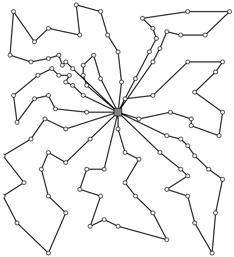
Heuristics for Vehicle Routing Problems: Current challenges and future prospects

Thibaut Vidal

Departamento de Informatica
Pontificia Universidade Catolica do Rio de Janeiro



SBPO'18

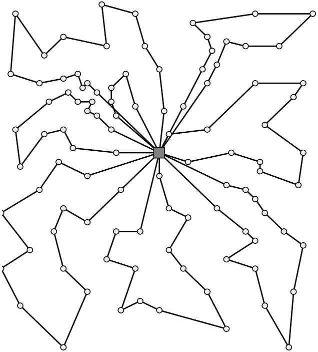
August 6-9th, Rio de Janeiro, Brasil

Contents

Contents

Multi-attribute vehicle routing problems (MAVRPs)

- Capacitated vehicle routing problems (VRP)
  - **INPUT :** n customers, with locations and demand quantity. All-pair distances. Homogeneous fleet of m vehicles with capacity Q located at a central depot.
  - **OUTPUT :** Least-cost delivery routes (at most one route per vehicle) to service all customers.

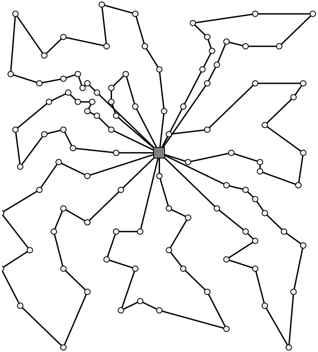


- NP-Hard problem
- recent breakthrough in exact methods enable to solve problems of moderate size with up to 300-400 customers (Uchoa et al., 2013).
- A Scopus search "Vehicle Routing" for 2007-2011 returns 1258 publications, including 566 journal papers.
- Massive research on heuristics
- Diverse and larger instances available (Uchoa et al., 2017)

Multi-attribute vehicle routing problems (MAVRPs)

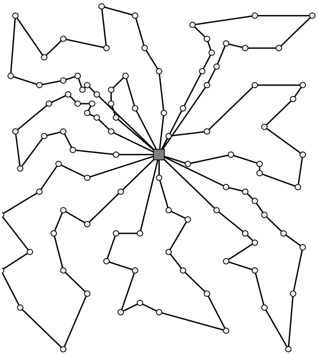
- Capacitated vehicle routing problems (VRP)
  - **Combinatorial explosion:** For a problem with **n=100 customers** and **a single vehicle**, the number of feasible solutions is:

    n! = 93326215443944152681699
    23885626670049071596826438162146859296389521759999322991
    56089414639761565182862536979208272237582511852109168640
    00000000000000000000000 $\approx 10^{158}$



Multi-attribute vehicle routing problems (MAVRPs)

- Even with a grid of computers which:
  - Contains as many CPUs as the estimated number of atoms in the universe : $n_{\text{CPU}} = 10^{80}$
  - Does one operation per Planck time: $t_P = 5.39 \times 10^{-44}$ seconds



- We would need $T = 10^{158} \times 5.39 \times 10^{-44}/1080 = 5.39 \times 10^{34}$ seconds to enumerate all solutions.
- Compare this to the estimated age of Universe : 4.33×10^{17} seconds

Multi-attribute vehicle routing problems (MAVRPs)

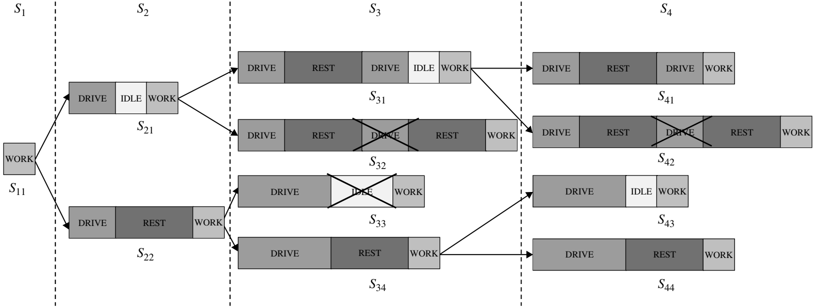
- **VRP "attributes":** Supplementary decisions, constraints and objectives combined with the classic VRP (Vidal et al., 2013b)
  - **Realistic objectives:** Profitability, equity, service Levels, persistence, compactness, robustness, externalities
  - **Integrated planning:** Multiple periods, depots, echelons, fleet mix, LRP, IRP, synchronization...
  - **Fine-grained modeling:** Time windows (soft or multiple), loading constraints (2D,3D), driver skills, time-dependent travel times, charging stations, engine modes, drones etc...

- **VRP "attributes":** Supplementary decisions, constraints and objectives combined with the classic VRP (Vidal et al., 2013b)

[SHOW EXAMPLE 1. SUBPROBLEM SOLVER]

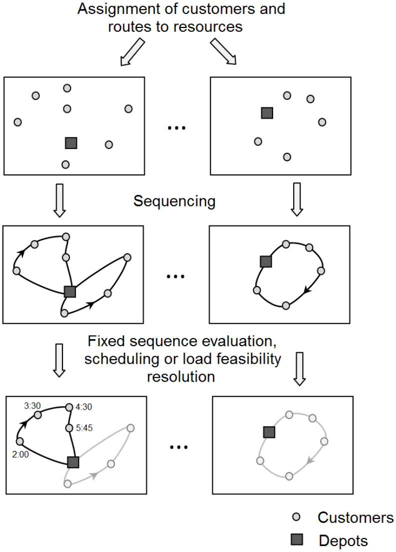
Multi-attribute vehicle routing problems (MAVRPs)

- **VRP "attributes":** Supplementary decisions, constraints and objectives which complement the classic VRP formulation (Vidal et al., 2013b)



Multi-attribute vehicle routing problems (MAVRPs)

- Three main resolution tasks and related problem attributes
- **ASSIGNMENT** (assignment of customers and routes to time-periods or depots)
  - *multi-period, multi-depot, heter. fleet, location routing...*
- **SEQUENCING** (choice of the sequence of visits)
  - *P&D, Backhauls, 2-echelon...*
- **ROUTE EVALUATION** (route feasibility/cost & other decisions)
  - *Time windows, time-dep travel time, loading constraints, HOS regulations, lunch breaks, load-dependent costs...*



Multi-attribute vehicle routing problems (MAVRPs)

- Challenges: **VARIETY** and **COMBINATION** of attributes
- *Over 200 attributes* have been proposed to this date...

  ...which often appear together ⇒ 2^{200} problems... 2^{200} different methods, and 2^{200} papers ?!!!
- **"Double" combinatorial explosion**: Combinatorial optimization problem and combinatorial *family of problems*

⇒ Progress towards unified solution concepts and methods

⇒ Solvers that can address a wide range of problems without need for extensive adaptation or user expertise.

⇒ Necessary tools for the timely application of current optimization methods to industrial settings.

[SHOW EXAMPLE 2. PROBLEM VARIETY]

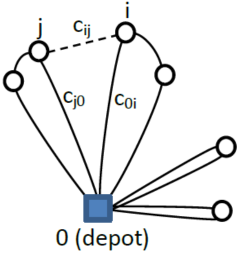
Contents

Contents

Constructive methods

- **Constructive methods:** mostly between 1960s and 1980s.
  - Making step-by-step definitive decisions, which cannot be revoked afterwards

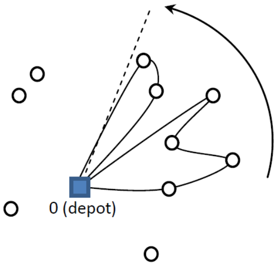
- Savings method (Clarke and Wright 1964)
  - Merge routes step by step based on a savings measure s_{ij} $s_{ij} = c_{i0} + c_{0j} - c_{ij}$
  - Some refinements by Gaskell (1967) and Yellow (1970): $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$
  - Mole and Jameson (1976) and Solomon (1987) later generalize the concepts and consider insertions inside the routes.



Constructive methods

- **Constructive methods:** mostly between 1960s and 1980s.
  - Making step-by-step definitive decisions, which cannot be revoked afterwards

- Sweep algorithm (Gillett and Miller, 1974)
  - Sweep the deliveries in circular order to create routes.
  - A new route is initiated each time the capacity is exceeded.
- Petal methods : generate several alternative routes, called petals, and select a subset of these by solving a set-covering linear program.



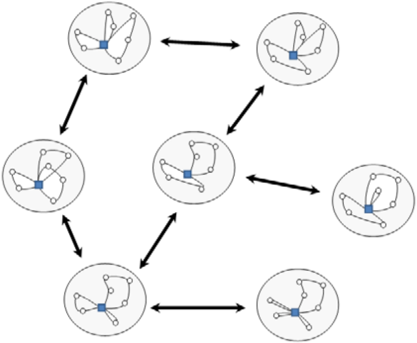
Constructive methods

- **Constructive methods:** mostly between 1960s and 1980s.
  - Making step-by-step definitive decisions, which cannot be revoked afterwards

- Route first cluster second (Newton and Thomas, 1974; Bodin and Berman, 1979; Beasley, 1983)
  - construct a giant circuit (TSP tour) that visits all customers.
  - Segmenting this tour into several routes. Optimal segmentation is assimilated to a shortest path problem in a auxiliary directed acyclic graph
  - Possible to solve the segmentation problem (Split) in $\mathcal{O}(n)$ (Vidal, 2016)

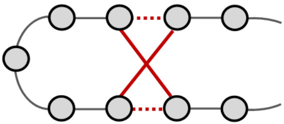
Classical Local Searches

- **Local-improvement procedures :**
  - From an *incumbent solution* s define a *neighborhood* $N(s)$ of solutions obtained by applying some changes.
  - The set of solutions, linked by neighborhood relationships = search space.
  - LS-improvement method progress from one solution to another in this search space as long at the cost improves.

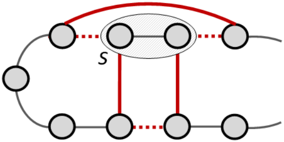


Classical Local Searches

- For optimizing a single route (TSP tour);
  - in the terminology of Lin (1965), λ-opt neighborhood = subset of moves obtained by deleting and reinserting λ arcs.
  - 2-opt and 3-opt are commonly used,
  - Or-opt which comes to relocate sequences of bounded size, and is a subset of 3-opt.



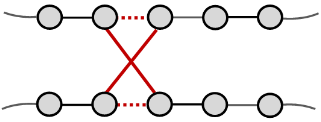
2-opt



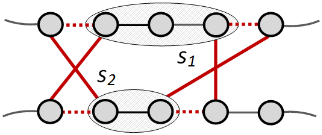
Or-exchange

Classical Local Searches

- For optimizing multiple routes together,
  - Insert neighborhood (relocate a delivery)
  - Swap neighborhoods (swap two deliveries from different routes)
  - CROSS-exchange (exchange two sequences of visits)
  - I-CROSS (exchange and reverse two sequences)
  - 2-opt* exchange two route tails (special case of CROSS)



2-opt*



CROSS

Classical Local Searches

- These neighborhoods contain a polynomial number of moves.
  - For all moves except CROSS and I-CROSS, the number of neighbors is $O(n^2)$
  - CROSS and I-CROSS are often limited of sequences of bounded size with less than k customers, in that case the number of neighbors becomes $O(k^2 n^2)$

- Other non-enumerative large-scale neighborhoods:
  - Heuristic of Lin and Kernighan (1973) – efficient implementation from Helsgaun (2000);
  - Ruin-and-recreate (Shaw, 1998; Schrimpf et al., 2000);
  - Ejection chains (Glover, 1992, 1996)

Classical Local Searches

- Efficient move evaluations and pruning procedures are critical to address large-scale problem instances
  - Neighborhood restrictions, granular search (Johnson and McGeoch, 1997; Toth and Vigo, 2003): restrain the subset of moves to spatially related customers
  - Sequential search (Christofides and Eilon, 1972; Irnich and Villeneuve, 2003): any profitable move can be broken down into a list of arc exchanges $(a_1, \ldots, a_\lambda)$ with gains $(g_1, \ldots, g_\lambda)$ such that for any $k \in \{1, \ldots, \lambda\}$, $g_1 + \cdots + g_k \geq 0$.
  - This condition allows to prune many non-promising moves.

Classical Metaheuristics

- Local-improvement methods $\Rightarrow$ local optima.
- Metaheuristics to escape and guide the search
- Main classes of methods:
  - **Neighborhood-centered** search – iterative improvement of one single solution – Tabu search, Simulated annealing, ILS, VNS...
  - **Population-based** search – improving a population of solutions – Hybrid GA, evolutionary algorithms, ACO, path relinking...
  - **Hybrid approaches** – often combining many successful strategies
- Hybrids are very common $\Rightarrow$ the limits between metaheuristics become blurred – Necessity to use a **simple and optimization-oriented terminology** to identify their common structures and success elements (Sörensen, 2015).

Unified Tabu Search – Cordeau et al. (1997, 2001)

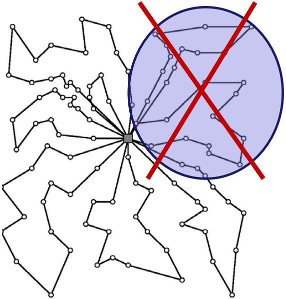
- Tabu search – choice of **best move** at each step (possibly non-improving).
- Neighborhood: single RELOCATE
- Short-term tabu memories to avoid cycling:
  - Moving Client i from route R_1 to $R_2 \Rightarrow$ Not allowed to insert i back into route R_1 for X iterations.
- Longer term **diversification strategies:**
  - Penalizing recurrent solution attributes in the objective function
  - Penalized infeasible solutions (excess load or duration)

TS with adaptive memory – Rochat and Taillard (1995)

- From 1995, but already contained most of the ideas used nowadays:
- **Diversification**
  - Tabu search based on SWAP and RELOCATE moves
  - *Probabilistic* selection of moves driven by measures of attractiveness
- **and Intensification**:
  - Detection of good fragments of solutions that consistently appear in elite solutions and creation of new solutions from these fragments to obtain new stating points
  - Decomposition phases based on spatial proximity
  - Exact solution of the TSPs at regular intervals
  - Set covering optimization as a post-optimization

ALNS – Pisinger and Ropke (2007)

- **Large neighborhoods** based on the *ruin-and-recreate* principle (Shaw, 1998; Schrimpf et al., 2000).
- Variety of operators to **partially destroy** the solutions
  - Based on randomness, cost metrics, relatedness, history...
  - Adaptive probabilities for operator selection
- Variety of operators to **reconstruct** the solutions
- Deteriorating solutions are accepted with some probability, as in a simulated annealing



Iterated Local Search – Subramanian et al. (2013)

- **Iterated local search:** at each iteration local search until a local optimum is encountered, **shaking** and local search again...

- A large diversity of neighborhoods is used
  - Relocate and Swap of one to three customers in different routes, 2-Opt, 2-Opt*, empty-route, swap depot...
  - Multiple shaking operators : multi-swap, multi-shift, double-bridge ...

Iterated Local Search – Subramanian et al. (2013)

- Set covering model to create new solutions out of a set of high-quality routes.
  Adaptation of the pool size.

ELITE ROUTES found during the search

$$
\begin{aligned}
\text{minimize} \quad & \sum_{k \in S} d_k x_k & \\
\text{subject to} \quad & \sum_{k \in S} a_{ik} x_k = 1 & i = 1, \dots, n \\
& x_k = 0 \text{ or } 1 & k \in S.
\end{aligned}
$$

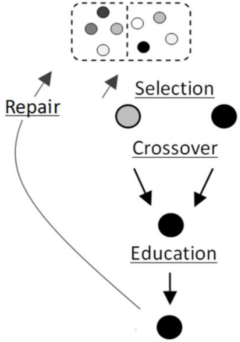
Solver for integer linear programming (Cplex)

BEST SOLUTION CREATED FROM THESE ROUTES

Hybrid Genetic Algorithm – Prins (2004)

- **First Genetic Algorithm (GA)** to achieve competitive results on some VRP variants.

- **Genetic algorithms** mimic natural evolution
  - Population of solutions
  - Selection
  - Crossover
  - Mutation
    (replaced here by a local search)

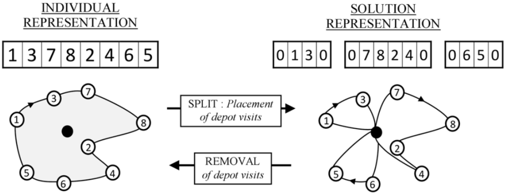


Hybrid Genetic Algorithm – Prins (2004)

- The algorithm of Prins (2004) includes a few important "tricks":

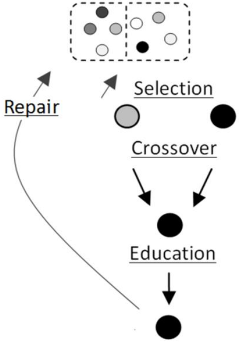
- Giant-tour solution representation
  - Polynomial Split algorithm to obtain a complete solution

- Simple Crossover
- Local search on the offspring
- Population management (spacing constraint)



Unif. Hybrid Genetic Search – Vidal et al. (2012, 2014)

- Classic hybrid genetic algorithm with:
  - Giant-tour solution representation in the crossover: the same as Prins (2004)
  - Efficient local search
  - Management of penalized infeasible solutions
  - Promotion of diversity in the population: **Biased fitness**
- Easily generalizable ⇒ Applied to over 50 VRP variants

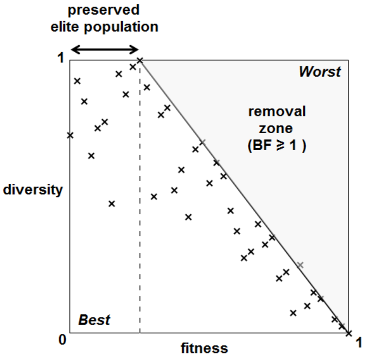


Unif. Hybrid Genetic Search – Vidal et al. (2012, 2014)

Biased fitness: combining ranks in terms of solution cost $C(I)$ and contribution to the population diversity $D(I)$, measured as a distance to other individuals :

$$BF(I) = C(I) + \left(1 - \frac{nbElite}{popSize - 1}\right) D(I)$$

- Used for parents selection
  $\Rightarrow$ Balancing quality with innovation to promote a more thorough exploration of the search space.
- Used during selection of survivors
  $\Rightarrow$ Removing individuals with worst $BF(I)$ still guarantees elitism



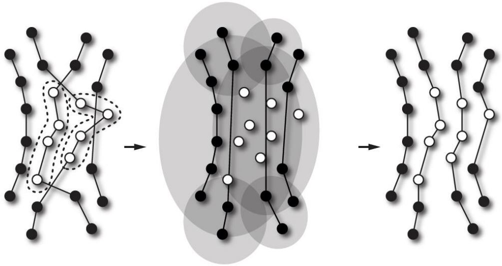
Knowledge-Guided LS Arnold and Sörensen (2018)

- Based on a Guided local search
  - **Detect and temporarily penalize *bad* edges**
  - Characterization of bad edges result from a prior study of defining features of good and bad solutions
- Three efficient types of neighborhoods
  - CROSS-exchanges
  - Ejection chains
  - Heuristic of Lin and Kernighan (1973)
- Leading to an efficient and fast method for the CVRP

[SHOW EXAMPLE 3. ALGO. ARNOLD & SORENSEN]

SISRs – Christiaens and Vanden Berghe (2018)

- Based on the ruin-and-recreate principle
- One ruin operator (adjacent string removals)
  - Aims to introduce capacity and spatial slack
- One recreate method (greedy insertion with blinks)
  - Skipping best insertions in a controlled manner



- Excellent results on the new large-scale CVRP instances of Uchoa et al. (2017)
- State-of-the-art results for multiple problem variants: CVRP, VRPTW, PDPTW...

Summary

- A very large variety of metaheuristics have been developed.
- Many existing concepts and methods... and many open questions:
- Why using a strategy or a metaheuristic of a given type
- For which problems some strategies are most successful
- Method: tradeoff between **solution quality**, **speed**, **ability to generalize**, **robustness** and **simplicity**

Contents

Successful strategies – Analysis

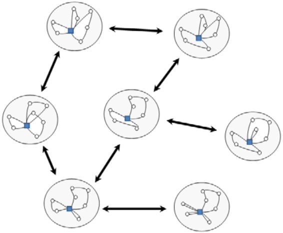
- Vidal et al. (2013b): analysis of solution concepts for multiple VRP variants

- **Protocole:**
  - Selection of 14 multi-attribute VRPs – Criterion : classic benchmark instances available + large number of heuristics).
  - Identification of the top 3 to 5 best metaheuristics for each problem
  - Analysis of the resulting 64 methods, to pinpoint successful methodological elements.

Successful strategies – Analysis

- 19 aspects of the methods have been examined:

Search space	1) presence of infeasible solutions
	2) use of indirect representations of solutions
Neighbourhoods	3) presence of multiple neighbourhoods
	4) use of polynomially enumerable neighbourhoods
	5) use of pruning procedures
	6) use of large neighbourhoods
	7) use of solution recombinations
Trajectory	8) presence of random components
	9) continuous aspect of trajectories
	10) discontinuous aspect
	11) mixed aspect
Control and memories	12) use of populations
	13) diversity management
	14) parameter adaptation
	15) advanced guidance mechanisms
Hybrid strategies	16) use of hybridization
	17) matheuristics with integer programming
Parallelism	18) use of parallelism or cooperation concepts
Problem decompositions	19) use of problem decompositions

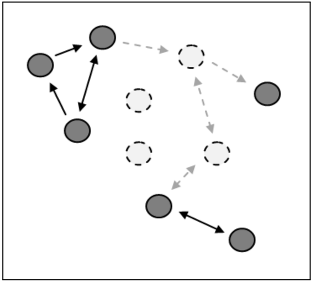


- 19 aspects of the methods have been examined:

		SP.		NEIGHBOUR.					TRAJEC.				CONTROL							
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
		RELAXATION	SOL. REPRESENT.	MULT NEIGHB.	ENUMERATIVE	PRUNING	LARGE	RECOMBIN.	RANDOMNESS	CONTINUOUS	DISCONTINOUS	MIXED	POPULATIONS	DIV. MANAG.	PARAM. ADAPT.	GUIDANCE	HYBRIDIZATION	MATHEUR.	PARALLELISM	DECOMPOSITION
BACKHAULS																				
Brandão (2006)	Tabu	X		X	X				X	X					X	X				
Ropke and Pisinger (2006a)	ALNS	X		X			X		X	X					X	X	X			
Gajpal and Abad (2009)	ACO			X	X	X			X		X					X	X			
Zachariadis and K. (2012)	Attrib. driven LS			X	X	X				X						X				
PICK-UP AND DELIVERIES																				
Bent and V.H. (2006)	SA + LNS			X	X	X	X		X	X							X	X		
Ropke and Pisinger (2006b)	ALNS	X		X			X		X	X					X	X	X			
Cordeau and Laporte (2003)	Tabu	X			X	X				X					X	X				
Parragh et al. (2010)	VNS	X		X	X		X		X	X					X					
Nagata and Kobayashi (2011)	HGA			X	X	X	X	X	X		X		X			X	X			
MULTIPLE TRIPS																				
Taillard et al. (1996)	Adapt. M. + Tabu	X		X	X			X	X	X	X	X	X			X	X		X	X
Olivera and Viera (2007)	Adapt. M. + Tabu	X		X	X	X		X	X	X	X	X	X			X	X			
Alonso et al. (2008)	Tabu	X			X					X					X	X				
TIME WINDOWS																				
Hashimoto et Y. (2008)	Path Relinking	X		X	X	X	X	X	X		X		X		X	X				
Repoussis et al. (2009)	Guided EA			X	X	X	X	X	X	X	X	X	X	X	X	X	X			
P.-Gagnon et al. (2009)	LNS + Col. Gen.			X			X		X	X							X	X		
Nagata et al. (2010)	HGA	X		X	X	X		X	X		X		X				X			

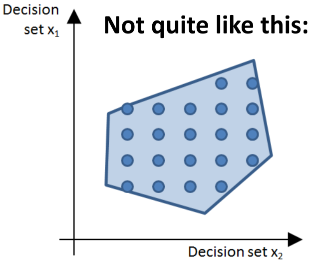
Successful strategies – Analysis

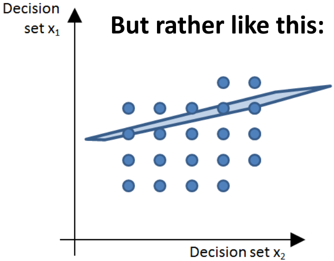
- **Search Space: Using infeasible solutions (31/64 methods).**
- Usual to relax route constraints and penalize violations: capacity, duration, TW...
- Enables to transition more easily in the search space between feasible solutions.
- Strategic oscillation (Glover, 1986): good solutions are known to be close to the borders of feasibility – Oscillating close to these borders by adapting the penalty coefficients.



Successful strategies – Analysis

- The space of feasible solutions is often...





- On problems with tight constraints, infeasible solutions are necessary to transition from one solution to another

Successful strategies – Analysis

- Experimental analysis of TW relaxations in Vidal et al. (2015a).
  - Solomon VRPTW instances – simple LS-improvement procedure.

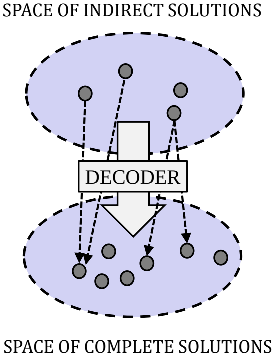
Inst.	SolI1	NoUnf	Late	Twarp	Flex	NoUnf	Late	Twarp	Flex
		SolomonI1 Initial Solution				Random Initial Solution			
R1	1431.97	1225.25	1219.11	1220.13	**1219.41**	1268.97	1231.70	1226.08	1229.73
R2	1326.64	963.53	957.11	947.77	942.87	982.44	**940.38**	947.41	940.79
C1	936.48	844.00	835.17	835.67	**834.26**	860.82	835.14	840.73	835.32
C2	696.57	605.62	603.62	603.08	**600.59**	709.37	650.52	645.45	649.61
RC1	1578.28	1401.49	1399.11	**1389.83**	1396.54	1482.79	1406.57	1401.53	1404.57
RC2	1653.61	1139.12	1077.93	1093.02	1079.40	1130.21	1072.63	1075.60	**1070.59**
CTD	71633	58067	57319	57275	**57125**	60361	57679	57678	57621
T(sec)	0.03	3.41	20.06	6.59	7.90	6.25	17.55	8.03	10.00

  - Similar conclusions regarding distance and load relaxations on CVRP, PVRP and MDVRP (Vidal et al 2012).

Successful strategies – Analysis

- **Search Space: Indirect solution representations (12/64)**

- Focus the heuristic search on a subset of the decision variables
- Use a decoder to determine the rest of the decisions – Exact algorithms may be used for decoding
- Examples:
  - Giant tours without trip delimiters in Prins (2004)
  - Storing a subset of the decision sets: visit-day choices for the PVRP, sequences of visits without visit-time information...



Successful strategies – Analysis

- **Neighborhoods**
  - All successful VRP metaheuristics use either local or large neighborhood search.
  - LS neighborhoods usally contain $\mathcal{O}(n^2)$ moves
  - Ruin-and-recreate is also frequently used
  - Covering at least the main families of moves (RELOCATE, SWAP, 2-OPT) is determining to achieve high-quality solutions.
  - Trade-off between neighborhood size and search speed
  - Optimizing **all attributes of the problems** (sequencing, assignment to depots,vehicles, days) is a key to success.

Successful strategies – Analysis

- Neighborhood search = bottleneck of most recent metaheuristics (but for a good reason)

- Speed-up techniques (used in 26/64 methods)
  - **Neighborhood restrictions**: granular or sequential search
  - **Memories**: matrices for move evaluations, hashtables for route evaluations.
  - **Preprocessing** on subsequences to speed move evaluations in presence of complicating attributes (discussed later in this talk).

Successful strategies – Analysis

- **Search Trajectory – Randomization (56/64)**
  - Prerequisite for some asymptotic convergence properties (e.g., SA, GA).
  - A simple way of avoiding cycling and bringing more diversity.
  - *"an intelligent use of randomization, which is not blindly uniform but embedded in probabilities that account for history and measures of attractiveness, offers a useful type of diversification that can substitute for more complex uses of memory"* (Rochat and Taillard, 1995)
  - If needed, *fix the seed* to obtain a deterministic algorithm.
  - Random order is not worse than any fixed customer-indices order obtained from the instance (often arbitrary).

Successful strategies – Analysis

- **Populations (28/64)**

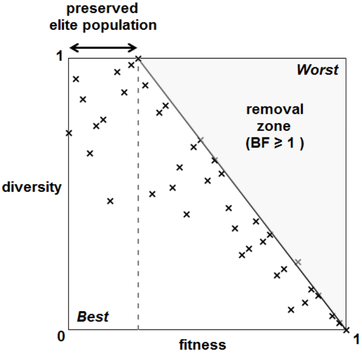
- Acquisition, management, and exploitation of problem-knowledge
  $\Rightarrow$ Core function of a metaheuristics.

- Glover (1989) discern two types of memories
  - Short term memories (e.g. tabu lists) – used to escape local optima
  - Medium and long-term memories – guide the overall exploration

- Other forms of memories: populations to store full solutions, routes or fragments of solutions, statistics on decision variables, pheromones, supported patterns...

- **Population management (14/28)**
- A need for diverse and high-quality solutions
  - Critical to avoid premature convergence. Needed to counterbalance the aggressive-improvement abilities of local search in hybrid GA.

  - Diversity management strategies (Prins, 2004; Sörensen and Sevaux, 2006)
  - Promotion of diversity in the objective (Vidal et al., 2012)
  - Based on distance measures, in objective or solution space.



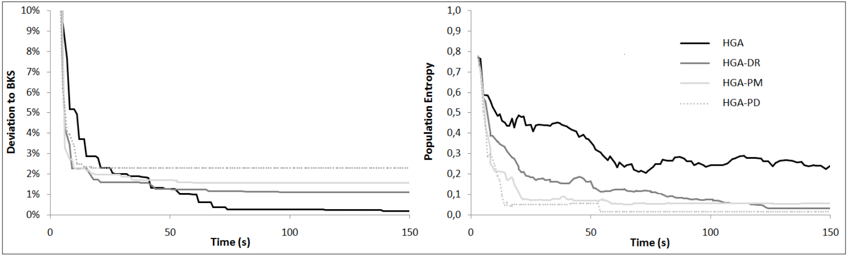
Successful strategies – Analysis

- **Memories and control – Population management (14/28)**
- Some experiments on this topic in Vidal et al. (2012), solution quality of HGSADC on standard PVRP, MDVRP, and MDPVRP instances:
  - HGA: No diversity management method
  - HGA-DR: Spacing rule in objective space (Prins, 2004)
  - HGA-PM: Spacing rule in solution space (Sörensen and Sevaux, 2006)
  - HGSADC: Promotion of diversity in the objective (Vidal et al., 2012)

Benchmark		**HGA**	**HGA-DR**	**HGA-PM**	**HGSADC**
PVRP	T	6.86 min	7.01 min	7.66 min	8.17 min
	%	+0.64%	+0.49%	+0.39%	+0.13%
MDVRP	T	7.93 min	7.58 min	9.03 min	8.56 min
	%	+1.04%	+0.87%	+0.25%	-0.04%
MDPVRP	T	25.32 min	26.68 min	28.33 min	40.15 min
	%	+4.80%	+4.07%	+3.60%	+0.44%

Successful strategies – Analysis

- Some experiments on this topic in Vidal et al. (2012), solution quality on standard PVRP, MDVRP, and MDPVRP instances:
  - HGA: No diversity management method
  - HGA-DR: Spacing rule in objective space (Prins, 2004)
  - HGA-PM: Spacing rule in solution space (Sörensen and Sevaux, 2006)
  - HGSADC: Promotion of diversity in the objective (Vidal et al., 2012)



Successful strategies – Analysis

- **Search Guidance**

- A very simple form of guidance: parameters adaptation (30/64)
  - Driving infeasibility penalties, mutation and crossover rates, frequency of use of some operators or strategies.
- More advanced forms of guidance: collect and analyze information on the past search to guide the method
  - Collect historical statistics on solution features, arcs, sets of arcs, routes, or problem specic attributes.
  - Mine **supported patterns** (Ribeiro et al., 2006; Santana, 2018)

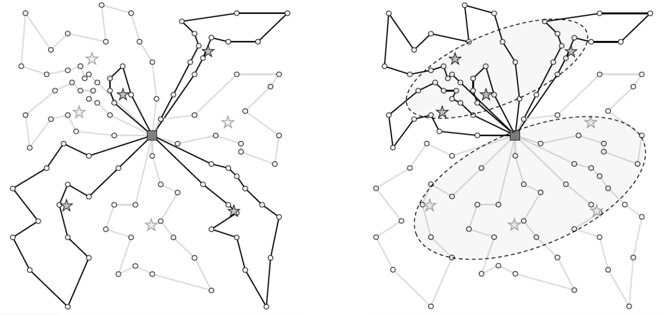
- **Exploitation of guidance information** :
  - Guidance actions to
    - Either **intensify** the search around promising solution features
    - Or **diversify** the search around promising unexplored areas.
  - Applying penalties or incentives on solution features
  - Restarts from elite solutions or fragments of solutions
  - Target solutions in path relinking
  - Neighborhood choice driven by pheromone matrices in ACO

- Continuous through the search, or punctually through a purposeful move

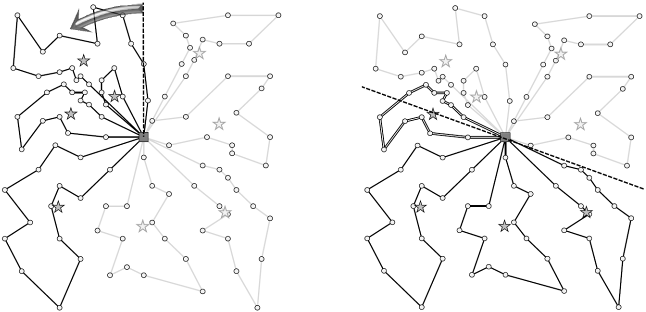
Successful strategies – Analysis

- **Hybridizations (39/64)**
  - Combining features of several methods
  - Most frequent in the heuristics surveyed : GA+LS, ACO+LS or ACO+LNS, Tabu + recombinations, ILS + VNS...
- Generally speaking, metaheuristics are inherently hybrids, described sometimes as “heuristics that guide other heuristics”
- **Matheuristics (9/64)**, blending metaheuristics with math. programming components:
  - Handling problem-attributes (e.g. loading constraints or split deliveries)
  - Exploring large neighborhoods
  - Recombining solution elements...

- **Decompositions**
  - Help to focus on subsets of decision variables: useful as an intensification procedure or to deal with large-scale problems
  - Some examples:



- **Decompositions**
  - Help to focus on subsets of decision variables: useful as an intensification procedure or to deal with large-scale problems
  - Some examples:



Some conclusions

- **Success is not related a single feature but rather to a combination of concepts**
- **Several search strategies are combined to achieve a good balance between search intensification and diversification**
- **Well-designed LS or LNS-based improvement methods are essential to refine the solutions**
  - **Computational complexity**. Do not confound *search* and *enumeration* (Bentley and Friedman, 1978; Bentley, 1992; De Berg et al., 2018)
  - Preprocessing, memories, neighborhood restrictions...

Some conclusions

- **Many components can contribute to increase performance**
  - ⇒ One can **always** improve a method by "adding more"...
  - ⇒ Success comes from a good tradeoff between performance and simplicity.
  - ⇒ To **gain methodological insights**, need to trim off all unnecessary component and avoid complex methodologies with only marginal contributions to performance.
  - ⇒ Computational experiments to assess the impact of each separate component

Some References

References:

1. T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Oper. Res. 60 (2012) 611–624.
2. T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, Heuristics for multi-attribute vehicle routing problems: A survey and synthesis, Eur. J. Oper. Res. 231 (2013) 1–21.
3. T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, Time-window relaxations in vehicle routing heuristics. Journal of Heuristics, 21(3) (2015) 329–358.

Contents

Contents

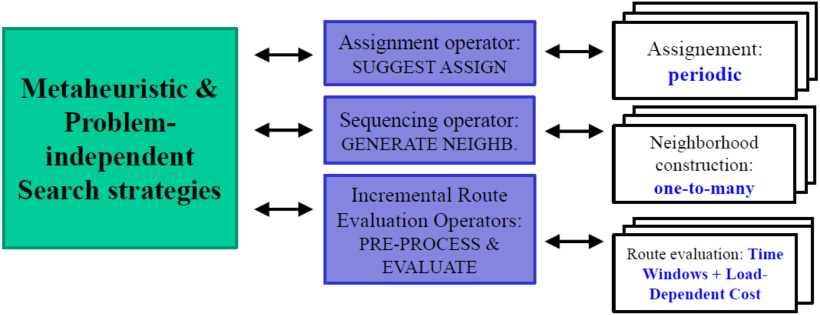
Unified Hybrid Genetic Search

- Unified Hybrid Genetic Search (UHGS): aiming to address the challenges related to the **variety of problem combinations** (Vidal et al., 2014)

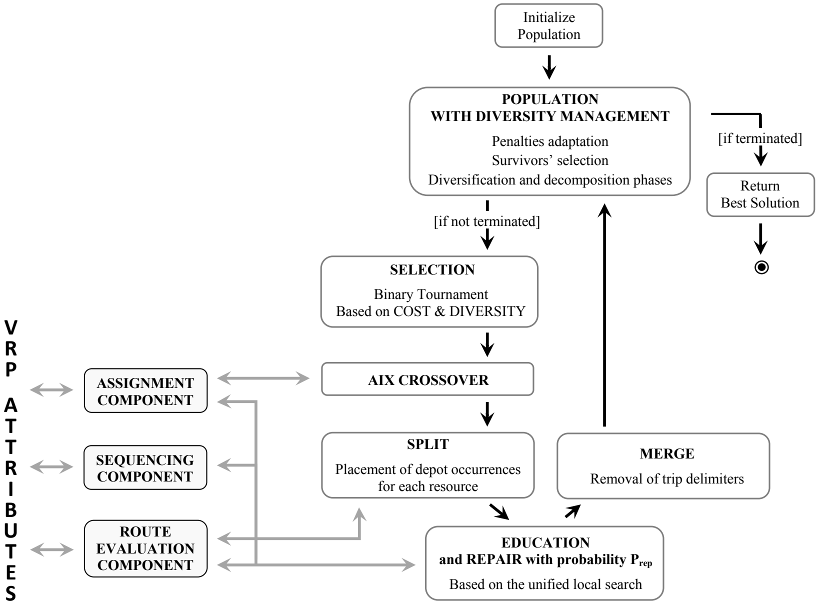
- Hybrid genetic search with Advanced Diversity Control (HGA):
  - Hybrid genetic Algorithm
  - Well-designed selection and crossover operators
  - High-performance local-improvement procedure ("education")
  - Management of penalized infeasible solutions in two subpopulations
  - **Diversity & Cost objective for individuals evaluations**

Unified Hybrid Genetic Search

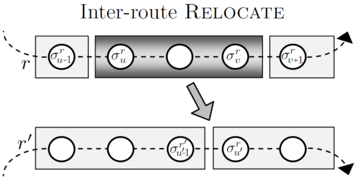
- The method relies on assignment, sequencing & route evaluation operators, which are selected and combined by the method relatively to the problem structure (using polymorphism and inheritance), to perform the assignment, sequencing and route evaluations.

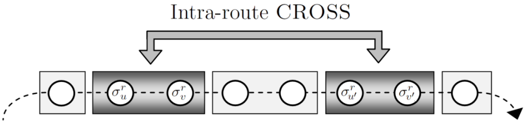


Unified Hybrid Genetic Search

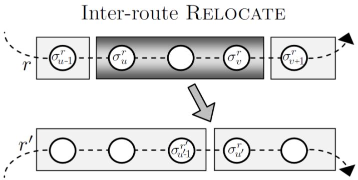


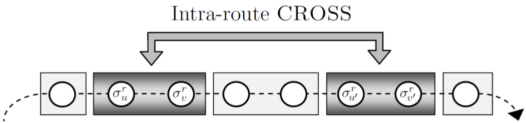
- One **important structural property of local searches** helps to progress towards **unified** and **efficient** metaheuristics:
  - Vidal et al. (2015b): Any LS move involving a bounded number of node relocations or arc exchanges can be assimilated to a recombination of a bounded number of consecutive visit sequences from the incumbent solution





Unified Hybrid Genetic Search





- Data preprocessing: compute auxiliary data on subsequences to speed up the search
- Evaluate moves by induction on the concatenation operator ($\oplus$)
- **Easy to adapt:**
  - Define all moves based on the concatenation operators
  - To deal with multiple problems: adapt the preprocessing and concatenation operators

Unified Hybrid Genetic Search

- Example 1) Distance and capacity constraints

Auxiliary data structures:

Partial loads $Q(\sigma)$ and distance $D(\sigma)$

Initialization

For a sequence σ_0 with a single visit v_i , $Q(\sigma_0) = q_i$ and $D(\sigma_0) = 0$

Induction Step:

$Q(\sigma_1 \oplus \sigma_2) = Q(\sigma_1) + Q(\sigma_2)$

$D(\sigma_1 \oplus \sigma_2) = D(\sigma_1) + d_{\sigma_1(|\sigma_1|)\sigma_2(1)} + D(\sigma_2)$

Unified Hybrid Genetic Search

- Example 2) Objectives based on cumulated arrival time objectives

Auxiliary data structures in use:

Travel time $D(\sigma)$, Cumulated arrival time $C(\sigma)$, Delay Cost $W(\sigma)$ associated to one unit of delay in starting time

Initialization

For a sequence σ_0 with a single visit v_i , $D(\sigma_0) = 0$ and $C(\sigma_0) = 0$, and $W(\sigma_0) = 1$ if v_i is a customer, and $W(\sigma_0) = 0$ if v_i is a depot visit.

Induction Step:

$$D(\sigma_1 \oplus \sigma_2) = D(\sigma_1) + d_{\sigma_1(|\sigma_1|)\sigma_2(1)} + D(\sigma_2)$$
$$C(\sigma_1 \oplus \sigma_2) = C(\sigma_1) + W(\sigma_2)(D(\sigma_1) + d_{\sigma_1(|\sigma_1|)\sigma_2(1)}) + C(\sigma_2)$$
$$W(\sigma_1 \oplus \sigma_2) = W(\sigma_1) + W(\sigma_2)$$

Unified Hybrid Genetic Search

- Example 3) Time windows (only feasibility check, see Vidal et al. 2013a for similar equations with penalized infeasibility)

Auxiliary data structures in use:

Travel time and service time $T(\sigma)$, earliest feasible completion time $E(\sigma)$, latest feasible starting date $L(\sigma)$, statement of feasibility $F(\sigma)$.

Initialization:

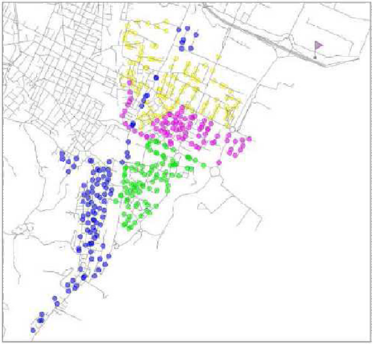
For a sequence σ_0 with a single visit v_i , $T(\sigma_0) = s_i$, $E(\sigma_0) = e_i + s_i$, $L(\sigma_0) = l_i$ and $F(\sigma_0) = true$.

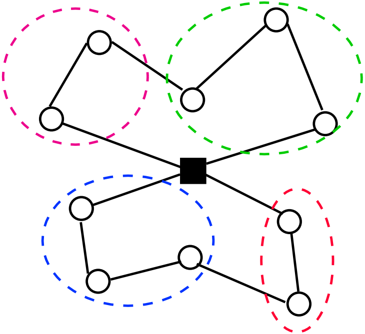
Induction Step:

$$
\begin{aligned}
T(\sigma_1 \oplus \sigma_2) &= T(\sigma_1) + d_{\sigma_1(|\sigma_1|)\sigma_2(1)} + T(\sigma_2) \\
E(\sigma_1 \oplus \sigma_2) &= \max\{E(\sigma_1) + d_{\sigma_1(|\sigma_1|)\sigma_2(1)} + T(\sigma_2), E(\sigma_2)\} \\
L(\sigma_1 \oplus \sigma_2) &= \min\{L(\sigma_1), L(\sigma_2) - d_{\sigma_1(|\sigma_1|)\sigma_2(1)} - T(\sigma_1)\} \\
F(\sigma_1 \oplus \sigma_2) &\equiv F(\sigma_1) \ \wedge \ F(\sigma_2) \ \wedge \ (E(\sigma_1) + d_{\sigma_1(|\sigma_1|)\sigma_2(1)} \leq L(\sigma_2))
\end{aligned}
$$

Unified Hybrid Genetic Search

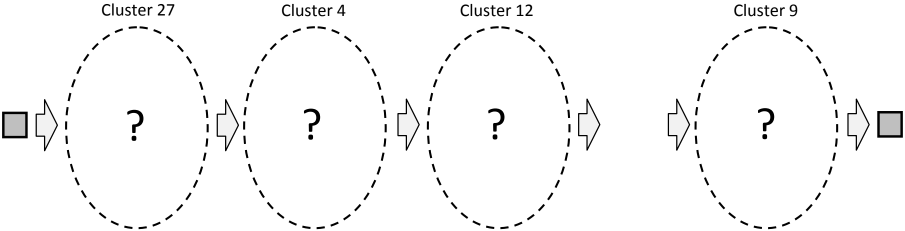
- Example 4) Clustered VRP (CluVRP)
- **Cluster constraint:** All visits of each cluster need to be **consecutive** and **in the same route**



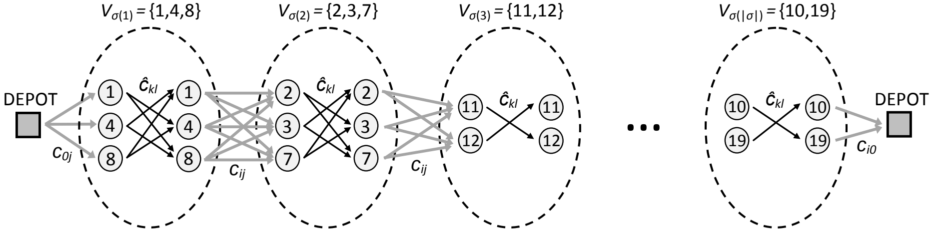


Unified Hybrid Genetic Search

- Example 4) Clustered VRP (CluVRP)
- We can work on solutions as sequences of clusters
  - ⇒ From the heuristic point of view, a solution looks like this:

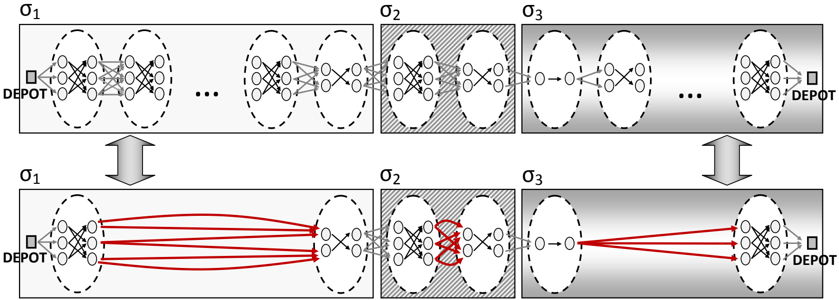


- Example 4) Clustered VRP (CluVRP)
- We can work on solutions as sequences of clusters
  - $\Rightarrow$ For route evaluation operator, it's a shortest path subproblem:



- Like this, a route evaluation would be in $\mathcal{O}(n)$, assuming that the number of customers in a cluster is bounded by a constant
  - Difficult to evaluate many solutions, need to do better.

- Example 4) Clustered VRP (CluVRP)
- Idea: Store shortest paths on partial sequences.
  - ⇒ To evaluate a move, solve a shortest path sub-problem with only $\mathcal{O}(1)$ arcs:



- Example 4) Clustered VRP (CluVRP)

Auxiliary data

Shortest path $S(\sigma)[i,j]$ inside sequence σ starting at the location i of the starting cluster and finishing at location j of the ending cluster

Initialization

For σ_0 with a single visit v_i , $S(\sigma_0)[i,j] = \begin{cases} +\infty & \text{if } i = j \\ \hat{c}_{ij} & \text{if } i \neq j \end{cases}$

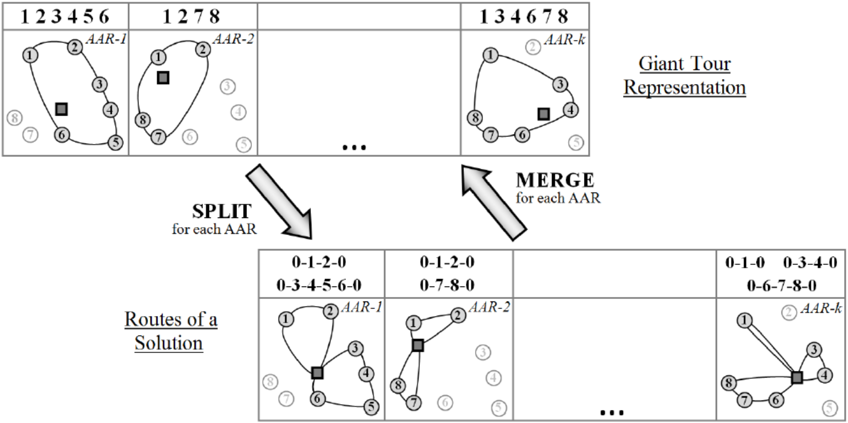
Induction step

By induction on the concatenation operator:

$$S(\sigma_1 \oplus \sigma_2)[i,j] = \min_{1 \leq x \leq \lambda_{\sigma_1(|\sigma_1|)}, 1 \leq y \leq \lambda_{\sigma_2(1)}} S(\sigma_1)[i,x] + c_{xy} + S(\sigma_2)[y,j]$$

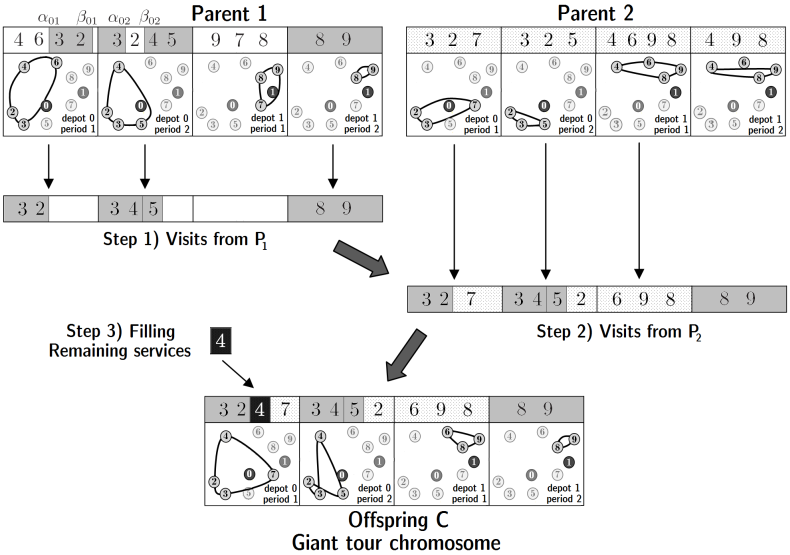
$$\forall i \in \{1, \ldots, \lambda_{\sigma_1(1)}\}, \forall j \in \{1, \ldots, \lambda_{\sigma_2(|\sigma_2|)}\}$$

- Solution representation and **Split**:



Unified Hybrid Genetic Search

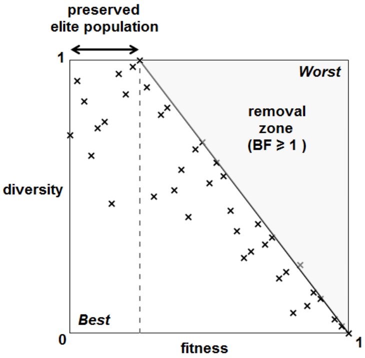
- Crossover operator:



Biased fitness: combining ranks in terms of solution cost $C(I)$ and contribution to the population diversity $D(I)$, measured as a distance to other individuals :

$$BF(I) = C(I) + \left(1 - \frac{nbElite}{popSize - 1}\right) D(I)$$

- Used for parents selection
  - $\Rightarrow$ Balancing quality with innovation to promote a more thorough exploration of the search space.
- Used during selection of survivors
  - $\Rightarrow$ Removing individuals with worst $BF(I)$ still guarantees elitism



Unified Hybrid Genetic Search

- **Computational Experiments:** UHGS has been tested on more than 2000 benchmark instances, and 50 different problems from the vehicle routing literature
- The method has been compared to over 240 previous algorithms
  - State-of-the-art results in the literature on all considered problems: VRP with capacity constraints, duration, backhauls, asymmetry, cumulative costs, simultaneous and mix pickup and deliveries, fleet mix, load dependency, multiple periods, depots, generalized deliveries, open routes, time windows, time-dependent travel time and costs, soft and multiple TW, truck driver scheduling regulations, many other problems and their combinations...
  - First method which addresses efficiently many problems and their combinations, equals or outperforms all available methods from the literature.

Unified Hybrid Genetic Search

Variant	Bench.	n	Obj.	State-of-the-art methods Author	Avg.%	Best%	T(min)	CPU
CVRP	CMT79	[50,199]	C	GG11:	—	+0.03%	8×2.38	8×Xe 2.3G
				MB07:	+0.03%	—	2.80	P-IV 2.8G
				UHGS*:	**+0.02%**	**+0.00%**	11.90	Opt 2.4G
CVRP	GWKC98	[200,483]	C	GG11:	—	+0.29%	8×5	8×Xe 2.3G
				NB09:	+0.27%	+0.16%	21.51	Opt 2.4G
				UHGS*:	**+0.15%**	**+0.02%**	71.41	Opt 2.4G
VRPB	GJ89	[25,200]	C	ZK12:	+0.38%	+0.00%	1.09	T5500 1.67G
				GA09:	+0.09%	+0.00%	1.13	Xe 2.4G
				UHGS:	**+0.01%**	**+0.00%**	0.99	Opt 2.4G
CCVRP	CMT79	[50,199]	C	NPW10:	+0.74%	+0.28%	5.20	Core2 2G
				RL12:	+0.37%	+0.07%	2.69	Core2 2G
				UHGS:	**+0.01%**	**-0.01%**	1.42	Opt 2.2G
CCVRP	GWKC98	[200,483]	C	NPW10:	+2.03%	+1.38%	94.13	Core2 2G
				RL12:	+0.34%	+0.07%	21.11	Core2 2G
				UHGS:	**-0.14%**	**-0.23%**	17.16	Opt 2.2G
VRPSDP	SN99	[50,199]	C	SDBOF10:	+0.16%	+0.00%	256×0.37	256×Xe 2.67G
				ZTK10:	—	+0.11%	—	T5500 1.66G
				UHGS:	**+0.01%**	**+0.00%**	2.79	Opt 2.4G
VRPSDP	MG06	[100,400]	C	SDBOF10:	+0.30%	+0.17%	256×3.11	256×Xe 2.67G
				UHGS:	+0.20%	+0.07%	12.00	Opt 2.4G
				S12 :	**+0.08%**	**+0.00%**	7.23	I7 2.93G

Unified Hybrid Genetic Search

Variant	Bench.	n	Obj.	State-of-the-art methods				
				Author	Avg.%	Best%	T(min)	CPU
VFMP-F	G84	[20,100]	C	ISW09:	—	+0.07%	8.34	P-M 1.7G
				SPUO12:	+0.12%	+0.01%	0.15	I7 2.93G
				UHGS:	**+0.04%**	**+0.01%**	1.13	Opt 2.4G
VFMP-V	G84	[20,100]	C	ISW09:	—	+0.02%	8.85	P-M 1.7G
				SPUO12:	+0.17%	+0.00%	0.06	I7 2.93G
				UHGS:	**+0.03%**	**+0.00%**	0.85	Opt 2.4G
VFMP-FV	G84	[20,100]	C	P09:	—	+0.02%	0.39	P4M 1.8G
				UHGS:	+0.01%	+0.00%	0.99	Opt 2.4G
				SPUO12:	**+0.01%**	**+0.00%**	0.13	I7 2.93G
LDVRP	CMT79	[50,199]	C	XZKX12:	+0.48%	+0.00%	1.3	NC 1.6G
				UHGS:	**-0.28%**	**-0.33%**	2.34	Opt 2.2G
LDVRP	GWKC98	[200,483]	C	XZKX12:	+0.66%	+0.00%	3.3	NC 1.6G
				UHGS:	**-1.38%**	**-1.52%**	23.81	Opt 2.2G
PVRP	CGL97	[50,417]	C	HDH09:	+1.69%	+0.28%	3.09	P-IV 3.2G
				UHGS*:	+0.43%	+0.02%	6.78	Opt 2.4G
				CM12:	**+0.24%**	**+0.06%**	64×3.55	64×Xe 3G
MDVRP	CGL97	[50,288]	C	CM12:	+0.09%	+0.03%	64×3.28	64×Xe 3G
				S12:	+0.07%	+0.02%	11.81	I7 2.93G
				UHGS*:	**+0.08%**	**+0.00%**	5.17	Opt 2.4G
GVRP	B11	[16,262]	C	BER11:	+0.06%	—	0.01	Opt 2.4G
				MCR12:	+0.11%	—	0.34	Duo 1.83G
				UHGS:	**+0.00%**	**-0.01%**	1.53	Opt 2.4G

Unified Hybrid Genetic Search

Variant	Bench.	n	Obj.	State-of-the-art methods Author	Avg.%	Best%	T(min)	CPU
OVRP	CMT79 &F94	[50,199]	F/C	RTBI10:	0%/+0.32%	—	9.54	P-IV 2.8G
				S12:	—/+0.16%	0%/+0.00%	2.39	I7 2.93G
				UHGS:	**0%/+0.11%**	**0%/+0.00%**	1.97	Opt 2.4G
OVRP	GWKC98	[200,480]	F/C	ZK10:	0%/+0.39%	0%/+0.21%	14.79	T5500 1.66G
				S12:	0%/+0.13%	0%/+0.00%	64.07	I7 2.93G
				UHGS:	**0%/-0.11%**	**0%/-0.19%**	16.82	Opt 2.4G
VRPTW	SD88	100	F/C	RTI09:	0%/+0.11%	0%/+0.04%	17.9	Opt 2.3G
				UHGS*:	0%/+0.04%	0%/+0.01%	2.68	Xe 2.93G
				NBD10:	**0%/+0.02%**	**0%/+0.00%**	5.0	Opt 2.4G
VRPTW	HG99	[200,1000]	F/C	RTI09b:	—	+0.16%/+3.36%	270	Opt 2.3G
				NBD10:	+0.20%/+0.42%	+0.10%/+0.27%	21,7	Opt 2.4G
				UHGS*:	**+0.18%/+0.11%**	**+0.08%/-0.10%**	141	Xe 2.93G
OVRPTW	SD88	100	F/C	RTI09a:	+0.89%/+0.42%	0%/+0.24%	10.0	P-IV 3.0G
				KTDHS12:	0%/+0.79%	0%/+0.18%	10.0	Xe 2.67G
				UHGS:	**+0.09%/-0.10%**	**0%/-0.10%**	5.27	Opt 2.2G
TDVRPTW	SD88	100	F/C	KTDHS12:	+2.25%	0%	10.0	Xe 2.67G
				UHGS:	**-3.31%**	**-3.68%**	21.94	Opt 2.2G
VFMPTW	LS99	100	D	BDHMG08:	—	+0.59%	10.15	Ath 2.6G
				RT10:	+0.22%	—	16.67	P-IV 3.4G
				UHGS:	**-0.15%**	**-0.24%**	4.58	Opt 2.2G
VFMPTW	LS99	100	C	BDHMG08:	—	+0.25%	3.55	Ath 2.6G
				BPDRT09:	—	+0.17%	0.06	Duo 2.4G
				UHGS:	-0.38%	-0.49%	4.82	Opt 2.2G

Unified Hybrid Genetic Search

Variant	Bench.	n	Obj.	State-of-the-art methods Author	Avg.%	Best%	T(min)	CPU
CVRP	CMT79	[50,199]	C	GG11:	—	+0.03%	8×2.38	8×Xe 2.3G
				MB07:	+0.03%	—	2.80	P-IV 2.8G
				UHGS*:	**+0.02%**	**+0.00%**	11.90	Opt 2.4G
CVRP	GWKC98	[200,483]	C	GG11:	—	+0.29%	8×5	8×Xe 2.3G
				NB09:	+0.27%	+0.16%	21.51	Opt 2.4G
				UHGS*:	**+0.15%**	**+0.02%**	71.41	Opt 2.4G
VRPB	GJ89	[25,200]	C	ZK12:	+0.38%	+0.00%	1.09	T5500 1.67G
				GA09:	+0.09%	+0.00%	1.13	Xe 2.4G
				UHGS:	**+0.01%**	**+0.00%**	0.99	Opt 2.4G
CCVRP	CMT79	[50,199]	C	NPW10:	+0.74%	+0.28%	5.20	Core2 2G
				RL12:	+0.37%	+0.07%	2.69	Core2 2G
				UHGS:	**+0.01%**	**-0.01%**	1.42	Opt 2.2G
CCVRP	GWKC98	[200,483]	C	NPW10:	+2.03%	+1.38%	94.13	Core2 2G
				RL12:	+0.34%	+0.07%	21.11	Core2 2G
				UHGS:	**-0.14%**	**-0.23%**	17.16	Opt 2.2G
VRPSDP	SN99	[50,199]	C	SDBOF10:	+0.16%	+0.00%	256×0.37	256×Xe 2.67G
				ZTK10:	—	+0.11%	—	T5500 1.66G
				UHGS:	**+0.01%**	**+0.00%**	2.79	Opt 2.4G
VRPSDP	MG06	[100,400]	C	SDBOF10:	+0.30%	+0.17%	256×3.11	256×Xe 2.67G
				UHGS:	+0.20%	+0.07%	12.00	Opt 2.4G
				S12 :	**+0.08%**	**+0.00%**	7.23	I7 2.93G

Contents

Tricks of the trade – Quick discussion

- A few lessons learned through the years:
  - **Neighborhood restrictions** or granular search – some "better ways" (Vidal et al., 2013a; Schneider et al., 2017)
  - When applicable: consider the RELOCATE, SWAP, 2-OPT, 2-OPT* moves as a single neighborhood – don't evaluate in different phases (Vidal et al., 2014)
  - Eliminate useless move re-evaluations: remember *when* a route was last modified and *when* a move was last tested (Vidal et al., 2014; Homsi et al., 2018)
  - Hash memories can help (Goel and Vidal, 2014; Toffolo et al., 2018)
  - **Move lower bounds – multi-phase evaluations** for harder problems (Vidal, 2017)

Some References

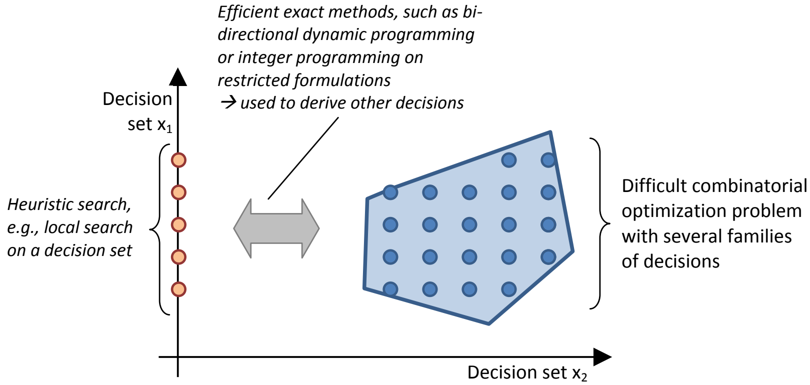
References:

1. T. Vidal, T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei, A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Oper. Res. 60 (2012) 611–624.
2. T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, Timing problems and algorithms: Time decisions for sequences of activities, Networks. 65 (2015) 102–128.
3. T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, Heuristics for multi-attribute vehicle routing problems: A survey and synthesis, Eur. J. Oper. Res. 231 (2013) 1–21.
4. T. Vidal, T.G. Crainic, M. Gendreau, C. Prins, A unified solution framework for multi-attribute vehicle routing problems, Eur. J. Oper. Res. 234 (2014) 658–673.

Contents

General Idea

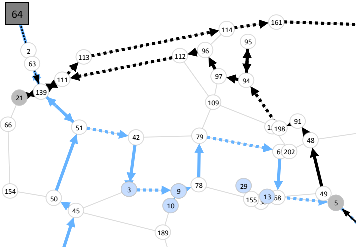
- Studying some other problems from the perspective of structural problem decomposition:

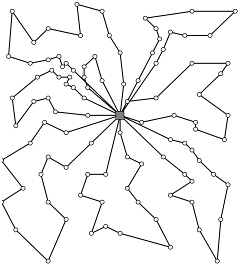


Contents

Challenges

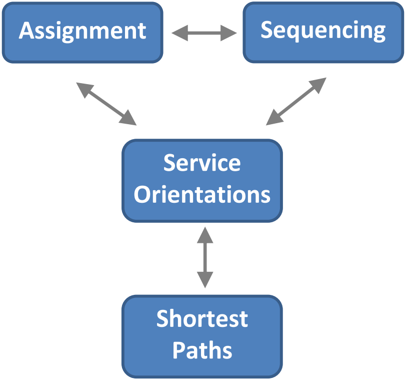
- Arc routing for home delivery, snow plowing, refuse collection, postal services, among others.
- Lead to additional challenges:
  - ⇒ *Deciding* on travel directions for services on edges
  - ⇒ Shortest path between services are *conditioned* by service orientations
    (may also need to include some additional aspects such as turn penalties or delays at intersections).





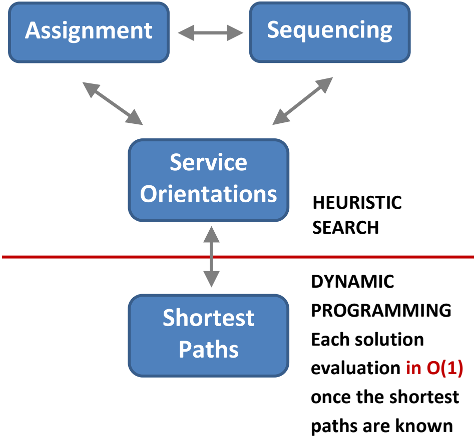
Challenges

- Arc routing for home delivery, snow plowing, refuse collection, postal services, among others.
- Lead to additional challenges:
  - ⇒ *Deciding* on travel directions for services on edges
  - ⇒ Shortest path between services are *conditioned* by service orientations
    (may also need to include some additional aspects such as turn penalties or delays at intersections).



A question of neighborhood

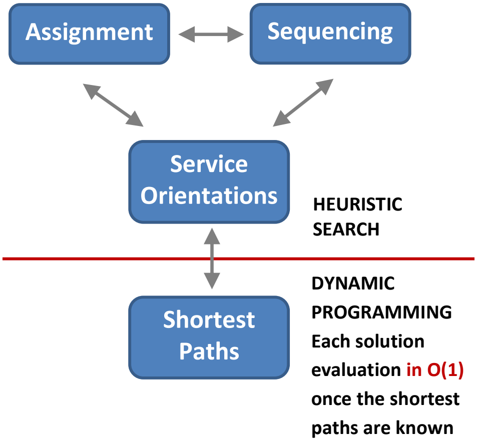
- Most **recent CARP heuristics rely on several enumerative neighborhood classes** to optimize assignment, sequencing and service orientation decisions
  - See, e.g. Brandão and Eglese (2008); Usberti et al. (2013); Dell'Amico et al. (2016)...
  - Shortest paths between node extremities have been pre-processed
  - Three decision classes are heuristically addressed



⇒ **This is, however, not the only option.**

A question of neighborhood

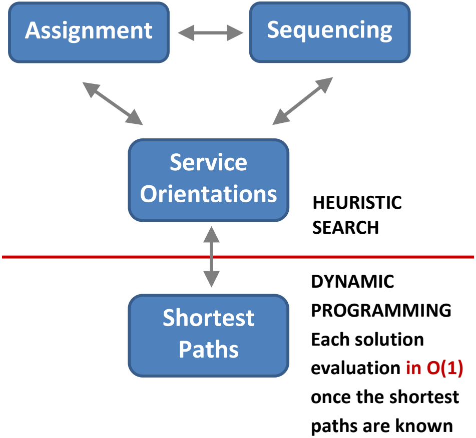
- Most **recent CARP heuristics rely on several enumerative neighborhood classes** to optimize assignment, sequencing and service orientation decisions
  - See, e.g. Brandão and Eglese (2008); Usberti et al. (2013); Dell'Amico et al. (2016)...
  - Shortest paths between node extremities have been pre-processed
  - Three decision classes are heuristically addressed



⇒ **This is, however, not the only option.**

A question of neighborhood

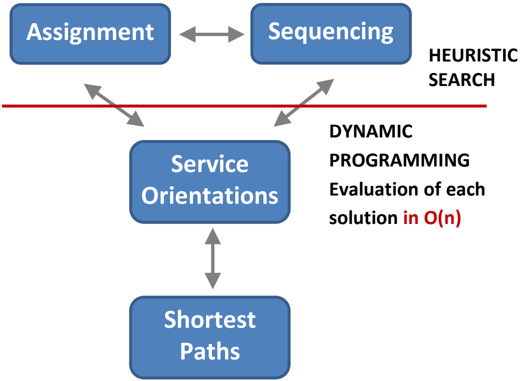
- Most **recent CARP heuristics rely on several enumerative neighborhood classes** to optimize assignment, sequencing and service orientation decisions
  - See, e.g. Brandão and Eglese (2008); Usberti et al. (2013); Dell'Amico et al. (2016)...
  - Shortest paths between node extremities have been pre-processed
  - Three decision classes are heuristically addressed



⇒ **This is, however, not the only option.**

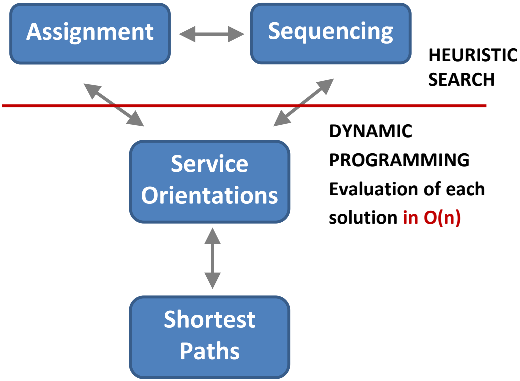
A question of neighborhood

- In Beullens et al. (2003) and Muyldermans et al. (2005), $O(n)$ dynamic-programming based optimization of service orientations:
- Combined in Irnich (2008) with the neighborhood of Balas and Simonetti (2001), leading to promising results on mail delivery applications.



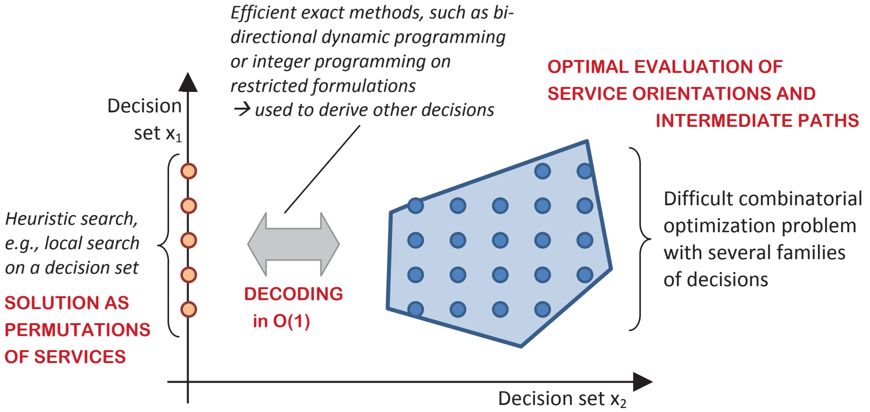
A question of neighborhood

- In Beullens et al. (2003) and Muyldermans et al. (2005), $O(n)$ dynamic-programming based optimization of service orientations:
- Combined in Irnich (2008) with the neighborhood of Balas and Simonetti (2001), leading to promising results on mail delivery applications.



A question of neighborhood

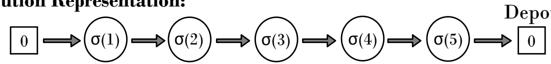
- Transferring several decision classes into exact dynamic-programming based components.
- This is a structural problem decomposition:



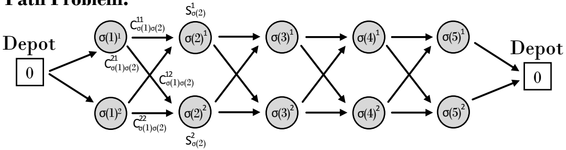
Solution representation and decoding

- How to decode/evaluate a solution = deriving optimal orientations for the services ?
  - ⇒ Simple dynamic programming subproblem (Beullens et al., 2003; Wøhlk, 2003, 2004):

Solution Representation:



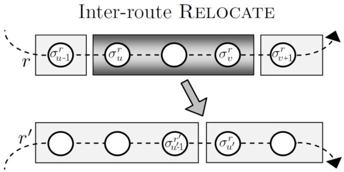
Shortest Path Problem:

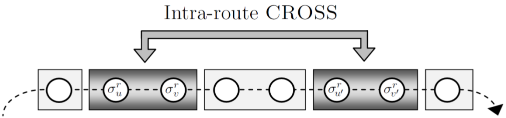


- Each service represented by two nodes, one for each orientation. Travel costs c_{ij}^{kl} between (i, j) are conditioned by the orientations (k, l) for departure and arrival.

Seeking low complexity for solution evaluations

- Modern neighborhood-centered heuristics evaluate millions/billions of neighbor solutions during one run.
- Back to our key property of classical routing neighborhoods:





Seeking low complexity for solution evaluations

Auxiliary data structures = partial shortest paths

Partial shortest path $C(\sigma)[k,l]$ between the first and last service in the sequence σ, for any (entry, exit) direction pair (k,l)

Initialization

For σ_0 with a single visit v_i , $S(\sigma_0)[k,l] = \begin{cases} 0 & \text{if } k = l \\ +\infty & \text{if } k \neq l \end{cases}$

Induction Step:

By induction on the concatenation operator:

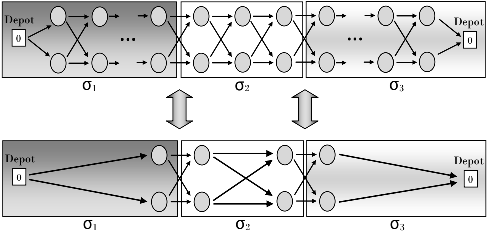
$$C(\sigma_1 \oplus \sigma_2)[k,l] = \min_{x,y} \left\{ C(\sigma_1)[k,x] + c^{xy}_{\sigma_1(|\sigma_1|)\sigma_2(1)} + C(\sigma_2)[y,l] \right\}$$

Seeking low complexity for solution evaluations

- **Pre-processing partial shortest paths in the incumbent solution** – in $\mathcal{O}(n^2)$ before the neighborhood exploration – dramatically simplifies the shortest paths:

Shortest path problem:

Shortest path problem on a reduced graph, using pre-processed labels:



- Only a constant number of edges

Lower bounds on moves

- Each move evaluation was still taking a bit more operations (constant of $4\times$) than in the classic CVRP.

- Even this can be avoided...
  $\Rightarrow$ by developing lower bounds on the cost of neighbors...

Lower bounds on moves

- Let $\bar{Z}(\sigma)$ be a lower bound on the cost of a route σ
- A move that modifies two routes: $\{\sigma_1, \sigma_2\} \Rightarrow \{\sigma'_1, \sigma'_2\}$ has a chance to be improving if and only if:

$$\Delta_\Pi = \bar{Z}(\sigma'_1) + \bar{Z}(\sigma'_2) - Z(\sigma_1) - Z(\sigma_2) < 0.$$

Lower bounds on moves

- Let $C^{\text{MIN}}(\sigma) = \min_{k,l}\{C(\sigma)[k,l]\}$ the shortest path for the sequence σ between any pair of origin/end orientations.
- Let $c_{ij}^{\text{MIN}} = \min_{k,l}\{c_{ij}^{kl}\}$ be the minimum cost of a shortest path between services i and j, for any orientation.
- Lower bound on the cost of a route $\sigma = \sigma_1 \oplus \dots \oplus \sigma_X$ composed of a concatenation of X sequences:

$$\bar{Z}(\sigma_1 \oplus \dots \oplus \sigma_X) = \sum_{j=1}^{X} C^{\text{MIN}}(\sigma_j) + \sum_{j=1}^{X-1} c_{\sigma_j,\sigma_{j+1}}^{\text{MIN}}.$$

- The bound helps to filter a lot of moves ($\geq 90\%$)
  - In practice : possible to evaluate a move with implicit service orientations for the CARP, using roughly the **same number of elementary operations as the same move for a CVRP!**

Lower bounds on moves

- Let $C^{\text{MIN}}(\sigma) = \min_{k,l} \{C(\sigma)[k,l]\}$ the shortest path for the sequence σ between any pair of origin/end orientations.
- Let $c_{ij}^{\text{MIN}} = \min_{k,l}\{c_{ij}^{kl}\}$ be the minimum cost of a shortest path between services i and j, for any orientation.
- Lower bound on the cost of a route $\sigma = \sigma_1 \oplus \cdots \oplus \sigma_X$ composed of a concatenation of X sequences:

$$\bar{Z}(\sigma_1 \oplus \cdots \oplus \sigma_X) = \sum_{j=1}^{X} C^{\text{MIN}}(\sigma_j) + \sum_{j=1}^{X-1} c_{\sigma_j,\sigma_{j+1}}^{\text{MIN}}.$$

- The bound helps to filter a lot of moves ($\geq 90\%$)
  - In practice : possible to evaluate a move with implicit service orientations for the CARP, using roughly the **same number of elementary operations as the same move for a CVRP!**

Experimental setting

- Initial experiments on CARP and MCGRP
- Literature on CARP and MCGRP built around several sets of well-known benchmark instances:

	#	Reference	$\|N_R\|$	$\|E_R\|$	$\|A_R\|$	n	Specificities
CARP:							
GDB	(23)	Golden et al. (1983)	0	[11,55]	0	[11,55]	Random graphs; Only required edges
VAL	(34)	Benavent et al. (1992)	0	[39,97]	0	[39,97]	Random graphs; Only required edges
BMCV	(100)	Beullens et al. (2003)	0	[28,121]	0	[28,121]	Intercity road network in Flanders
EGL	(24)	Li and Eglese (1996)	0	[51,190]	0	[51,190]	Winter-gritting application in Lancashire
EGL-L	(10)	Brandão and E. (2008)	0	[347,375]	0	[347,375]	Larger winter-gritting application
MCGRP:							
MGGDB	(138)	Bosco et al. (2012)	[3,16]	[1,9]	[4,31]	[8,48]	From CARP instances GBD
MGVAL	(210)	Bosco et al. (2012)	[7,46]	[6,33]	[12,79]	[36,129]	From CARP instances VAL
CBMix	(23)	Prins and B. (2005)	[0,93]	[0,94]	[0,149]	[20,212]	Randomly generated planar networks
BHW	(20)	Bach et al. (2013)	[4,50]	[0,51]	[7,380]	[20,410]	From CARP instances GDB, VAL, & EGL
DI-NEARP	(24)	Bach et al. (2013)	[120,347]	[120,486]	0	[240,833]	Newspaper and media product distribution

Experimental setting

- For each benchmark set, we collected the best three solution methods in the literature (some are heavily tailored for specific benchmark sets).

BE08	Brandão and Eglese (2008)	HKSG12	Hasle et al. (2012)	MTY09	Mei et al. (2009)
BLMV14	Bosco et al. (2014)	LPR01	Lacomme et al. (2001)	PDHM08	Polacek et al. (2008)
BMCV03	Beullens et al. (2003)	MLY14	Mei et al. (2014)	TMY09	Tang et al. (2009)
DHDI14	Dell'Amico et al. (2016)	MPS13	Martinelli et al. (2013)	UFF13	Usberti et al. (2013)

- Comparison with the proposed metaheuristics, which are searching the space of service permutations (our methods are not fine-tuned for any of these instance sets).

Comparison with previous literature

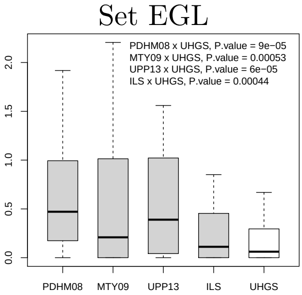
Variant	Bench.	n	Author	Runs	Avg.	Best	T	T*	CPU
CARP	GDB	[11,55]	TMY09	*30*	0.009%	0.000%	0.11	—	Xe 2.0G
			BMCV03	*1*	0.000%	—	—	0.03	P-II 500M
			MTY09	*1*	0.000%	—	—	0.01	Xe 2.0G
			ILS	*10*	0.002%	0.000%	0.16	0.03	Xe 3.07G
			UHGS	*10*	**0.000%**	**0.000%**	0.22	0.01	Xe 3.07G
	VAL	[39,97]	MTY09	*1*	0.142%	—	—	0.11	Xe 2.0G
			LPR01	*1*	0.126%	—	2.00	—	P-III 500M
			BMCV03	*1*	0.060%	—	—	1.36	P-II 500M
			ILS	*10*	0.054%	0.024%	0.68	0.16	Xe 3.07G
			UHGS	*10*	**0.048%**	**0.021%**	0.82	0.08	Xe 3.07G
	BMCV	[28,121]	BE08	*1*	0.156%	—	—	1.08	P-M 1.4G
			MTY09	*1*	0.073%	—	—	0.35	Xe 2.0G
			BMCV03	*1*	0.036%	—	2.57	—	P-II 450M
			ILS	*10*	0.027%	0.000%	0.82	0.22	Xe 3.07G
			UHGS	*10*	**0.007%**	**0.000%**	0.87	0.11	Xe 3.07G
	EGL	[51,190]	PDHM08	*10*	0.624%	—	30.0	8.39	P-IV 3.6G
			UFF13	*15*	0.560%	0.206%	13.3	—	I4 3.0G
			MTY09	*1*	0.553%	—	—	2.10	Xe 2.0G
			ILS	*10*	0.236%	0.106%	2.35	1.33	Xe 3.07G
			UHGS	*10*	**0.153%**	**0.058%**	4.76	3.14	Xe 3.07G
	EGL-L	[347,375]	BE08	*1*	4.679%	—	—	17.0	P-M 1.4G
			MPS13	*10*	2.950%	2.523%	20.7	—	I5 3.2G
			MLY14	*30*	1.603%	0.895%	33.4	—	I7 3.4G
			ILS	*10*	0.880%	0.598%	23.6	15.4	Xe 3.07G
			UHGS	*10*	**0.645%**	**0.237%**	36.5	27.5	Xe 3.07G

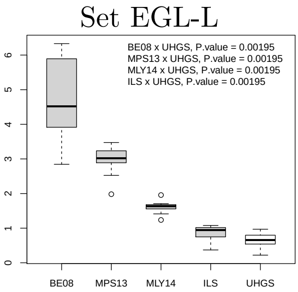
Comparison with previous literature

Variant	Bench.	n	Author	Runs	Avg.	Best	T	T*	CPU
MCGRP	MGGDB	[8,48]	BLMV14	*1*	1.342%	—	0.31	—	Xe 3.0G
			DHDI14	*1*	0.018%	—	60.0	0.86	CPU 3G
			ILS	*10*	0.010%	0.000%	0.13	0.03	Xe 3.07G
			UHGS	*10*	**0.015%**	**0.000%**	0.16	0.01	Xe 3.07G
	MGVAL	[36,129]	BLMV14	*1*	2.620%	—	16.7	—	Xe 3.0G
			DHDI14	*1*	0.071%	—	60.0	3.69	CPU 3G
			ILS	*10*	0.067%	0.019%	1.18	0.32	Xe 3.07G
			UHGS	*10*	**0.045%**	**0.011%**	1.20	0.17	Xe 3.07G
	CBMix	[20,212]	HKSG12	*2*	—	3.076%	120	56.9	CPU 3G
			BLMV14	*1*	2.697%	—	44.7	—	Xe 3.0G
			DHDI14	*1*	0.884%	—	60.0	19.6	CPU 3G
			ILS	*10*	0.733%	0.363%	2.46	1.48	Xe 3.07G
			UHGS	*10*	**0.381%**	**0.109%**	4.56	3.08	Xe 3.07G
	BHW	[20,410]	HKSG12	*2*	—	1.949%	120	60.1	CPU 3G
			DHDI14	*1*	0.555%	—	60.0	21.4	CPU 3G
			ILS	*10*	0.440%	0.196%	5.22	2.90	Xe 3.07G
			UHGS	*10*	**0.208%**	**0.077%**	7.95	5.87	Xe 3.07G
	DI-NEARP	[240,833]	HKSG12	*2*	—	1.639%	120	93.0	CPU 3G
			DHDI14	*1*	0.536%	—	60.0	36.3	CPU 3G
			ILS	*10*	0.199%	0.084%	30.0	21.3	Xe 3.07G
			UHGS	*10*	**0.139%**	**0.055%**	29.6	16.7	Xe 3.07G

Comparison with previous literature

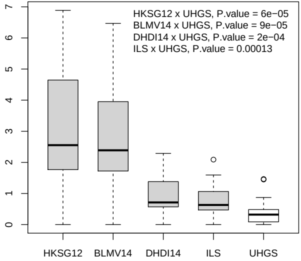
- Boxplot visualizations of Gap(%) of various methods on large-scale instances:
- Gray colors indicate a significant difference of performance, as highlighted by pairwise Wilcoxon tests with adequate correction for multiplicity



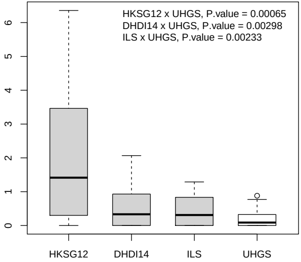


Comparison with previous literature

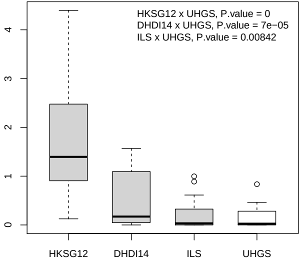
Set CBMix



Set BHW

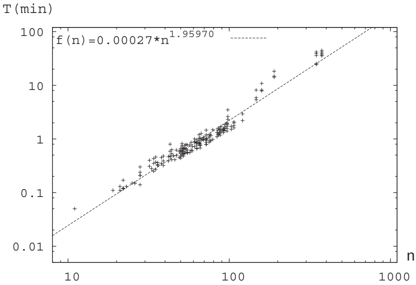


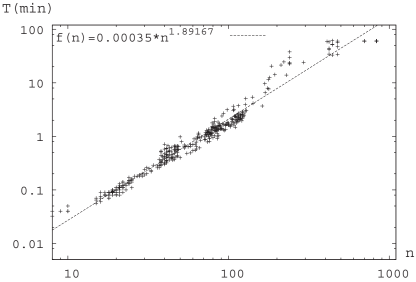
Set DI-NEARP



Scalability

- Growth of the CPU time of UHGS as a function of the number of services, for the CARP instances (left figure) and MCGRP instances (right figure). Log-log scale.





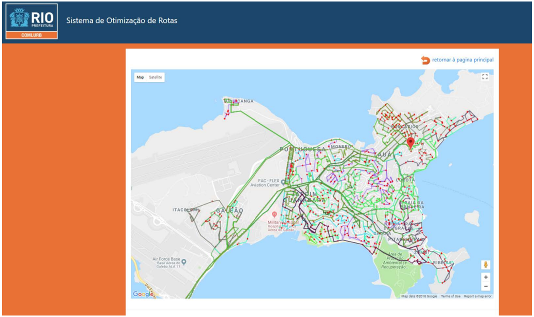
- A linear fit, with a least square regression, has been performed on the sample after logarithmic transformation:
  $\Rightarrow$ CPU time appears to grow in $\mathcal{O}(n^2)$

Real-case application

- Currently being used as the optimization core for a refuse collection application in Rio de Janeiro

⇒ Multiple periods, multiple trips, heterogeneous vehicle types, access restrictions, risk areas, congestion...

- 5 minute CPU time for graphs containing thousands of requests



Some References

References:

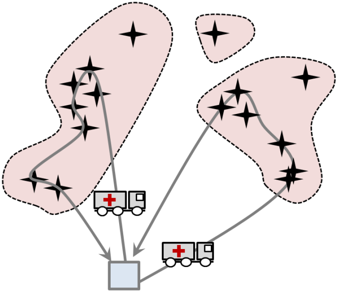
1. T. Vidal, Node, edge, arc routing and turn penalties : Multiple problems – One neighborhood extension, Oper. Res. 65 (2017) 992–1010.

Contents

Large Neighborhoods: team-orienteering problem

- **Team-orienteering problem**:
  - Each customer i is associated with a prize p_i. Not all customers are to be serviced.
  - Each route must have a distance of less than D.
  - The goal is to generate m feasible routes while maximizing the total amount of prizes

- Numerous applications, including:
  - Logistics, third party providers, secondary market (Tricoire et al., 2010; Aksen et al., 2012)
  - Humanitary relief (Campbell et al., 2008)
  - Robotics, maintenance & military surveillance (Falcon et al., 2012; Mufalli et al., 2012).



Large Neighborhoods: Team-orienteering Problem

- Large amount of literature on TOP heuristics and metaheuristics

Acronym	Authors	Methodology
CGW	Chao et al. (1996)	Tabu Search
TMH	Tang and Miller-Hooks (2005)	Tabu Search
GTF	Archetti et al. (2007)	Tabu Search & VNS
ASe	Ke et al. (2008)	Ant colony optimization
BDM	Bouly et al. (2009)	Memetic Algorithm
GLS	Vansteenwegen et al. (2009)	Guided Local Search
SVNS	Vansteenwegen and Souffriau (2009)	Skewed VNS
SPR	Souffriau et al. (2010)	Path Relinking
DGM	Dang et al. (2011)	Particle Swarm Optimization
MSA	Lin (2013)	Multi-Start Simulated Annealing

Table: Metaheuristics for team-orienteering problems

Large Neighborhoods: Team-orienteering Problem

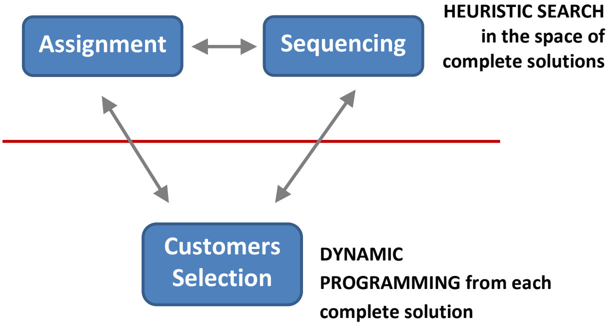
- **Main Idea : always work on a full solution with all visits**
  - Q : How will customers be selected ?
  - A : Directly during separate route evaluations

- The problem of optimally selecting the customers from a complete solution can be assimilated to a shortest path with maximum profit under distance constraints for each route.

- We propose efficient techniques to solve this problem, combining
  - bi-directional dynamic programming,
  - graph sparsification,
  - and data preprocessing techniques.

Large Neighborhoods: Team-orienteering Problem

- Again a structural decomposition:

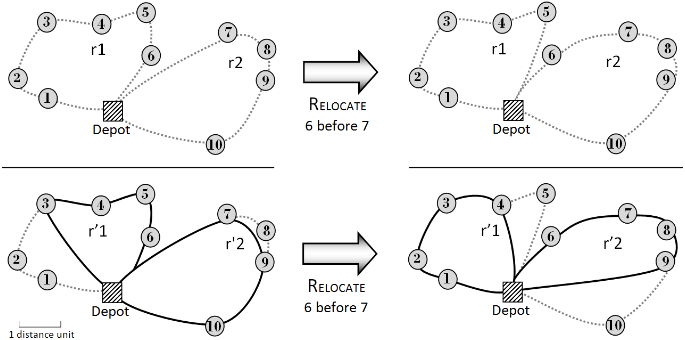


Large Neighborhoods: Team-orienteering Problem

- **Main interest:** Classic VRP neighborhoods on the complete solution representation ⇔ large neighborhoods with an exponential number of implicit insertions and removal of visits.
- SELECT algorithm at each move ⇔ resource-constrained SP

i	$d_{0,i}$	$d_{i-1,i}$	p_i
1	15	–	10
2	25	30	15
3	15	20	15
4	15	20	10
5	20	25	12
6	15	10	15
7	20	15	15
8	25	15	12
9	25	20	15
10	15	35	15
$D_{max} = 100$			
$d_{7,9} = 25$			
all other distances $= +\infty$			

σ	$D(\sigma)$	$P(\sigma)$
(3,4,5,6)	85	52
(7,9,10)	95	45
(1,2,3,4)	100	50
(6,7,8,9)	90	57



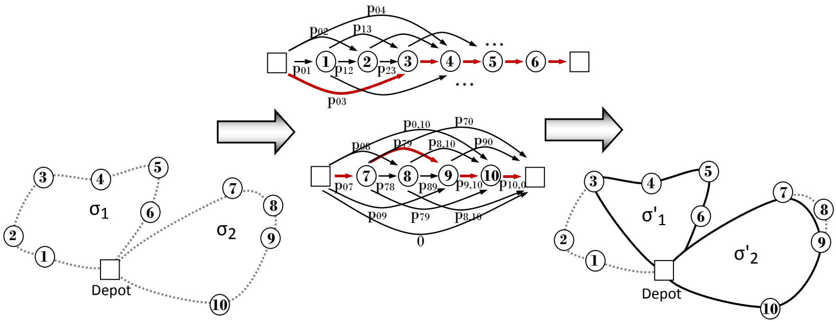
Large Neighborhoods: Team-orienteering Problem

Proposition

Let B be an upper bound on the number of labels per node. Then, the Select *algorithm is pseudo-polynomial, with a complexity of*

$$O(n^2 B). \tag{4.1}$$

- In practice the number of labels remains very small, i.e., $B \leq 10$.



Large Neighborhoods: Team-orienteering Problem

- Using a particular hierarchical cost function which considers in priority the Team-Orienteering cost (with only selected customers), and then the VRP cost with all customers.

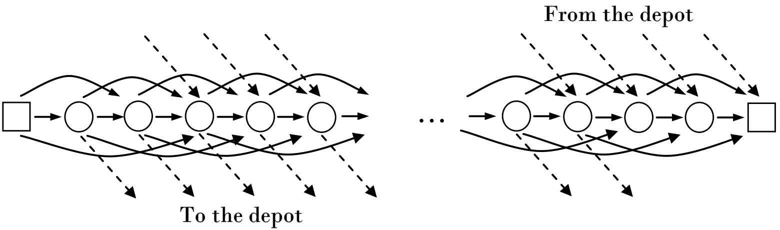
$$Z' = \max \sum_{\sigma \in \mathcal{R}} Z^{\text{SELECT}}(\sigma) - \omega \sum_{\sigma \in \mathcal{R}} \sum_{i \in \{1,\dots,|\sigma|-1\}} d_{\sigma(i)\sigma(i+1)}$$

- As a consequence, when the method is unable to improve the primary objective, moves may still be performed to improve the second objective = the positions of unserviced customers.
- This may lead in turn to a new repartition of customers and new opportunities of improvement of the main objective.

- Speed-ups for move evaluations – 1. Graph Sparsification
  - For a given *sparsification parameter* $H \in \{1, \dots, n\}$, only the arcs (i, j), with $(i < j)$ satisfying Equation (4.2) are kept.

$$j < i + H \text{ or } i = 0 \text{ or } j = |\sigma| \tag{4.2}$$

  - H is a sparsification parameter, usually small, e.g. $H = 3$.
  - Thus there are only O(Hn) arcs



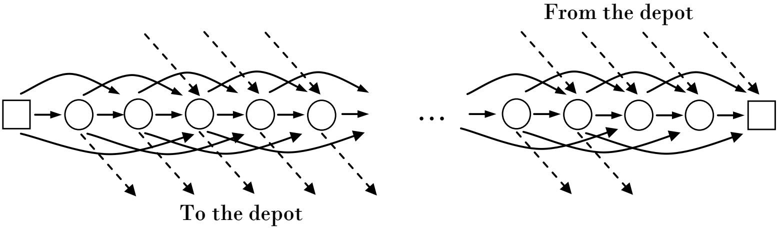
Large Neighborhoods: Team-orienteering Problem

- Speed-ups for move evaluations – 1. Graph Sparsification

Proposition

After sparsification, the number of arcs $|\mathcal{A}'|$ in the new graph becomes $O(nH)$, and the complexity of SELECT*, in terms of number of elementary operations, is*

$$O(nHB). \tag{4.3}$$



Large Neighborhoods: Team-orienteering Problem

- Speed-ups for move evaluations – 2. Evaluation by Concatenation
- For any sequence σ of successive nodes from the incumbent solution, we propose to pre-process the following information :

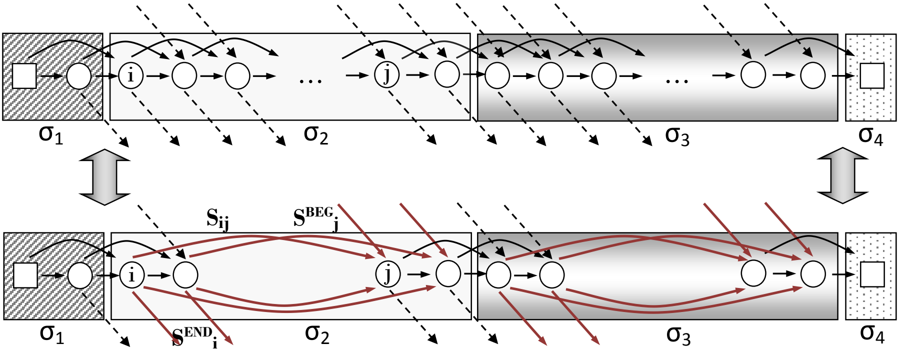
Auxiliary data structures in use:

- Set of labels $S_{ij}(\sigma)$ associated to each resource-constrained path (i, j) between any node among the H first of σ, and any node among the H last of σ.
- Set of labels $S_i^{\text{END}}(\sigma)$ associated to each resource-constrained path between any node among the H first nodes of σ and the ending depot.
- Set of labels $S_j^{\text{BEG}}(\sigma)$ associated to each resource-constrained path between the beginning depot and any node among the H last of σ.
- Best profit $Z(\sigma)$ of a inside resource-feasible path in σ, starting from the depot, visiting a subset of customers in σ, and coming back to the depot.

Large Neighborhoods: Team-orienteering Problem

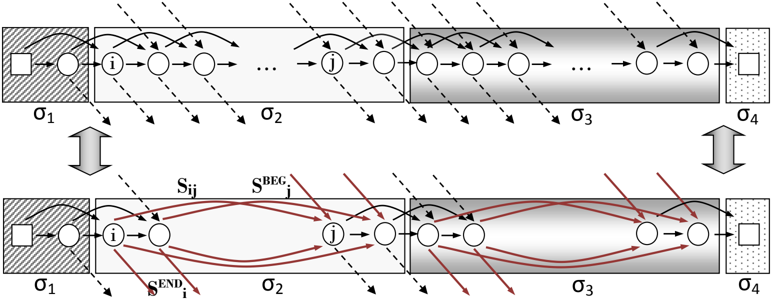
Initialization and Pre-processing:

Preprocessing these values for a sequence σ requires $O(n^2HB)$ elementary operations



Initialization and Pre-processing:

Preprocessing these values for a sequence σ requires $O(n^2 HB)$ elementary operations



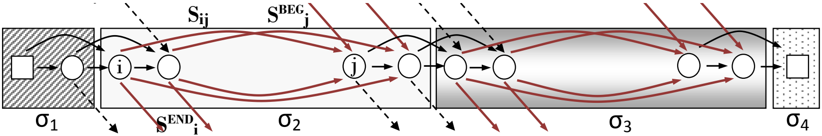
- The resulting *reduced* multi-graph $\mathcal{G}'' = (\mathcal{V}'', \mathcal{A}'')$ is such that $|\mathcal{A}''| = O(MH^2)$ arcs and $|\mathcal{V}''| = O(MH)$ nodes. M is the number of subsequences.

Large Neighborhoods: Team-orienteering Problem

Proposition (**Concatenation – general**)

The optimal profit $Z(\sigma_1 \oplus \cdots \oplus \sigma_M)$ *of* Select, *for a combination of* M *sequences is the maximum between the profit* $\bar{Z}(\sigma_1 \oplus \cdots \oplus \sigma_M)$ *of the resource-constrained shortest path in* $\mathcal{G}''$, *and the maximum profit* $Z(\sigma_i)$ *of an inside resource-feasible path in* σ_i *for* $i \in \{1, \ldots, M\}$. *Furthermore,* $\bar{Z}(\sigma_1 \oplus \cdots \oplus \sigma_M)$ *can be evaluated in*

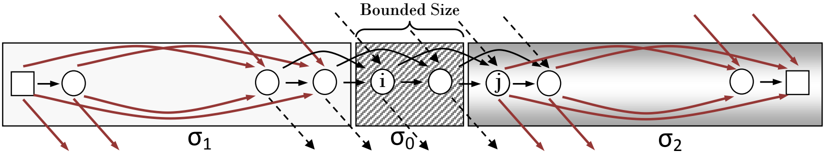
$$\Phi_{\text{C-M}} = O(MH^2B^2). \tag{4.4}$$



Proposition (Concatenation – 2 or 3 subsequences)

The optimal cost $Z(\sigma_1 \oplus \sigma_0 \oplus \sigma_2)$ of Select*, for a route assimilated to a recombination of three subsequences σ_1, σ_0 and σ_2 such that σ_0 contains a bounded number of customers can be evaluated using bi-directional dynamic programming for a complexity of*

$$\Phi_{\text{C-3}} = O(H^2 B). \tag{4.5}$$



- The same complexity is achieved for a concatenation of two sequences σ_1 and σ_2.

Computational Experiments

- Experimental analysis of three heuristics and metaheuristics based on our large-neighborhood concepts
  - A simple local search (LS), restarted 100 times.
  - An Iterated Local Search (ILS), based on Prins (2009)
  - Unified Hybrid Genetic Search (UHGS) of Vidal et al. (2014)
- Benchmark instances:
  - Chao et al. (1996) for the TOP : 7 groups of instances. Groups 4-7 are the largest with 64 to 102 customers.
  - Bolduc et al. (2008) for a variant called VRP with private fleet and common carrier. These instances are derived from the CVRP instances of Christofides et al. (1979) and Golden et al. (1998).
- Tests conducted on a single Xeon 3.0GHz processor.
- Method performance evaluated relatively to Gap to Best Known Solutions BKS and CPU time.

Computational Experiments

Table: Summary of results on TOP benchmark instances

	CGW	**TMH**	**GTF**	**SVF**	**ASe**	**SVNS**	**SPR**	**MSA**	**UHGS**	**ILS**	**LI**
Best Gap 4	4.36%	1.99%	0.48%	0.06%	0.30%	1.46%	0.11%	0.06%	**0.01%**	0.05%	0.09%
Best Gap 5	1.36%	1.38%	0.01%	0.03%	0.04%	0.61%	0.05%	0.01%	**0.00%**	0.01%	0.01%
Best Gap 6	0.37%	0.79%	0.04%	0.00%	0.00%	0.52%	0.00%	0.00%	**0.00%**	0.00%	0.00%
Best Gap 7	2.68%	1.15%	0.29%	0.06%	0.00%	1.31%	0.04%	0.03%	**0.00%**	0.00%	0.07%
Avg Time 4	796.70	105.30	22.50	11.40	32.00	36.70	367.40	81.00	298.57	301.54	76.72
Avg Time 5	71.30	69.50	34.20	3.50	15.10	11.20	119.90	6.60	222.92	193.97	11.31
Avg Time 6	45.70	66.30	8.70	4.30	14.10	9.00	89.60	1.40	184.60	138.25	6.86
Avg Time 7	432.60	160.00	10.30	12.10	24.60	27.30	272.80	32.20	306.35	309.62	50.22

- Equaled or improved 380 of the 387 best known solutions.
- 4 new BKS, quite surprising since the problems have been studied by dozens of previous papers

Table: Highlight of the results on some of the most difficult problems

Inst	CGW	TMH	GTF	SVF	ASe	SVNS	SPR	MSA	UHGS	ILS	LI	BKS
p4.2.o	1147	1175	1192	**1218**	1215	1195	**1218**	1217	**1218**	**1218**	**1218**	1218
p4.2.p	1199	1208	1239	1241	**1242**	1237	**1242**	1241	1241	1241	1241	1242
p4.2.q	1242	1255	1255	1263	1263	1239	1263	1259	**1267**	1265	1265	1265
p4.2.r	1199	1277	1283	1285	1288	1279	1286	**1290**	1286	1281	1285	1290
p4.2.s	1286	1294	1299	1301	**1304**	1295	1296	1300	1302	1297	1301	1304
p4.2.t	1299	**1306**	**1306**	**1306**	**1306**	1305	**1306**	**1306**	**1306**	**1306**	**1306**	1306
p4.3.o	1078	1151	1157	**1172**	1170	1136	1170	1170	**1172**	**1172**	1170	1172
p4.3.p	1115	1218	1221	**1222**	1221	1200	1220	**1222**	**1222**	**1222**	**1222**	1222
p4.3.q	1222	1249	1241	1245	1252	1236	**1253**	1251	**1253**	**1253**	1251	1253
p4.3.r	1225	1265	1269	1273	1267	1250	1272	1265	**1273**	1272	1269	1272
p4.3.s	1239	1282	1294	**1295**	1293	1280	1287	1293	**1295**	**1295**	**1295**	1295
p4.3.t	1285	1288	1304	1304	**1305**	1299	1299	1299	**1305**	**1305**	1299	1305
p4.4.o	995	1014	1057	**1061**	1036	1030	1057	**1061**	**1061**	**1061**	**1061**	1061
p4.4.p	996	1056	1120	1120	1111	1120	1122	**1124**	**1124**	**1124**	**1124**	1124
p4.4.q	1084	1124	1157	**1161**	1145	1149	1160	**1161**	**1161**	**1161**	1157	1161
p4.4.r	1155	1165	1211	1207	1200	1193	1213	**1216**	**1216**	**1216**	1211	1216
p4.4.s	1230	1243	1256	1260	1249	1213	1250	1256	**1260**	**1260**	**1260**	1259
p4.4.t	1253	1255	**1285**	**1285**	1281	1281	1280	**1285**	**1285**	**1285**	**1285**	1285
Best Gap	4.36%	1.99%	0.48%	0.06%	0.30%	1.46%	0.11%	0.06%	0.01%	0.05%	0.09%	
Avg Time	796.70	105.30	22.50	457.90	32.00	36.70	367.40	81.00	298.57	301.54	76.72	

Some References

References:

1. T. Vidal, N. Maculan, L.S. Ochi, P.H.V. Penna, Large neighborhoods with implicit customer selection for vehicle routing problems with profits, Transp. Sci. 50 (2016) 720–734.

Contents

CVRP = Assignment + Sequencing

- Most state-of-the-art CVRP metaheuristics built on a combination of inter- and intra-route neighborhoods, usually simple variations of Swap, Relocate, Cross-Echanges, 2-opt and 2-opt*

- These neighborhoods alone are generally sufficient to obtain TSP-optimal routes for classical benchmark instances (rarely contain over 20 customers per route)
  $\Rightarrow$ Larger intra-route neighborhoods are not commonly used

- Does this mean that we should consider Sequencing optimization a "solved case" and focus on Assignment optimization in majority ?

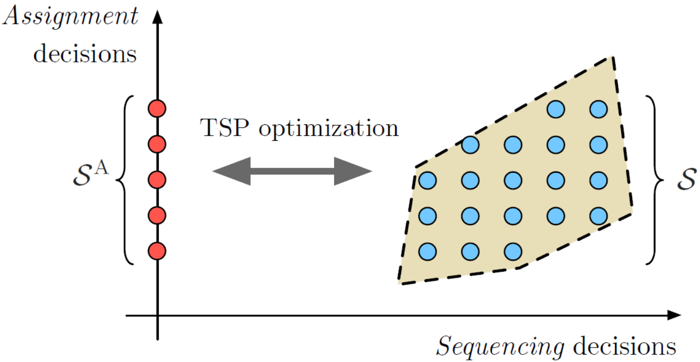
CVRP = Assignment + Sequencing

- Does this mean that we should consider Sequencing optimization a "solved case" and focus on Assignment optimization in majority ?

- Inter-route moves often lead to TSP-suboptimal tours which are **rejected** due to their higher cost, but could be **accepted** if the tours were simultaneously optimized
  $\Rightarrow$ (see, e.g., GENI – Gendreau et al. 1994).

CVRP = Assignment + Sequencing

- The two decision sets – Assignments and Sequencing – can be used to decompose the problem and define search space $\mathcal{S}^A$
- Assignments decisions are decoded into complete solutions by a TSP solver:



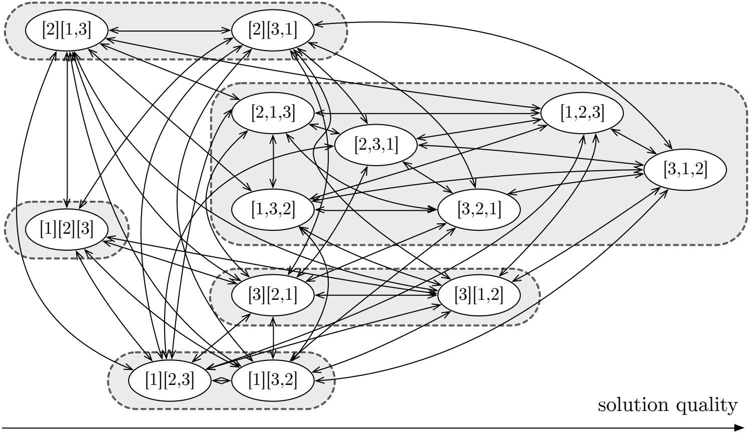
Our quest...

- **Question 1: Is it practical and worthwhile to search in $\mathcal{S}^A$ rather than $\mathcal{S}$?**

- **Question 2: If searching in $\mathcal{S}^{\mathrm{A}}$ requires too much effort, can we define an intermediate search space with some properties of $\mathcal{S}^{\mathrm{A}}$ but easier to explore?**

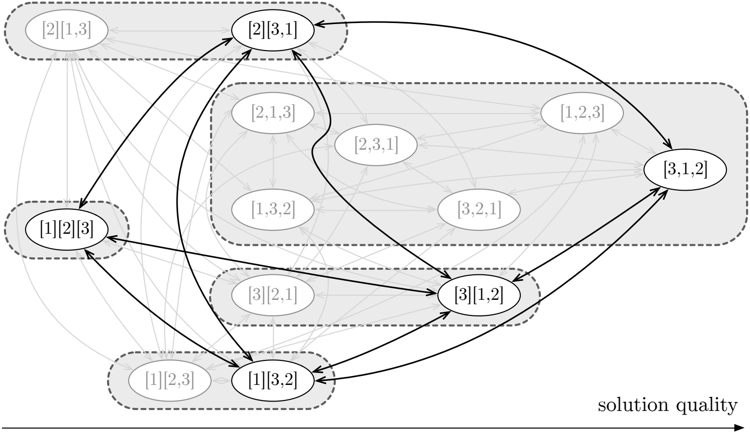
Small example with 3 customers

- From $\mathcal{S}$ to $\mathcal{S}^A$: a much smaller search space



Small example with 3 customers

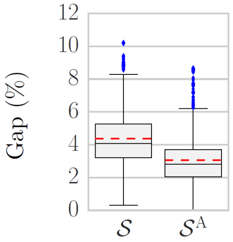
- From $\mathcal{S}$ to $\mathcal{S}^A$: a much smaller search space



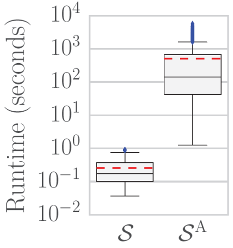
Using Concorde to search in $\mathcal{S}^{\mathrm{A}}$

- The CONCORDE solver was used for TSP optimizations (Applegate et al., 2006)
- Experiments consider a single local search execution. Results consider 100 executions for each instance.
- The initial solution was produced by the *savings* algorithm of Clarke and Wright (1964).

Computational experiments

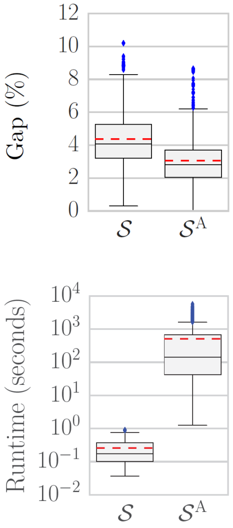


- Idea seems promising!
- Local search on the space of assignments ($\mathcal{S}^{\mathrm{A}}$) resulted in improved solutions



- However... runtime was prohibitive (even with efficient exploration strategies)
- We went from less than a second to over an hour

Computational experiments



- Idea seems promising!
- Local search on the space of assignments ($\mathcal{S}^{\mathrm{A}}$) resulted in improved solutions

- However... runtime was prohibitive (even with efficient exploration strategies)
- We went from less than a second to over an hour

An intermediate approach?

- In a recent conference presentation, Irnich (2013) proposed using the B&S neighborhood *in combination* with some classical CVRP moves.

- B&S Neighborhood (Balas and Simonetti, 2001)
  - Given a range parameter k and an initial tour, the B&S algorithm finds, in $\mathcal{O}(k^2 2^{k-2} n)$ operations, the vertex sequence with minimum cost such that no vertex is displaced by more than k positions.

- $\Rightarrow$ B^k–optimal tour
  - A tour σ is B^k–optimal if there exists no other permutation of its visits $\pi \circ \sigma$ with a shorter total distance such that $\pi(1) = 1$ and $\pi(i) \leq \pi(j)$ for all $i, j \in \{1, \ldots, n\}$ with $i + k \leq j$.

An intermediate approach?

- In a recent conference presentation, Irnich (2013) proposed using the B&S neighborhood *in combination* with some classical CVRP moves.

- B&S Neighborhood (Balas and Simonetti, 2001)
  - Given a range parameter k and an initial tour, the B&S algorithm finds, in $\mathcal{O}(k^2 2^{k-2} n)$ operations, the vertex sequence with minimum cost such that no vertex is displaced by more than k positions.

- $\Rightarrow$ B^k–optimal tour
  - A tour σ is B^k–optimal if there exists no other permutation of its visits $\pi \circ \sigma$ with a shorter total distance such that $\pi(1) = 1$ and $\pi(i) \leq \pi(j)$ for all $i, j \in \{1, \ldots, n\}$ with $i + k \leq j$.

An alternative search space

- We decided to investigate a systematic use of B&S in combination with *every* move of a LS.
- **Search Space $\mathcal{S}_k^{\mathrm{B}}$:**
  - Set of primitive solutions Y is a subset of the complete solutions, those containing only B^k–optimal tours;
  - A nontrivial decoder f is used, consisting of applying B&S multiple times to each route with a fixed k-range until tours are B^k–optimal;

  Properties:
  - From an initial solution containing B^k–optimal tours, a local search in the space $\mathcal{S}_k^{\mathrm{B}}$ explores only B^k–optimal tours.
  - For a fixed range k, each move evaluation and subsequent solution decoding is done in polynomial time as a function of n and the number of applications of B&S.
  - The search space $\mathcal{S}_k^{\mathrm{B}}$ is such that $\mathcal{S}_0^{\mathrm{B}} = \mathcal{S}$ and $\mathcal{S}_{n-1}^{\mathrm{B}} = \mathcal{S}^{\mathrm{A}}$.

An alternative search space

- We decided to investigate a systematic use of B&S in combination with *every* move of a LS.
- **Search Space $\mathcal{S}_k^{\mathrm{B}}$:**
  - Set of primitive solutions Y is a subset of the complete solutions, those containing only B^k–optimal tours;
  - A nontrivial decoder f is used, consisting of applying B&S multiple times to each route with a fixed k-range until tours are B^k–optimal;

Properties:

- From an initial solution containing B^k–optimal tours, a local search in the space $\mathcal{S}_k^{\mathrm{B}}$ explores only B^k–optimal tours.
- For a fixed range k, each move evaluation and subsequent solution decoding is done in polynomial time as a function of n and the number of applications of B&S.
- The search space $\mathcal{S}_k^{\mathrm{B}}$ is such that $\mathcal{S}_0^{\mathrm{B}} = \mathcal{S}$ and $\mathcal{S}_{n-1}^{\mathrm{B}} = \mathcal{S}^{\mathrm{A}}$.

An alternative search space

- We decided to investigate a systematic use of B&S in combination with *every* move of a LS.
- **Search Space $\mathcal{S}_k^{\mathrm{B}}$:**
  - Set of primitive solutions Y is a subset of the complete solutions, those containing only B^k–optimal tours;
  - A nontrivial decoder f is used, consisting of applying B&S multiple times to each route with a fixed k-range until tours are B^k–optimal;

Properties:

- From an initial solution containing B^k–optimal tours, a local search in the space $\mathcal{S}_k^{\mathrm{B}}$ explores only B^k–optimal tours.
- For a fixed range k, each move evaluation and subsequent solution decoding is done in polynomial time as a function of n and the number of applications of B&S.
- The search space $\mathcal{S}_k^{\mathrm{B}}$ is such that $\mathcal{S}_0^{\mathrm{B}} = \mathcal{S}$ and $\mathcal{S}_{n-1}^{\mathrm{B}} = \mathcal{S}^{\mathrm{A}}$.

An alternative search space

- We decided to investigate a systematic use of B&S in combination with *every* move of a LS.
- **Search Space $\mathcal{S}_k^{\mathrm{B}}$:**
  - Set of primitive solutions Y is a subset of the complete solutions, those containing only B^k–optimal tours;
  - A nontrivial decoder f is used, consisting of applying B&S multiple times to each route with a fixed k-range until tours are B^k–optimal;

Properties:

- From an initial solution containing B^k–optimal tours, a local search in the space $\mathcal{S}_k^{\mathrm{B}}$ explores only B^k–optimal tours.
- For a fixed range k, each move evaluation and subsequent solution decoding is done in polynomial time as a function of n and the number of applications of B&S.
- The search space $\mathcal{S}_k^{\mathrm{B}}$ is such that $\mathcal{S}_0^{\mathrm{B}} = \mathcal{S}$ and $\mathcal{S}_{n-1}^{\mathrm{B}} = \mathcal{S}^{\mathrm{A}}$.

An alternative search space

- We decided to investigate a systematic use of B&S in combination with *every* move of a LS.
- **Search Space $\mathcal{S}_k^{\mathrm{B}}$:**
  - Set of primitive solutions Y is a subset of the complete solutions, those containing only B^k–optimal tours;
  - A nontrivial decoder f is used, consisting of applying B&S multiple times to each route with a fixed k-range until tours are B^k–optimal;

Properties:

- From an initial solution containing B^k–optimal tours, a local search in the space $\mathcal{S}_k^{\mathrm{B}}$ explores only B^k–optimal tours.
- For a fixed range k, each move evaluation and subsequent solution decoding is done in polynomial time as a function of n and the number of applications of B&S.
- The search space $\mathcal{S}_k^{\mathrm{B}}$ is such that $\mathcal{S}_0^{\mathrm{B}} = \mathcal{S}$ and $\mathcal{S}_{n-1}^{\mathrm{B}} = \mathcal{S}^{\mathrm{A}}$.

An alternative search space

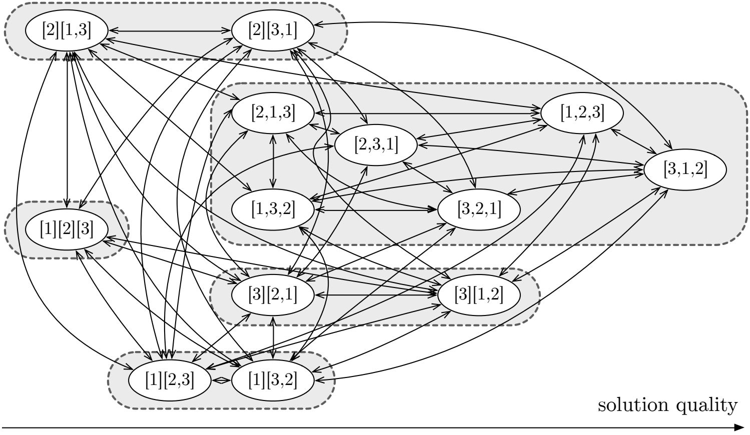
- We decided to investigate a systematic use of B&S in combination with *every* move of a LS.
- **Search Space $\mathcal{S}_k^{\mathrm{B}}$:**
  - Set of primitive solutions Y is a subset of the complete solutions, those containing only B^k–optimal tours;
  - A nontrivial decoder f is used, consisting of applying B&S multiple times to each route with a fixed k-range until tours are B^k–optimal;

Properties:

- From an initial solution containing B^k–optimal tours, a local search in the space $\mathcal{S}_k^{\mathrm{B}}$ explores only B^k–optimal tours.
- For a fixed range k, each move evaluation and subsequent solution decoding is done in polynomial time as a function of n and the number of applications of B&S.
- The search space $\mathcal{S}_k^{\mathrm{B}}$ is such that $\mathcal{S}_0^{\mathrm{B}} = \mathcal{S}$ and $\mathcal{S}_{n-1}^{\mathrm{B}} = \mathcal{S}^{\mathrm{A}}$.

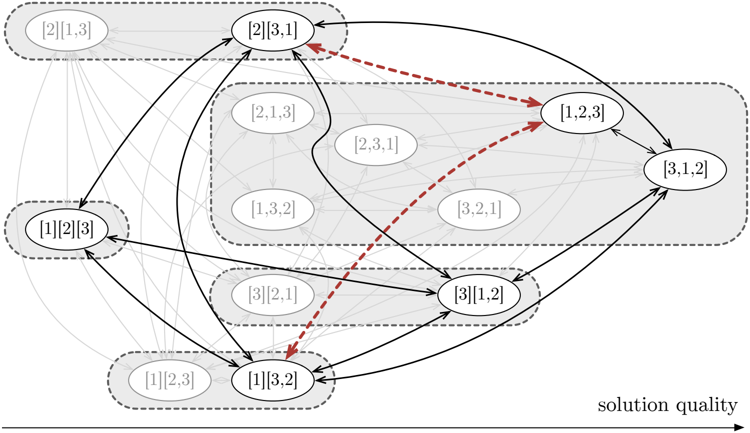
Small example with 3 customers

- Beginning from $\color{red}{\mathcal{S} = \mathcal{S}_0^{\mathrm{B}}}$, then $\mathcal{S}_1^{\mathrm{B}}$, and finally $\mathcal{S}^A = \mathcal{S}_2^{\mathrm{B}}$:



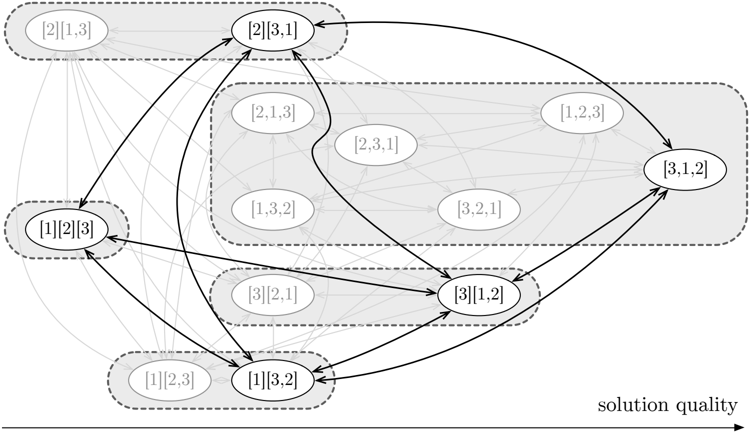
Small example with 3 customers

- Beginning from $\mathcal{S} = \mathcal{S}_0^{\mathrm{B}}$, then $\mathcal{S}_1^{\mathrm{B}}$, and finally $\mathcal{S}^A = \mathcal{S}_2^{\mathrm{B}}$:



Small example with 3 customers

- Beginning from $\mathcal{S} = \mathcal{S}_0^{\mathrm{B}}$, then $\mathcal{S}_1^{\mathrm{B}}$, and finally $\mathcal{S}^A = \mathcal{S}_2^{\mathrm{B}}$:



Efficient Local Search

- LS based on classical Relocate and Swap, for single vertices or generalized to consecutive vertex pairs, along with 2-opt and 2-opt* moves.

- Speedup techniques to reduce the search effort: **static neighborhood restrictions**, **dynamic move filters**, **concatenation techniques** and **memory structures**.

Static Neighborhood Restrictions

- As detailed in Vidal et al. (2013a), and in a similar way as Johnson and McGeoch (1997) and Toth and Vigo (2003): we restrict the search to the subset of moves that reconnect at least one vertex i with a vertex j belonging to the Γ closest vertices of i.

$\Rightarrow$ Number of moves is $\mathcal{O}(\Gamma n)$

Dynamic move filters

- Decoding solutions (by applying B&S multiple times) **is time consuming**.
- It is thus important to **reduce** the number of decoded solutions, filtering **infeasible** and **non-promising** moves.

- **Dynamic move filters**:
  - Only solutions resulting from moves that increased the distance by a factor $1 + \phi$ or less are decoded:

$$z(\phi(x^t)) \leq (1 + \psi) \times z(x^t)$$

  - Parameter ϕ plays an important role defining the percentage of evaluated moves.

Dynamic move filters

- Decoding solutions (by applying B&S multiple times) **is time consuming**.
- It is thus important to **reduce** the number of decoded solutions, filtering **infeasible** and **non-promising** moves.

- **Dynamic move filters**:
  - Only solutions resulting from moves that increased the distance by a factor $1 + \phi$ or less are decoded:

$$z(\phi(x^t)) \leq (1 + \psi) \times z(x^t)$$

  - Parameter ϕ plays an important role defining the percentage of evaluated moves.

Dynamic move filters

- Parameter ϕ is dynamically adjusted given a target range $[\xi^-, \xi^+]$ for the fraction of filtered moves.
- After each 1,000 move evaluations, the fraction ξ of filtered moves is collected and ϕ is updated.

$$\psi = \begin{cases} \psi \times \alpha & \text{if } \xi \leq \xi^-, \\ \psi \ / \ \alpha & \text{if } \xi \geq \xi^+, \\ \psi & \text{otherwise.} \end{cases}$$

Constant-time Evaluations

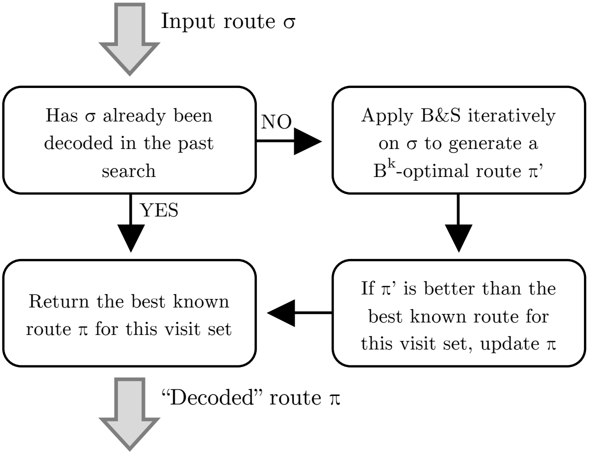
- **Constant-time feasibility checks and computations of hash functions** play an important role in the algorithm, exploiting the same concepts as Vidal et al. (2014):

$$
\begin{aligned}
Q(\sigma^1 \oplus \sigma^2) &= Q(\sigma^1) + Q(\sigma^2) \\
C(\sigma^1 \oplus \sigma^2) &= C(\sigma^1) + d_{\sigma^1(|\sigma^1|),\sigma^2(1)} + C(\sigma^2) \\
H^p(\sigma^1 \oplus \sigma^2) &= H^p(\sigma^1) + \rho^{|\sigma_1|} \times H^p(\sigma^2) \\
H^s(\sigma^1 \oplus \sigma^2) &= H^s(\sigma^1) + H^s(\sigma^2).
\end{aligned}
$$

Reshaping the neighborhood

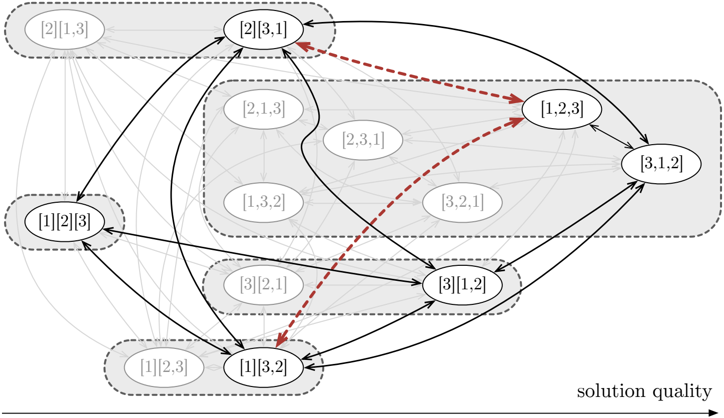
- Moreover, can we transform the search space $\mathcal{S}$ or $\mathcal{S}_k^{\mathrm{B}}$ so that it converges towards $\mathcal{S}^A$ as the optimization is run?
  - Definitely, using long-term memories to implement a **tunneling** strategy!

Tunneling strategy



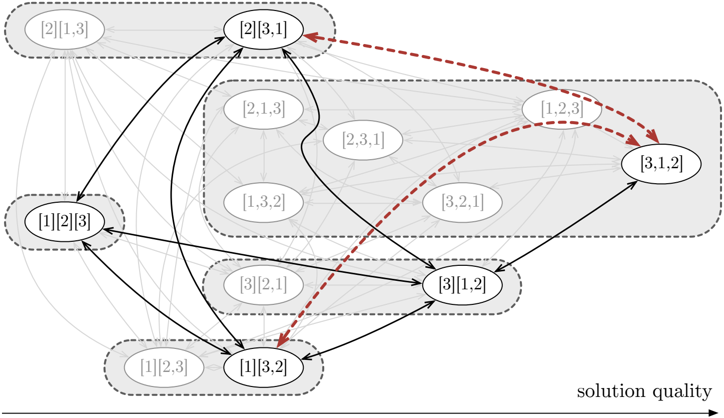
Small example with 3 customers

- Tunneling effect on search space $\mathcal{S}_1^{\mathrm{B}}$: in this example, search space converges to $\mathcal{S}^A$ when solution [3,1,2] is discovered



Small example with 3 customers

- Tunneling effect on search space $\mathcal{S}_1^{\mathrm{B}}$: in this example, search space converges to $\mathcal{S}^A$ when solution [3,1,2] is discovered



Input: An initial complete solution x^0, an evaluation threshold ψ and a granularity threshold Γ

$t \leftarrow 0$

repeat

    // Enumerating $\mathcal{O}(\Gamma n)$ moves - candidate lists based on vertex proximity

    for each move $\phi(x^t) \in \mathcal{N}(x^t)$ involving a vertex pair (i,j), $j \in \Gamma(i)$

        The move ϕ modifies up to two routes of x^t. Let z_{BEFORE} be the sum of the costs of these two routes, and let $(\sigma_1^1, \ldots, \sigma_{b_1}^1)$ and $(\sigma_1^2, \ldots, \sigma_{b_2}^2)$ be the new routes in $\phi(x^t)$.

        // First, filter infeasible moves with respect to capacity constraints in $\mathcal{O}(1)$:

        if $Q(\sigma_1^1 \oplus \cdots \oplus \sigma_{b_1}^1) > Q$ **or** $Q(\sigma_1^2 \oplus \cdots \oplus \sigma_{b_2}^2) > Q$ **then**

            continue.

        // Second, consider the cost of the classical CVRP move to filter non-promising solutions in $\mathcal{O}(1)$:

        if $z(x^t) + C(\sigma_1^1 \oplus \cdots \oplus \sigma_{b_1}^1) + C(\sigma_1^2 \oplus \cdots \oplus \sigma_{b_2}^2) - z_{\text{BEFORE}} > (1+\psi) \times z(x^t)$ **then**

            continue.

        // Third, *decode* the routes σ^1 and σ^2 to evaluate the move ϕ in $\mathcal{S}_k^{\mathrm{B}}$:

        $z_{\text{MOVE}} \leftarrow 0$

        for each route σ^i with $i \in \{1,2\}$

            // Compute hash key in $\mathcal{O}(1)$ and check memory in $\mathcal{O}(1)$:

            $(\bar{\sigma}^i, \bar{z}_i) \leftarrow \textsc{Lookup}(H(\sigma_1^i \oplus \cdots \oplus \sigma_{b_i}^i))$

            if $(\bar{\sigma}^i, \bar{z}_i) = \textsc{Not Found}$ **then**

                $(\bar{\sigma}^i, \bar{z}_i) \leftarrow \textsc{Balas-Simonetti}(\sigma_1^i \oplus \cdots \oplus \sigma_{b_i}^i)$

                $\textsc{Store}((\bar{\sigma}^i, \bar{z}_i), H(\sigma_1^i \oplus \cdots \oplus \sigma_{b_i}^i))$

            $z_{\text{MOVE}} \leftarrow z_{\text{MOVE}} + \bar{z}_i$

        // Filter non-improving moves:

        if $z_{\text{MOVE}} \geq z_{\text{BEFORE}}$ **then**

            continue.

        // At this stage, apply ϕ since it is an improving move in $\mathcal{S}_k^{\mathrm{B}}$:

        Set $x^{t+1} = \phi(x)$; $t = t+1$

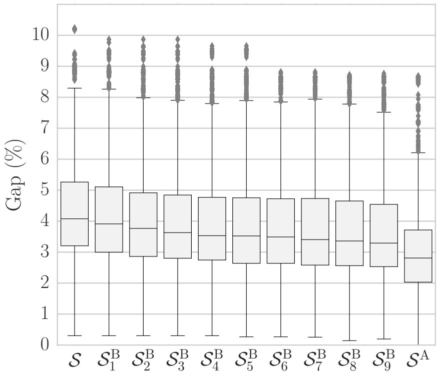
        Replace the routes (σ^1,σ^2) by $(\bar{\sigma}^1,\bar{\sigma}^2)$ in x^{t+1}

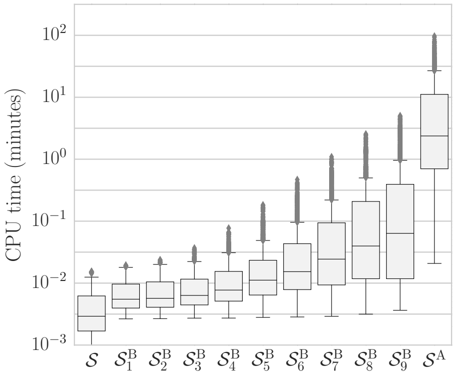
until x^t is a local minimum

return x^t

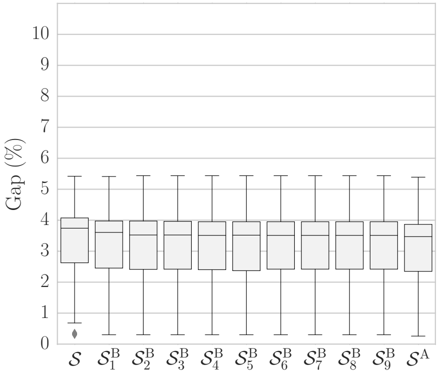
Computational experiments

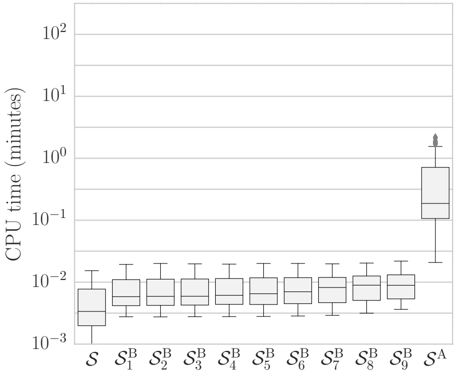
- Experiments with simple local search – on all instances



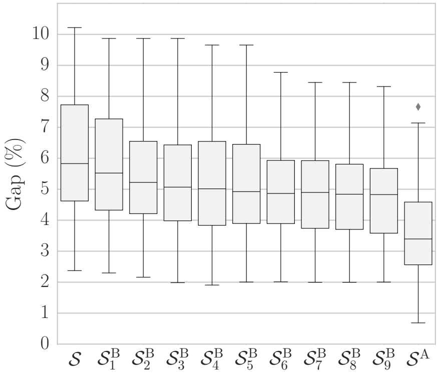


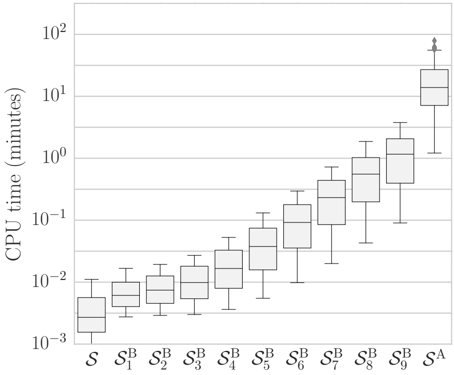
- Experiments with simple local search – instances with route cardinality in range [3.0, 4.55]:



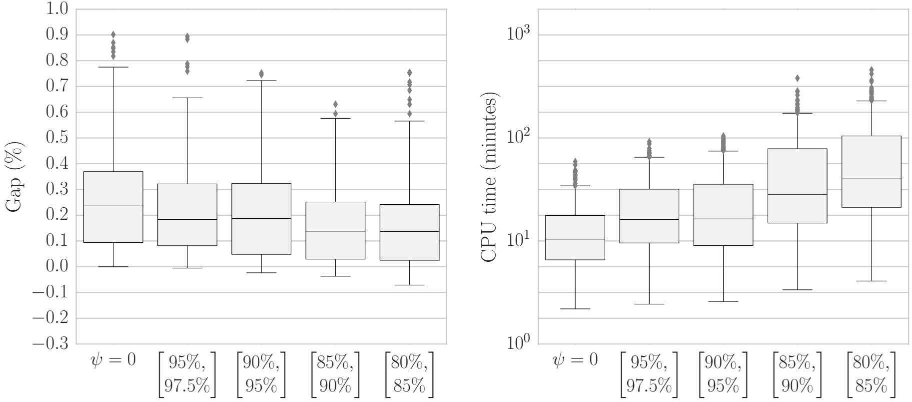


- Experiments with simple local search – instances with route cardinality in range [16.47, 24.43]:



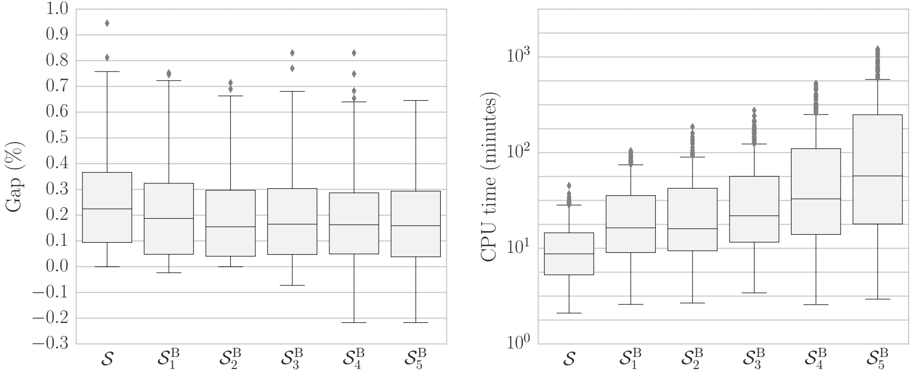


- Results with different target target intervals $[\xi^-, \xi^+]$ (desired quantity of filtered moves):



Computational experiments

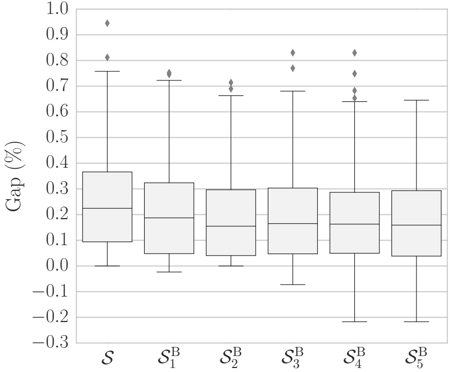
- Experiments with UHGS-BS and different values for parameter k:

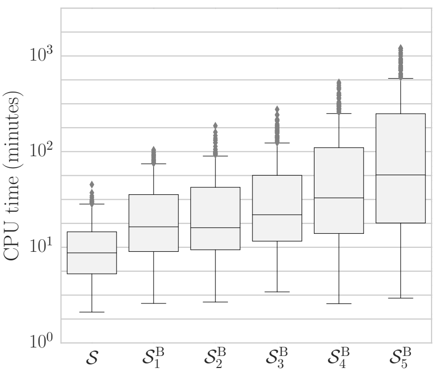


- Wilcoxon tests show that statistically significant differences exist between the results of $\mathcal{S}$, $\mathcal{S}_1^{\mathrm{B}}$ and $\mathcal{S}_2^{\mathrm{B}}$ (p-values < 0.05)

Computational experiments

- Experiments with UHGS-BS and different values for parameter k:





- Wilcoxon tests show that statistically significant differences exist between the results of $\mathcal{S}$, $\mathcal{S}_1^B$ and $\mathcal{S}_2^B$ (p-values < 0.05)

Final results (small instances)

#	Instance	ILS Time	ILS Average	ILS Best	UHGS Time	UHGS Average	UHGS Best	UHGS-BS Time	UHGS-BS Average	UHGS-BS Best
1	X-n101-k25	0.1	**27591.0**	**27591**	1.4	**27591.0**	**27591**	2.4	**27591.0**	**27591**
2	X-n106-k14	2.0	26375.9	**26362**	4.0	26381.8	26378	17.6	**26374.3**	**26362**
3	X-n110-k13	0.2	**14971.0**	**14971**	1.6	**14971.0**	**14971**	3.9	**14971.0**	**14971**
4	X-n115-k10	0.2	**12747.0**	**12747**	1.8	**12747.0**	**12747**	6.4	**12747.0**	**12747**
5	X-n120-k6	1.7	13337.6	**13332**	2.3	**13332.0**	**13332**	38.3	**13332.0**	**13332**
6	X-n125-k30	1.4	55673.8	**55539**	2.7	55542.1	**55539**	6.1	**55540.0**	**55539**
7	X-n129-k18	1.9	28998.0	28948	2.7	28948.5	**28940**	8.4	**28940.0**	**28940**
8	X-n134-k13	2.1	10947.4	**10916**	3.3	10934.9	**10916**	20.2	**10916.0**	**10916**
9	X-n139-k10	1.6	13603.1	**13590**	2.3	**13590.0**	**13590**	8.9	**13590.0**	**13590**
10	X-n143-k7	1.6	15745.2	15726	3.1	15700.2	**15700**	33.1	**15700.0**	**15700**
11	X-n148-k46	0.8	43452.1	**43448**	3.2	**43448.0**	**43448**	5.1	**43448.0**	**43448**
12	X-n153-k22	0.5	21400.0	21340	5.5	21226.3	**21220**	15.2	**21225.6**	21225
13	X-n157-k13	0.8	**16876.0**	**16876**	3.2	**16876.0**	**16876**	28.4	**16876.0**	**16876**
14	X-n162-k11	0.5	14160.1	**14138**	3.3	14141.3	**14138**	12.2	**14138.0**	**14138**
15	X-n167-k10	0.9	20608.7	20562	3.7	20563.2	**20557**	36.1	**20557.0**	**20557**
16	X-n172-k51	0.6	45616.1	**45607**	3.8	**45607.0**	**45607**	5.7	**45607.0**	**45607**
17	X-n176-k26	1.1	48249.8	48140	7.6	47957.2	**47812**	16.2	**47830.7**	**47812**
18	X-n181-k23	1.6	25571.5	**25569**	6.3	25591.1	**25569**	13.9	**25569.4**	**25569**
19	X-n186-k15	1.7	24186.0	**24145**	5.9	24147.2	**24145**	20.2	**24145.0**	**24145**
20	X-n190-k8	2.1	17143.1	17085	12.1	16987.9	**16980**	161.9	**16985.3**	**16980**
21	X-n195-k51	0.9	**44234.3**	**44225**	6.1	44244.1	**44225**	9.3	44283.8	**44225**
22	X-n200-k36	7.5	58697.2	58626	8.0	58626.4	**58578**	12.0	**58615.1**	**58578**
23	X-n204-k19	1.1	19625.2	19570	5.4	19571.5	**19565**	15.7	**19567.0**	**19565**
24	X-n209-k16	3.8	30765.4	30667	8.6	30680.4	**30656**	35.7	**30671.3**	**30656**
25	X-n214-k11	2.3	11126.9	10985	10.2	10877.4	**10856**	52.3	**10872.1**	**10856**
26	X-n219-k73	0.9	**117595.0**	**117595**	7.7	117604.9	**117595**	18.2	117600.5	**117595**
27	X-n223-k34	8.5	40533.5	40471	8.3	40499.0	**40437**	18.6	**40478.4**	**40437**
28	X-n228-k23	2.4	25795.8	25743	9.8	25779.3	**25742**	29.0	**25768.0**	25743
29	X-n233-k16	3.0	19336.7	19266	6.8	19288.4	**19230**	34.0	**19276.5**	**19230**

Final results (small instances)

#	Instance	ILS Time	ILS Average	ILS Best	UHGS Time	UHGS Average	UHGS Best	UHGS-BS Time	UHGS-BS Average	UHGS-BS Best	
30	X-n237-k14	3.5	27078.8	**27042**	8.9	27067.3	**27042**	32.2	**27048.8**	**27042**	
31	X-n242-k48	17.8	**82874.2**	82774	12.4	82948.7	82804	18.1	82920.9	**82751**	
32	X-n247-k47	2.1	37507.2	37289	20.4	**37284.4**	**37274**	27.7	37388.9	**37274**	
33	X-n251-k28	10.8	38840.0	38727	11.7	38796.4	38699	20.2	**38778.7**	**38684**	
34	X-n256-k16	2.0	18883.9	18880	6.5	18880.0	18880	23.0	**18867.7**	**18839**	⊛
35	X-n261-k13	6.7	26869.0	26706	12.7	26629.6	**26558**	48.6	**26618.1**	**26558**	
36	X-n266-k58	10.0	**75563.3**	**75478**	21.4	75759.3	75517	29.9	75710.7	**75478**	
37	X-n270-k35	9.1	35363.4	35324	11.3	35367.2	**35303**	18.9	**35314.6**	**35303**	
38	X-n275-k28	3.6	21256.0	**21245**	12.0	21280.6	**21245**	22.7	**21255.0**	**21245**	
39	X-n280-k17	9.6	33769.4	33624	19.1	33605.8	33505	136.2	**33587.9**	**33503**	
40	X-n284-k15	8.6	20448.5	20295	19.9	20286.4	**20227**	97.7	**20282.1**	20228	
41	X-n289-k60	16.1	95450.6	95315	21.3	95469.5	95244	41.7	**95447.2**	**95211**	
42	X-n294-k50	12.4	**47254.7**	47190	14.7	47259.0	47171	27.0	47272.7	**47161**	⊛
43	X-n298-k31	6.9	34356.0	34239	10.9	34292.1	**34231**	20.7	**34276.3**	**34231**	
44	X-n303-k21	14.2	21895.8	21812	17.3	21850.9	21748	48.4	**21811.2**	**21744**	
45	X-n308-k13	9.5	26101.1	25901	15.3	**25895.4**	**25859**	112.8	25897.3	25861	
46	X-n313-k71	17.5	94297.3	94192	22.4	**94265.2**	94093	30.6	94280.4	**94045**	
47	X-n317-k53	8.6	**78356.0**	**78355**	22.4	78387.8	**78355**	50.3	78385.3	**78355**	
48	X-n322-k28	14.7	29991.3	29877	15.2	29956.1	29870	27.7	**29892.5**	**29834**	⊛
49	X-n327-k20	19.1	27812.4	27599	18.2	27628.2	27564	68.7	**27590.8**	**27532**	⊛
50	X-n331-k15	15.7	31235.5	31105	24.4	31159.6	**31103**	102.1	**31126.7**	**31103**	
	Average gap:		0.37%	0.13%		0.14%	0.02%		0.10%	0.00%	

Final results (large instances)

#	Instance	ILS			UHGS			UHGS-BS			
		Time	Average	Best	Time	Average	Best	Time	Average	Best	
51	X-n336-k84	21.4	139461.0	**139197**	38.0	139534.9	139210	66.0	**139460.1**	139303	
52	X-n344-k43	22.6	42284.0	42146	21.7	42208.8	42099	39.7	**42156.1**	**42056**	⊛
53	X-n351-k40	25.2	26150.3	26021	33.7	26014.0	25946	51.5	**25981.8**	**25938**	
54	X-n359-k29	48.9	52076.5	51706	34.9	51721.7	**51509**	112.0	**51640.7**	51555	
55	X-n367-k17	13.1	23003.2	22902	22.0	**22838.4**	**22814**	117.3	22876.2	**22814**	
56	X-n376-k94	7.1	**147713.0**	**147713**	28.3	147750.2	147717	70.3	147740.5	147714	
57	X-n384-k52	34.5	66372.5	66116	40.2	66270.2	66081	56.8	**66170.3**	**65997**	
58	X-n393-k38	20.8	38457.4	38298	28.7	38374.9	38269	49.3	**38309.3**	**38260**	⊛
59	X-n401-k29	60.4	66715.1	66453	49.5	66365.4	66243	110.2	**66359.0**	**66212**	
60	X-n411-k19	23.8	19954.9	19792	34.7	19743.8	**19718**	126.0	**19736.7**	19721	
61	X-n420-k130	22.2	**107838.0**	**107798**	53.2	107924.1	**107798**	87.7	107913.7	**107798**	
62	X-n429-k61	38.2	65746.6	65563	41.5	**65648.5**	65501	65.6	65661.6	**65470**	
63	X-n439-k37	39.6	36441.6	**36395**	34.6	36451.1	**36395**	57.1	**36410.1**	**36395**	
64	X-n449-k29	59.9	56204.9	55761	64.9	55553.1	55378	132.6	**55432.7**	**55330**	
65	X-n459-k26	60.6	24462.4	24209	42.8	24272.6	24181	92.9	**24226.0**	**24145**	⊛
66	X-n469-k138	36.3	**222182.0**	**221909**	86.7	222617.1	222070	142.3	222427.5	222235	
67	X-n480-k70	50.4	89871.2	89694	67.0	89760.1	89535	73.1	**89744.7**	**89513**	
68	X-n491-k59	52.2	67226.7	66965	71.9	66898.0	66633	81.9	**66794.1**	**66607**	
69	X-n502-k39	80.8	69346.8	69284	63.6	69328.8	69253	177.7	**69277.1**	**69247**	
70	X-n513-k21	35.0	24434.0	24332	33.1	24296.6	**24201**	99.4	**24256.2**	**24201**	
71	X-n524-k137	27.3	155005.0	**154709**	80.7	**154979.5**	154774	207.3	155038.1	154787	
72	X-n536-k96	62.1	95700.7	95524	107.5	**95330.6**	95122	144.5	95335.4	**95112**	
73	X-n548-k50	64.0	**86874.1**	**86710**	84.2	86998.5	86822	136.6	86881.0	86778	
74	X-n561-k42	68.9	43131.3	42952	60.6	42866.4	42756	77.2	**42860.0**	**42733**	
75	X-n573-k30	112.0	51173.0	51092	188.2	50915.1	**50780**	782.4	**50876.9**	50801	
76	X-n586-k159	78.5	190919.0	190612	175.3	190838.0	190543	234.3	**190752.4**	**190442**	
77	X-n599-k92	73.0	109384.0	109056	125.9	109064.2	108813	166.9	**108993.3**	**108576**	
78	X-n613-k62	74.8	60444.2	60229	117.3	59960.0	59778	103.6	**59859.7**	**59654**	
79	X-n627-k43	162.7	62905.6	62783	239.7	62524.1	62366	543.1	**62442.9**	**62254**	

Final results (large instances)

#	Instance	ILS			UHGS			UHGS-BS		
		Time	Average	Best	Time	Average	Best	Time	Average	Best
80	X-n641-k35	140.4	64606.1	64462	158.8	64192.0	**63839**	304.4	**64105.6**	63859
81	X-n655-k131	47.2	**106782.0**	**106780**	150.5	106899.1	106829	253.2	106855.6	106804
82	X-n670-k126	61.2	147676.0	147045	264.1	**147222.7**	**146705**	267.7	147663.9	147163
83	X-n685-k75	73.9	68988.2	68646	156.7	68654.1	**68425**	177.0	**68596.0**	68496
84	X-n701-k44	210.1	83042.2	82888	253.2	82487.4	82293	368.0	**82409.2**	**82174**
85	X-n716-k35	225.8	44171.6	44021	264.3	43641.4	43525	437.2	**43599.9**	**43498**
86	X-n733-k159	111.6	137045.0	136832	244.5	**136587.6**	**136366**	334.2	136607.4	136424
87	X-n749-k98	127.2	78275.9	77952	313.9	77864.9	77715	308.3	**77862.8**	**77605**
88	X-n766-k71	242.1	115738.0	115443	383.0	115147.9	**114683**	330.5	**115115.9**	114812
89	X-n783-k48	235.5	73722.9	73447	269.7	73009.6	72781	351.2	**72892.4**	**72738**
90	X-n801-k40	432.6	74005.7	73830	289.2	73731.0	73587	424.0	**73651.6**	**73466**
91	X-n819-k171	148.9	159425.0	159164	374.3	158899.3	158611	675.6	**158849.0**	**158592**
92	X-n837-k142	173.2	195027.0	194804	463.4	**194476.5**	**194266**	634.9	194504.0	194356
93	X-n856-k95	153.7	89277.6	89060	288.4	89238.7	89118	314.6	**89220.0**	**89020**
94	X-n876-k59	409.3	100417.0	100177	495.4	99884.1	99715	543.1	**99780.3**	**99610**
95	X-n895-k37	410.2	54958.5	54713	321.9	54439.8	**54172**	500.2	**54407.4**	54254
96	X-n916-k207	226.1	330948.0	330639	560.8	330198.3	**329836**	1082.5	**330153.2**	329866
97	X-n936-k151	202.5	134530.0	133592	531.5	**133512.9**	**133140**	1022.2	133729.3	133376
98	X-n957-k87	311.2	85936.6	85697	432.9	85822.6	85672	307.9	**85681.5**	**85555**
99	X-n979-k58	687.2	120253.0	119994	554.0	**119502.1**	119194	928.4	119527.7	**119188**
100	X-n1001-k43	792.8	73985.4	73776	549.0	72956.0	72742	952.8	**72903.3**	**72629**
	Average gap:		0.74%	0.42%		0.30%	0.06%		0.24%	0.03%

Contents

Conclusions and Perspectives

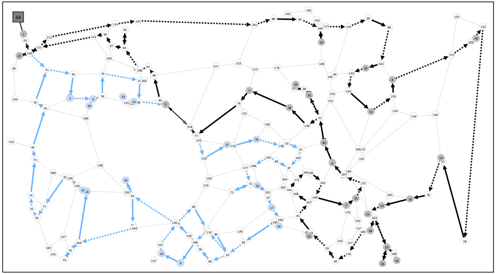
- Unified methods for vehicle routing problems, no need to reinvent the wheel for each new variant. **Generality does not necessarily impede efficiency for a large class of problems**.
- **Understanding the structure of the problems** is critical for the design of efficient methods
- Structural problem decompositions allow to relegate difficult decision classes (e.g., customer selection, edge orientations etc...) inside (modular) route-evaluation operators
- Efficient move evaluation strategies (e.g., pre-processing and dynamic programming) can lead to **considerable speedups**.
- Structural problem decompositions can be used to explore exponential-sized neighborhoods

Conclusions and Perspectives

- Perspectives: keep on focusing **problem structure**, **computational complexity** and **neighborhood search**. **Major breakthroughs are still possible around those research lines**.
- Following the recent advances of Arnold and Sörensen (2018) and Christiaens and Vanden Berghe (2018), design advanced inter-route moves which efficiently optimize the assignment decisions.
- Exploit pattern mining, machine learning and guidance to a larger extent...
- ...and many other promising perspectives

Thank you

THANK YOU FOR YOUR ATTENTION !



Articles, instances, detailed results and slides available at:

`http://w1.cirrelt.ca/~vidalt/`

Source code available at:

`https://github.com/vidalt/HGS-CARP` – Node, edge, and arc routing

`https://github.com/vidalt/HGS-CVRP` – Simple CVRP version

For further reading I

Aksen, D., O. Kaya, F. S. Salman, Y. Akça. 2012. Selective and periodic inventory routing problem for waste vegetable oil collection. *Optimization Letters* **6**(6) 1063–1080.

Applegate, D., R.E. Bixby, V. Chvatal, W. Cook. 2006. CONCORDE TSP Solver.

Archetti, C., A. Hertz, M.G. Speranza. 2007. Metaheuristics for the team orienteering problem. *Journal of Heuristics* **13**(1) 49–76.

Arnold, F., K. Sörensen. 2018. Knowledge-guided local search for the Vehicle Routing Problem. Tech. rep.

Bach, L., G. Hasle, S. Wøhlk. 2013. A lower bound for the node, edge, and arc routing problem. *Computers & Operations Research* **40**(4) 943–952.

Balas, E., N. Simonetti. 2001. Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study. *INFORMS Journal on Computing* **13**(1) 56–75.

Beasley, J.E. 1983. Route first-cluster second methods for vehicle routing. *Omega* **11**(4) 403–408.

Benavent, E., V. Campos, A. Corberán, E. Mota. 1992. The capacitated arc routing problem: Lower bounds. *Networks* **22**(7) 669–690.

Bentley, J.J. 1992. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing* **4**(4) 387–411.

For further reading II

Bentley, J.L., J.H. Friedman. 1978. Fast Algorithms for Constructing Minimal Spanning Trees in Coordinate Spaces. *IEEE Transactions on Computers* **C-27**(2).

Beullens, P., L. Muyldermans, D. Cattrysse, D. Van Oudheusden. 2003. A guided local search heuristic for the capacitated arc routing problem. *European Journal of Operational Research* **147**(3) 629–643.

Bodin, L.D., L. Berman. 1979. Routing and scheduling of school buses by computer. *Transportation Science* **13**(2) 113.

Bolduc, M.-C., J. Renaud, F. Boctor, G. Laporte. 2008. A perturbation metaheuristic for the vehicle routing problem with private fleet and common carriers. *Journal of the Operational Research Society* **59**(6) 776–787.

Bosco, A., D. Laganà, R. Musmanno, F. Vocaturo. 2012. Modeling and solving the mixed capacitated general routing problem. *Optimization Letters* **7**(7) 1451–1469.

Bosco, A., D. Laganà, R. Musmanno, F Vocaturo. 2014. A matheuristic algorithm for the mixed capacitated general routing problem. *Networks* **64**(4) 262–281.

Bouly, H., D.-C. Dang, A. Moukrim. 2009. A memetic algorithm for the team orienteering problem. *4OR* **8**(1) 49–70.

Brandão, J., R. Eglese. 2008. A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research* **35**(4) 1112–1126.

Campbell, A.M., D. Vandenbussche, W. Hermann. 2008. Routing for relief efforts. *Transportation Science* **42**(2) 127–145.

For further reading III

Chao, I., B. Golden, E.A. Wasil. 1996. The team orienteering problem. *European Journal of Operational Research* **88**(3) 464–474.

Christiaens, J., G. Vanden Berghe. 2018. Slack Induction by String Removals for Vehicle Routing Problems. Tech. rep.

Christofides, N., S. Eilon. 1972. Algorithms for Large-Scale Travelling Salesman Problems. *Operational Research Quarterly* **23**(4) 511.

Christofides, N., A. Mingozzi, P. Toth. 1979. The vehicle routing problem. N. Christofides, A. Mingozzi, P. Toth, C. Sandi, eds., *Combinatorial Optimization*. Wiley, Chichester, 315–338.

Clarke, G., J. W. Wright. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* **12**(4) 568–581.

Cordeau, J.-F., M. Gendreau, G. Laporte. 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks* **30**(2) 105–119.

Cordeau, J.-F., G. Laporte, A. Mercier. 2001. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society* **52**(8) 928–936.

Dang, D.-C., R. Guibadj, A. Moukrim. 2011. A PSO-based memetic Algorithm for the team orienteering problem. *Applications of Evolutionary Computation*, *LNCS*, vol. 6625. Springer Berlin Heidelberg, 471–480.

For further reading IV

De Berg, M., K. Buchin, B.M.P. Jansen, G. Woeginger. 2018. Fine-grained complexity analysis of two classic TSP variants. Tech. rep.

Dell'Amico, Mauro, José Carlos Díaz Díaz, Geir Hasle, Manuel Iori. 2016. An Adaptive Iterated Local Search for the Mixed Capacitated General Routing Problem. *Transportation Science* .

Falcon, R., X. Li, A. Nayak, I. Stojmenovic. 2012. A harmony-seeking firefly swarm to the periodic replacement of damaged sensors by a team of mobile robots. *ICC'12*. 4914–4918.

Gaskell, T.J. 1967. Bases for vehicle fleet scheduling. *Operational Research Quarterly* **18**(3) 281–295.

Gendreau, M., A. Hertz, G. Laporte. 1994. A tabu search heuristic for the vehicle routing problem. *Management Science* **40**(10) 1276–1290.

Gillett, B.E., L.R. Miller. 1974. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research* **22**(2) 340–349.

Glover, F. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* **13**(5) 533–549.

Glover, F. 1989. Tabu Search – Part I. *ORSA Journal on Computing* **1**(3) 190–206.

Glover, F. 1992. New ejection chain and alternating path methods for traveling salesman problems. O. Balci, R. Sharda, S. Zenios, eds., *Computer Science and Operations Research: New Developments in Their Interfaces*. Pergamon Press, 449–509.

For further reading V

Glover, F. 1996. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics* **65**(1-3) 223–253.

Goel, A., T. Vidal. 2014. Hours of service regulations in road freight transport: an optimization-based international assessment. *Transportation Science* **48**(3) 391–412.

Golden, B.L., J.S. DeArmon, E.K. Baker. 1983. Computational experiments with algorithms for a class of routing problems. *Computers & Operations Research* **10**(1) 47–59.

Golden, B.L., E.A. Wasil, J.P. Kelly, I. Chao. 1998. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. T.G. Crainic, G. Laporte, eds., *Fleet management and Logistics*. Kluwer, Boston, 33–56.

Hasle, G., O. Kloster, M. Smedsrud, K. Gaze. 2012. Experiments on the node, edge, and arc routing problem. Tech. rep., SINTEF, Oslo, Norway.

Helsgaun, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. *European Journal of Operational Research* **126**(1) 106–130.

Homsi, G., R. Martinelli, T. Vidal, K. Fagerholt. 2018. Industrial and Tramp Ship Routing Problems: Closing the Gap for Real-Scale Instances. Tech. rep.

Irnich, S. 2008. Solution of real-world postman problems. *European Journal of Operational Research* **190**(1) 52–67.

Irnich, S. 2013. Efficient local search for the CARP with combined exponential and classical neighborhood. *VEROLOG*. Southampton, U.K.

For further reading VI

Irnich, Stefan, Daniel Villeneuve. 2003. *The Shortest-Path Problem with Resource Constraints and k -Cycle Elimination for k >= 3*, vol. 18. Groupe d'études et de recherche en analyse des décisions, HEC Montréal.

Johnson, D.S., L.A. McGeoch. 1997. The traveling salesman problem: A case study in local optimization. E.H.L. Aarts, J.K. Lenstra, eds., *Local search in Combinatorial Optimization*. University Press, Princeton, NJ, 215–310.

Ke, L., C. Archetti, Z. Feng. 2008. Ants can solve the team orienteering problem. *Computers & Industrial Engineering* **54**(3) 648–665.

Lacomme, P., C. Prins, W. Ramdane-Chérif. 2001. A genetic algorithm for the capacitated arc routing problem and its extensions. *Applications of Evolutionary Computing* 473–483.

Li, L.Y.O., R.W. Eglese. 1996. An interactive algorithm for vehicle routeing for winter-gritting. *Journal of the Operational Research Society* **47**(2) 217–228.

Lin, S., B.W. Kernighan. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research* **21**(2) 498–516.

Lin, S.-W. 2013. Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing* **13**(2) 1064–1073.

Martinelli, R., M. Poggi, A. Subramanian. 2013. Improved bounds for large scale capacitated arc routing problem. *Computers & Operations Research* **40**(8) 2145–2160.

For further reading VII

Mei, Y., X. Li, X. Yao. 2014. Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation* **18**(3) 435–449.

Mei, Y., K. Tang, X. Yao. 2009. A global repair operator for capacitated arc routing problem. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics* **39**(3) 723–734.

Mole, R.H., S.R. Jameson. 1976. A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion. *Operational Research Quarterly* **27**(2) 503–511.

Mufalli, F., R. Batta, R. Nagi. 2012. Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans. *Computers & Operations Research* **39**(11) 2787–2799.

Muyldermans, L., P. Beullens, D. Cattrysse, D. Van Oudheusden. 2005. Exploring variants of 2-opt and 3-opt for the general routing problem. *Operations Research* **53**(6) 982–995.

Newton, R.M., W.H. Thomas. 1974. Bus routing in a multi-school system. *Computers & Operations Research* **1**(2) 213–222.

Pisinger, D., S. Ropke. 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research* **34**(8) 2403–2435.

Polacek, M., K.F. Doerner, R.F. Hartl, V. Maniezzo. 2008. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics* **14**(5) 405–423.

For further reading VIII

Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research* **31**(12) 1985–2002.

Prins, C. 2009. A GRASP - evolutionary local search hybrid for the vehicle routing problem. F.B. Pereira, J. Tavares, eds., *Bio-inspired Algorithms for the Vehicle Routing Problem*. Springer, 35–53.

Prins, C., S. Bouchenoua. 2005. A memetic algorithm solving the VRP, the CARP, and more general routing problems with nodes, edges and arcs. W. Hart, N. Krasnogor, J. Smith, eds., *Recent advances in memetic algorithms*. Springer, 65–85.

Ribeiro, Marcos Henrique, Alexandre Plastino, Simone L. Martins. 2006. Hybridization of GRASP metaheuristic with data mining techniques. *Journal of Mathematical Modelling and Algorithms* **5**(1) 23–41.

Rochat, Y., E.D. Taillard. 1995. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics* **1**(1) 147–167.

Santana, I.G. 2018. Improving a state-of-the-art heuristic for the minimum latency problem with data mining. Ph.D. thesis, Universidade Federal Fluminense.

Schneider, Michael, Fabian Schwahn, Daniele Vigo. 2017. Designing granular solution methods for routing problems with time windows. *European Journal of Operational Research* **263**(2) 493–509.

Schrimpf, G., J. Schneider, H. Stamm-Wilbrandt, G. Dueck. 2000. Record Breaking Optimization Results Using the Ruin and Recreate Principle. *Journal of Computational Physics* **159**(2) 139–171.

For further reading IX

Shaw, P. 1998. Using constraint programming and local search methods to solve vehicle routing problems. M. Maher, J.-F. Puget, eds., *Principles and Practice of Constraint Programming - CP98*, *LNCS*, vol. 1520. Springer Berlin Heidelberg, 417–431.

Solomon, M.M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* **35**(2) 254–265.

Sörensen, K. 2015. Metaheuristics-the metaphor exposed. *International Transactions in Operational Research* **22**(1) 3–18.

Sörensen, K., M. Sevaux. 2006. MAPM: memetic algorithms with population management. *Computers & Operations Research* **33**(5) 1214–1225.

Souffriau, W., P. Vansteenwegen, G. Vanden Berghe, D. Van Oudheusden. 2010. A path relinking approach for the team orienteering problem. *Computers & Operations Research* **37**(11) 1853–1859.

Subramanian, A., E. Uchoa, L.S. Ochi. 2013. A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research* **40**(10) 2519–2531.

Tang, H, E. Miller-Hooks. 2005. A TABU search heuristic for the team orienteering problem. *Computers & Operations Research* **32**(6) 1379–1407.

Tang, K., Y. Mei, X. Yao. 2009. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation* **13**(5) 1151–1166.

For further reading X

Toffolo, T.A.M., T. Vidal, T. Wauters. 2018. Heuristics for vehicle routing problems: Sequence or set optimization? Tech. rep., PUC-Rio, Rio de Janeiro.

Toth, P., D. Vigo. 2003. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing* **15**(4) 333–346.

Tricoire, F., M. Romauch, K.F. Doerner, R.F. Hartl. 2010. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research* **37**(2) 351–367.

Uchoa, E., D. Pecin, A. Pessoa, M. Poggi. 2013. Advances on the Exact Approaches for the CVRP. *EURO'13 Conference, Rome, Italy*.

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

Usberti, F.L., P.M. França, A.L.M. França. 2013. GRASP with evolutionary path-relinking for the capacitated arc routing problem. *Computers & Operations Research* **40**(12) 3206–3217.

Vansteenwegen, P, W Souffriau. 2009. Metaheuristics for tourist trip planning. K. Sörensen, M. Sevaux, W. Habenicht, M.J. Geiger, eds., *Metaheuristics in the Service Industry*. LNEMS, Springer Berlin Heidelberg, 15–31.

Vansteenwegen, P., W. Souffriau, G.V. Berghe, D.V. Oudheusden. 2009. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research* **196**(1) 118–127.

For further reading XI

Vidal, T. 2016. Technical note: Split algorithm in O(n) for the capacitated vehicle routing problem. *Computers & Operations Research* **69** 40–47.

Vidal, T. 2017. Node, edge, arc routing and turn penalties : Multiple problems – One neighborhood extension. *Operations Research* **65**(4) 992–1010.

Vidal, T., T.G. Crainic, M. Gendreau, N. Lahrichi, W. Rei. 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research* **60**(3) 611–624.

Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2013a. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research* **40**(1) 475–489.

Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2013b. Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. *European Journal of Operational Research* **231**(1) 1–21.

Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research* **234**(3) 658–673.

Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2015a. Time-window relaxations in vehicle routing heuristics. *Journal of Heuristics* **21**(3) 329–358.

Vidal, T., T.G. Crainic, M. Gendreau, C. Prins. 2015b. Timing problems and algorithms: Time decisions for sequences of activities. *Networks* **65**(2) 102–128.

For further reading XII

Wøhlk, S. 2003. Simulated annealing for the capacitated arc routing problem, using an online formulation. Tech. rep., University of Southern Denmark.

Wøhlk, S. 2004. Combining dynamic programming and simulated annealing. Tech. rep., University of Southern Denmark.

Yellow, P.C. 1970. A Computational Modification to the Savings Method of Vehicle Scheduling. *Operational Research Quarterly* **21**(2) 281–283.