The Fascinating History of the Vehicle Routing Problem

Gilbert Laporte

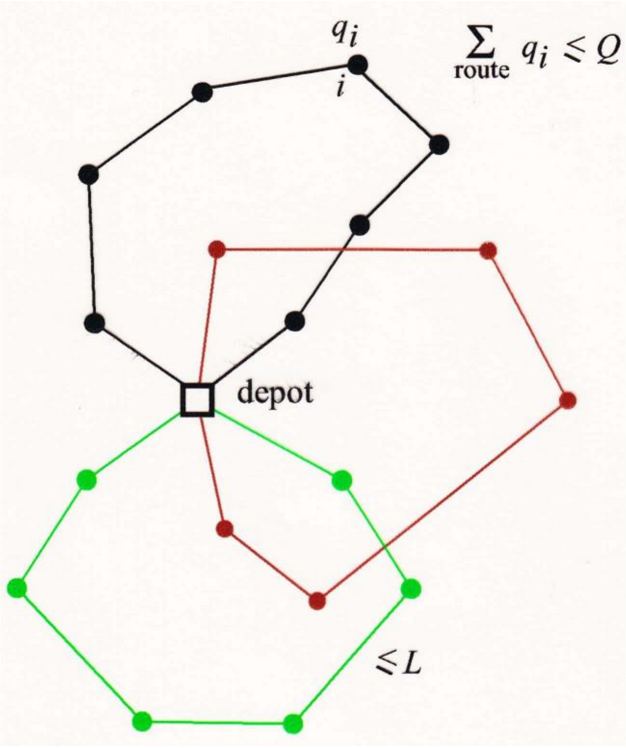
Canada Research Chair in Distribution Management

GERAD CIRRELT HEC MONTRÉAL

Outline

1. Introduction
2. Exact algorithms
3. Heuristic algorithms
4. Conclusions

Introduction



- Problem introduced by Dantzig and Ramser (Management Science, 1959)
- NP-hard
- Has multiple applications
- Exact algorithms: relatively small instances
- In practice heuristics are used

- Several variants

  e.g.

  - Heterogeneous vehicle fleet (Gendreau et al., 1999)
  - Time windows (Cordeau et al., 2002)
  - Pickup and deliveries (Desaulniers et al., VRP book, 2002)
  - Periodic visit (Cordeau et al., Networks, 1997)
  - Inventory-routing (Coelho et al., Transportation Science, 2014)
  - Green vehicle routing (Demir et al., European Journal of Operational Research, 2014)
  - Etc.

References

- Laporte, G., "Fifty Years of Vehicle Routing", *Transportation Science*, 43, 408-416, 2009.
- Toth, P., Vigo, D., "Vehicle Routing: Problems, Methods and Applications", P. Toth and D. Vigo, eds., MOS-SIAM Series on Optimization, SIAM, Philadelphia, 2014.
- Laporte, G., Ropke, S., Vidal, T., "Heuristics for the Vehicle Routing Problem", in *Vehicle Routing: Problems, Methods and Applications*, P. Toth and D. Vigo, eds., MOS-SIAM Series on Optimization, SIAM, Philadelphia, 2014.

2. Exact algorithms

2.1 Branch-and-bound

2.2 Dynamic programming

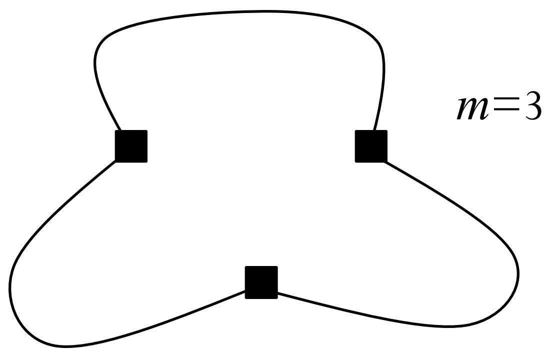
2.3 Vehicle flow formulations and algorithms

2.4 Commodity flow formulations and algorithms

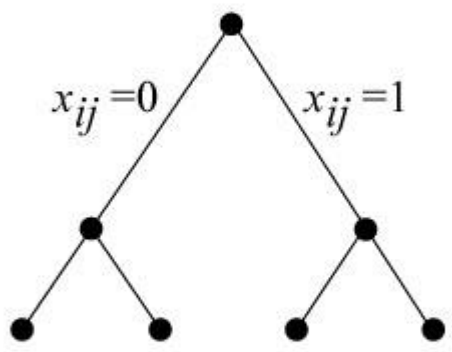
2.5 Set partitioning formulations and algorithms

2.1 Branch-and-bound

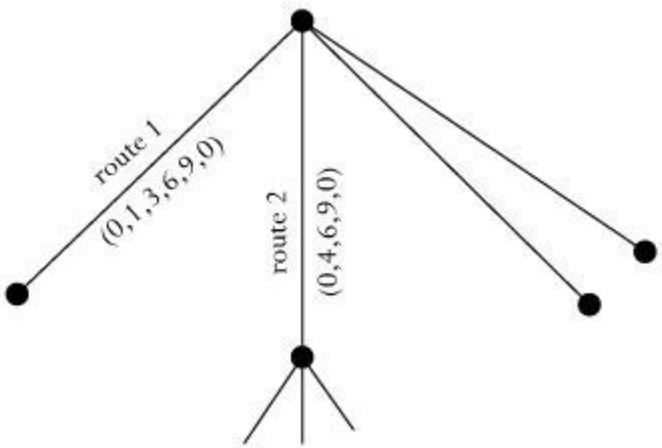
- Christofides and Eilon (1971)
  - Disregard constraints: m-TSP
  - Add m-1 artificial depots: TSP
  - Solve the TSP by branch-and-bound (Little et al., 1963)
  - Impose side constraints in the branching



- Laporte and Nobert (1986)
  - Same transformation
  - Use Carpaneto and Toth (1980) to solve the TSP



- Christofides (1976)
  - Branch on routes: broad tree with m levels



2.3 Vehicle flow formulations and algorithms

- Based on the Dantzig, Fulkerson and Johnson (1954) formulation for the TSP.
- Laporte, Nobert and Desrochers (1983, 1985)
  $x_{ij} = 0,1,2$: number of times edge (i,j) is used.

$$
\begin{array}{llll}
\text{(VF)} & \text{minimize} \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} & & \\
\text{subject to} & \displaystyle\sum_{j=1}^{n} x_{0j} = 2m & & \\
 & \displaystyle\sum_{i<k} x_{ik} + \sum_{j>k} x_{kj} = 2 & & (k \in V\backslash\{0\}) \\
 & \displaystyle\sum_{i,j\in S} x_{ij} \le |S| - v(S) & & (S \subseteq V\backslash\{0\}) \\
 & x_{0j} = 0,1,2 & & (j \in V\backslash\{0\}) \\
 & x_{ij} = 0,1 & & (i,j \in V\backslash\{0\})
\end{array}
$$

  where $v(S)$ is a lower bound on the number of vehicles needed to serve S.
  For example, $v(S) = \lceil \sum_{i\in S} q_i/Q \rceil$ or solution of a bin packing problem.
- Branch-and-cut algorithm: $n \le 60$

2.5 Set partitioning formulations and algorithms

- Balinski and Quandt (1965)

  r : a route

  $$a_{ij} = \begin{cases} 1 \text{ if } i \in V\{0\} \text{ is in route } r \\ 0 \text{ otherwise} \end{cases}$$

  c_r^* : optimal cost of route r

  $$y_r = \begin{cases} 1 \text{ if route } r \text{ is in the solution} \\ 0 \text{ otherwise} \end{cases}$$

  $$\begin{array}{lll} \text{(SP)} & \text{minimize} \sum_r c_r^* y_r & \\ \text{subject to} & \sum_r a_{ir} = 1 & (i \in V\backslash\{0\}) \\ & \sum_r y_r = m & \\ & y_r = 0, 1 & (\text{all } r) \end{array}$$

  Drawbacks: - very large number of routes
  - computing c_r^* is NP-hard (TSP)

- Early column generation algorithms
  - Rao and Zionts (1968) (not tested)
  - Foster and Ryan (1976) (not run to completion)
  - Agarwal, Mathur and Salkin (1989) $(15 \leq n \leq 25)$.

- Baldacci, Christofides and Mingozzi (2008)
  - (SP) formulation augmented with some valid inequalities from (VF) using the identity

$$x_{ij} = \sum_r a_{ijr} y_r \quad ((i,j) \in E)$$

  - Clique inequalities: let H be a graph whose vertices are vehicle routes, two routes conflict if they share some customers. For any clique C of H, $\sum_{r \in C} y_r \leq 1$.
  - Lower bounds computed on dual of linear relaxation by using three ascent heuristics.
  - Final dual solution used to generate a reduced problem of reduced cost between lower and upper bound achieved.
  - $37 \leq n \leq 121$ slightly better than the algorithm of Fukasawa et al. (2006)

3. Heuristic algorithms

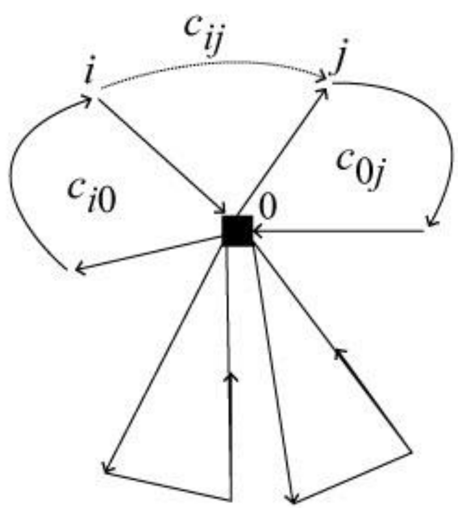
- 3.1 Constructive heuristics
- 3.2 Classical improvement heuristics
- 3.3 Metaheuristics (sophisticated improvement heuristics)
- 3.4 Computational results: all heuristics tested on three sets of benchmark instances:
  - Christofides, Mingozzi and Toth (1979): 14 instances, $51 \leq n \leq 199$
  - Golden, Wasil, Kelly and Chao (1998): 20 instances, $240 \leq n \leq 483$
  - Uchoa, Pecin, Pesso, Poggi, Subramanian and Vidal (2014): 100 new instances, $100 \leq n \leq 1000$

  Results are compared in terms of the deviation from the value of the best known solution (a moving target).

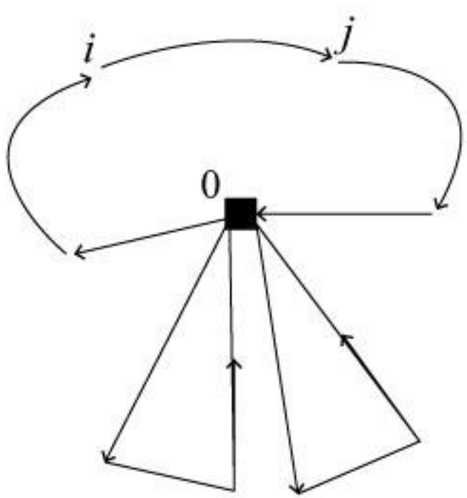
3.1 Constructive heuristics

- Dantzig and Ramser (1959)
  - Simple heuristics based on the Dantzig, Fulkerson and Johnson (1954) formulation for the Traveling Salesman Problem (TSP)
  - Successive matchings of vertices through the solution of linear programs
  - Elimination of fractional solutions by trial and error
  - Example on an eight-vertex graph
  - May have inspired matching based heuristics (Altinkemer and Gavish, 1991; Desrochers and Verhoog, 1989; Wark and Holt, 1994).

- Clarke and Wright (1964) savings heuristic
  - Construct n back-and-forth routes $(0, i, 0)$ $(i = 1, \dots, n)$
  - Merge route $(0, \dots, i, 0)$ with route $(0, j, \dots, 0)$ yielding largest saving $s_{ij} = c_{io} + c_{oj} - c_{ij}$.
  - Repeat while feasible
  - Several variants are available. Fast and simple, reasonably acurate (7%), not flexible.
  - Highly popular.



Before merge

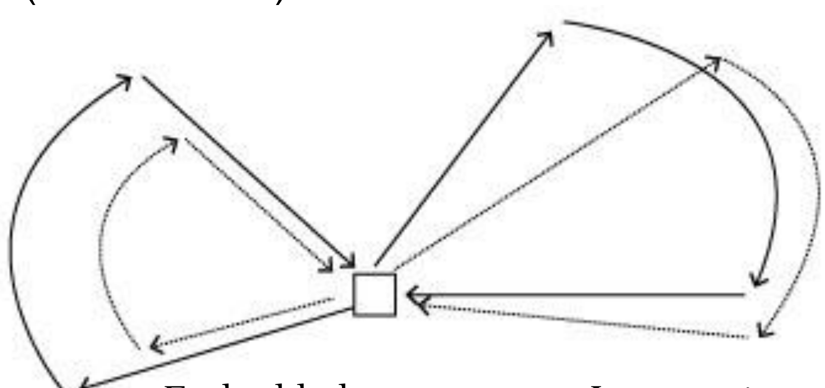


After merge

- Petal heuristics
  Generate several good routes and combine them by solving a set partionning problem.

$$
\begin{array}{llr}
\text{(SP)} & \text{minimize} \sum_r c_r^* y_r & \\
\text{subject to} & \sum_r a_{ir} = 1 & (i \in V\backslash\{0\}) \\
& \sum_r y_r = m & \\
& y_r = 0, 1 & (\text{all } r)
\end{array}
$$

  The sweep algorithm (Gillett and Miller, 1974) is a primitive implementation of this idea.
  Better implementations: Foster and Ryan (1976), Ryan, Hjorring and Glover(1993).
  Renaud, Boctor and Laporte (1996) generate embedded and overlapping routes (≤ 2.38%).



Embedded routes

Intersecting routes

3.2 Classical improvement heuristics: intra-route and inter-route moves

- Intra- route moves
  - λ-opt exchanges (Lin, 1965)
  - Dynamic λ-opt (Lin and Kernighan, 1973)
  - Very efficient implementation (Helsgaun, 2000), used in Concorde (Applegate, Bixby, Chvátal and Cook, 2006).

- Inter-route moves
  - RELOCATE: move *k* consecutive vertices in a different route
  - SWAP: swap consecutive vertices between different routes
  - 2-opt*: remove two edges from different routes and reconnect the routes differently

3.3 Metaheuristics

- Local search algorithms (examples)
  - **Simulated annealing** (Osman, 1993): accepts worsening move with some probability
  - **Tabu search** (Taillard, 1993; Gendreau, Hertz and Laporte, 1994; and many others): accept best solution in neighbourhood; sometimes accept intermediary infeasible solutions.

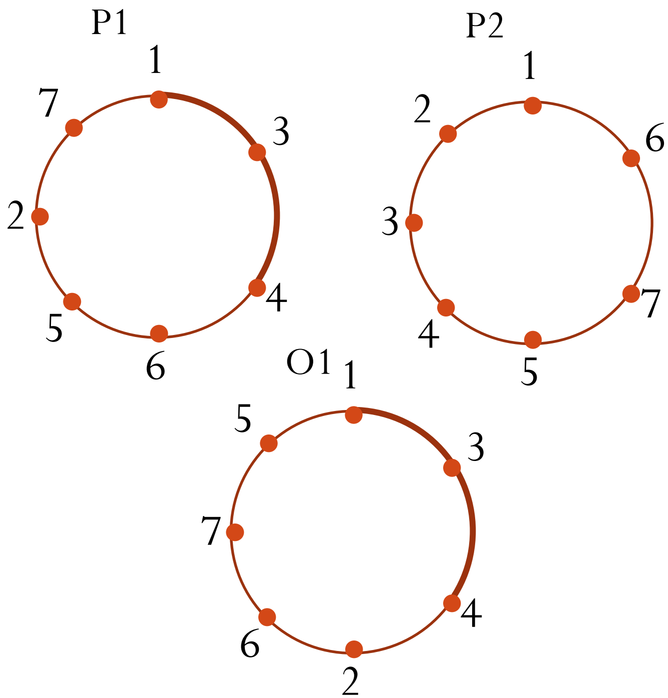
- **Genetic algorithms** (rooted in the work of Holland, 1975). Often combined with local search: memetic algorithms (Moscato and Cotta, 2010).

  Three excellent implementations

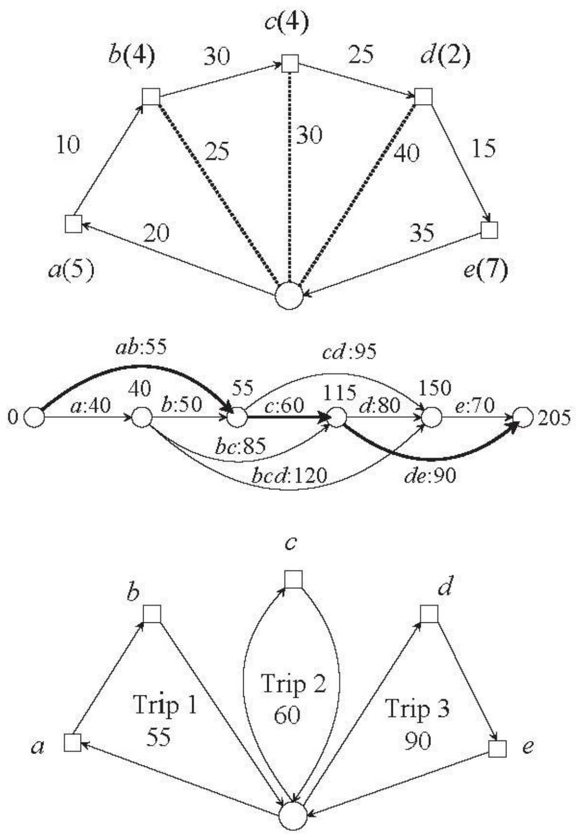
  1. Prins (2004). Work on population of TSP solutions, combined them, improve them, apply a SPLIT algorithm to regain a VRP solution (shortest path).

  2. Nagata and Bräysy (2009). Based on edge-assembly mechanism for the TSP. Remove some edges from parent P1 and replace them with those of parent P2. Solution may be infeasible for the VRP. Apply a repair procedure. Apply local search to the solution.

  3. Vidal, Crainic, Gendreau, Lahrichi and Rei (2012, 2013). The fitness of a solution to be inserted in the population is measured by its **cost** and the **diversity** it brings to the population (Hamming distance).



Creating an offspring (O1)
from two parents (P1 and P2)



Regaining a VRP solution

- **Hybridization**: the main tendency over the past 10 years.
- Heuristics combining several concepts initially developed independently of each other (e.g., tabu search, genetic algorithm)
- Integration of several strategies
  - Exotic large neighbourhoods
  - Exact mathematical programming techniques
  - Decomposition
  - Cooperation
- Hybridization of scope: flexible methods capable of solving several variants with the same parameters (Cordeau, Laporte and Mercier, 2001; Vidal, Crainic, Gendreau and Prins, 2013; Subramanian, Uchoa and Ochi, 2013).

3.4 Computational results

- Benchmark instances:
  - Christofides, Mingozzi and Toth (1979): 14 instances, $51 \leq n \leq 199$.
  - Golden, Wasil, Kelly and Chao (1998): 20 instances, $240 \leq n \leq 483$.
- Criteria: accuracy, speed, simplicity, flexibility. 15 of the best metaheuristics:

Heuristic	Configuration	Gap (%)	Time (s)	Time* (s)
NB09	Best of 10 runs	0.00	506	831
SUO13	Best of 10 runs	0.00	1008	1008
GGW11	129 threads, best of 5 runs	0.00	138899	193500
VCGLR12	Average of 10 runs – 50,000 it.	0.02	356	585
NB09	Average of 10 runs	0.03	51	83
MB07	Best configuration	0.03	87	163
GGW11	8 threads, best of 5 runs	0.03	4102	5714
VCGLR12	Average of 10 runs – 10,000 it.	0.05	81	133
GGW11	4 threads, best of 5 runs	0.05	2051	2857
P09	-	0.07	9	16
MB07	Fast configuration	0.08	2	3
SUO13	Average of 10 runs	0.08	101	101
PR07	Best of 10 runs – 50,000 it.	0.11	593	1046
T05	Standard configuration	0.18	21	338
GGW10	Set partitioning	0.28	7	9
GGW10	Ejection - random	0.31	17	22
PR07	Average of 10 runs – 50,000 it.	0.31	59	105
CLM01	-	0.56	529	1477
TV03	-	0.64	5	230

Table 4.3. *Computational results for the CMT instances. Note that the result for the CLM01 metaheuristic is from the survey by Cordeau et al. [13].*

Heuristic	Configuration	Gap (%)	Time (s)	Time* (s)
GGW11	129 threads, best of 5 runs	0.12	138899	193500
VCGLR12	Average of 10 runs, 50,000 iterations	0.16	3387	5563
NB09	Best of 10 runs	0.17	13006	21359
ZK10	Best of 10 runs	0.22	14128	24300
VCGLR12	Average of 10 runs, 10,000 iterations	0.27	1042	1712
NB09	Average of 10 runs	0.27	1301	2136
GGW11	8 threads, best of 5 runs	0.30	8470	11800
MB07	Best configuration	0.33	782	1461
JCL12	8 threads, best of 10 runs	0.35	200978	200978
SUO13	Best of 10 runs	0.40	39382	39382
ZK10	Average of 10 runs	0.43	1413	2430
GGW11	4 threads, best of 5 runs	0.44	4235	5900
CM12	Best of 10, 10^6 iterations	0.56	18770	18770
SUO13	Average of 10 runs	0.56	3938	3938
P09	-	0.63	233	436
JCL12	8 threads, average of 10 runs	0.60	20098	20098
PR07	Best of 10 runs, 50,000 iterations	0.82	3662	6457
T05	Standard	0.93	169	2729
RDH04	-	0.93	599	2960
CM12	Average of 10 runs, 10^6 iterations	0.94	1877	1877
GGW10	Ejection - random	1.19	43	55
MB07	Fast configuration	1.23	7	13
GGW10	Set partitioning	1.27	10	13
PR07	Average of 10 runs, 50,000 iterations	1.35	366	646
CM12	Average of 10 runs, 10^5 iterations	1.46	188	188
CLM01	-	1.79	1207	3366.6
TV03	-	3.21	21	1053

Table 4.4. *Computational results for the GWKC data set. We note that the time spent for the CM12 metaheuristic probably is underestimated since the paper only report time averaged over both the GWKC and CMT data sets and the CMT instances typically are solved faster than the GWKC instances. We also note that the results for the CLM01 and RDH04 metaheuristics are from the survey by Cordeau et al. [13].*

- Solution quality vs running time.

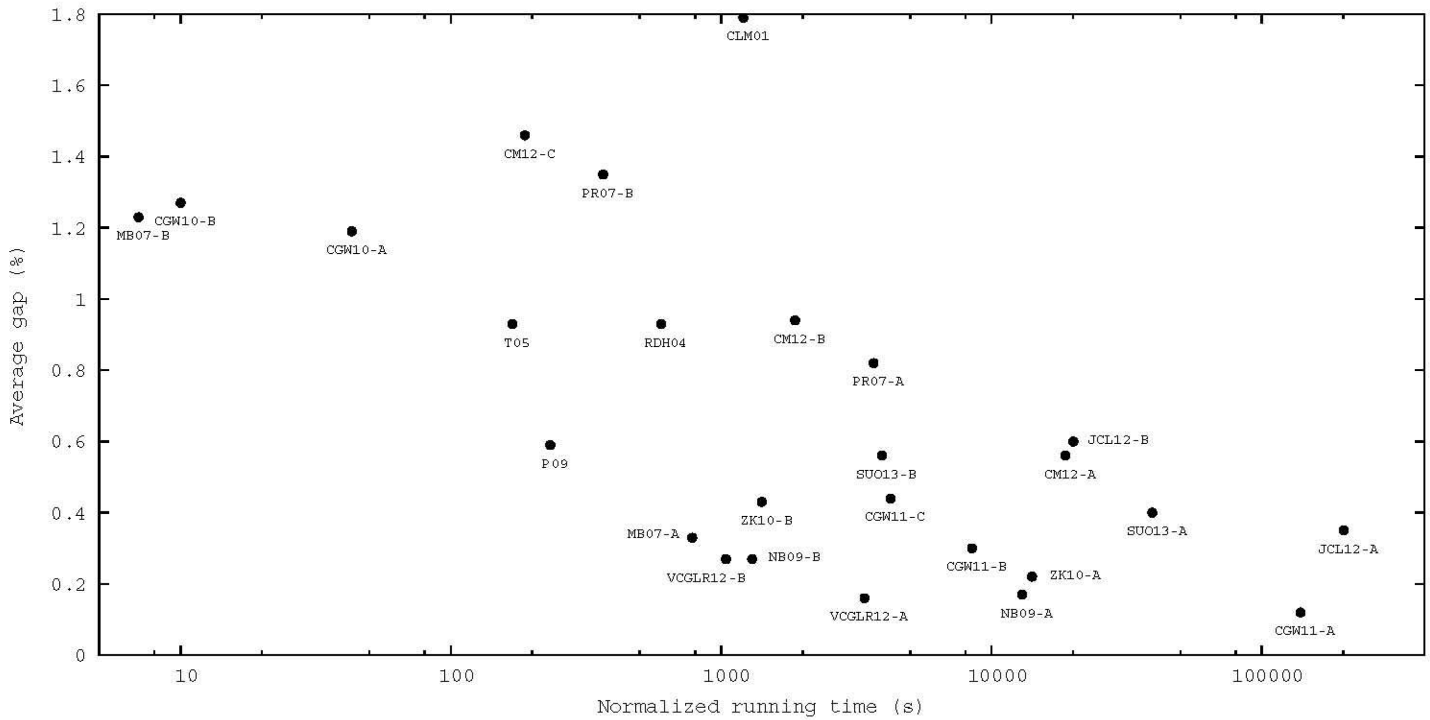


Figure 4.3. *Solution quality vs. running time for the GWKC instances. The x-axis shows the average computation time per instance in seconds (logarithmic scale), while the y-axis shows the solution quality measured as the average percent deviation from best known solution. Each label identifies a heuristic using the identifiers defined in Table 4.2. If several configurations of a heuristic is used then each configuration is indicated by a letter "A", "B" or "C" after the heuristic identifier, with "A" being the most powerful configuration. The data for the table can be found in table 4.4.*

4. Conclusions

- Exact algorithms have become very sophisticated but also complicated.
- Practical limit: $50 \leq n \leq 120$, with high variance.
- Recently, some larger instances $(n = 360)$ have been solved optimally (Pecin, Pessoa, Poggi and Uchoa, 2014).
- Heuristics have also be come very sophisticated:
  - Metaheuristics
  - With hybridizations
  - Accuracy: within 0.5% of best known solution values, within reasonable computing times (often less than 1000 seconds), $n \leq 500$.
  - Too much sophistication
  - Some over-engineering

* But essentially, the problem is now solved.

- Future algorithms:
  - Simplex algorithms
  - Fewer parameters
  - Same solution quality within less time
  - Matheuristics are promising (CPLEX, Gurobi)
- New areas:
  - Green logistics
  - City logistics
  - Humanitarian logistics
  - Synchronization
  - Service consistency
  - Balance workload allocation
  - Congestion and speed optimization.