A large neighborhood search algorithm for the vehicle routing problem with time windows

Guy Desaulniers
Eric Prescott-Gagnon
Louis-Martin Rousseau
Ecole Polytechnique, Montreal

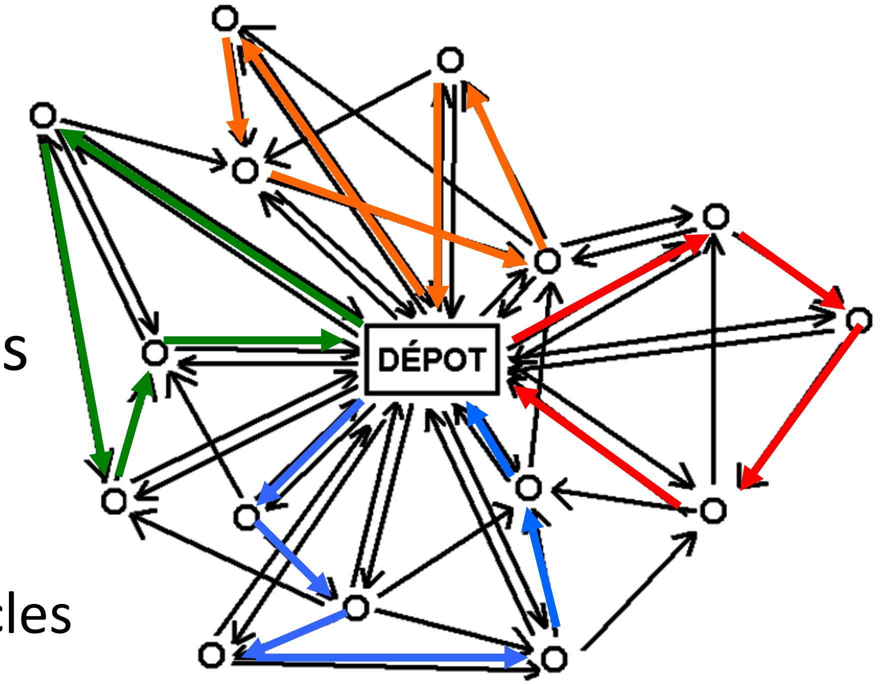
Column Generation, Aussois, June 2008

Overview

- Introduction
  - Vehicle routing problem with time windows
  - Motivation
  - Large neighborhood search
- Hybrid LNS and Column Generation
- Computational results
- Conclusion

Vehicle routing problem with time windows (VRPTW)

- 1 depot
- N customers
  - Time windows $[a_i, b_i]$
  - Demands d_i
- Unlimited number of vehicles
  - Capacity
- Objectives
  - First, minimize number of vehicles
  - Second, minimize total mileage



Motivation

- Real industrial problems are very large
- Successful exact method (from early 1990s)
  - Column generation – Branch-and-price
  - Feillet et al. (2004), Jepsen et al. (2006), Desaulniers et al. (2006)
  - Limited to relatively small problem (100-200 customers)
- Successful metaheuristics (from mid 80s)
  - Large neighborhood search
    - Pisinger & Ropke (2007)
  - Evolutionary algorithms
    - Gehring and Homberger (2001), Mester and Bräysy (2004)
  - Somewhat myopic when problem size increases

Objective

Combining column generation and LNS

Intuition:

LNS needs a good reconstruction method

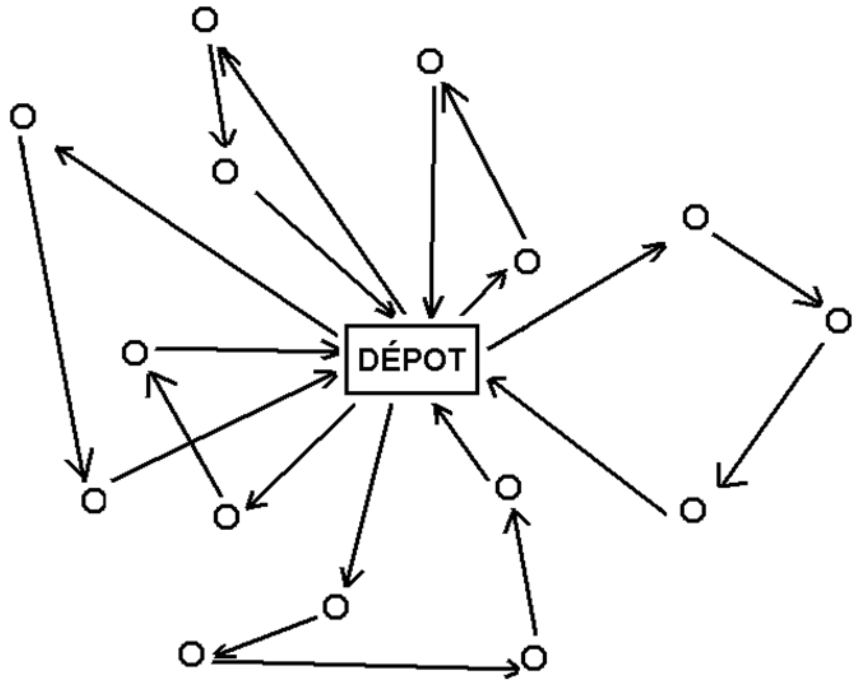
CG yields very good results when size is limited

Bonus:

The combination yields an evolutionary behaviour

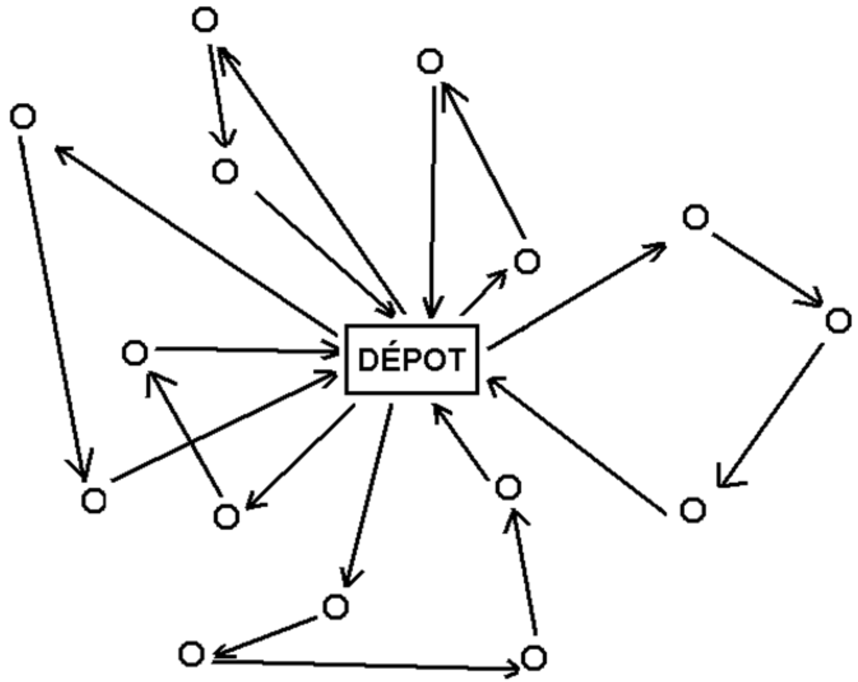
Large neighborhood search

- Iterative method



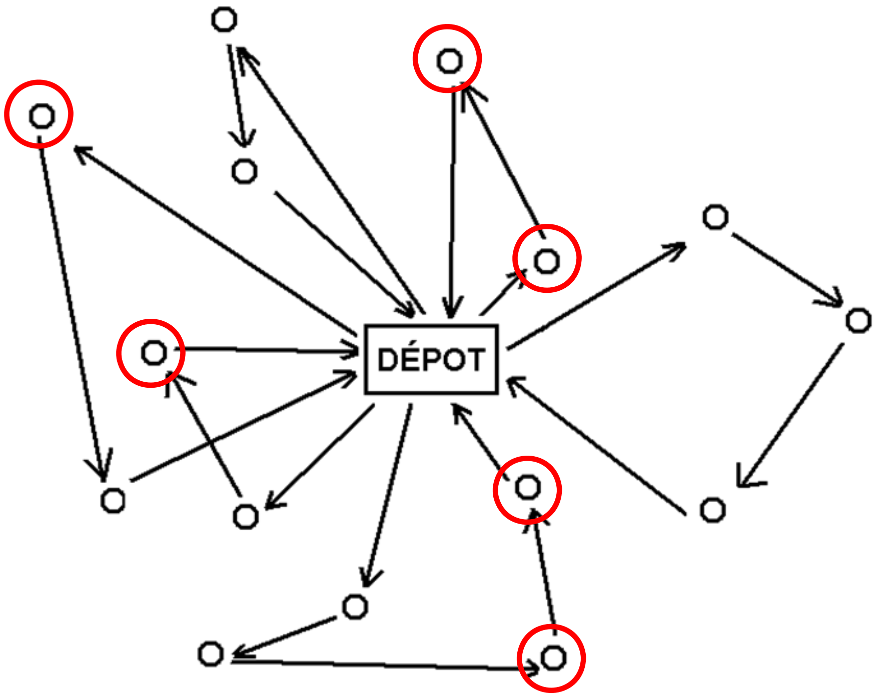
Large neighborhood search

- Iterative method
  - Current solution



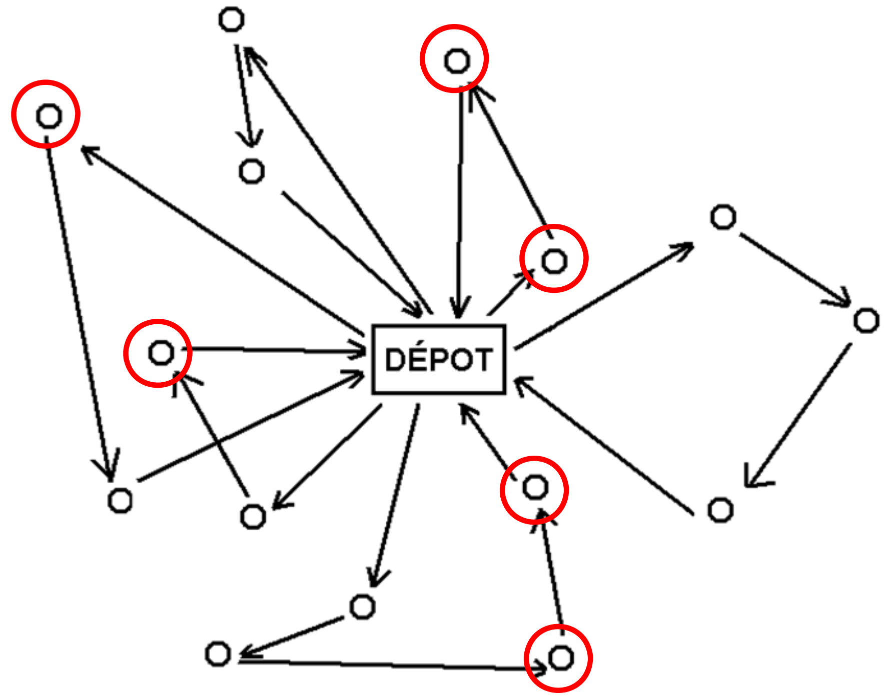
Large neighborhood search

- Iterative method
  - Current solution
  - Destruction



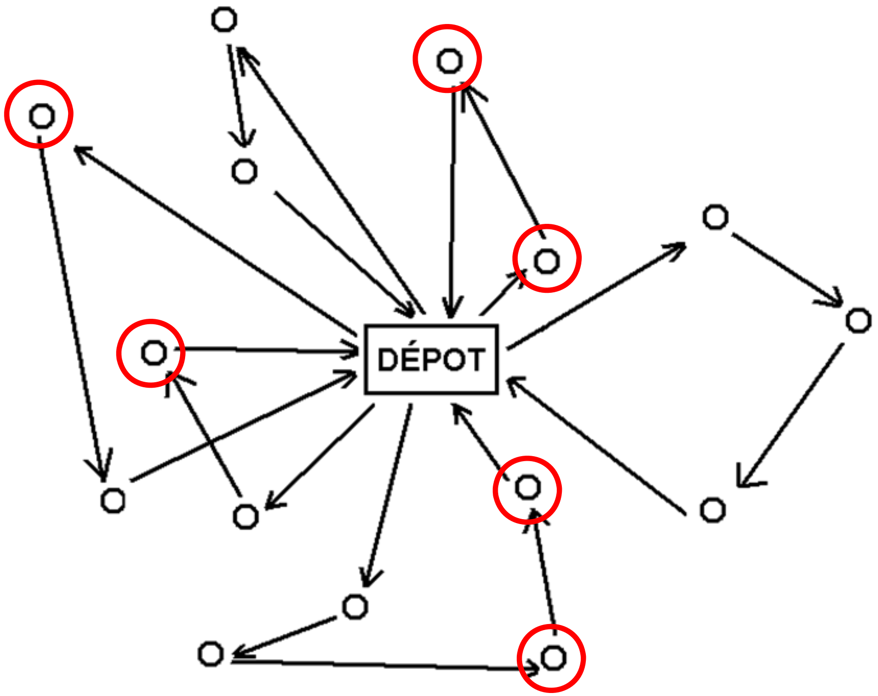
Large neighborhood search

- Iterative method
  - Current solution
  - Destruction



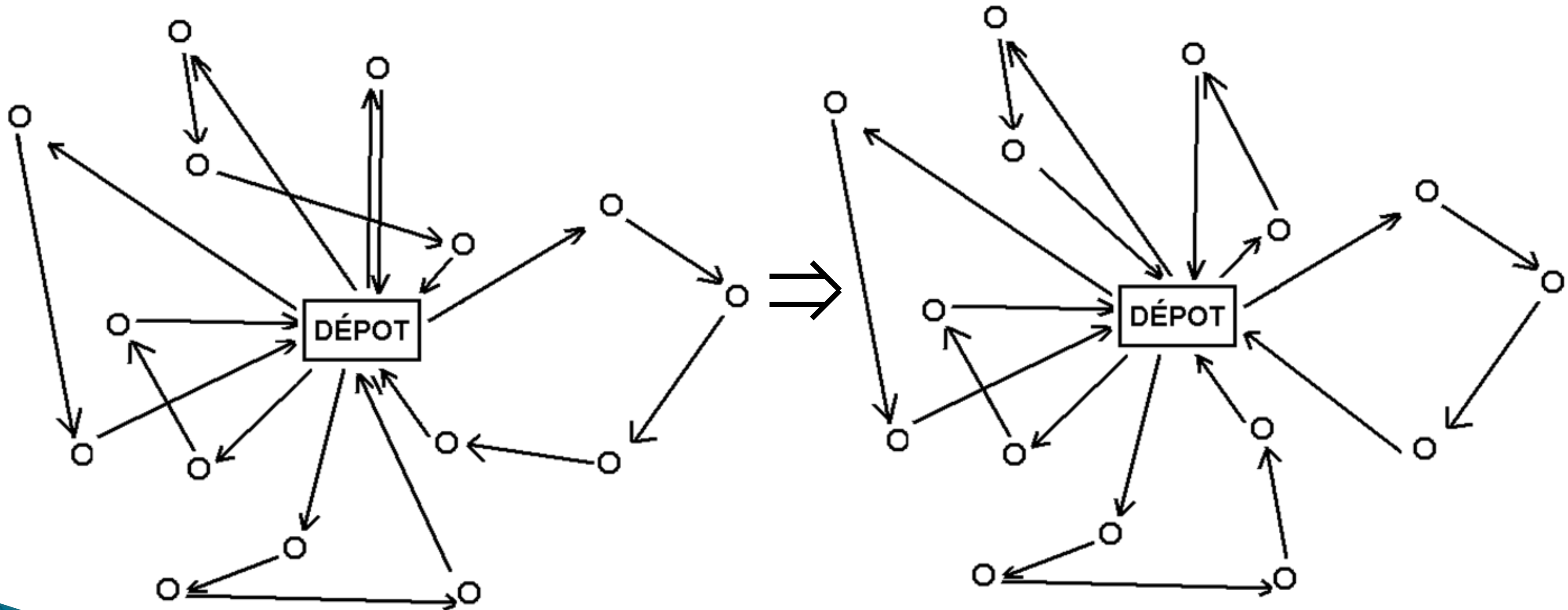
Large neighborhood search

- Iterative method
  - Current solution
  - Destruction
  - Reconstruction



Large neighborhood search

- New solution



Hybrid LNS-CG method outline

- Destruction
  - A roulette-wheel selection of known operators (ALNS of Pisinger and Ropke, 2007)
- Reconstruction
  - Heuristic version of the column generation method of Desaulniers et al. (2006)
- Two-phase approach
  - Reducing the number of vehicles
  - Reducing the traveled distance

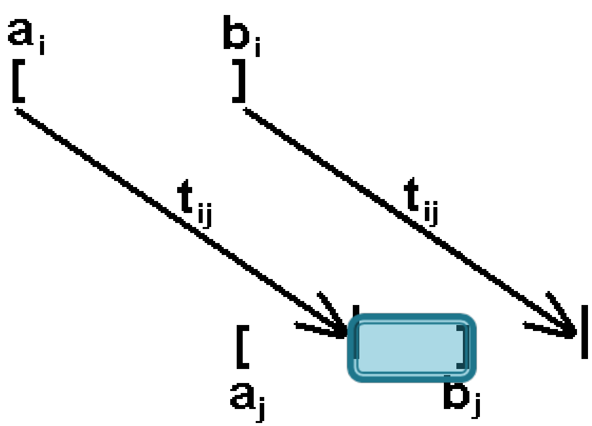
Destruction

- Neighborhood operators based on:
  - Proximity
  - Route portion
  - Longest detour
  - Time

- Roulette-wheel selection based on performance

Proximity operator
(Shaw, 1998)

- Select randomly a customer i
- Order the remaining customers according to their proximity to i



$$R(i,j) = \frac{1}{(c_{ij} + T_{ij})},$$

c_{ij} normalized distance ([0...1]) from i to j

T_{ij} normalized temporal proximity

$$T_{ij} = \frac{1}{min(b_j, b_i + t_{ij}) - max(a_j, a_i + t_{ij})}$$

Proximity operator
(Shaw, 1998)

- Select randomly a customer i
- Order the remaining customers according to their proximity to i
- Select randomly a new customer i’ favoring those having a greater proximity
- Select each subsequent customer according to its proximity to an already selected customer, which is chosen at random

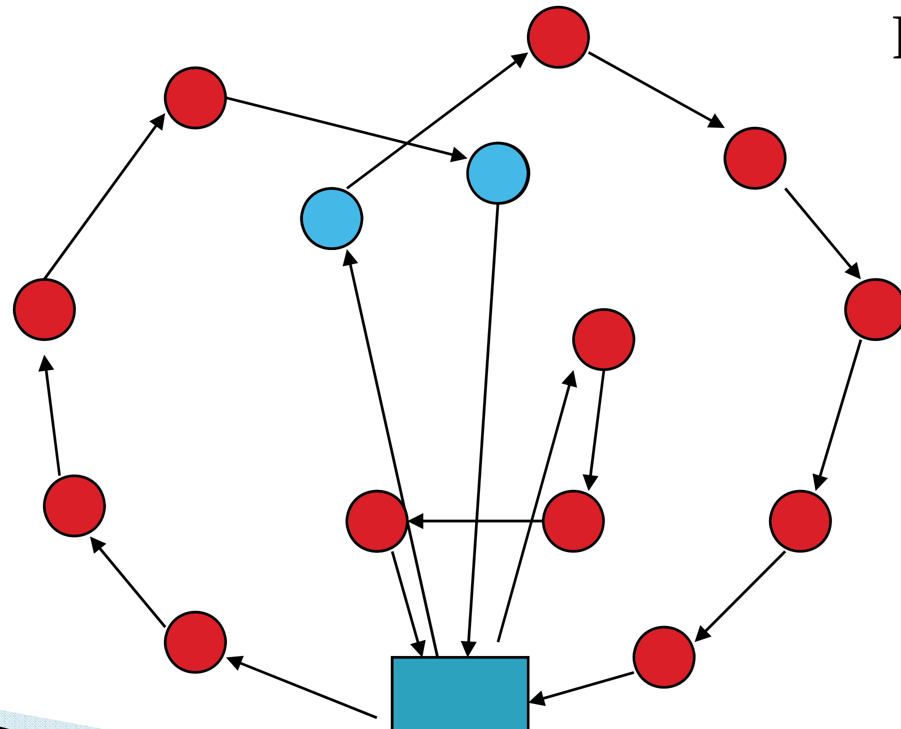
Route portion operator
(Rousseau et al., 2002)

Identify a seed customer

Remove preceding and succeeding arcs on same route

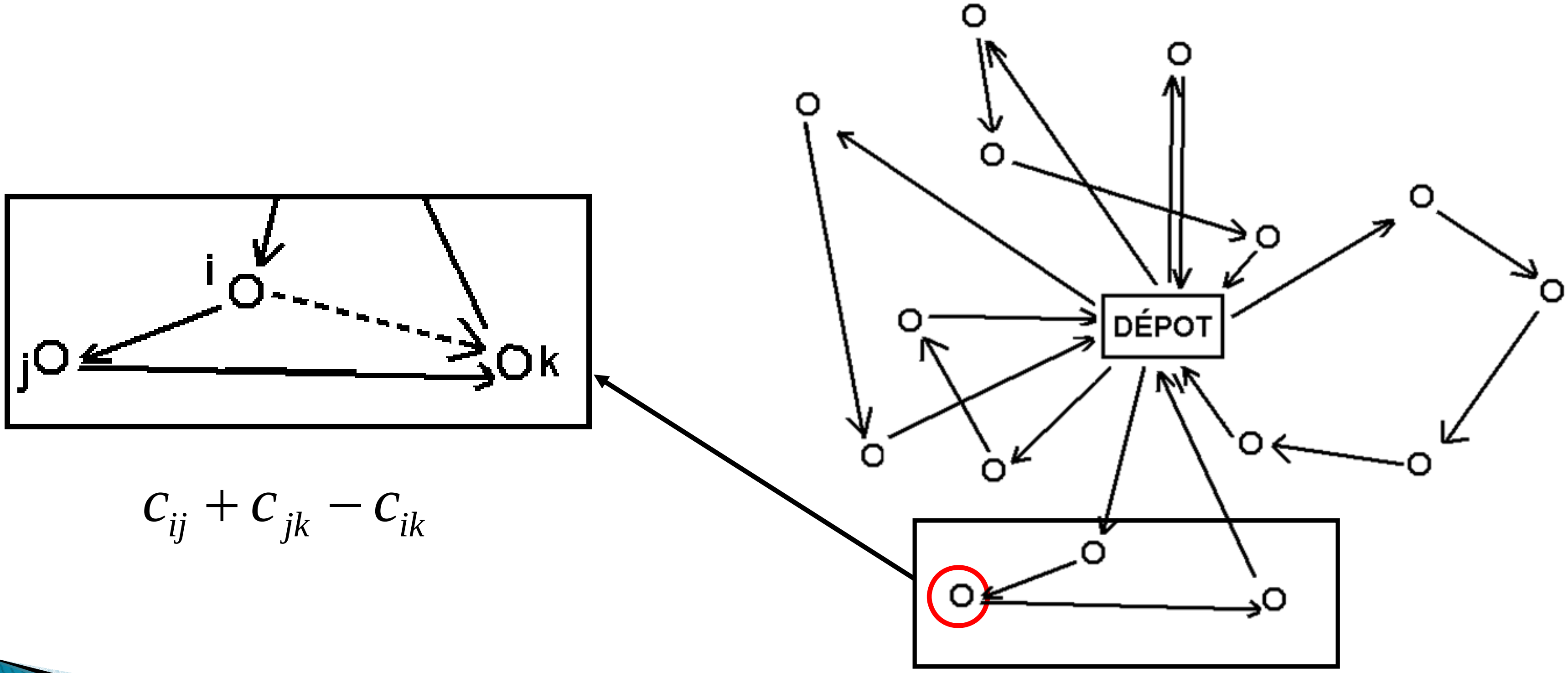
Identify a secondary seed customer

Remove other arcs



Longest detour operator
(Rousseau et al., 2002)

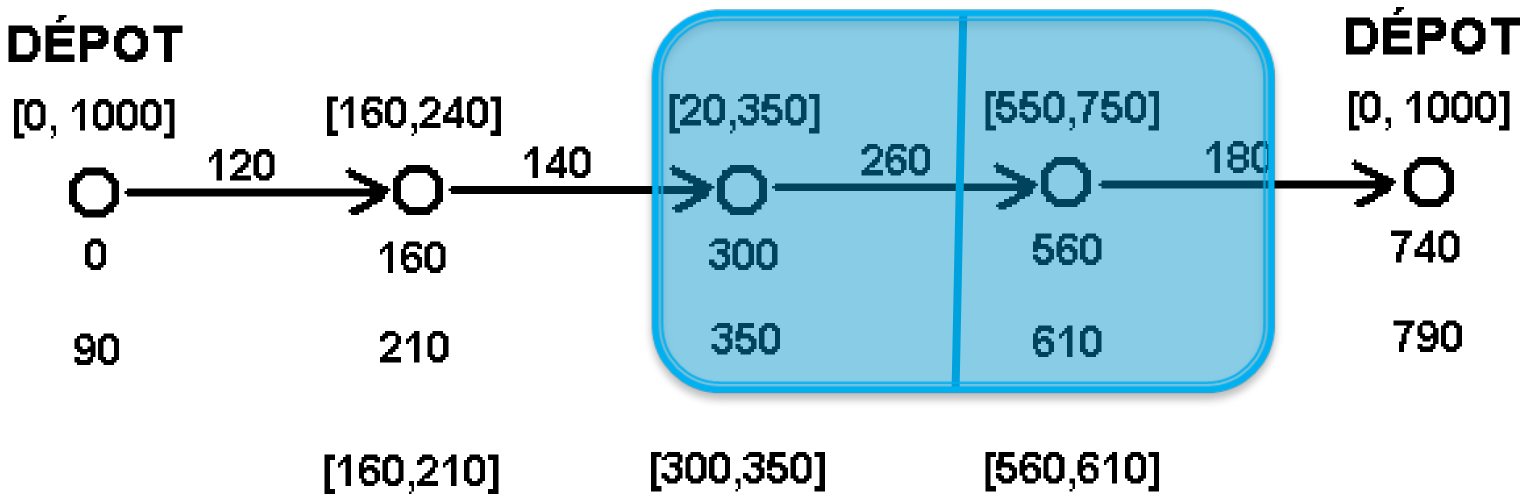
- Select randomly customers, favoring those generating longer detours



$$c_{ij} + c_{jk} - c_{ik}$$

Time operator

- Select randomly a specific time
- Select customers whose possible visiting time is closest to selected time



Roulette-wheel selection
(Pisinger & Ropke, 2007)

- Each operator i has an associated value π_i
- If operator i finds a better solution: $\pi_i = \pi_i + 1$
- Probability of choosing operator i = $\pi_i / \Sigma_j \pi_j$
- π_i values are reset to 5 every 100 iterations

Reconstruction

- Column generation made heuristic

1. Fixing part of the problem (remaining arcs)
2. Solving the subproblem with local search
3. Column generation is stopped after performing a number of iterations without significant improvement
4. Fixing column to obtain integer solutions
5. Keeping columns throughout LNS iterations

Solving the subproblem
Tabu search (Desaulniers et al., 2006)

- For each route in the current master problem basis

1. Set as initial solution
2. Apply local operator: Insert or remove a customer (or sequence of customers) from the current route
3. Maintain feasibility
4. If iteration limit is reached, move on to next column

Column fixing

- When tabu method cannot generate any column and solution is fractional

1. Fix one column
2. Re-start column generation
3. No backtracking (branch and dive ?)
4. May deteriorate solution cost (diversify search)

Keeping columns in memory

- Columns are kept in memory and reused when they are compatible with a given LNS iteration
- Total number of columns kept is limited to avoid memory shortage
- Interesting links to be made with adaptive and long term memory metaheuristics.
  - Traditional memory based metaheuristics have intricate search mechanisms that use a simple pool of known good routes.
  - Here the master problem is a kind of *intelligent* pool of routes that gives insightful guidance to a very simple search.
  - Some relations to evolutionary algorithms since the pools of columns implicitly represents a set of solutions

A two-phase approach

Recall that the VRPTW has a hierarchal objective

1. Vehicle reduction (VR)
   - Enforce an upper bound on the number of vehicles
   - Allow uncovered customers (large penalty)
   - Up to k_{VR} iterations to find a feasible solution
   - Switch to next phase if lower bound reached
   - Special version of the operators and parameters
2. Distance reduction (DR)
   - k_{DR} iterations to lower the distance

Modifications to destruction operator during vehicle reduction

- Proximity operator
  - Select an uncovered customer as first seed
- Route portion operator
  - Select an uncovered customer as first seed
- Longest detour operator
  - Select uncovered customers according to their proximity to longest detour customers
- Time operator
  - Visiting time of uncovered customers is the whole time window
- Roulette-wheel
  - Bonus to operators reducing the number of uncovered customers
- Tabu search
  - Only positive-valued columns are used as initial solutions
  - Number of iterations per column depends on the number of positive-valued columns

Computational experiments

- Benchmark problems
  - Solomon (1987) with 100 customers
  - Gehring & Homberger (1999) with 200 to 1000 customers

- Hierarchical objective function
  1. CNV: Cumulative number of vehicles
  2. CTD: Cumulative total distance

- 5 runs for each instance

Parameters

- k_{VR} = 400 iterations to reduce by one vehicle in VR phase
- k_{DR} = 800 ite
- 5 iterations R phase
- For *n* - cust
  - 60 custo
  - Total of 3 tion in VR phase
  - Column g t improvement
- For *n* - cust
  - 100 custo
  - Total of 1.5 iteration in VR phase
  - Column generation stopped after 5 iterations without improvement

All parameters behave like sliders that trade CPU time against solution quality

Computational results

- 100 customers (Solomon)

	PDR(best)	PDR(avg)	PR	BVH	B	I etal
CNV	405	406.6	405	405	405	405
CTD	57256	57101	57332	57273	57710	57444
Time (min)		18	2.5	120	82.5	250

PDR : Prescott-Gagnon, Desaulniers & Rousseau (2007)

PR: Pisinger & Ropke (2007)

BVH: Bent & Van Hentenryck (2004)

B: Bräysy (2003)

I etal: Ibaraki et al. (2002)

Computational results

- 200 customers (Gehring & Homberger)

	PDR(best)	PDR(avg)	PR	GH	MB	LCK
CNV	694	695	694	696	694	694
CTD	168553	168786	169042	179328	168572	169959
Time (min)		26	7.7	4x2.1	8	5x10

30 new best solutions

out of 60. According to
http://www.sintef.no/static/am/opti/projects/top/

PDR : Prescott-Gagnon, Desaulniers & Rousseau (2007)
PR: Pisinger & Ropke (2007)
GH: Gehring & Homberger (2001)
MB: Mester & Bräysy (2004)
LCK: Le Bouthillier, Crainic & Kropf (2005)

Computational results

- 400 customers (Gehring & Homberger)

	PDR(best)	PDR(avg)	PR	GH	MB	LCK
CNV	1385	1388.8	1385	1392	1389	1389
CTD	389011	390071	393210	428489	390386	396611
Time (min)		75	15.8	4x7.1	17	5x20

39 new best solutions

out of 60. According to http://www.sintef.no/static/am/opti/projects/top/

PDR : Prescott-Gagnon, Desaulniers & Rousseau (2007)
PR: Pisinger & Ropke (2007)
GH: Gehring & Homberger (2001)
MB: Mester & Bräysy (2004)
LCK: Le Bouthillier, Crainic & Kropf (2005)

Computational results

- 600 customers (Gehring & Homberger)

	PDR(best)	PDR(avg)	PR	GH	MB	LCK
CNV	2071	2074.4	2071	2079	2082	2086
CTD	800797	805325	807470	890121	796172	809493
Time (min)		88	18.3	4x12.9	40	5x30

29 new best solutions

out of 60. According to http://www.sintef.no/static/am/opti/projects/top/

PDR : Prescott-Gagnon, Desaulniers & Rousseau (2007)
PR: Pisinger & Ropke (2007)
GH: Gehring & Homberger (2001)
MB: Mester & Bräysy (2004)
LCK: Le Bouthillier, Crainic & Kropf (2005)

Computational results

- 800 customers (Gehring & Homberger)

	PDR(best)	PDR(avg)	PR	GH	MB	LCK
CNV	2745	2750.6	2758	2760	2765	2761
CTD	1391344	1401569	1358291	1535849	1361586	1443399
Time (min)		108	22.7	4x23.2	145	5x40

32 new best solutions

out of 60. According to http://www.sintef.no/static/am/opti/projects/top/

PDR : Prescott-Gagnon, Desaulniers & Rousseau (2007)
PR: Pisinger & Ropke (2007)
GH: Gehring & Homberger (2001)
MB: Mester & Bräysy (2004)
LCK: Le Bouthillier, Crainic & Kropf (2005)

Computational results

- 1000 customers (Gehring & Homberger)

	PDR(best)	PDR(avg)	PR	GH	MB	LCK
CNV	3432	3437.8	3438	3446	3446	3442
CTD	2096823	2110187	2110925	2290367	2078110	2133644
Time (min)		135	26.6	4x30.1	600	5x50

15 new best solutions

out of 60. According to http://www.sintef.no/static/am/opti/projects/top/

PDR : Prescott-Gagnon, Desaulniers & Rousseau (2007)
PR: Pisinger & Ropke (2007)
GH: Gehring & Homberger (2001)
MB: Mester & Bräysy (2004)
LCK: Le Bouthillier, Crainic & Kropf (2005)

Conclusion

- Column-generation-based Large Neighborhood Search
- Built with mostly known LNS operators
- Relies on a heuristic version of a powerful exact method
- Very effective
  - Best or close to best solution on all benchmarks
  - Improved 106 of 356 best known solutions

    (145 throughout the whole project)
- But not the fastest algorithm (e.g. Pisinger and Ropke)

Questions ?