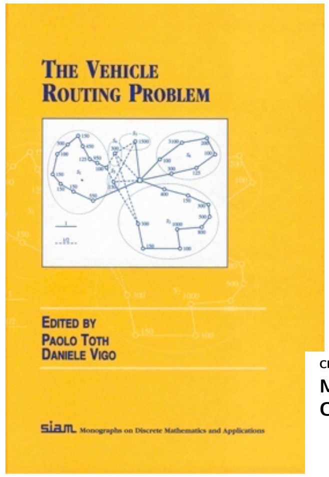
Customer i
requires q_i goods
(x_i, y_i)
(x_0, y_0)
Depot 0
∞ num. of vehicles with capacity Q
Euclidean distance (cost)
Undirected and complete graph

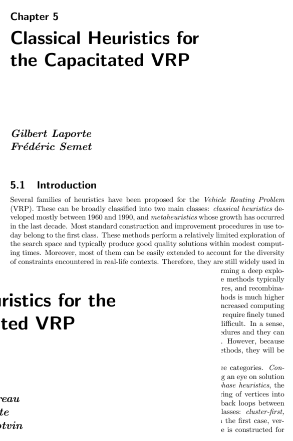
A CVRP Solution



Main references

- Classical Methods (1960-2000)
- Recent Methods (>2000)


THE VEHICLE ROUTING PROBLEM
EDITED BY
PAOLO TOTH
DANIELE VIGO
siam. Monographs on Discrete Mathematics and Applications



Chapter 5

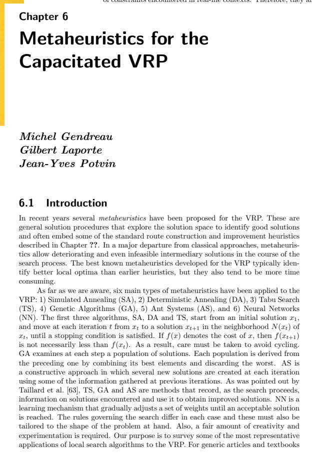
Classical Heuristics for the Capacitated VRP

Gilbert Laporte
Frédéric Semet

5.1 Introduction

Several families of heuristics have been proposed for the *Vehicle Routing Problem* (VRP). These can be broadly classified into two main classes: *classical heuristics* developed mostly between 1960 and 1990, and *metaheuristics* whose growth has occurred in the last decade. Most standard construction and improvement procedures in use today belong to the first class. These methods perform a relatively limited exploration of the search space and typically produce good quality solutions within modest computing times. Moreover, most of them can be easily extended to account for the diversity of constraints encountered in real-life contexts. Therefore, they are still widely used in
rming a deep explo-
e methods typically
res, and recombina-
hods is much higher
ncreased computing
require finely tuned
lifficult. In a sense,
edures and they can
. However, because
ethods, they will be

ee categories. *Con-*
g an eye on solution
hase heuristics, the
ring of vertices into
back loops between
lasses: *cluster-first,*
the first case, ver-
e is constructed for



Chapter 6

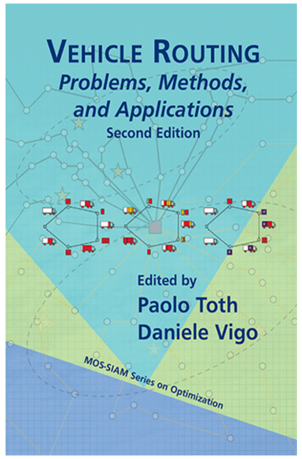
Metaheuristics for the Capacitated VRP

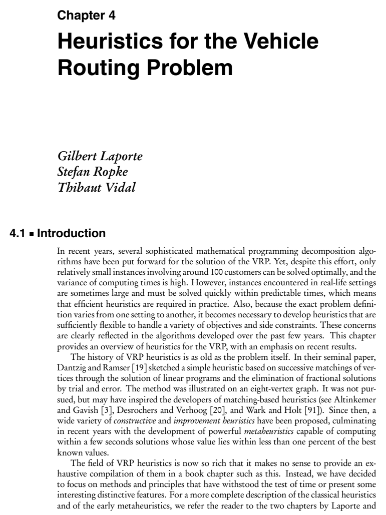
Michel Gendreau
Gilbert Laporte
Jean-Yves Potvin

6.1 Introduction

In recent years several *metaheuristics* have been proposed for the VRP. These are general solution procedures that explore the solution space to identify good solutions and often embed some of the standard route construction and improvement heuristics described in Chapter ??. In a major departure from classical approaches, metaheuristics allow deteriorating and even infeasible intermediary solutions in the course of the search process. The best known metaheuristics developed for the VRP typically identify better local optima than earlier heuristics, but they also tend to be more time consuming.

As far as we are aware, six main types of metaheuristics have been applied to the VRP: 1) Simulated Annealing (SA), 2) Deterministic Annealing (DA), 3) Tabu Search (TS), 4) Genetic Algorithms (GA), 5) Ant Systems (AS), and 6) Neural Networks (NN). The first three algorithms, SA, DA and TS, start from an initial solution x_1, and move at each iteration t from x_t to a solution x_{t+1} in the neighborhood $N(x_t)$ of x_t, until a stopping condition is satisfied. If $f(x)$ denotes the cost of x, then $f(x_{t+1})$ is not necessarily less than $f(x_t)$. As a result, care must be taken to avoid cycling. GA examines at each step a population of solutions. Each population is derived from the preceding one by combining its best elements and discarding the worst. AS is a constructive approach in which several new solutions are created at each iteration using some of the information gathered at previous iterations. As was pointed out by Taillard et al. [63], TS, GA and AS are methods that record, as the search proceeds, information on solutions encountered and use it to obtain improved solutions. NN is a learning mechanism that gradually adjusts a set of weights until an acceptable solution is reached. The rules governing the search differ in each case and these must also be tailored to the shape of the problem at hand. Also, a fair amount of creativity and experimentation is required. Our purpose is to survey some of the most representative applications of local search algorithms to the VRP. For generic articles and textbooks


VEHICLE ROUTING
Problems, Methods, and Applications
Second Edition
Edited by
Paolo Toth
Daniele Vigo
MOS-SIAM Series on Optimization



Chapter 4

Heuristics for the Vehicle Routing Problem

Gilbert Laporte
Stefan Ropke
Thibaut Vidal

4.1 ▪ Introduction

In recent years, several sophisticated mathematical programming decomposition algorithms have been put forward for the solution of the VRP. Yet, despite this effort, only relatively small instances involving around 100 customers can be solved optimally, and the variance of computing times is high. However, instances encountered in real-life settings are sometimes large and must be solved quickly within predictable times, which means that efficient heuristics are required in practice. Also, because the exact problem definition varies from one setting to another, it becomes necessary to develop heuristics that are sufficiently flexible to handle a variety of objectives and side constraints. These concerns are clearly reflected in the algorithms developed over the past few years. This chapter provides an overview of heuristics for the VRP, with an emphasis on recent results.

The history of VRP heuristics is as old as the problem itself. In their seminal paper, Dantzig and Ramser [19] sketched a simple heuristic based on successive matchings of vertices through the solution of linear programs and the elimination of fractional solutions by trial and error. The method was illustrated on an eight-vertex graph. It was not pursued, but may have inspired the developers of matching-based heuristics (see Altinkemer and Gavish [3], Desrochers and Verhoog [20], and Wark and Holt [91]). Since then, a wide variety of *constructive* and *improvement heuristics* have been proposed, culminating in recent years with the development of powerful *metaheuristics* capable of computing within a few seconds solutions whose value lies within less than one percent of the best known values.

The field of VRP heuristics is now so rich that it makes no sense to provide an exhaustive compilation of them in a book chapter such as this. Instead, we have decided to focus on methods and principles that have withstood the test of time or present some interesting distinctive features. For a more complete description of the classical heuristics and of the early metaheuristics, we refer the reader to the two chapters by Laporte and

Background

some previous works on integrating
ML within CVRP heuristics

Previous/ongoing work on using ML for VRP

- Arnold & Sorensen (C&OR 2018): What makes a VRP solution good? The generation of problem-specific knowledge for heuristics
- seminal paper trying to determine features which characterize good/bad solutions
- such features may be favoured/penalized in a heuristic solver

Solution and instance features

- 18 features characterizing the instance and the solution were identified

Instance features

1. Number of customers
2. Number of routes
3. Degree of capacity utilization
4. Average distance between each pair of customers
5. Standard deviation of the pairwise distance between customers
6. Average distance from customers to the depot
7. Standard deviation of the distance from customers to the depot
8. Standard deviation of the radians of customers towards the depot

Solution features

1. Average number of intersections per customers
2. Longest distance between two connected customers, per route
3. Average distance between depot to directly-connected customers
4. Average distance between routes (their centers of gravity)
5. Average width per route
6. Average span in radian per route
7. Average compactness per route, measured by width
8. Average compactness per route, measured by radian
9. Average depth per route
10. Standard deviation of the number of customers per route

Training set generation

- A set of 5000 small (20-50 cust.) and 1000 larger (70-100 cust.) random instances is generated with different capacity and depot positions (8 classes)
- For each instance they generated
  - a good solution by using the Arnold and Sorensen C&OR 2017 heuristic
  - two less good solutions within 2% and 4% from the good one
  - by using the A&S heuristics (H1) and a randomized Clarke&Wright (H2)
  - in total 4 datasets {2%,4%} X {H1,H2} with 10000 small and 2000 large datapoints per class
  - 192,000 solutions in total labeled near_optimal/non_optimal

Classification

- they used different classifiers (DT, RF, SVM) to predict solution quality using 5-fold Cross Validation and obtained 60-90% prediction accuracy
- Using simple Decision Tree based on just one solution feature they identified the most predictive features

Solution features

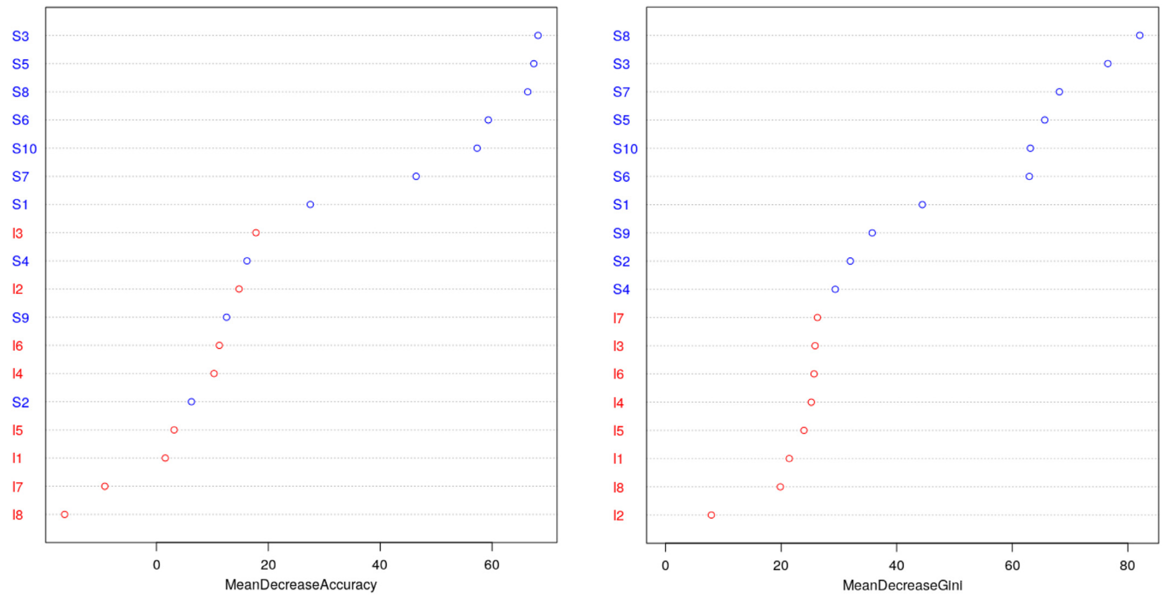
1. Average number of intersections per customers
2. Longest distance between two connected customers, per route
3. Average distance between depot to directly-connected customers
4. Average distance between routes (their centers of gravity)
5. Average width per route
6. Average span in radian per route
7. Average compactness per route, measured by width
8. Average compactness per route, measured by radian
9. Average depth per route
10. Standard deviation of the number of customers per route

Use of knowledge to guide heuristics

- The outcome of previous analyses identified solution width (S5) as a feature with high predictive value
- They used a Guided LS effective heuristics in which edges which increase solution width are penalized
- The overall GLS is very good on the famous X benchmark and other benchmark
- Actually the knowledge contribution is quite marginal (the GLS is indeed already very good !)

More on A&S features

- Recently, Lucas, Billot, Sevaux (C&OR 2019) analyzed in depth the features proposed by A&S
- They used random forests to rank the expalantory importance of the various features



Experiments on the utility of I1. . .I8.

Features	Random forest	Support-Vector machine
I1...I8, S1...S10	75.16%	77.28%
S1...S10	76.10%	77.16%

More on A&S Features

- they also studied the correlation between S features
- … and performed Principal Component Analysis
- I1-I8 and S2 can be removed "a priori" slighlty improving the overall accuracy

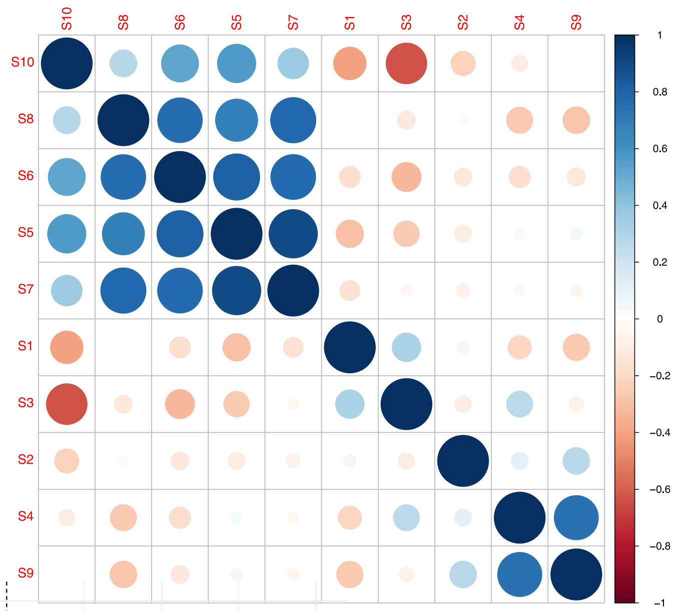
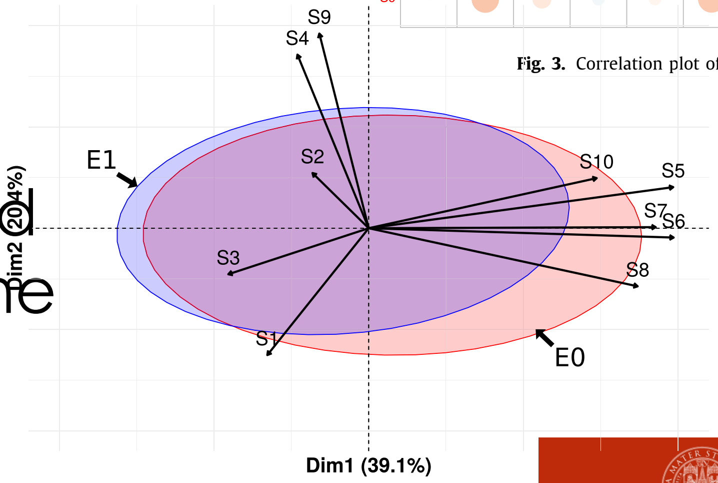


Fig. 3. Correlation plot of the solution metrics.



Other works on using ML for VRP

- Schrö̈der, Gauthier, Schneider, Irnich (Odysseus 2018)
  - use of ML to define the sparsification factor in a granular search
  - use of ML to accelerate sequential local search
- Arnold, Vidal, Santana, Sorensen (Verolog 2019):
  - frequent pattern mining in a set of elite solutions
  - during a "pattern injected local search" (PILS) patterns are used to define moves in which:
    - incompatible edges are removed, pattern edges are reconnected and remaining routes are optimally reconnected

Attempt #1

Neighborhood ranking in VND

joint work with L. Accorsi, M. Lombardi, M. Milano

Local Search

- All existing heuristic approaches for the CVRP (as well as for most Combinatorial Problems) rely on local search
- For CVRP several relatively simple neighborhoods are widely used and examined hundreds of thousands of times.
  - 2-opt and 2-opt*
  - relocate (one or more customers)
  - exchange (one or more customers)
  - ...
  - cardinality is typically $O(n^2)$

Local Search

- to achieve acceptable computing times
  - Granular neighborhoods (Toth&V., 2005),
  - Sequential search (Irnich et al. 2006)
  - Static Move Descriptors (Zachariadis&Kiranoudis, ‘10, Beek et al. ‘18, Accorsi and V., 2020)

- Simple (granular) neighborhoods are often combined and searched together to achieve better performance
- A common approach to combine neighborhoods is Variable Neighborhood Descent (Mladenovich&Hansen ‘97)

Variable Neighborhood Descent

- Variable Neighborhood Descent (VND):
  - k neighborhoods $N_1,\dots,N_k$

```
determine initial candidate solution s
i := 1
Repeat:
        choose a most improving neighbor $s_0$ of s in $N_i$
        If $g(s_0) < g(s)$ then s := $s_0$ ; i := 1
        Else i := i + 1
Until i > k
```

N. are typically ordered according to increasing size/effectiveness or randomly (RVND)

Our goal

- in a VND setting,
  - using ML techniques to identify the most promising neighborhood (or that none of them will be able to improve a given solution)
  - enables computational savings that could be used to perform a more fruitful search.
- focus on the CVRP and train an Artificial Neural Network for ranking neighborhoods at search time.
- preliminary experimental results show that using an informed neighborhood selection strategy for local search helps in avoiding the exploration of unpromising neighborhoods.

12 Neighborhoods considered

- relocate/exchange
  1. one_zero
  2. one_one,
  3. two_zero,
  4. two_one,
  5. two_two,
  6. three_zero,
  7. three_one,
  8. three_two,
  9. three_three,
- 3 intra and inter-route 2-opt exchanges (split, tail, intra)
- 2 different shaking:
  - random 1-0 exchanges
  - removal of some routes and reinsertion of customers

The value of a neighborhood exploration: A preliminary analysis

- We run a VND with 12 neighborhoods + shaking and stop after a given # of non improving iterations
- We simulate the following strategies
  - Randomized VND (baseline)
    - random N. permutation
  - Best VND (optimal classifier without errors)
    - evaluate all N. and always select the most improving one
  - Probabilistic BVND (sub-optimal classifier)
    - evaluate all N. and roulette-wheel selection with probability proportional to the improvement

The value of a neighborhood exploration
A preliminary analysis

- We computed the average value of a neighborhood exploration

Strategy	Total improvement	# applications	Ratio	Appl. Savings wrt RVND
RVND	7,5 * 10^5	3,6*10^7	0,02%	--
BVND	8,2 * 10^5	2,1 * 10^6	0,22%	1600%
PBVND	8,1 * 10^5	3,1 * 10^6	0,21%	1000%

Values are averaged across #runs=10 and all instances of X dataset

- Ratio express what is the average improvement of any neighborhood on any solution
- BVND and PVND can identify local optima and thus stop prematurely the VND

Definition of the training set

- We randomly selected 60 out of 100 instances from the X dataset introduced by Uchoa et al (2014)
- Generate a set of solutions
  - starting from Clarke&Wright '64 solution S
  - use an ILS-like framework on the current S
    - applying a single descent with each one of the 12 neighborhoods
    - select the best improving operator and the corresponding solution S'
    - iterate to a local optimum and possibly update S
    - shaking on the globally best solution S
- This way we collected a set of 22,113,348 solutions together with the best improving operator

Definition of the training set

- for each solution we store the 18 f.p. features proposed by Arnold&Sorensen + class (best improving operator)
- class = shaking if solution is a LO for all N.

Instance features

1. Number of customers
2. Number of routes
3. Degree of capacity utilization
4. Average distance between each pair of customers
5. Standard deviation of the pairwise distance between customers
6. Average distance from customers to the depot
7. Standard deviation of the distance from customers to the depot
8. Standard deviation of the radians of customers towards the depot

Solution features

1. Average number of intersections per customers
2. Longest distance between two connected customers, per route
3. Average distance between depot to directly-connected customers
4. Average distance between routes (their centers of gravity)
5. Average width per route
6. Average span in radian per route
7. Average compactness per route, measured by width
8. Average compactness per route, measured by radian
9. Average depth per route
10. Standard deviation of the number of customers per route

Classifier based on Artificial Neural Network

- We trained a fully connected neural network with 4 total layers
  - input layer with 18 input units
  - hidden layer with 64 hidden units and relu activation function
  - hidden layer with 32 hidden units and relu activation function
  - output layer with 13 output units and softmax activation function: 12 local search operator and 1 shaking operator (= Local Optimum)
- Training using the categorical crossentropy loss function (stardard choice for multiclass classification with neural networks) and lasted 10 epochs.
  - Accuracy results were around 40% (most probably related to the features we use).
  - More epochs did not allow to obtain better accuracy results.

Use within optimization

- VND with 12 neighborhoods where order is

a) Static (statistics-based):
   - N. are sorted according to their success in the training set

b) Random

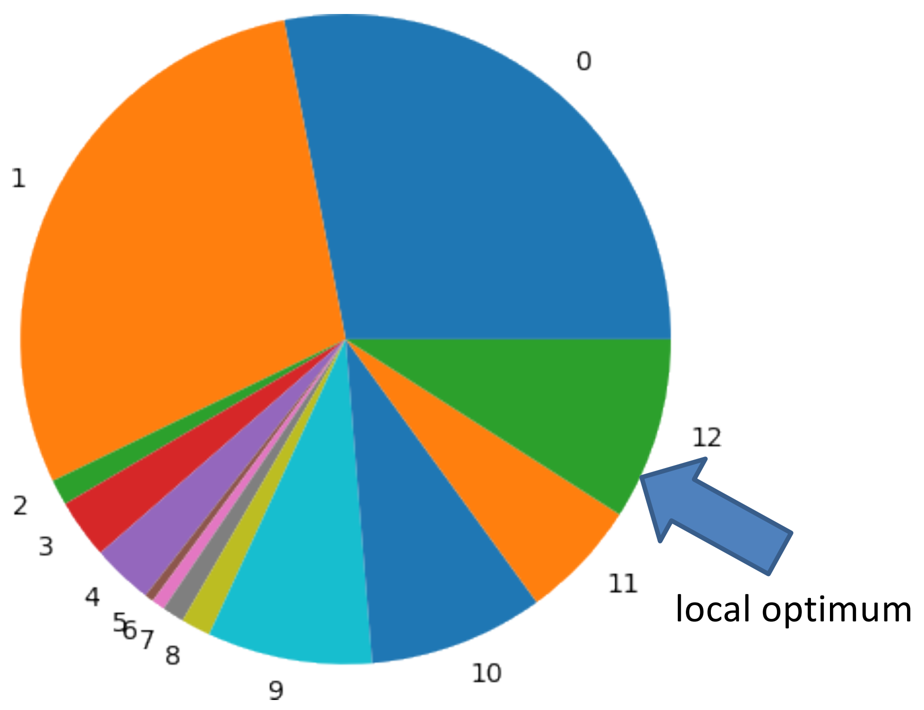
c) ML-based:
   - The trained neural network is used to predict, given a solution, the probability of each neighborhood to be the most improving one on that solution.
   - Neighborhoods are sorted according to their probability in a decreasing order and used in that order

Experiments

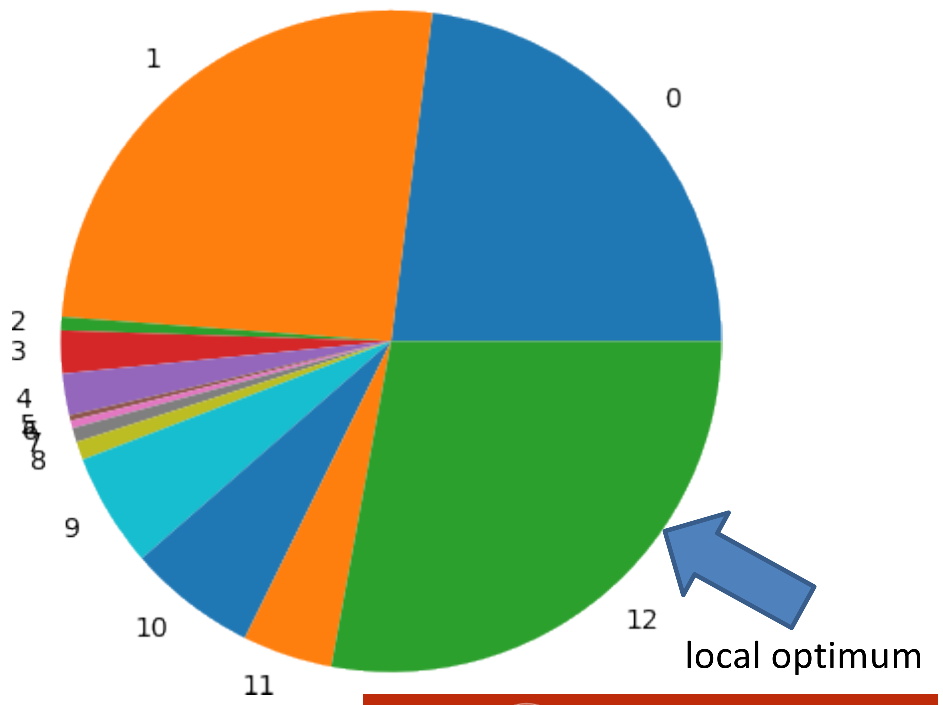
- Goal: compare the effectiveness of the three different approaches.
- To have a fair comparison, we evaluate the number of attempts to find an improving solution/operator (and the quality of such improvement) on the same set of starting solutions.
- solution generator (to create 1,000 solutions per test instance)
  - initialized with C&W solution,
  - subsequent solutions are obtained with an ILS based on the same neighborhoods executed in a RVND fashion and run for a number of iterations randomly chosen between 1 and 10.
  - The final solution that will be returned is shaken and has a probability of 0.5^h to be a local optimum for h operators.
  - We used the same generator both in experiment 1 and 2.

Best operator ranking

impr. threshold = 0



impr. threshold = 0.001



Experiment #1

- Given 40% solutions of the test set
- for each instance we generated 1000 solutions
- we compared the 3 VNDs and computed
  - how many attempts were necessary to find an improving N.
  - what was the improvement of the first improving N. in terms of % gap.

VND ordering	# Attempts	% savings	% Gap
Static	1.98	-	2.88
Random	2.39	+21%	2.58
NN-based (Thr=0)	1.44	-27%	2.88
NN-based (Thr=0.001)	1.37	-31%	2.65

Experiment #2

- Given 40% solutions of the test set
- we compared the 3 VNDs and computed
  - how many neighborhoods were explored to reach a local optimum (in a VND using the different neighborhoods)
  - what was the improvement in terms of % gap.

VND ordering	# N. explored	% savings	% Gap
Static	27.79	-	3.70
Random	62.84	+126%	3.62
NN-based (Thr=0)	18.76	-32%	3.47
NN-based (Thr=0.001)	11.62	-59%	3.16

Attempt 1: Conclusions

- ML may enable consistent time savings in multi-neighborhood local search while preserving solution quality
- Very promising results obtained

… However:

- we performed a preliminary "In field" testing within a high-quality ILS
- Current features extraction is $O(n^2)$ ➔ the associated overhead is not compensated by the potential saving of NN-based VND

Attempt 1: future work

- incremental feature evaluation or use just a selection of features (following Lucas et al. findings)
- better features (linear-time extractable, see later) ?
- compute NN-based only every H iterations and then keep it fixed
- similar testing on the shaking step which is of crucial importance for ILS

Intermezzo:

Aggregate Bounding

Joint work with L. Accorsi

Can we estimate the value of the Optimal Solution?

- To prematurely stop the search in heuristic algorithms once we are close enough
- In (heuristic) B&B algorithms to cut unpromising paths?

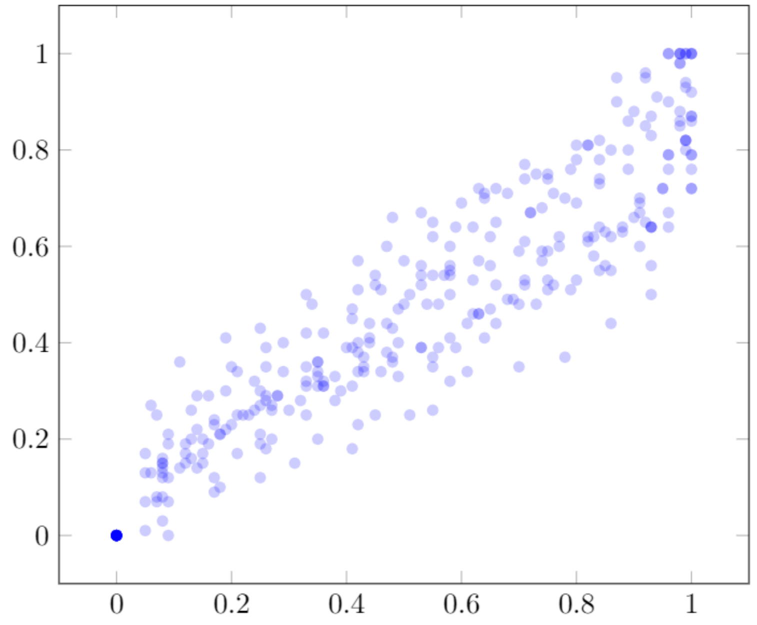
The ML way

- Train a neural network to predict the value of the optimal solution of a given instance
  - A one-shot task
  - Extremely difficult!

- What we try to do instead:
- Train a neural network to predict the gap of a solution from the optimal solution
  - We can easily retrieve the value of the optimal solution from the gap and the solution cost
  - For each instance we can have a lot of predictions
  - Aggregate them and hope they are going towards the right direction!
  - A wrong prediction will harm but less than in the first scenario

Training

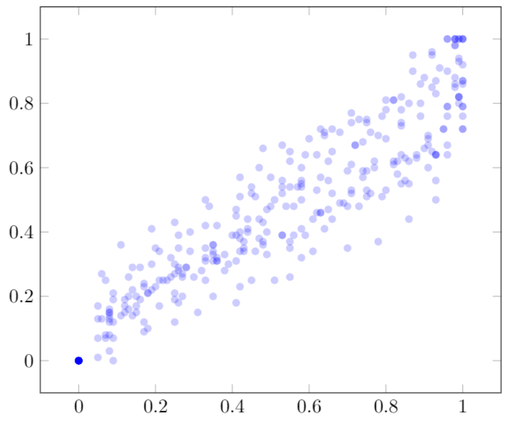
- Train a Convolutional Neural Network on a "top secret" 2D CVRP (linearly extractable) solution representations



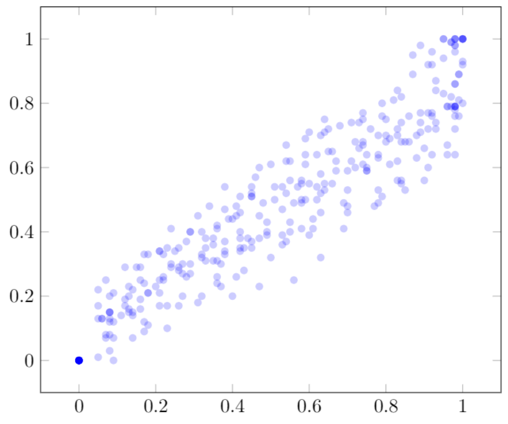
- It's somehow a "picture" of a solution
- NN are succesfully used in image recognition
- are pictures of good and bad solutions "different"?

Visualizing solutions with different qualities

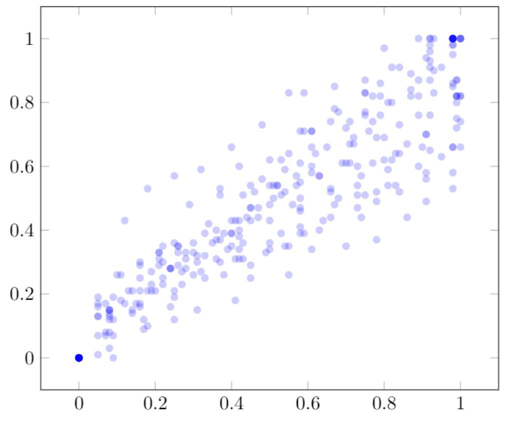
- Images tend to became larger on worse solutions because containing worse routes
- However, it is not always that clear for a human eye!



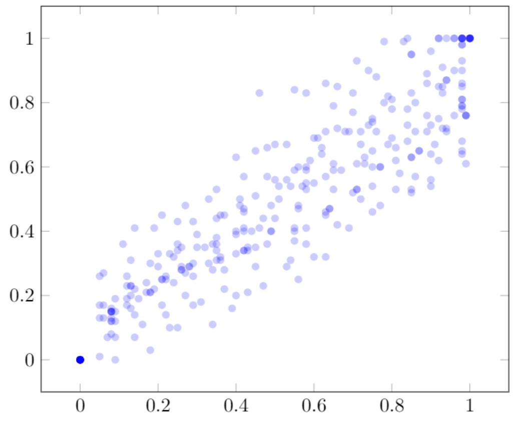
Gap = 0.01%



Gap = 3%



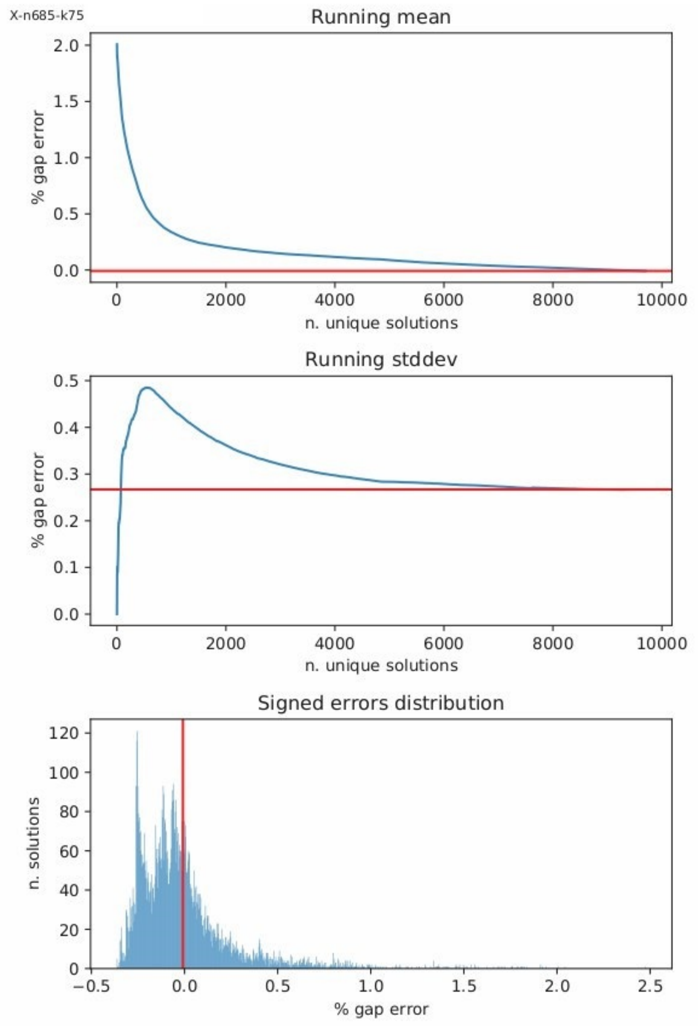
Gap = 6%



Gap = 8%

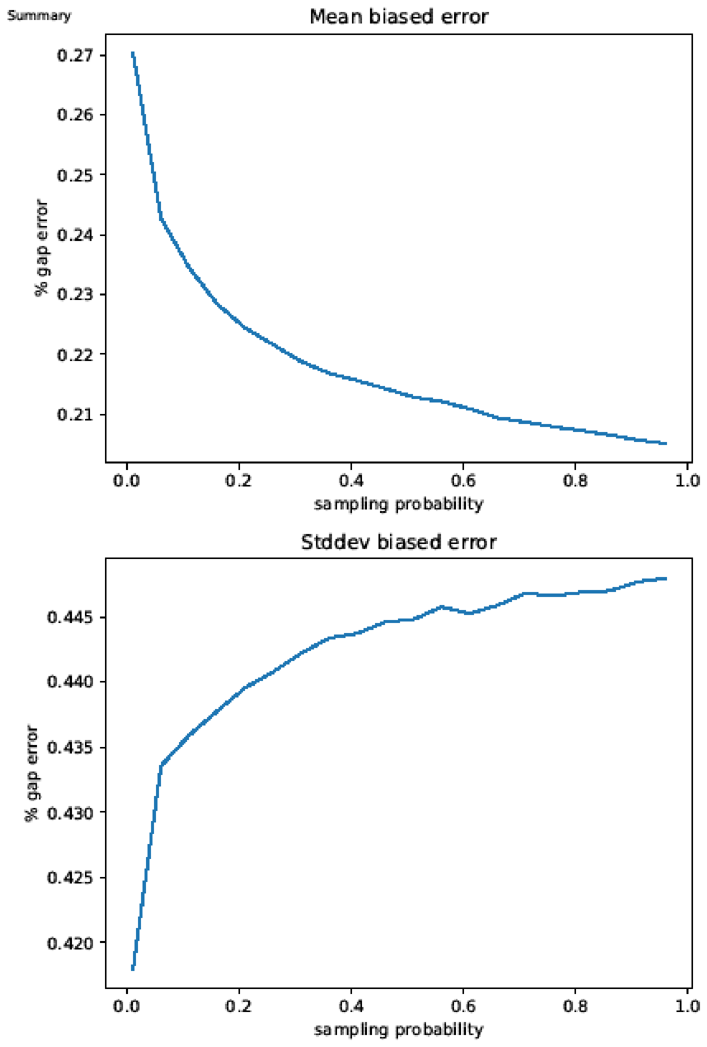
Aggregate bounding on a single instance

- Train a NN to estimate, given a solution s, its gap wrt the optimal/BK solution
- How to use that on an unknown CVRP instance:
  1. Sample some solutions during the search
  2. For each sampled solution s compute an estimated value opt(s)
  3. Aggregate the estimations to approximate the optimal solution value for the instance during the search



Preliminary testing on a test set

- Sample local optima during algorithm evolution
  - Features computation and prediction has a cost
  - Try to minimize their impact
- Incrementally compute the mean optimal value
  - Mean error of about 0.20!
  - High std dev



Conclusions on aggregate bounding

- preliminary experiments on the new features are promising
- further testing on their use for other purposes
- gap estimation may be useful for early stop of a heuristic

Attempt #2:

Data-based Guided Optimization for the CVRP

joint work with L. Accorsi

Goal

- Can we study characteristics of high-quality solutions to design better algorithms?

- The Machine Learning/Data Mining Approach:
  - Let a model identify recurring patterns found in high-quality solutions

Not as easy as a Standard ML Task

- We are not interested in maximizing a ML metric (e.g., accuracy) but in guiding an algorithm possibly composed of several complex interconnected components
- Any change in an existing algorithm does not necessarily produces better final outcome
  - Particularly in algorithms already producing high-quality results
- Analysis of what a change implies on the overall algorithm may not be trivial

Challenges

1. The power of randomization (probability)
   - Relying on randomness typically generalizes much better to different populations than any deterministic choice
2. High baseline
   - Improving results generated by state-of-the-art algorithms is difficult
   - Years of high-quality human knowledge vs biased dataset of raw data
3. Prediction overhead
   - Additional computation is required for features extraction and inference
   - Guided decisions must substantially improve quality (difficult because of 2) or cut computing time to be useful in practice

Arnold & Sorensen Solution’s Characterization

- Featurize a solution by using aggregate values
  - Average routes width
  - Average routes depth
  - …
- For randomly generated instances with 20-100 customers classify solutions as near-optimal or sub-optimal (2%-4%)
- Interesting accuracy results of up to 90% obtained with a decision tree on a cluster of similar instances with 70-100 customers to classify near-optimal solutions from sub-optimal (4%)
- Accuracy decreases when classifying near-optimal solutions from slightly better sub-optimal solutions (2%)
  - They probably share a lot of characteristics!

Solutions are Complex Objects

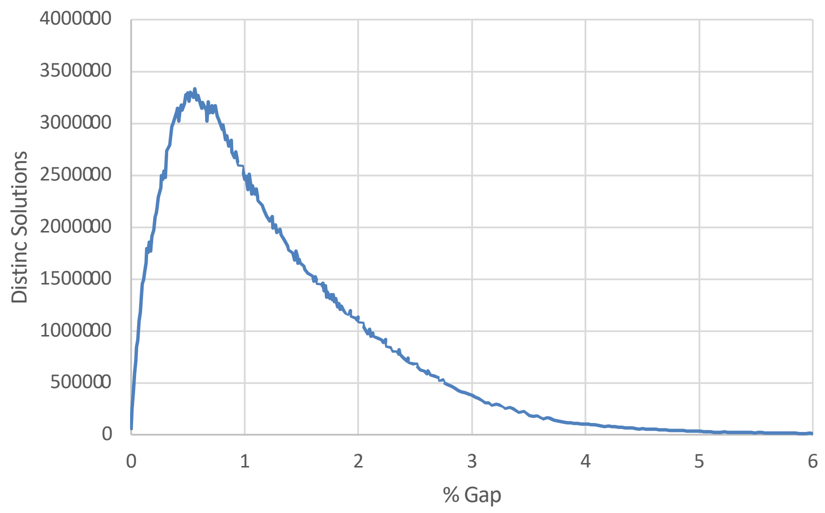
- Can you discriminate a near-optimal solution from a sub-optimal one by using aggregate characterizations?
  - Everything gets extremely diluted
  - Think of an optimal solution having 100 routes for a CVRP instance in which 1 or 2 route(s) are changed
    - Quality can degrade at will
    - Aggregate features would not significantly change
- Overcoming the near-optimal and sub-optimal solution classification
  - If not trivially bad, a sub-optimal solution will just be such because defined by the wrong set of routes

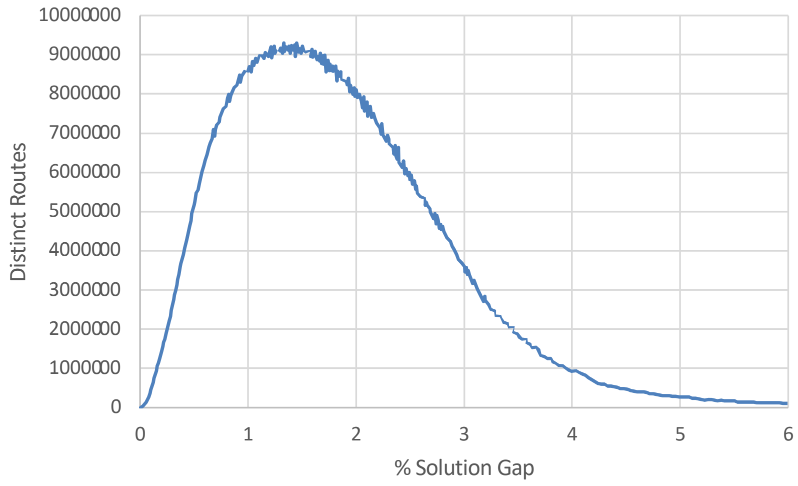
Solutions as Set of Routes

- Work instead with routes situated in their context
- Solution features as the set of features of individual routes normalized according to the solution itself
  - Load ratio
  - Route cost contribution to the solution
  - Mean distance between interconnected customers
  - ...

Full Solution Dataset

- From the X instances we computed 500 millions distinct solutions describing a broad range of qualities composed of 450 millions distinct routes





About 50% distinct solutions have gap <= 1%

Sampled Solution Dataset

- 10 million high-quality (<=1%) distinct solutions
- 10 million low-quality (>1%) distinct solutions
- Total number of about 600 million not necessarily distinct routes classified as
  - Shared: when occurring in solutions both in high-quality and low-quality solutions
  - High-quality: when occurring in high-quality solutions only
  - Low-quality: when occurring in low-quality solutions only
- Note that labels strictly depend on the sampled dataset

Shared, High-quality and Low-quality routes

- About 80% of the routes are shared
- Just 10% of them are peculiar of high-quality and low-quality solutions

- Is an aggregate solution characterization really meaningful?

Sampled Dataset Analysis

- About 94% low-quality solutions contain at least one low-quality route
- About 6% low-quality solutions contain only shared routes
  - Those are solutions very close to the hard threshold of 1% gap
  - Average gap 1.24 ± 0.24
- About 49% high-quality solutions are entirely made of shared routes
  - Possibly be related to the routes' distribution
  - There are less ways of composing high-quality solutions and a lot more to define low-quality ones

Sampled Dataset Analysis

- About 94% low-quality solutions contain at least one low-quality route
- Idea: get rid of low-quality routes!
- Applications:
  - Guide a heuristic optimization processes
  - Skip low-quality routes in SP-based matheuristics

Sampled Route Dataset
(out of the 600 million non distinct routes)

- 1 million low-quality routes
- 500 thousand high-quality routes
- 250 thousand shared routes from low-quality solutions
- 250 thousand shared routes from high-quality solutions

- A binary classification
  - Interesting routes
  - Not interesting routes

Use-oriented ML model selection

- Model purpose is to be embedded into state-of-the-art algorithm
  - Fast features extraction from solution
    - Simple handcrafted features linearly extractable
  - Fast prediction
- Let's use the simplest model having good accuracy results
  - Decision tree with at most 5 levels
  - Has accuracy slightly worse than more complex models but it has a very fast prediction time
  - 10-fold cross validation: mean accuracy 77%

Testing the idea

- To have a convincing validation of the idea we must plug iit into a state-of-the-art method …
- … improving an average algorithm is not so difficult
- … and see if it allows for:
  - final solution quality improvement, or
  - faster convergence to good solutions, or
  - … at least, some speedup while preserving quality !

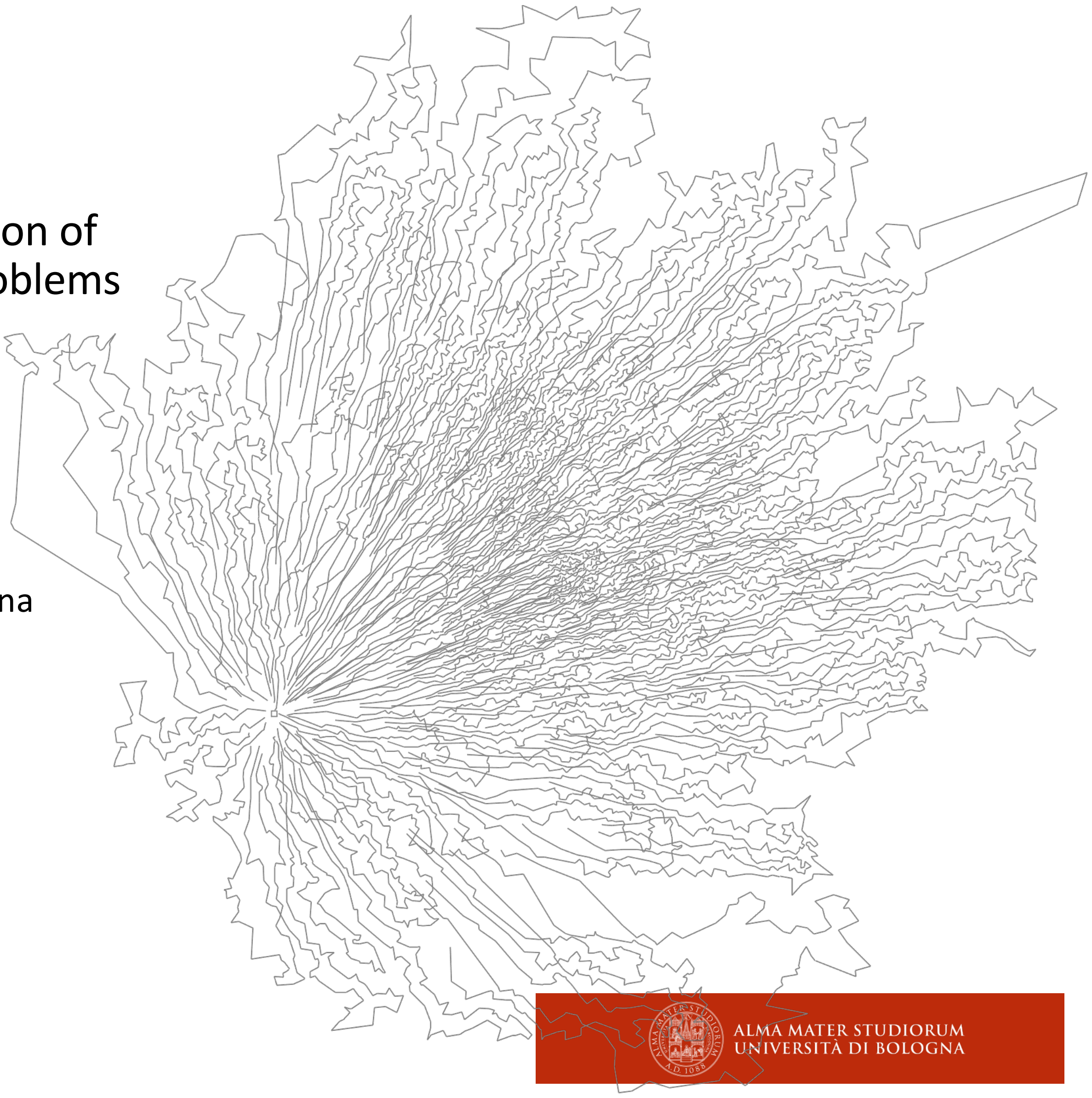
- We used as benchmark FILO, by Accorsi and Vigo 2020 (submitted 🤞)

FILO

A Fast and Scalable Heuristic for the Solution of Large-Scale Capacitated Vehicle Routing Problems

Luca Accorsi[1] and Daniele Vigo[1,2]

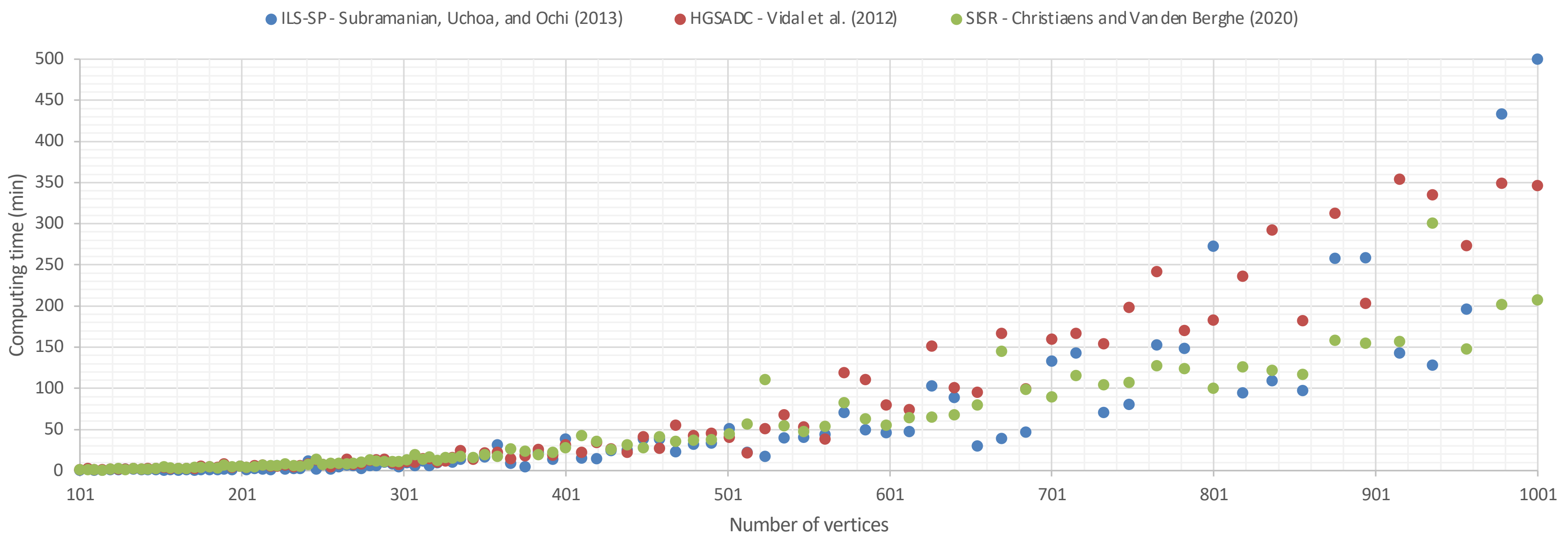
[1] DEI «Guglielmo Marconi», University of Bologna
[2] CIRI ICT, University of Bologna



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

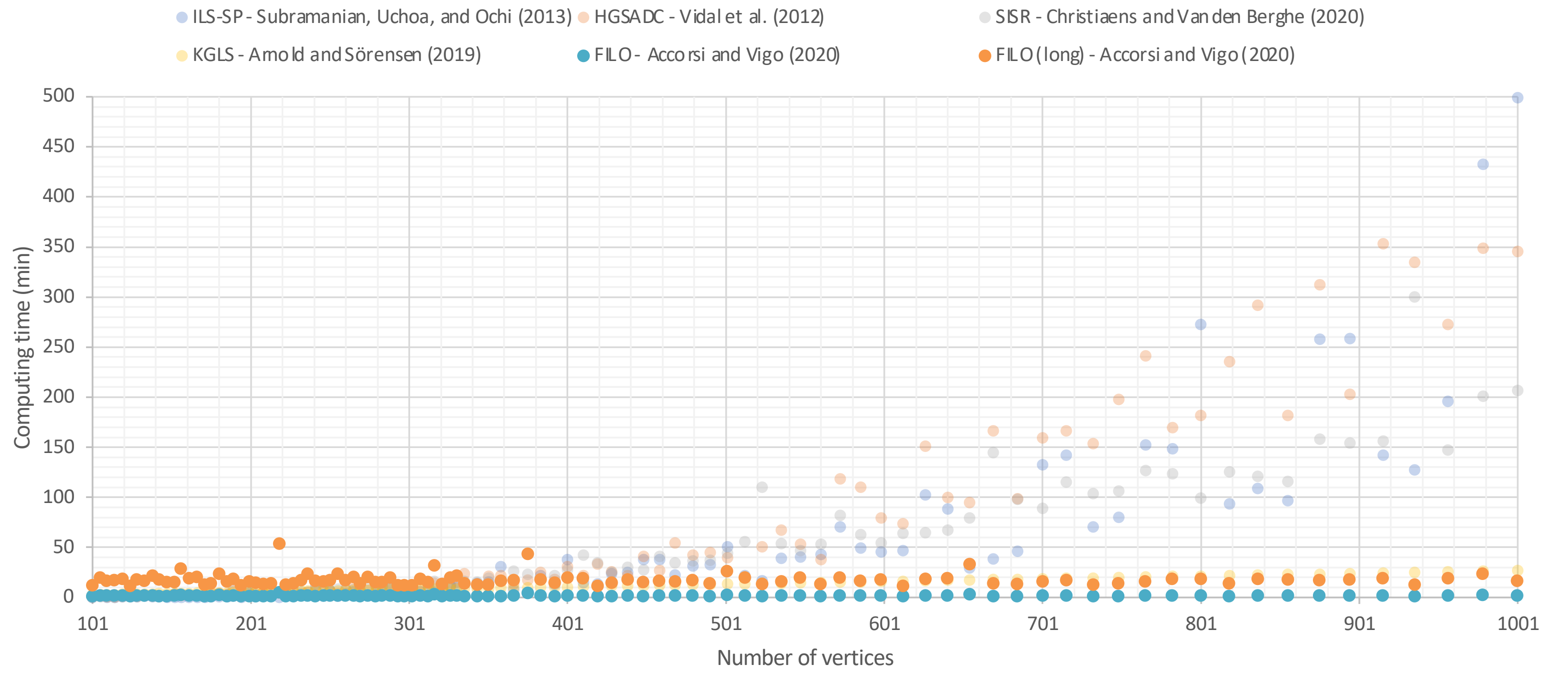
Motivation

- State-of-the-art (heuristic) CVRP algorithms often exhibit a quadratic growth



Goal

- Designing a fast, naturally scalable and effective heuristic

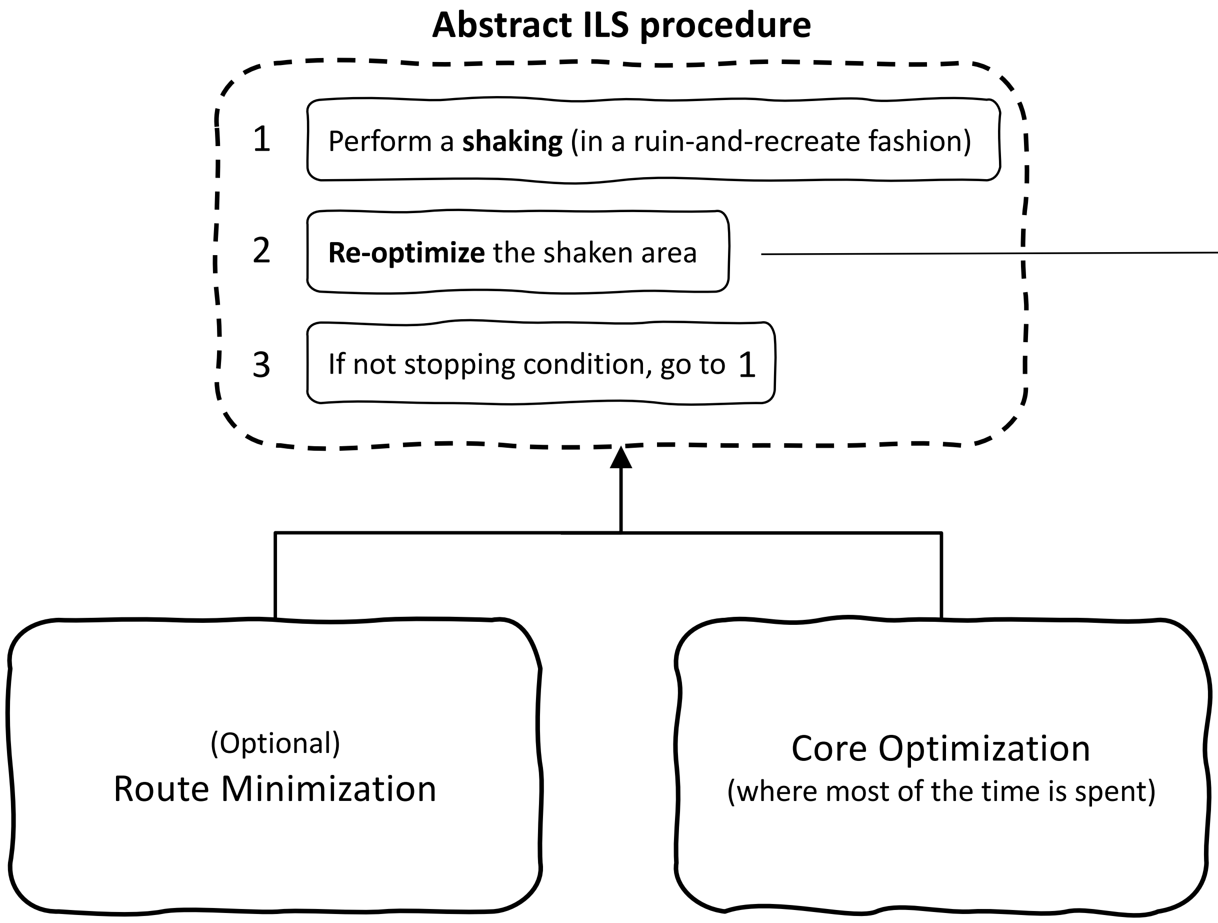


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fast ILS Localized Optimization (FILO) recipe

- ILS-based framework
- Local Search Acceleration Techniques
- Pruning Techniques
- Careful Design
- Careful Implementation
- Careful Parameters Tuning

Improvement procedures


Abstract ILS procedure
1
Perform a **shaking** (in a ruin-and-recreate fashion)
2
Re-optimize the shaken area
3
If not stopping condition, go to 1
(Optional)
Route Minimization
Core Optimization
(where most of the time is spent)

By using a sophisticated **Local Search Engine**

Local search engine

- Several operators explored in a VND fashion
  - Hierarchical Randomized Variable Neighborhood Descent
- Acceleration techniques for neighborhood exploration
  - Static Move Descriptors
- Pruning techniques
  - Granular Neighborhoods and Selective Vertex Caching

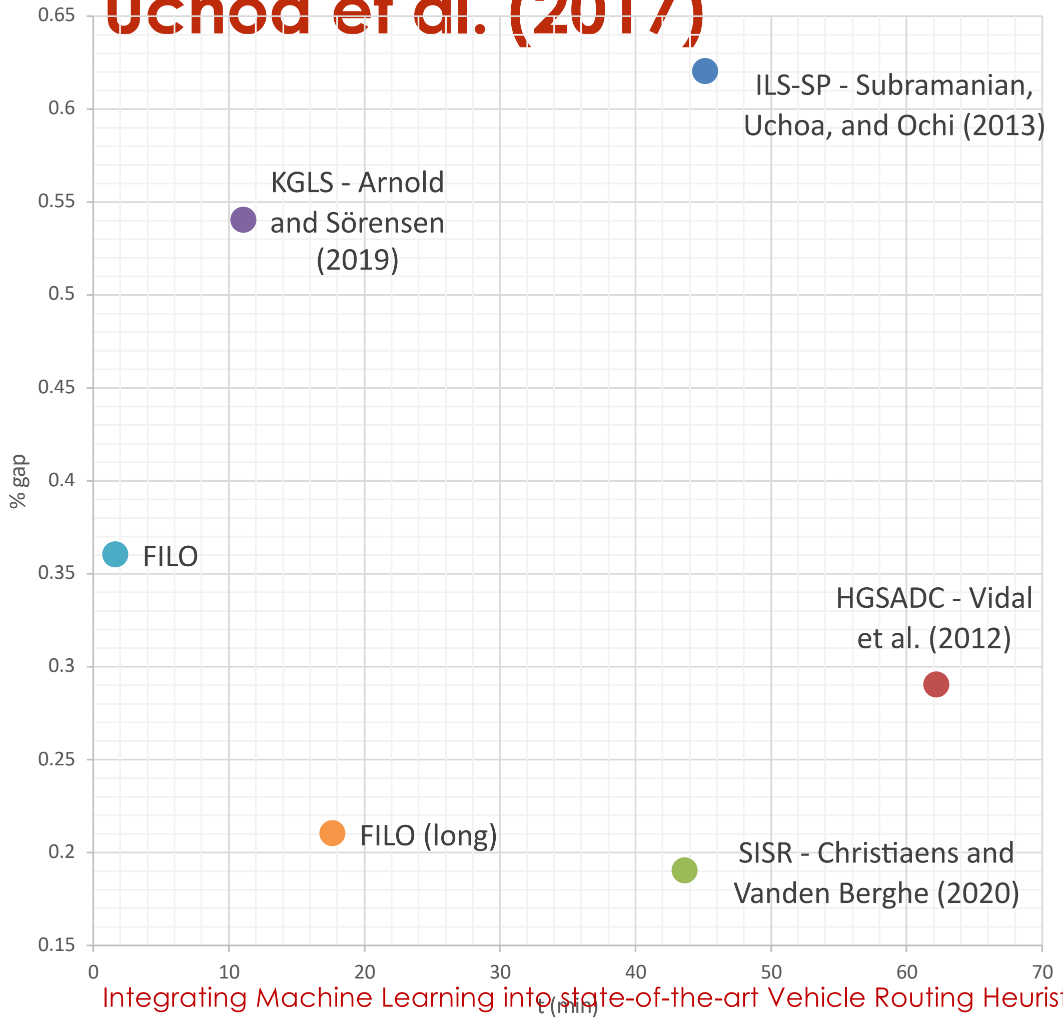
Computational results

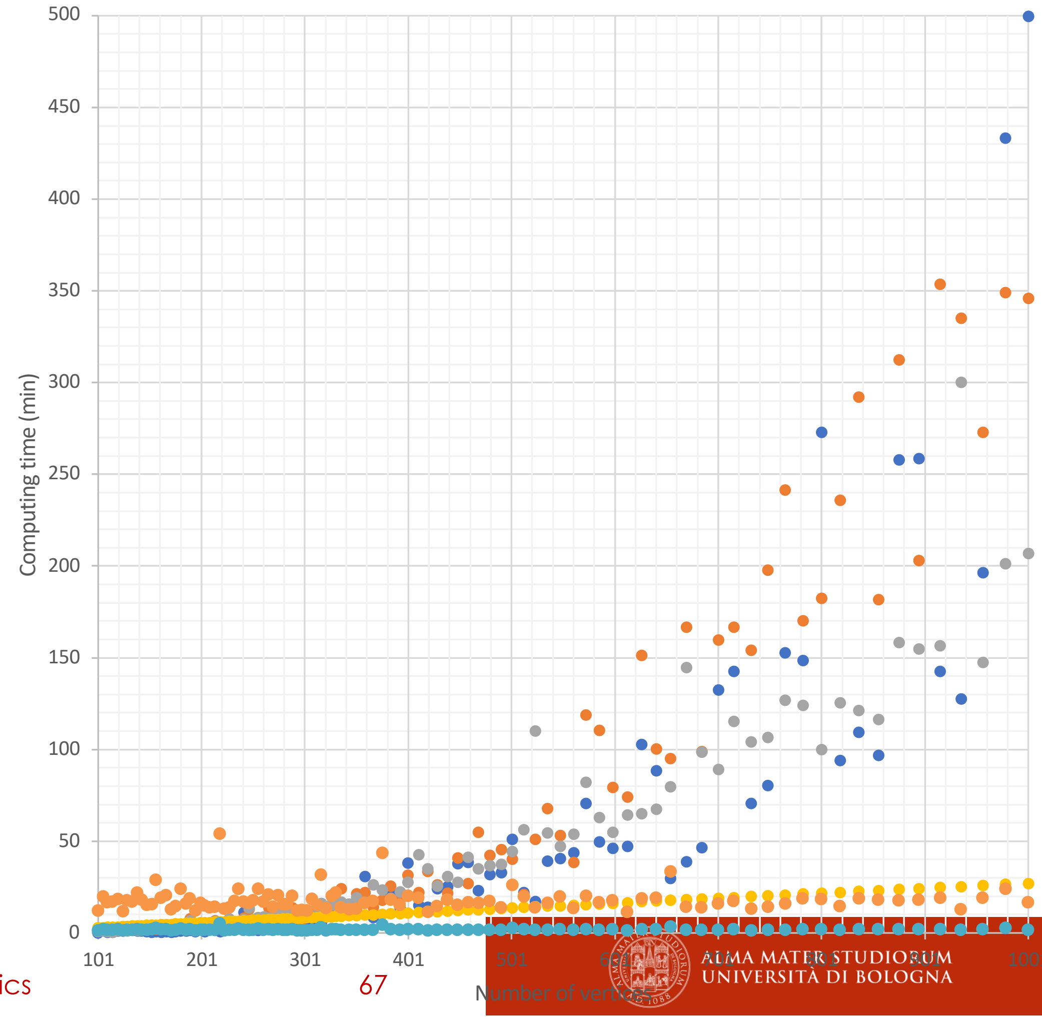
- Two versions of FILO
  - FILO $100K$ core optimization iterations
  - FILO (long) $1M$ core optimization iterations

- On *standard* instances
  - **X** dataset by **Uchoa et al. (2017)**

- On very large-scale instances
  - **B** dataset by **Arnold, Gendreau, and Sörensen (2019)**
  - **K** dataset by **Kytöjoky et al. (2007)**
  - **Z** dataset by **Zachariadis and Kiranoudis (2010)**

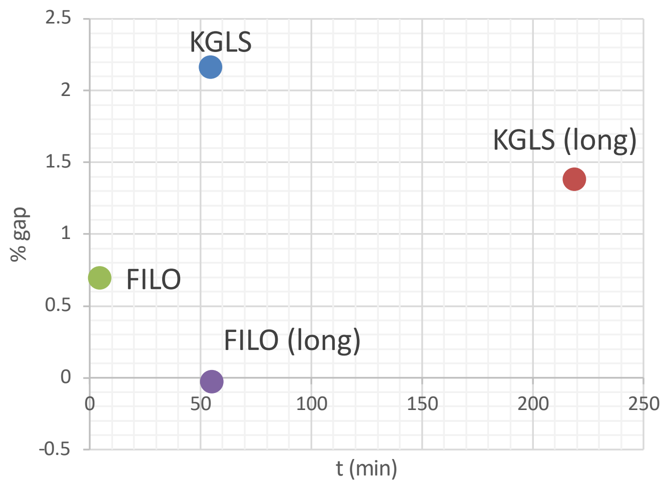
X
Uchoa et al. (2017)



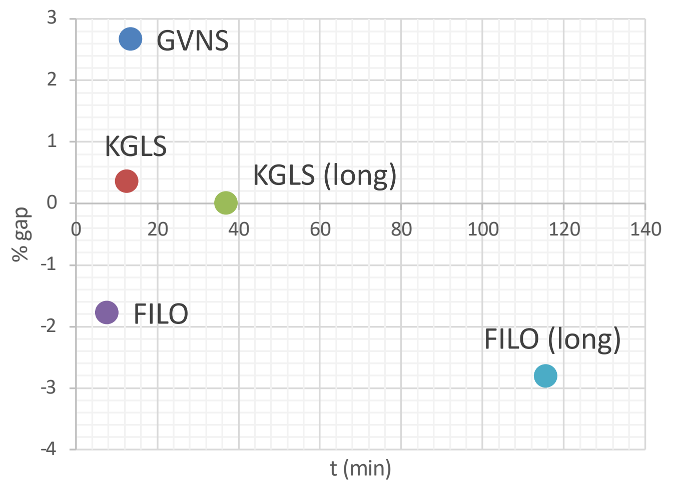


Very large instances

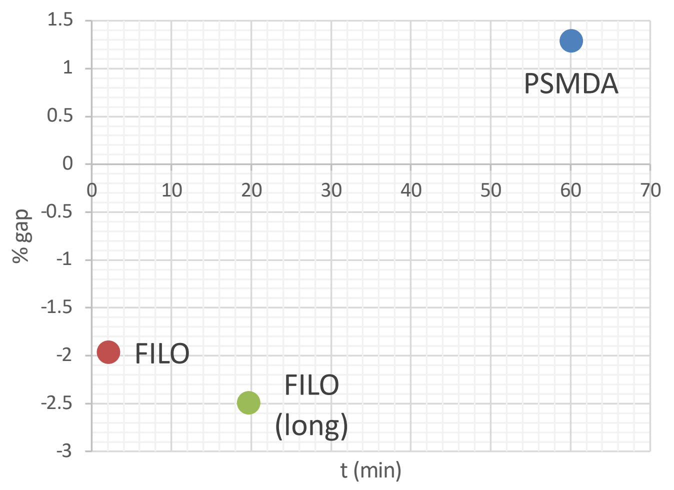
B (3K – 30K)
Arnold, Gendreau, and Sörensen (2019)



K (≈8K – 12K)
Kytöjoky et al. (2007)



Z (3K)
Zachariadis and Kiranoudis (2010)



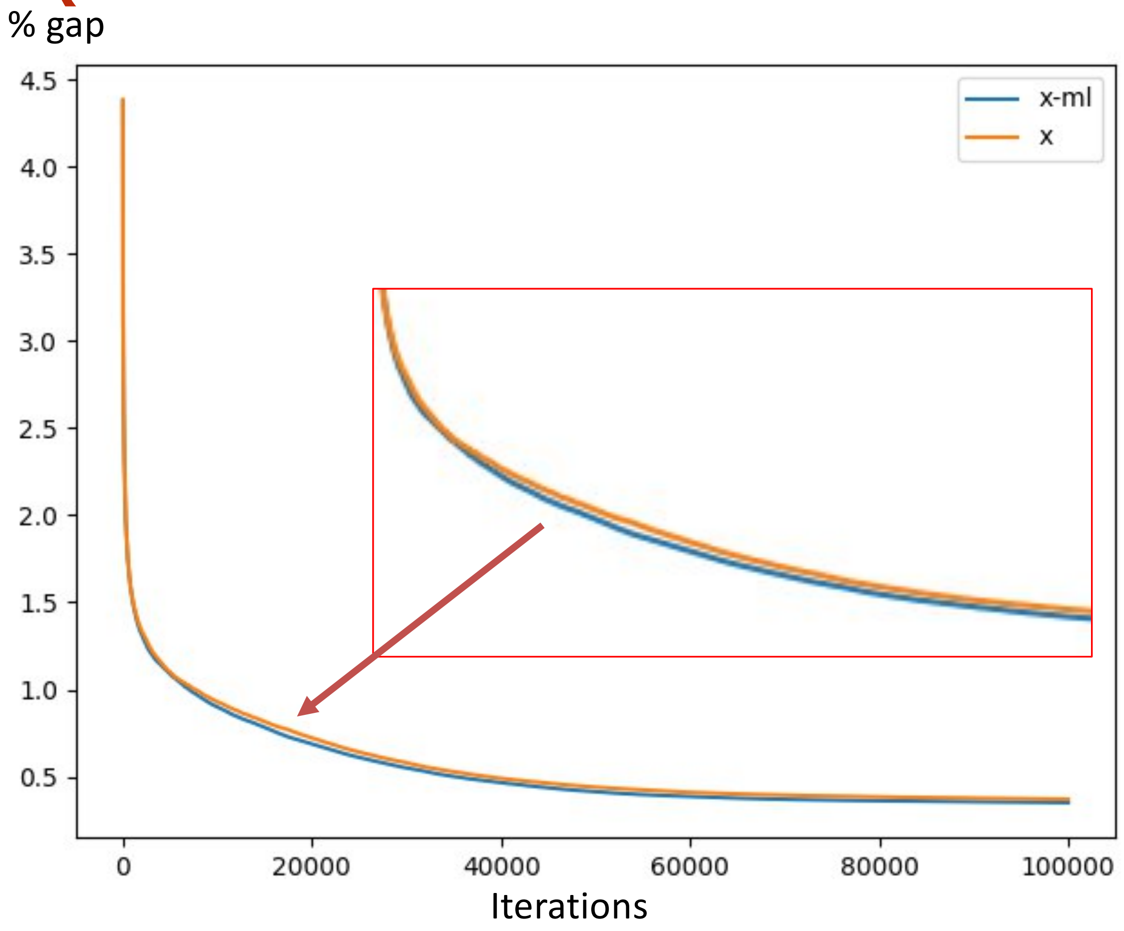
Algorithms

- KGLS, KGLS (long) - Arnold, Gendreau, and Sörensen (2019)
- GVNS - Kytöjoky et al. (2007)
- PSMDA - Zachariadis and Kiranoudis (2010)

Plugging the idea into FILO

- Shaking performed in ruin-and-recreate fashion
  - Select a "seed customer"
  - remove a set of customers belonging to routes "close" to the seed
- Standard: shaking seed is a randomly selected customer
- Guided: bias the seed selection towards customers belonging to low-quality routes

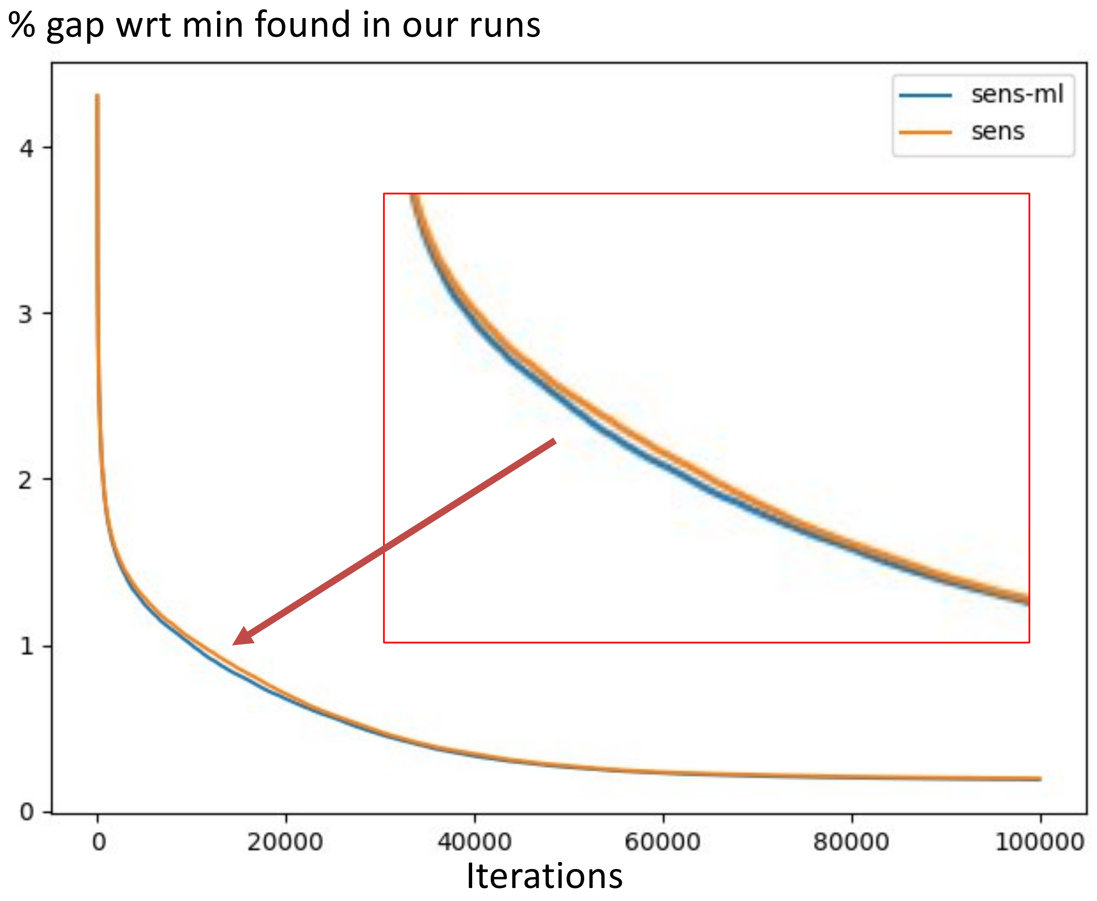
Preliminary Experiments
(on the X instances from which we extracted the training routes)



Standard: 0.36%
Guided: 0.34%
Both in 2.12 minutes!
During the algorithm we do not recompute features if not needed

At least it does not harm!

Preliminary Experiments
(on the 100 largest instances from the Extended Benchmark of Uchoa et al. 2017)



BKS values not available

Using Standard as baseline
Standard: solved in 2.08 minutes
Guided: -0.007% in 2.11 minutes

Not statistically significant results but:

- Better final solutions value in 57 instances out of 100
- Average faster convergence to better solutions

Attempt 2: conclusions

- Route feature analysis has a better explanatory power than aggregate solution features in distinguishing good and bad solutions
- Identifying non-interesting routes may be profitably used in high-quality heuristics
- … and it works ! (or at least it does not harm !)

Attempt 2: future work

- complete and submit the paper as soon as possible !

Overall conclusions

- ML has great potential in the improvement of optimization algorithms
- Feature selection is of paramount importance to obtain meaningful results and keep at bay the computational overhead
- It is important that the computational validation is performed with high quality heuristics

- … there is a lot of work to do !

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Daniele Vigo

Department of Electrical, Electronic and Information Engineering «Guglielmo Marconi»
CIRI-ICT

daniele.vigo@unibo.it

www.unibo.it