Designing a heuristic the modern way

Or: how to solve **very** large vehicle routing problems

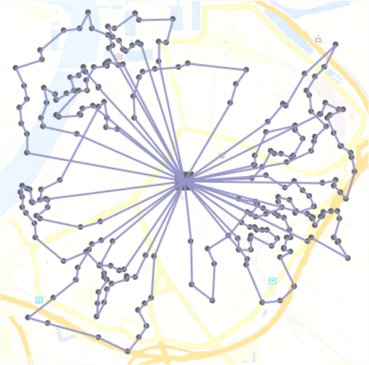
Kenneth Sörensen kenneth.sorensen@uantwerpen.be
Florian Arnold florian.arnold@uantwerpen.be

International Spring School on Integrated Operational Problems - Troyes - 14-16 may 2018

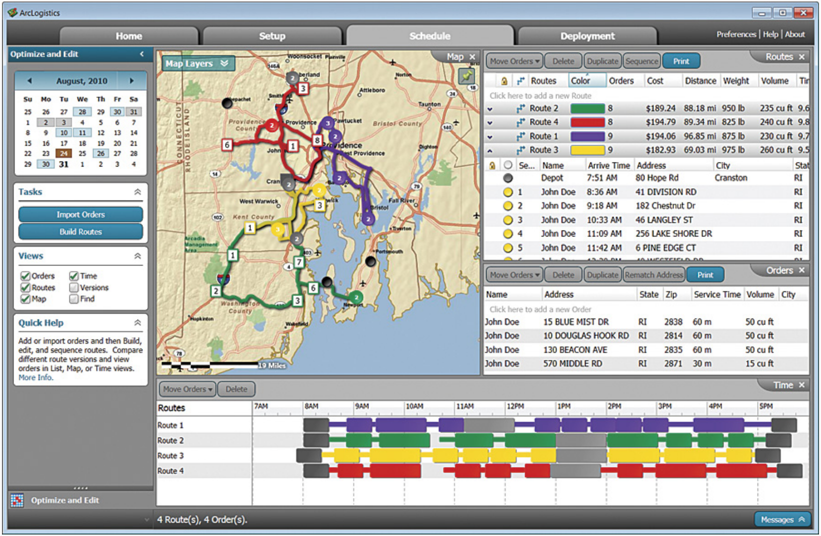
University of Antwerp
Operations Research Group
ANT/OR

The vehicle routing problem

- One of the most studied problems in OR
- Google Scholar:
  - 780.000 entries
  - 20.000 new entries every year
  - 10.000 on heuristics
- Huge practical relevance



Practical relevance



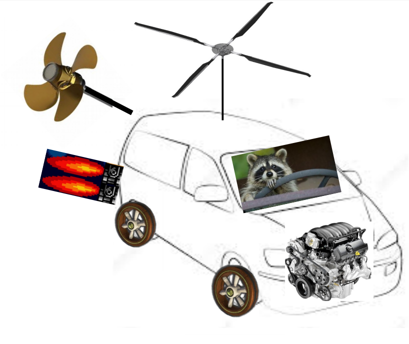
Relevance

- A lot of extensions
  - Time windows
  - Pick up and delivery
  - Arc routing
  - ...
- Integral part of many other problems
  - Location–routing
  - Inventory–routing
  - School bus routing
  - ...
- All rely on effective algorithms for the canonical CVRP

State of the art

- Use as many local search (constructive) operators as possible
- Either VNS or LNS
- Fit in a metaheuristic framework
  - This is your Unique Selling Point
  - But it really does not matter all that much
- Beware of “Frankenstein” algorithms

State of the art



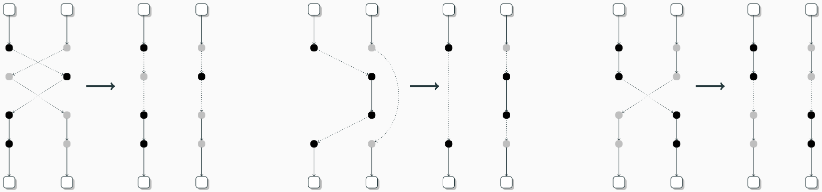
- Use as many local search (constructive) operators as possible
- Either VNS or LNS
- Fit in a metaheuristic framework
  - This is your Unique Selling Point
  - But it really does not matter all that much
- Beware of “Frankenstein” algorithms

Local search for the VRP

Operator	Complexity	Description
2-opt	$O(n^2)$	Swap 2 edges
3-opt	$O(n^3)$	Swap 3 edges
Insert / Relocate	$O(n^2)$	Relocate a customer
Swap	$O(n^2)$	Exchange two customers
Crossover	$O(n^2)$	Exchange route ends
CROSS-exchange	$O(n^4)$	Exchange any two customer sequences

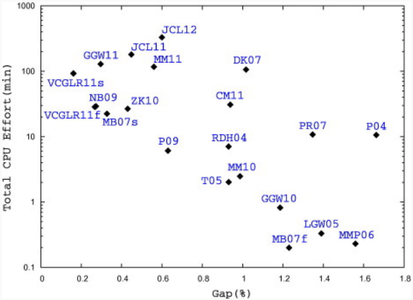
$$Power \sim \frac{1}{Speed}$$

Local search operators

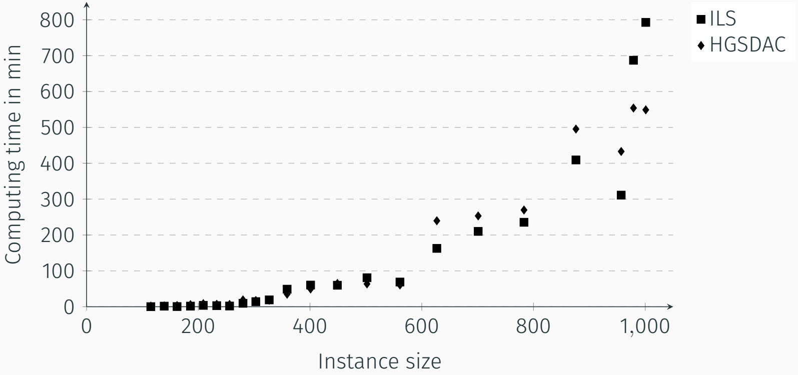


State of the art

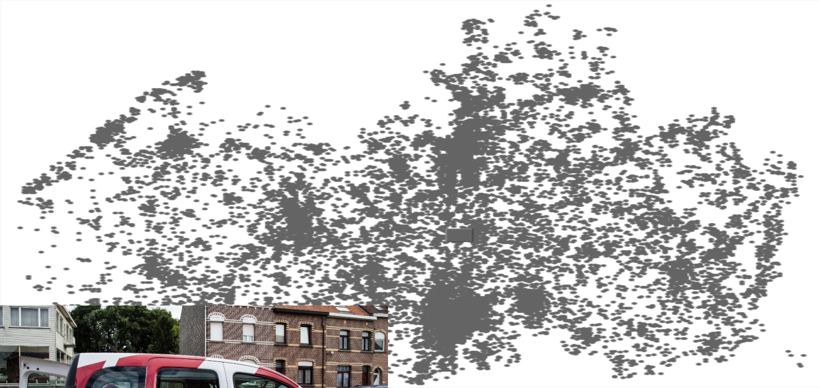
- Many algorithms with more or less equivalent performance
- Stuck at around 1000 customers ("very large scale")
- Larger problems exist and smaller problems should be solved more efficiently
- Can we go further?



Heuristic performance



Extra extra large scale vehicle routing — can we do it?



Some fresh ideas

1. Develop a small set of **powerful**, **complementary** local search operators
2. Learn the **properties of good solutions** and use this knowledge
3. Focus the power of the heuristic to make it **efficient**

Idea #1

A (simple yet efficient) heuristic based on complementary local search operators

A fresh look at local search

- Two ways to solve VRPs in the literature
  - “Multiple neighborhood search”
  - Large Neighborhood Search (i.e., “multiple constructive heuristics”)
- General sentiment: “it does not hurt to try”
  (i.e., implement *a lot* of operators)

However

- There is an **overhead** for every operator
- Many operators have **overlapping domains**
- **Powerful** operators tend to be **slow**
  (complexity based on searching the entire operator space)

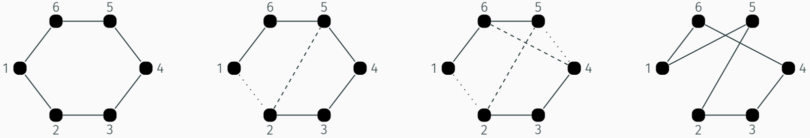
Our heuristic: complementary local search operators

- One route: Lin Kernighan
- Two routes: CROSS exchange
- Many routes: Relocation Chain

Careful

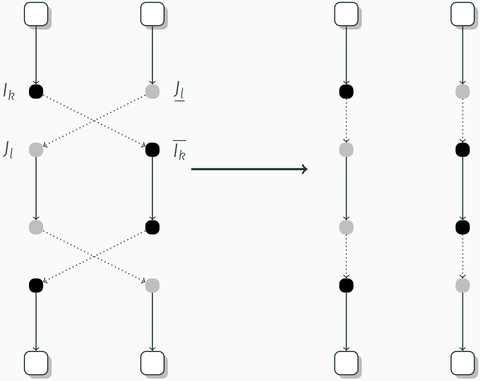
- Each operator is very powerful
- Each operator is very complex

One route: Lin Kernighan



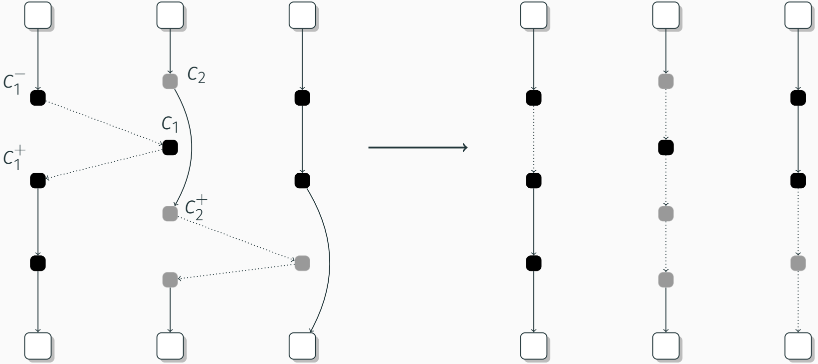
- Solves a TSP by edge exchanges (n-opt)
- Edge exchanges best restricted to nearest neighbors
- Routes in VRPs are generally smaller
  - We can try more neighbors
  - We can do steepest descent (instead of first-improving)

Two routes: CROSS exchange



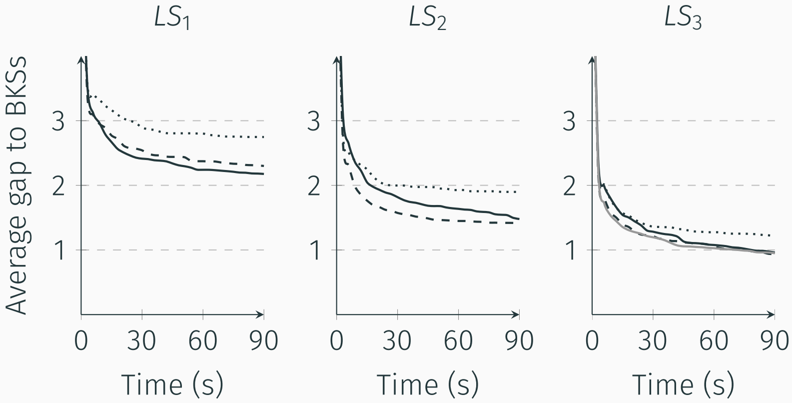
- Exchanges two sub-routes
- Complexity $O(n^4)$
- Length of substrings best restricted

Three routes: Relocation chain



- Chain of relocations
- Depth of chain best restricted

Performance of neighborhoods



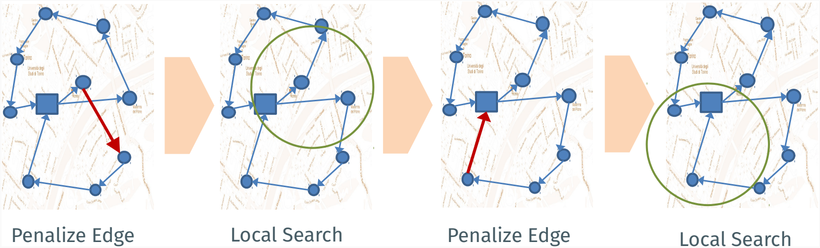
	Intra-route LS	Inter-route LS
LS_1	2-opt	relocate, swap, Or-exchange
LS_2	LK	CE
LS_3	LK	CE, RC

Metaheuristic framework: guided local search

- Idea: penalize bad edges

$$c^g(i,j) = c(i,j) + \lambda p(i,j)L$$

- Alternate penalization and local search



- Question: what is a “bad” edge?

Idea #2

Learn the properties of good solutions

What makes a solution good?

+0.14%
"near-optimal"

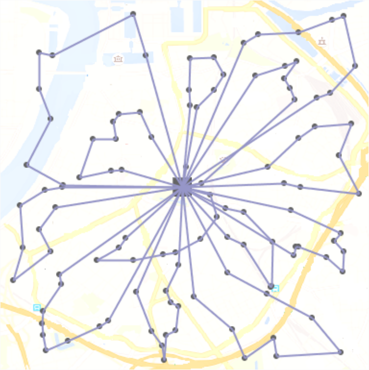
+2.03%
"non-optimal"

Question

Is there a relationship between solution characteristics, instance characteristics, and solution quality?

What makes a solution good?

+0.14%
“near-optimal”



+2.03%
“non-optimal”

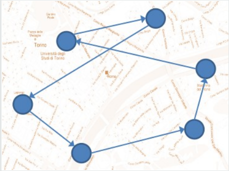
Question

Can we tell whether a solution is good or not without looking at the objective function value?

What makes a solution good?

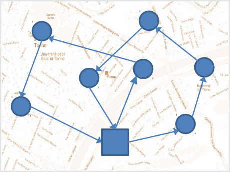
Problem-specific information is rare ($\neq$ intuition)

TSP



?

VRP



Quotes

- “[...] make use of any problem-specific information that you have.”
- “[...] the perturbation can incorporate as much problem-specific information as the developer is willing to put into it.”
- “Exploiting problem-specific knowledge [...] are key ingredients for leading optimization algorithms.”

Methodology

1 Random instance

2 **Near**-optimal solution (**O**)

Non-optimal solution (**N**)

3

intersections	9
average width	5.4
...	...

intersections	12
average width	6.3
...	...

4 Train and predict **O** versus **N**

5 Extract rules

Table 1: Instance parameters for the different instance classes

Class	Customers	Depot	Demand	Routes
1	20-50	Center	[1,1]	3-6
2	20-50	Center	[1,10]	3-6
3	20-50	Edge	[1,1]	3-6
4	20-50	Edge	[1,10]	3-6
5	70-100	Center	[1,1]	6-10
6	70-100	Center	[1,10]	6-10
7	70-100	Edge	[1,1]	6-10
8	70-100	Edge	[1,10]	6-10

Solution generation

“Near optimal”	“Non optimal”
Own heuristic (see before)	H1: weak version of own heuristic
	H2: Modified Clarke-Wright
Very powerful	Rather weak
0.20% gap on Augerat A	2% and 4% gap

Intermezzo

RESEARCH ARTICLE

ScienceAsia 38 (2012): **307–318**

doi: 10.2306/scienceasia1513-1874.2012.38.307

An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem

Tantikorn Pichpibul[a], **Ruengsak Kawtummachai**[b,*]

[a] School of Manufacturing Systems and Mechanical Engineering, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani 12121 Thailand

[b] Faculty of Business Administration, Panyapiwat Institute of Management, Chaengwattana Road, Nonthaburi 11120 Thailand

*Corresponding author, e-mail: ruengsakkaw@pim.ac.th

Received 1 Aug 2011
Accepted 20 Jun 2012

ABSTRACT: In this paper, we have proposed an algorithm that has been improved from the classical Clarke and Wright savings algorithm (CW) to solve the capacitated vehicle routing problem. The main concept of our proposed algorithm is to hybridize the CW with tournament and roulette wheel selections to determine a new and efficient algorithm. The objective is to find the feasible solutions (or routes) to minimize travelling distances and number of routes. We have tested the proposed algorithm with 84 problem instances and the numerical results indicate that our algorithm outperforms CW and the optimal solution is obtained in 81% of all tested instances (68 out of 84). The average deviation between our solution and the optimal one is always very low (0.14%).

KEYWORDS: heuristics, optimization, tournament selection, roulette wheel selection

INTRODUCTION

The capacitated vehicle routing problem (CVRP) was initially introduced by Dantzig and Ramser[1] in their article on a truck dispatching problem and, consequently became one of the most important and widely

branch-and-bound algorithm[6], a branch-and-cut algorithm[7–9], and a branch-and-cut-and-price algorithm[10]. In these algorithms, CVRP instances involving more than 100 customers can rarely be solved to optimality due to a huge amount of computation time. Second, a heuristic algorithm, which is an algorithm that

Clarke–Wright algorithm for the VRP

- Create a separate route per customer
- Connect routes according to the largest possible *savings*
- Repeat while routes can be connected

Saving

$s(i,j) = d(D,i) + d(D,j) - d(i,j)$

"Improved" Clarke and Wright

Add some randomization ("GRASP") $\rightarrow$ *unbelievably* effective

Intermezzo

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 00 (2017) 1–10
DOI: 10.1111/itor.12443

A critical analysis of the "improved Clarke and Wright savings algorithm"

Kenneth Sörensen, Florian Arnold and Daniel Palhazi Cuervo

ANT/OR – Operations Research Group, Department of Engineering Management, University of Antwerp, Belgium
E-mail: kenneth.sorensen@uantwerpen.be [Sörensen]; florian.arnold@uantwerpen.be [Arnold]; daniel.palhazicuervo@uantwerpen.be [Palhazi Cuervo]

Received 16 February 2017; accepted 23 June 2017

Abstract

In their paper "An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem," published in *ScienceAsia* (38, 3, 307–318, 2012), Pichpibul and Kawtummachai developed a simple stochastic extension of the well-known Clarke and Wright savings heuristic for the capacitated vehicle routing problem. Notwithstanding the simplicity of the heuristic, which they call the "improved Clarke and Wright savings algorithm" (ICW), the reported results are among the best heuristics ever developed for this problem. Through a careful reimplementation, we demonstrate that the results published in the paper could not have been produced by the ICW heuristic. Studying the reasons how this paper could have passed the peer review process to be published in an ISI-ranked journal, we have to conclude that the necessary conditions for a thorough examination of a typical paper in the field of optimization are generally lacking. We investigate how this can be improved and come to the conclusion that disclosing source code to reviewers should become a

Methodology

1 Random instance

2 **Near**-optimal solution (**O**)

Non-optimal solution (**N**)

3

intersections	9
average width	5.4
...	...

intersections	12
average width	6.3
...	...

4 Train and predict **O** versus **N**

5 Extract rules

Solution metrics

S1 - Average number of intersections per customer

$$\frac{\sum_{i=1}^{|R|-1} \sum_{j=i+1}^{|R|} I(r_i, r_j)}{N}$$

S2 - Longest distance between two connected customers, per route

$$\frac{\sum_{r \in R} \max_{i \in \{1,\ldots,|r|-1\}} d(n_i^r, n_{i+1}^r)}{|R|}$$

S3 - Average distance between depot to directly-connected customers

$$\frac{\sum_{r \in R} \left(d(D, n_1^r) + d(n_{|r|}^r, D)\right)}{2|R|}$$

Solution metrics

S4 - Average distance between routes (their centers of gravity)

$$\frac{\sum_{r_1 \in R} \sum_{r_2 \in R \setminus r_1} d(G_{r_1}, G_{r_2})}{|R| \cdot (|R| - 1)}$$

S5 - Average width per route

$$\frac{\sum_{r \in R} \left(\max_{i \in \{1,\ldots,|r|\}} d(L_{G_r}, n_i) - \min_{i \in \{1,\ldots,|r|\}} d(L_{G_r}, n_i) \right)}{|R|}$$

S6 - Average span in radian per route

$$\frac{\sum_{r \in R} \max_{i,j \in \{1,\ldots,|r|\}} rad(n_i^r, n_j^r)}{|R|}$$

Solution metrics

S7 - Average compactness per route, measured by width

$$\frac{\sum_{r\in R}\sum_{i=1}^{|r|}\left(d(L_{G_r}, n_i)\right)^+}{N}$$

S8 - Average compactness per route, measured by radian

$$\frac{\sum_{r\in R}\sum_{i=1}^{|r|} rad(G_r, n_i)}{N}$$

S9 - Average depth per route

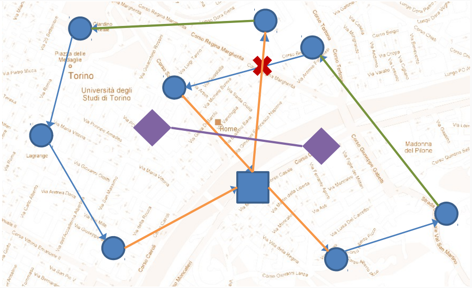
$$\frac{\sum_{r\in R}\max_{i\in\{1,\ldots,|r|\}} d(n_i^r, D)}{|R|}$$

S10 - Standard deviation of the number of customers per route

$$\sqrt{\frac{\sum_{r\in R}(|r| - \frac{N}{|R|})^2}{|R|}}$$

Solution metrics

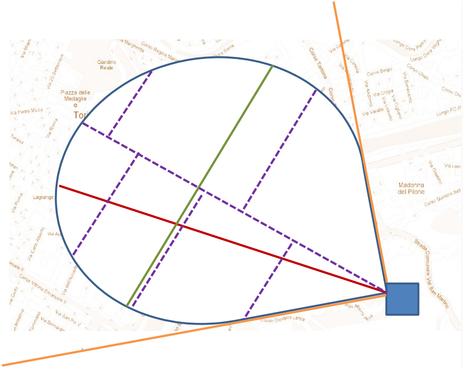
- #Intersections
- Longest Edge
- First Edges
- Inter-Route Distance
- #Customers



Metrics

- Properties of solutions that might influence quality
- Some creativity is required

- Depth
- Width
- Angle Variation
- Compactness



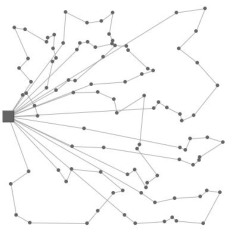
Metrics

- Properties of solutions that might influence quality
- Some creativity is required

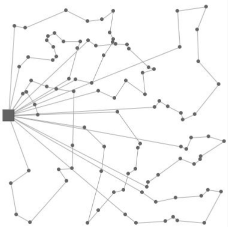
Normalization is necessary

Near-optimal solution | Non-optimal solution

Instance 1

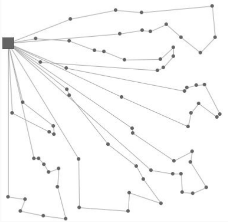


Average Width: 295

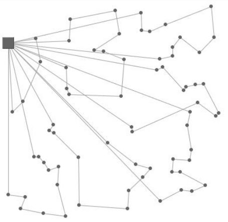


Average Width: 323

Instance 2



Average Width: 204



Average Width: 234

Instance characteristics

- I1 - Number of customers
- I2 - Minimum number of routes
- I3 - Degree of capacity utilisation
- I4 - Average distance between each pair of customers
- I5 - Standard deviation of the pairwise distance between customers
- I6 - Average distance from customers to the depot
- I7 - Standard deviation of the distance from customers to the depot
- I8 - Standard deviation of the radians of customers towards the depot

Methodology

1 Random instance

2 **Near**-optimal solution (**O**)

Non-optimal solution (**N**)

3

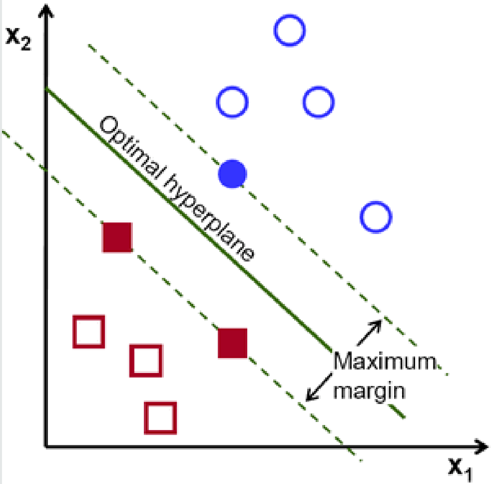
intersections	9
average width	5.4
...	...

intersections	12
average width	6.3
...	...

4 Train and predict **O** versus **N**

5 Extract rules

Support Vector Machines (SVM)



Data mining

Table 2: Prediction accuracies with linear SVM for each dataset

			2% gap		4% gap	
		#data points	H1	H2	H1	H2
20-50 cust.	Class 1	10.000	65%	62%	76%	64%
	Class 2	10.000	67%	61%	77%	63%
	Class 3	10.000	67%	68%	76%	75%
	Class 4	10.000	66%	65%	74%	71%
70-100 cust.	Class 5	2.000	81%	81%	89%	89%
	Class 6	2.000	80%	80%	89%	89%
	Class 7	2.000	85%	85%	90%	91%
	Class 8	2.000	81%	82%	88%	89%

What causes the prediction accuracy

Table 3: Solution metrics with an individual prediction accuracy of higher than 55% per instance class (largest per class in bold)

	2% gap										4% gap									
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Class 1								**58**			57	56	56		56	59	57	**61**		
Class 2								**57**			57	57	56			56	56	**62**		
Class 3	58				**60**		60	57			61		56		**65**	59	64	60		
Class 4	57				**58**		58	56			59		56		**62**	57	62	61		
Class 5			62		67	**68**	67	67			60		71		78	77	**79**	76	59	
Class 6	57		62		65	66	68	**70**			60		67		74	73	74	**75**		
Class 7	66	57	60		**79**	65	75	65			71		66		**84**	72	80	72		
Class 8	64				**72**	61	70	66			68		58		**79**	67	77	72		

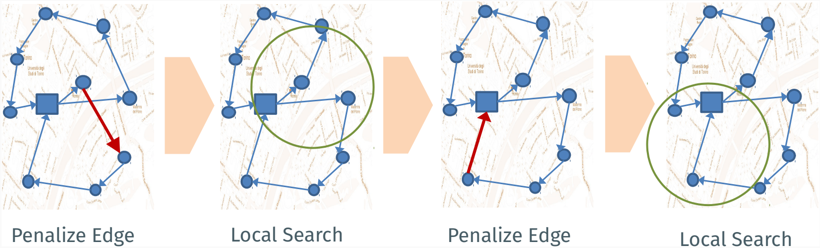
Most effect: S1 (intersections), S3 (edges from depot), S5 (width), S6 (width in radian), S7 (compactness), S8 (compactness by radian)

Metaheuristic framework: guided local search

- Idea: penalize bad edges

$$c^g(i,j) = c(i,j) + \lambda p(i,j)L$$

- Alternate penalization and local search

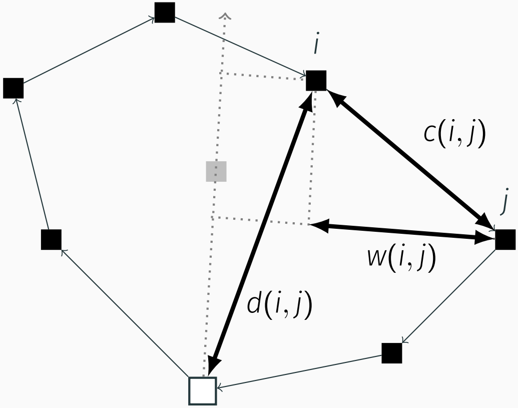


- Question: what is a “bad” edge?

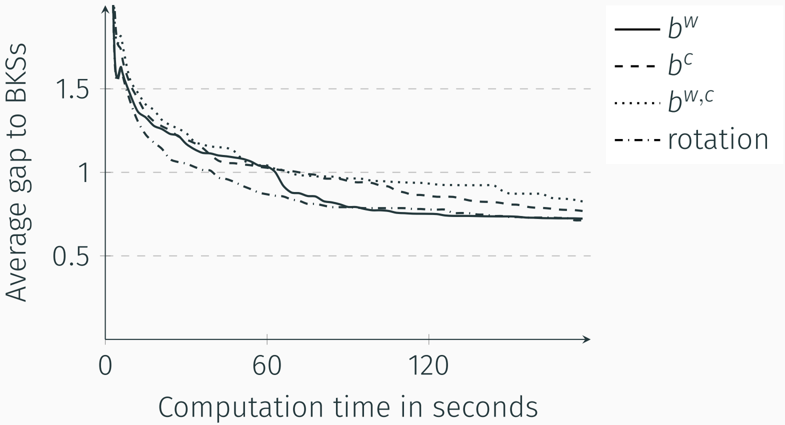
Idea #3

Focus the power of the heuristic to make it efficient

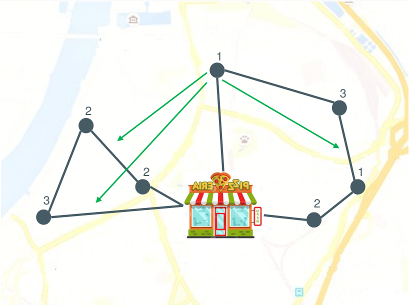
Badness of an edge



Penalization criterion



Linearizing the performance

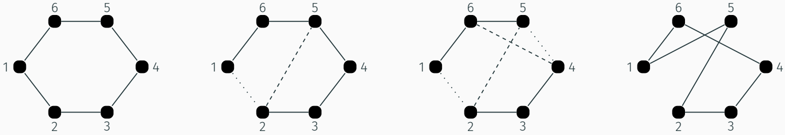


- Try to relocate next to each customer: $O(n^2)$
- Try to relocate next to closest a customers: $O(a \times n)$

Heuristic pruning

Can we restrict a without hurting performance?

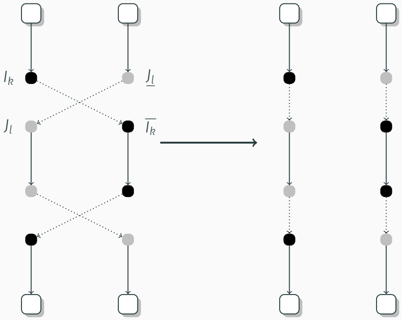
Heuristic pruning



Lin Kernighan (one route)

- Already very efficient
- Restrict to 10 nearest neighbors
- Restrict to 4-opt

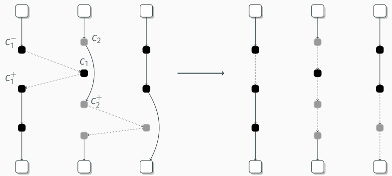
Heuristic pruning



CROSS exchange (two routes)

- Start from most penalized edge
- Restrict to 30 nearest neighbors
- Restrict size of subroute to 100

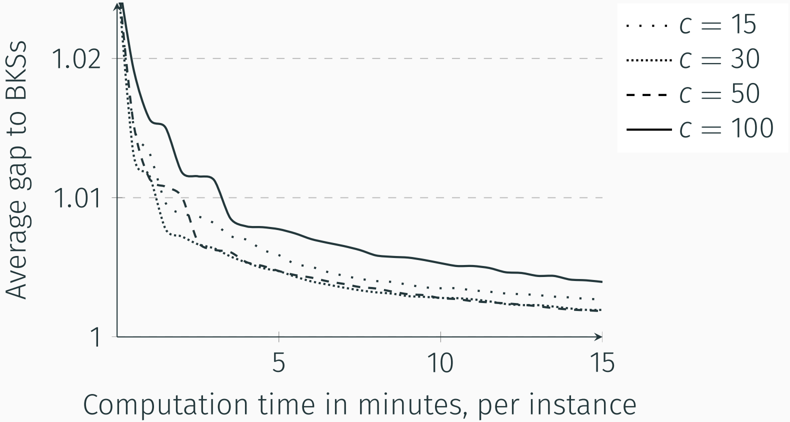
Heuristic pruning



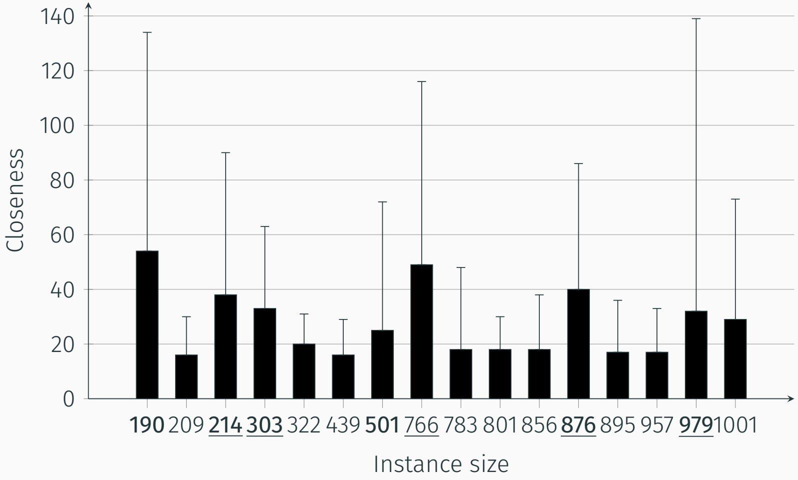
Relocation chain (>two routes)

- Start from most penalized edge
- Restrict to 30 nearest neighbors
- Restrict size of chain to 2

Effect of pruning tightness



Closeness of customers in high-quality solutions



Memory issues

Only distances between close neighbors need to be loaded

Our algorithm

1. Construct an initial solution (Clarke–Wright)
2. **Repeat** until stopping criterion
    2.1 **Repeat** (GLS)
        2.1.1 Penalize worst edge w

$$\text{largest value of “badness”: } b = \frac{f(w, c, d, \cdots)}{1 + p}$$

        2.1.2 Apply LS starting from w using $c^g(.)$ as evaluation function
    2.2 Global optimization: apply LS on all routes that where changed by GLS, using $c(.)$ as evaluation function

Our algorithm

1. Construct an initial solution (Clarke–Wright)
2. **Repeat** until stopping criterion
   2.1 **Repeat** (GLS)
      2.1.1 Penalize worst edge w

$$\text{largest value of "badness": } b = \frac{f(w, c, d, \cdots)}{1 + p}$$

      2.1.2 Apply LS starting from w using $c^g(.)$ as evaluation function
   2.2 Global optimization: apply LS on all routes that where changed by GLS, using $c(.)$ as evaluation function

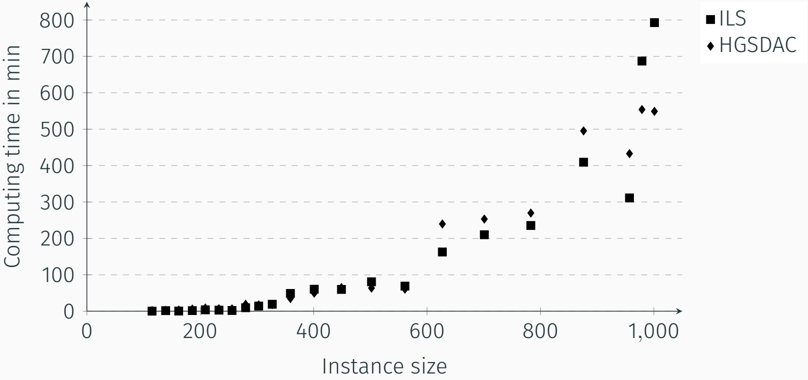
Important note

Completely deterministic

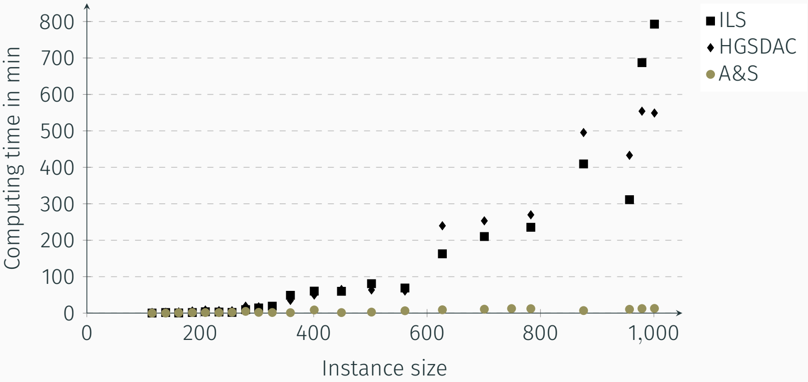
Movie Time

Results

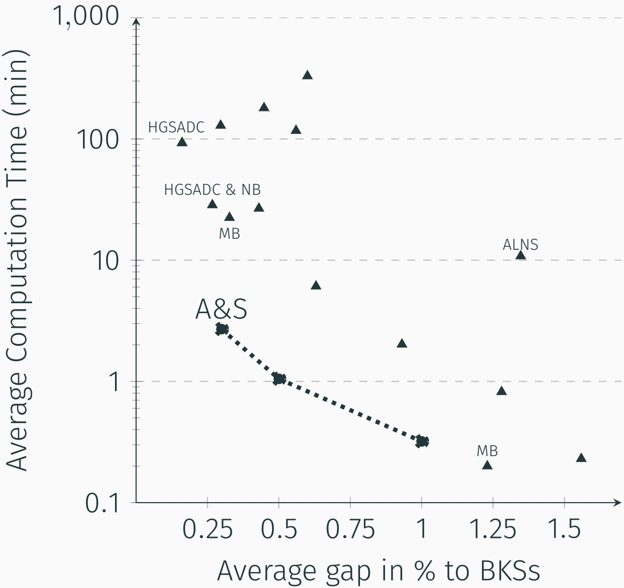
Comparison to other algorithms



Comparison to other algorithms



Comparison to other algorithms



Results on XXL instances

Instance	GVNS			AGS (short runtime)			AGS (long runtime)		
	Value	Gap	Time	Value	Gap	Time	Value	Gap	Time
W (7,798)	4,559,986	7.37	34.5	4,294,216	1.12	7.8	4,246,802	0.00	39.0
E (9,516)	4,757,566	4.17	83.9	4,639,775	1.59	9.5	4,567,080	0.00	47.5
S (8,454)	3,333,696	3.97	56.2	3,276,189	2.18	8.5	3,206,380	0.00	42.5
M (10,217)	3,170,932	4.35	77.6	3,064,272	0.84	10.2	3,038,828	0,00	51.0
R3 (3,000)	186,220	1.87	4.8	183,184	0.21	3.0	182,808	0.00	15.0
R6 (6,000)	352,702	1.49	24.4	348,225	0.20	6.0	347,533	0.00	30.0
R9 (9,000)	517,443	1.05	57.7	512,530	0.09	9.0	512,051	0.00	45.0
R12 (12,000)	680,833	1.12	108.4	674,732	0.22	12.0	673,260	0.00	60.0
Average		3.17	55.8		0.80	8.3		0.00	41.3

30.000 customers

An unexpected benchmark

from: Keld Helsgaun <keld@ruc.dk>

[...]

My aim was to see how close, given plenty of time, my LKH-3 solver could get to the best solutions found by your extremely fast VRP solver. Now, after more than a month of computation, LKH-3 has been able to find tours that are from 0.4 to 1.1 percent shorter than yours. I attach a table with the results together with the solutions found.

[...]

An unexpected benchmark

from: Keld Helsgaun <keld@ruc.dk>

[...]

My aim was to see how close, given plenty of time, my LKH-3 solver could get to the best solutions found by your extremely fast VRP solver. Now, **after more than a month of computation**, LKH-3 has been able to find tours that are from 0.4 to 1.1 percent shorter than yours. I attach a table with the results together with the solutions found.

[...]

An unexpected benchmark

Results for Belgium instances (CVRP)
Keld Helsgaun, February 16, 2018

Instance	*n*	*m*	BKS	LKH-3	Gap (%)
L1	3000	203	195239	**194381**	**-0.439**
L2	4000	46	114833	**113484**	**-1.175**
A1	6000	343	483606	**481338**	**-0.469**
A2	7000	120	299398	**297478**	**-0.641**
G1	10000	485	476489	**474164**	**-0.488**
G2	11000	110	267935	**265763**	**-0.811**
B1	15000	512	512089	**509457**	**-0.514**
B2	16000	182	360760	**357382**	**-0.936**
F1	20000	684	7321847	**7300772**	**-0.288**
F2	30000	256	4526789	**4499422**	**-0.605**

Conclusions

Conclusions

Designing heuristics the modern way

- Use powerful complementary local search heuristics
- Make them efficient using knowledge on the properties of good solutions
- Make them even more efficient using heavy pruning

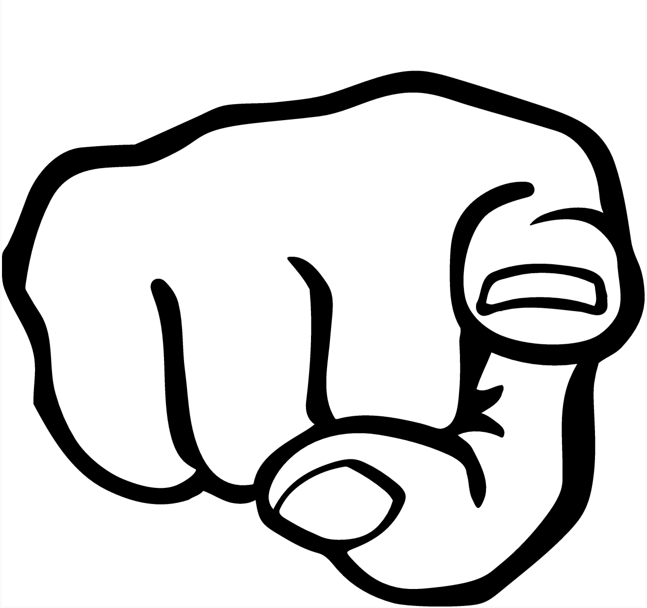
Conclusions

Designing heuristics the modern way

- Use powerful complementary local search heuristics
- Make them efficient using knowledge on the properties of good solutions
- Make them even more efficient using heavy pruning

Challenge

Works for VRP, what about other problems?



University of Antwerp
Operations Research Group
ANT/OR

antor.uantwerpen.be

kenneth.sorensen@uantwerpen.be