Linear-time Split algorithm and applications

Thibaut Vidal

Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22451-900, Brazil
vidalt@inf.puc-rio.br

Seminar, University of Brescia
September 21$^{\text{th}}$, 2016

Contents

1. Giant-tour representations and the VRP
2. Bellman-based Split algorithm
3. Linear-time Split algorithm
   - Properties of the shortest-path graph
   - Unlimited fleet
   - Limited fleet
   - Soft capacity constraints
   - Computational experiments
4. Application: VRP with intermediate facilities
   - Problem Statement
   - Methodology
   - Computational experiments
5. Perspectives and Conclusions

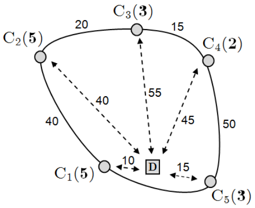
Contents

1. Giant-tour representations and the VRP
2. Bellman-based Split algorithm
3. Linear-time Split algorithm
   - Properties of the shortest-path graph
   - Unlimited fleet
   - Limited fleet
   - Soft capacity constraints
   - Computational experiments
4. Application: VRP with intermediate facilities
   - Problem Statement
   - Methodology
   - Computational experiments
5. Perspectives and Conclusions

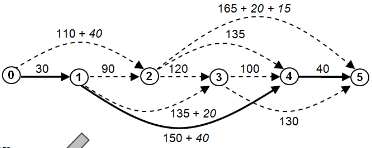
Giant-tour representations and the VRP

- Prins (2004) $\Rightarrow$ Important milestone for the VRP, first HGA to outperform classical Tabu searches
- A key ingredient of success: the giant-tour solution representation, allowing to use much simpler crossovers

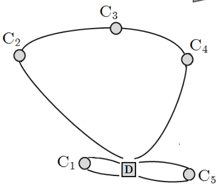
Giant tour representation with distances and demands :



Graph H & shortest path solution :



Optimal segmentation into routes :



Giant-tour representations and the VRP

- Ten years on ⇒ extensive growth of population-based methods.
- Efficient GAs with a complete solution representation and more advanced crossover operators now exist (Nagata and Bräysy, 2009)
- But the approach of Prins (2004) remains simple and generic
- Many generalizations (see the survey of Prins et al., 2014): capacity and duration limits, time windows, choices of depots, vehicle types, edges orientations in CARP, or profitable customers in each route...

Contents

1. Giant-tour representations and the VRP
2. Bellman-based Split algorithm
3. Linear-time Split algorithm
   - Properties of the shortest-path graph
   - Unlimited fleet
   - Limited fleet
   - Soft capacity constraints
   - Computational experiments
4. Application: VRP with intermediate facilities
   - Problem Statement
   - Methodology
   - Computational experiments
5. Perspectives and Conclusions

Problem and notations

- The "Splitting" problem:
- **INPUT:**
  - A giant tour of n customers with demands $q_1, \ldots, q_n$
  - A vehicle capacity limit Q
  - $d_{i,i+1}$ be the distances between two successive customers
  - d_{0i} and d_{i0} the distances from and to the depot
- **FIND:** a best segmentation of the tour into feasible routes which originate and return to the depot, and contain consecutive visits from the giant tour

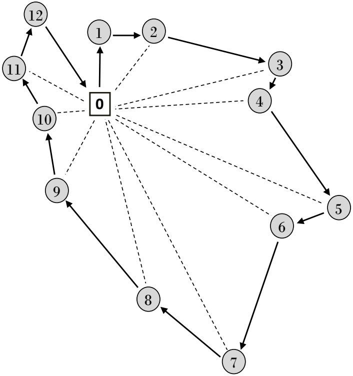
Problem and notations

- Classical formulation as the search for a shortest path between 0 and n in an acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$:
  - $\mathcal{V} = (0, \dots, n)$
  - each arc $(i, j) \in \mathcal{A}$ for $i < j$ corresponds to a feasible route starting at the depot, visiting customers $i + 1$ to j, and returning to the depot (Beasley, 1983; Prins, 2004).

Illustrative Example

Node	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
$d_{i-1,i}$	—	4	3	7	2	7	3	8	6	8	4	3	3
$d_{0,i}$	—	4	5	10	9	14	12	16	11	5	3	5	6
q_i	—	11	3	6	5	7	8	1	7	3	7	3	6
$p[i]$	**0**	**8**	**12**	**24**	**25**	**43**	**44**	56	67	69	75	80	84

with $\mathbf{Q = 30}$.

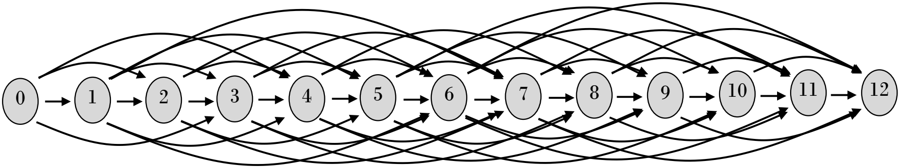


Illustrative Example

Node	0	1	2	3	4	5	6	7	8	9	10	11	12
$d_{i-1,i}$	—	4	3	7	2	7	3	8	6	8	4	3	3
$d_{0,i}$	—	4	5	10	9	14	12	16	11	5	3	5	6
q_i	—	11	3	6	5	7	8	1	7	3	7	3	6
$p[i]$	**0**	**8**	**12**	**24**	**25**	**43**	**44**	56	67	69	75	80	84

with $\mathbf{Q} = \mathbf{30}$.

Auxiliary Graph for Split:



with the cost of an arc (i,j):

$$c(i,j) = d_{0,i+1} + \sum_{k=i+1,\ldots,j-1} d_{k,k+1} + d_{j,0}$$

Bellman-based Split algorithm

$p[0] \leftarrow 0$;
for $t = 1$ *to* n **do**
    $p[t] \leftarrow \infty$;
for $t = 0$ *to* $n - 1$ **do**
    $load \leftarrow 0$;
    $i \leftarrow t + 1$;
    while $i \leq n$ ***and*** $load + q_i \leq Q$ **do**
        $load \leftarrow load + q_i$;
        if $i = t + 1$ **then**
            $cost \leftarrow d_{0,i}$;
        else
            $cost \leftarrow cost + d_{i-1,i}$;
        if $p[t] + cost + d_{i0} < p[i]$ **then**
            $p[i] = p[t] + cost + d_{i0}$;
            $pred[i] = t$;
        $i \leftarrow i + 1$;

- $O(n^2)$ complexity $\Rightarrow$ in practice $O(nB)$ if the average number of customers in a feasible route is bounded by a constant B.

- **Question 1: Can we do better?**
- **Question 2: If we have a better Split, what can we do with it?**

Bellman-based Split algorithm

- **Question 1: Can we do better?**
- **Question 2: If we have a better Split, what can we do with it?**

Contents

1. Giant-tour representations and the VRP
2. Bellman-based Split algorithm
3. Linear-time Split algorithm
   - Properties of the shortest-path graph
   - Unlimited fleet
   - Limited fleet
   - Soft capacity constraints
   - Computational experiments
4. Application: VRP with intermediate facilities
   - Problem Statement
   - Methodology
   - Computational experiments
5. Perspectives and Conclusions

Monge property

- Some $O(n)$ algorithms are, in fact, already known for this shortest path (see Burkard et al., 1996; Bein et al., 2005, and the references therein) since the graph $\mathcal{G}$ satisfies the Monge property:

$$\begin{aligned} c(i_1, j_1) + c(i_2, j_2) \leq c(i_1, j_2) + c(i_2, j_1) \\ \text{for all } 0 \leq i_1 < i_2 < j_1 < j_2 \leq n \\ \text{such that } (i_1, j_2) \in \mathcal{A}, \end{aligned} \tag{3.1}$$

- But this was not used to this date in the VRP literature...

An Even Stronger Property

- The Split graph satisfies in fact **an even stronger property:**

$$\text{for all } 0 \leq i_1 < i_2 < n, \text{ there exists } K \in \mathbb{R} \text{ such that}$$
$$c(i_1, j) - c(i_2, j) = K \text{ for all } j > i_2 \text{ such that } (i_1, j) \in \mathcal{A}.$$

- This property will be used to **eliminate dominated predecessors** and retain only good candidates
- $\Rightarrow$ leading to a very simple labeling algorithm in $\mathcal{O}(n)$ which can be efficiently used in practice.

An Even Stronger Property

- The Split graph satisfies in fact **an even stronger property:**

$$\text{for all } 0 \le i_1 < i_2 < n, \text{ there exists } K \in \mathbb{R} \text{ such that}$$
$$c(i_1, j) - c(i_2, j) = K \text{ for all } j > i_2 \text{ such that } (i_1, j) \in \mathcal{A}.$$

- This property will be used to **eliminate dominated predecessors** and retain only good candidates
- $\Rightarrow$ leading to a very simple labeling algorithm in $\mathcal{O}(n)$ which can be efficiently used in practice.

Towards a very simple algorithm

- **Some notations:** For $i \in \{1, \ldots, n\}$, define the cumulative distance $D[i]$ and cumulative load $Q[i]$:

$$D[i] = \sum_{k=1}^{i-1} d_{k,k+1} \tag{3.2}$$

$$Q[i] = \sum_{k=1}^{i} q_k. \tag{3.3}$$

- Then, the cost can be accessed as:

$$c(i,j) = d_{0,i+1} + D[j] - D[i+1] + d_{j,0}, \tag{3.4}$$

- and the arc (i,j) exists if and only if the route is feasible, i.e., $Q[j] - Q[i] \leq Q$.

Towards a very simple algorithm

- We also rely on a double-ended queue Λ, which supports the following operations in $\mathcal{O}(1)$:

front – accesses the oldest element in the queue;
front2 – accesses the second-oldest element in the queue;
back – accesses the most recent element in the queue;
push_back – adds an element to the queue;
pop_front – removes the oldest element in the queue;
pop_back – removes the newest element in the queue.

We refer to the elements of the queue as $(\lambda_1, \ldots, \lambda_{|\Lambda|})$, from the front λ_1 to the back $\lambda_{|\Lambda|}$.

Towards a very simple algorithm

We propose the following linear time Split algorithm:

```
p[0] ← 0 ;
Λ ← (0) ;
for t = 1 to n do
    p[t] ← p[front] + f(front, t) ;
    pred[t] ← front ;
    if t < n then
        if not dominates(back, t) then
            while |Λ| > 0 and dominates(t, back) do
                popBack() ;
            pushBack(t)
        while Q[t + 1] > Q + Q[front] do
            popFront() ;
```

With the boolean function $dominates(i, j) \equiv$

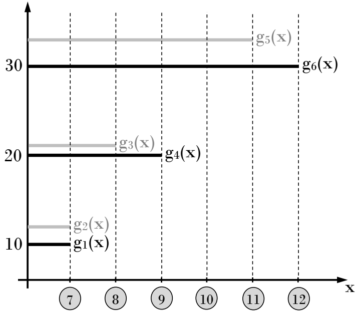
$$\begin{cases} p[i] + d_{0,i+1} - D[i+1] \leq p[j] + d_{0,j+1} - D[j+1] \textbf{ and } Q[i] = Q[j] & \text{if } i \leq j \\ p[i] + d_{0,i+1} - D[i+1] \leq p[j] + d_{0,j+1} - D[j+1] & \text{if } i > j \end{cases}$$

Towards a very simple algorithm

Correctness of the algorithm: Define $f(i,x)$ the cost when extending the label of a predecessor i to a node $x \in \{i+1,\ldots,n\}$:

$$f(i,x) = \begin{cases} p[i] + c(i,x) & Q[x] - Q[i] \leq Q \\ \infty & otherwise \end{cases}$$

...and the auxiliary function $g_i(x) = f(i,x) - D[x] - d_{x0}$. **This function of x takes a constant value as long as the label extension is feasible.**



(if $Q[x] - Q[i] \leq Q$, then

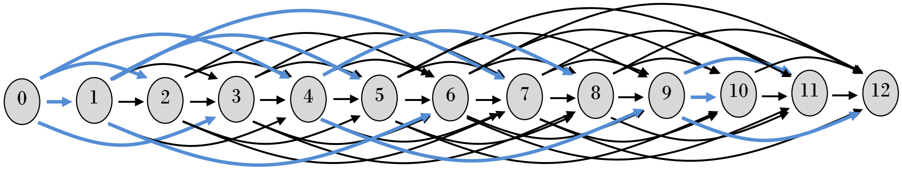
$g_i(x) = p[i] + d_{0,i+1} + D[x] - D[i+1] + d_{x0} - D(x) - d_{x0} = p[i] + d_{0,i+1} - D[i+1]$

Illustrative Example

Node	0	1	2	3	4	5	6	7	8	9	10	11	12
$d_{i-1,i}$	—	4	3	7	2	7	3	8	6	8	4	3	3
$d_{0,i}$	—	4	5	10	9	14	12	16	11	5	3	5	6
q_i	—	11	3	6	5	7	8	1	7	3	7	3	6
$p[i]$	**0**	**8**	**12**	**24**	**25**	**43**	**44**	56	67	69	75	80	84

with $\mathbf{Q} = \mathbf{30}$.

Those were the arcs (in blue) explored in practice on the illustrative example:



Extension to limited fleets

- Split considering a limited fleet of m vehicles in $O(nm)$ (instead of $O(nBm)$)

```
for k = 1 to m do
    for t = 0 to n do
        p[k, t] = ∞ ;
p[0, 0] ← 0 ;
for k = 0 to m − 1 do
    clear(Λ) ;
    Λ ← (k) ;
    for t = k + 1 to n s.t. |Λ| > 0 do
        p[k + 1, t] ← p[k, front] + f(front, t) ;
        pred[k + 1][t] ← front ;
        if t < n then
            if not dominates(k, back, t) then
                while |Λ| > 0 and dominates(k, t, back) do
                    popBack() ;
                pushBack(t)
            while |Λ| > 0 and Q[t + 1] > Q + Q[front] do
                popFront() ;
```

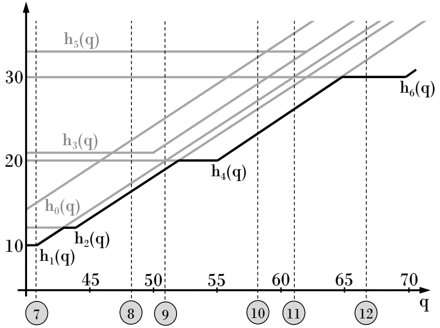
Management of soft capacity constraints

- Soft capacity constraints can also be addressed via a change of the function $dominates(i,j) \equiv$

$$\begin{cases} p[i] + d_{0,i+1} - D[i+1] + \alpha \times (Q[j] - Q[i]) \leq p[j] + d_{0,j+1} - D[j+1] & \text{if } i < j \\ p[i] + d_{0,i+1} - D[i+1] \leq p[j] + d_{0,j+1} - D[j+1] & \text{if } i > j. \end{cases}$$

- The rule for eliminating the front label also requires a minor adaptation (see paper)

- The complexity remains $O(n)$.



Computational experiments

- 105 benchmark instances based on the TSPLib
- 29 to 71,009 nodes
- 10 vehicle capacities: $Q \in \{10^2, 2{\times}10^2, 4{\times}10^2, 10^3, 2{\times}10^3, 4{\times}10^3, 10^4, 2{\times}10^4, 4{\times}10^4, 10^5\}$
- Comparing the speed of the classical Bellman-based Split algorithm with the linear Split for the three problem settings
- Xeon 3.07 GHz CPU, using a single thread.

Computational experiments

We compare the following algorithms:

Algorithm:	**Complexity:**
Bellman–Based Split algorithm	$O(nB)$
Bellman–Based Split algorithm with a fleet–size limit m	$O(nBm)$
Bellman–Based Split algorithm with soft capacity constraints	$O(n^2)$
Linear Split algorithm	$O(n)$
Linear Split algorithm with a fleet–size limit m	$O(nm)$
Linear Split algorithm with soft capacity constraints	$O(n)$

Computational experiments

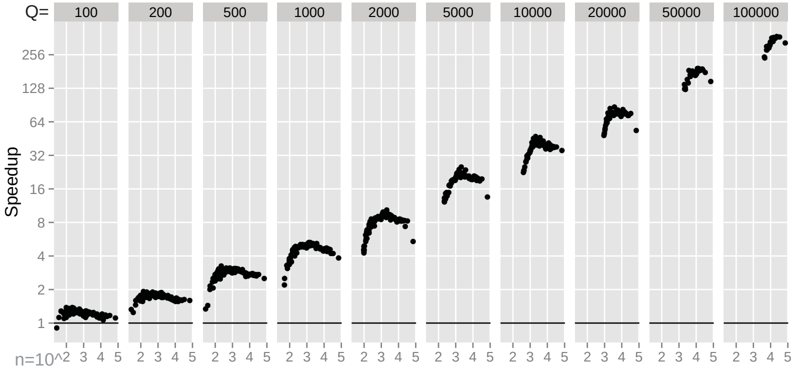


Figure : Speedups of the linear Split over the Bellman-based algorithm for all 105 instances. Hard capacity constraints, unlimited fleet.

Computational experiments

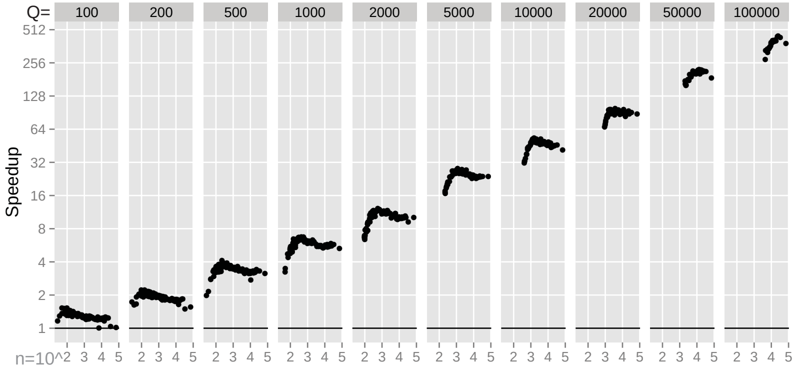


Figure : Speedup factors for the case with a limited fleet.

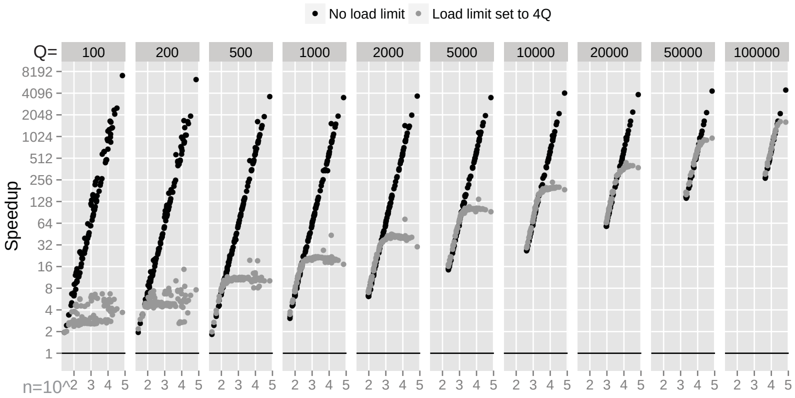


Figure : Speedups for soft capacity constraints. Two sets of results: the speedups relative to the Bellman algorithm with no limit on the excess capacity (black dots), and those relative to the Bellman algorithm with a limit of $4Q$ on the total demand of a route (gray dots).

Contents

1 Giant-tour representations and the VRP

2 Bellman-based Split algorithm

3 Linear-time Split algorithm
- Properties of the shortest-path graph
- Unlimited fleet
- Limited fleet
- Soft capacity constraints
- Computational experiments

4 Application: VRP with intermediate facilities
- Problem Statement
- Methodology
- Computational experiments

5 Perspectives and Conclusions

The VRP with intermediate facilities

- **The VRP with intermediate facilities** (see, e.g. Crevier et al., 2007; Tarantilis et al., 2008; Hemmelmayr et al., 2013; Schneider et al., 2015):

- Classical duration-constrained CVRP
- With the possibility to reload at a subset of intermediate facilities locations
  - Docking time at the intermediate facilities
  - Service time at the customers
  - Duration constraint is global on the whole route

- Generalizes the multi-trip VRP
- Close connections to green VRPs with choices of recharging stations

The VRP with intermediate facilities

- **The VRP with intermediate facilities** (see, e.g. Crevier et al., 2007; Tarantilis et al., 2008; Hemmelmayr et al., 2013; Schneider et al., 2015):

- Classical duration-constrained CVRP
- With the possibility to reload at a subset of intermediate facilities locations
  - Docking time at the intermediate facilities
  - Service time at the customers
  - Duration constraint is global on the whole route

- Generalizes the multi-trip VRP
- Close connections to green VRPs with choices of recharging stations

The VRP with intermediate facilities

- **The VRP with intermediate facilities** (see, e.g. Crevier et al., 2007; Tarantilis et al., 2008; Hemmelmayr et al., 2013; Schneider et al., 2015):

- Classical duration-constrained CVRP
- With the possibility to reload at a subset of intermediate facilities locations
  - Docking time at the intermediate facilities
  - Service time at the customers
  - Duration constraint is global on the whole route

- Generalizes the multi-trip VRP
- Close connections to green VRPs with choices of recharging stations

The VRP with intermediate facilities

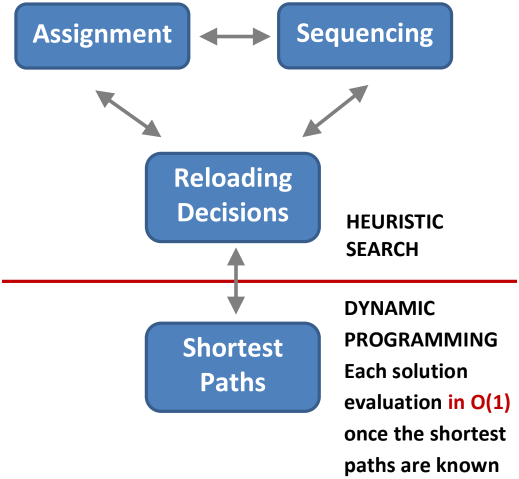
- **The VRP with intermediate facilities** (see, e.g. Crevier et al., 2007; Tarantilis et al., 2008; Hemmelmayr et al., 2013; Schneider et al., 2015):

- Classical duration-constrained CVRP
- With the possibility to reload at a subset of intermediate facilities locations
  - Docking time at the intermediate facilities
  - Service time at the customers
  - Duration constraint is global on the whole route

- Generalizes the multi-trip VRP
- Close connections to green VRPs with choices of recharging stations

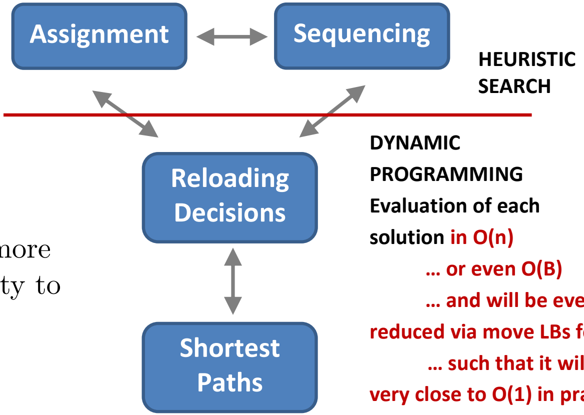
A question of search space

- 3–4 main decision sets
- and a classical way to deal with them:



⇒ **This is, however, not a unique option.**

A question of search space



DYNAMIC PROGRAMMING
Evaluation of each solution in O(n)
… or even O(B)
… and will be even further reduced via move LBs for filtering
… such that it will become very close to O(1) in practice

⇒ Let's give more responsibility to DP:

Move evaluations

- Evaluating any neighbor solution, defined as sequences of services without visits to intermediate facilities, requires to solve an optimization problem for the choice of visits to intermediate facilities.

- Can be transformed into an instance of Split problem

  (with some pre-processing prior to routing optimization: find for any customer pair (i, j) the facility which leads to the smallest detour).

- Now solved in $O(n)$

Move LBs

- This solution evaluation procedure is more time consuming than usual.

- To save some computational effort, rely on lower bounds on solution cost to filter non-promising 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 move evaluations

- In the VRP-IF, the cost of a route σ is always greater than
  - the total travel distance (without recharging), plus
  - the minimum number of necessary visits
    $\times$ shortest detour $S(\sigma)$ to a facility

$$\bar{Z}(\sigma) = \sum_{i=1}^{|\sigma|-1} d_{\sigma_i \sigma_{i+1}} + \left\lfloor \frac{\sum_{i=1}^{|\sigma|} q_{\sigma_i}}{Q} \right\rfloor \times S(\sigma)$$

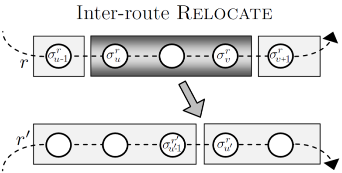
- And this bound helps, in practice, to filter a significant subset of the moves

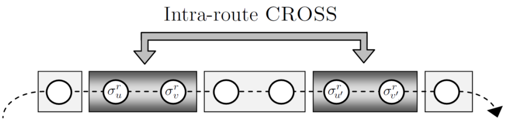
(**Experiments of today**)

Preprocessing and bidirectional search

- To improve further the move evaluations, it is even possible to avoid solving each SP subproblem independently in $O(n)$
  - $\Rightarrow$ Rely instead on pre-processed shortest paths for partial routes.

- Key property of classical routing neighborhoods:
  - Any local-search move involving a bounded number of node relocations or arc exchanges can be assimi-
lated to a concatenation of a bounded number of sub-sequences.

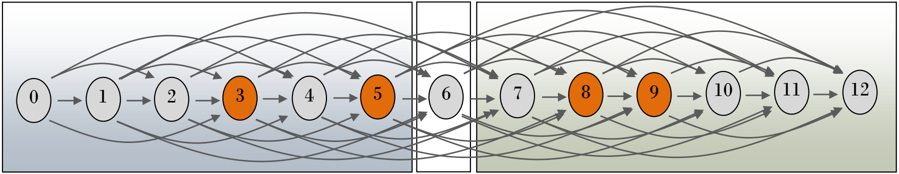




  - To decrease the computational complexity, compute auxiliary data on subsequences by induction on concatenation ($\oplus$).

Preprocessing and bidirectional search

- Now, consider an inter-route move, which inserts or replaces a bounded number of customers in a route.
  - ⇒ New route obtained by the concatenation of 3 services sequences
  - ⇒ Prior to move evaluations, we pre-process the shortest paths from the node 0 to the subsequent nodes, and from the end (backwards) to each node, in $O(n)$.



⇒ Reusing the preprocessed information allows to evaluate each classical inter-route move in $O(B)$.

⇒ We discuss later about intra-route moves...

Computational experiments

- **Some Preliminary experiments with:**
- The ILS variant of Prins (2009)
  - Produces iteratively n_C offspring from the incumbent solution (via shaking and LS) and selects the best. Search is restarted n_P times until n_I consecutive generations without improvement. Shaking done by 1 or 2 random swaps, with equal probability.
- The unified hybrid genetic search (UHGS) of Vidal et al. (2012, 2014)

Computational experiments

- LS based on the classical routing neighborhoods (but applied on solutions represented without intermediate-facility visits): Relocate, Swap, CROSS, 2-opt and 2-opt*.
  - Exploration in random order
  - First improvement policy
  - Restrictions of moves to the Γ^{TH} closest services

    $\Rightarrow$ Number of neighbors in $\mathcal{O}(n)$

Computational experiments

- Using a short termination criterion: $(n_P, n_C, n_I) = (5, 10, 50)$ for ILS, and $It_{\text{MAX}} = 5,000$ for UHGS
- Single core: Xeon 3.07 GHz CPU with 16 GB of RAM
- Reporting the average and best solutions on 10 runs.
- All Gap(%) values measured from the best known solutions (BKS)

Computational experiments

- Comparing with the previous methods for this problem:

  CCL07: Hybrid TS with Adaptive Memory Programming and Integer Programming of Crevier et al. (2007)

  TZK08: Hybrid guided local search of Tarantilis et al. (2008)

  HDHR13: Variable neighborhood search of Hemmelmayr et al. (2013)

  SSH15: Adaptive VNS of Schneider et al. (2015)

Computational experiments

				CCL07		TZK08			HDHR13			SSH15			ILS			BKS
Inst	n	m	r	Avg-10	T	Avg-10	Best-10	T	Avg-10	Best-10	T	Avg-10	Best-10	T	Avg-10	Best-10	T	
a1	48	6	3	1211.28	4.58	1189.70	**1179.79**	3.38	1180.57	**1179.79**	1.42	1184.57	**1179.79**	0.64	**1179.79**	**1179.79**	1.46	1179.79
b1	96	4	3	1232.67	9.17	1225.08	**1217.07**	7.80	**1217.07**	**1217.07**	6.39	1218.21	**1217.07**	4.19	**1217.07**	**1217.07**	5.20	1217.07
c1	192	5	3	1893.01	36.22	1898.92	1883.05	34.21	1867.96	1866.76	20.40	1925.41	1882.46	32.98	1869.20	1866.76	30.05	1866.76
d1	48	5	4	1076.31	8.55	1064.29	**1059.43**	5.87	**1059.43**	1059.43	1.57	1061.5	**1059.43**	0.55	**1059.43**	**1059.43**	1.34	1059.43
e1	96	5	4	1311.60	13.52	**1309.12**	**1309.12**	8.62	**1309.12**	**1309.12**	6.22	1312.75	**1309.12**	5.08	**1309.12**	**1309.12**	3.47	1309.12
f1	192	4	4	1601.54	41.41	1585.83	1572.17	38.81	1573.05	**1570.41**	25.60	1601.4	1577.63	34.99	1571.86	**1570.41**	30.04	1570.41
g1	72	5	5	1202.00	55.22	1190.21	**1181.13**	5.79	1183.32	**1181.13**	3.38	1183.75	**1181.13**	1.69	**1181.13**	**1181.13**	5.84	1181.13
h1	144	4	5	1598.51	32.07	1577.54	1547.25	11.06	1548.61	**1545.50**	14.61	1567.22	1553.75	14.08	1547.23	**1545.50**	22.54	1545.50
i1	216	4	5	1976.11	51.01	1956.17	1925.99	42.50	1923.52	**1922.18**	33.58	1974.97	1934.08	35.11	1925.72	**1922.18**	30.07	1922.18
j1	72	4	6	1161.77	58.90	1128.86	1117.20	5.52	**1115.78**	**1115.78**	2.78	1116.82	**1115.78**	2.02	**1115.78**	**1115.78**	2.35	1115.78
k1	144	4	6	1618.45	64.61	1591.74	1580.39	12.07	1577.96	1576.36	14.56	1600.42	1577.98	10.74	1577.89	1573.21	20.93	1576.36
l1	216	4	6	1917.08	104.27	1904.39	1880.60	51.39	1869.70	**1863.28**	35.48	1916.07	1894.69	40.59	1873.37	1868.70	30.08	1863.28
a2	48	4	5	1005.16	6.39	–	–	–	**997.94**	**997.94**	1.23	**997.94**	**997.94**	0.72	**997.94**	**997.94**	0.70	997.94
b2	96	4	5	1333.20	14.72	–	–	–	**1291.19**	**1291.19**	6.41	1300.42	**1291.19**	4.83	1292.95	1292.95	5.51	1291.19
c2	144	4	5	1792.46	61.68	–	–	–	1715.84	**1715.60**0	15.01	1741.55	**1715.60**	18.32	1716.40	1716.40	18.56	1715.60
d2	192	3	5	1898.21	40.54	–	–	–	1860.92	1856.84	30.14	1903.15	1874.12	30.64	1862.19	1858.81	30.06	1856.84
e2	240	3	5	1995.75	73.78	–	–	–	1922.81	1919.38	49.31	1957.8	1937.84	41.6	1930.04	1919.23	30.14	1919.38
f2	288	3	5	2312.15	162.22	–	–	–	2233.43	**2230.32**	71.24	2313.08	2268.54	42.8	2255.59	2238.26	30.21	2230.32
g2	72	4	7	1185.93	29.51	–	–	–	1153.17	**1152.92**	3.71	1158.21	**1152.92**	2.2	**1152.92**	**1152.92**	2.76	1152.92
h2	144	4	7	1611.75	160.79	–	–	–	**1575.28**	**1575.28**	15.66	1586.24	1576.86	21.2	1575.67	**1575.28**	16.85	1575.28
i2	216	3	7	1998.20	322.41	–	–	–	1922.24	**1919.74**	41.92	1971.27	1944.74	41.1	1928.80	1920.75	30.08	1919.74
j2	288	3	7	2325.18	256.85	–	–	–	2250.21	**2247.70**	73.38	2303.67	2281.86	41.93	2262.16	2249.79	30.19	2247.70
Gap(%)				2.63%		1.14%	0.22%		0.09%	0.00%		1.44%	0.49%		0.20%	0.04%		
T(min)					73.11			18.92			21.55			19.46			17.20	
CPU				Prosys 2GHz		PIV 2.4 GHz			2.4 GHz			I5 2.67 GHz			Xe 3.07G			

Computational experiments

Inst	n	m	r	CCL07		TZK08			HDHR13			SSH15			UHGS			BKS
				Avg-10	T	Avg-10	Best-10	T	Avg-10	Best-10	T	Avg-10	Best-10	T	Avg-10	Best-10	T	
a1	48	6	3	1211.28	4.58	1189.70	**1179.79**	3.38	1180.57	**1179.79**	1.42	1184.57	**1179.79**	0.64	**1179.79**	**1179.79**	2.80	1179.79
b1	96	4	3	1232.67	9.17	1225.08	**1217.07**	7.80	**1217.07**	**1217.07**	6.39	1218.21	**1217.07**	4.19	**1217.07**	**1217.07**	10.13	1217.07
c1	192	5	3	1893.01	36.22	1898.92	1883.05	34.21	1867.96	1866.76	20.40	1925.41	1882.46	32.98	1866.62	**1863.49**	30.01	1866.76
d1	48	5	4	1076.31	8.55	1064.29	**1059.43**	5.87	**1059.43**	1059.43	1.57	1061.5	**1059.43**	0.55	**1059.43**	**1059.43**	2.64	1059.43
e1	96	5	4	1311.60	13.52	**1309.12**	**1309.12**	8.62	**1309.12**	**1309.12**	6.22	1312.75	**1309.12**	5.08	**1309.12**	**1309.12**	8.36	1309.12
f1	192	4	4	1601.54	41.41	1585.83	1572.17	38.81	1573.05	**1570.41**	25.60	1601.4	1577.63	34.99	1572.19	**1570.41**	30.02	1570.41
g1	72	5	5	1202.00	55.22	1190.21	**1181.13**	5.79	1183.32	**1181.13**	3.38	1183.75	**1181.13**	1.69	**1181.13**	**1181.13**	12.31	1181.13
h1	144	4	5	1598.51	32.07	1577.54	1547.25	11.06	1548.61	**1545.50**	14.61	1567.22	1553.75	14.08	1545.56	**1545.50**	30.01	1545.50
i1	216	4	5	1976.11	51.01	1956.17	1925.99	42.50	1923.52	**1922.18**	33.58	1974.97	1934.08	35.11	1924.51	1923.62	30.02	1922.18
j1	72	4	6	1161.77	58.90	1128.86	1117.20	5.52	**1115.78**	**1115.78**	2.78	1116.82	**1115.78**	2.02	**1115.78**	**1115.78**	5.13	1115.78
k1	144	4	6	1618.45	64.61	1591.74	1580.39	12.07	1577.96	1576.36	14.56	1600.42	1577.98	10.74	1576.30	**1573.21**	30.01	1576.36
l1	216	4	6	1917.08	104.27	1904.39	1880.60	51.39	1869.70	**1863.28**	35.48	1916.07	1894.69	40.59	1871.83	1865.27	30.02	1863.28
a2	48	4	5	1005.16	6.39	–	–	–	**997.94**	**997.94**	1.23	**997.94**	**997.94**	0.72	**997.94**	**997.94**	1.50	997.94
b2	96	4	5	1333.20	14.72	–	–	–	**1291.19**	**1291.19**	6.41	1300.42	**1291.19**	4.83	1292.95	1292.95	10.35	1291.19
c2	144	4	5	1792.46	61.68	–	–	–	1715.84	**1715.60**0	15.01	1741.55	**1715.60**	18.32	1716.40	1716.40	30.01	1715.60
d2	192	3	5	1898.21	40.54	–	–	–	1860.92	1856.84	30.14	1903.15	1874.12	30.64	1858.87	**1853.86**	30.01	1856.84
e2	240	3	5	1995.75	73.78	–	–	–	1922.81	1919.38	49.31	1957.8	1937.84	41.6	1923.74	**1919.23**	30.02	1919.38
f2	288	3	5	2312.15	162.22	–	–	–	2233.43	**2230.32**	71.24	2313.08	2268.54	42.8	2248.85	2230.95	30.04	2230.32
g2	72	4	7	1185.93	29.51	–	–	–	1153.17	**1152.92**	3.71	1158.21	**1152.92**	2.2	**1152.92**	**1152.92**	5.01	1152.92
h2	144	4	7	1611.75	160.79	–	–	–	**1575.28**	**1575.28**	15.66	1586.24	1576.86	21.2	1575.60	**1575.28**	29.75	1575.28
i2	216	3	7	1998.20	322.41	–	–	–	1922.24	**1919.74**	41.92	1971.27	1944.74	41.1	1926.76	1920.75	30.03	1919.74
j2	288	3	7	2325.18	256.85	–	–	–	2250.21	**2247.70**	73.38	2303.67	2281.86	41.93	2263.89	2253.18	30.05	2247.70
Gap(%)				2.63%		1.14%	0.22%		0.09%	0.00%		1.44%	0.49%		0.14%	0.01%		
T(min)					73.11			18.92			21.55			19.46			20.37	
CPU				Prosys 2GHz		PIV 2.4 GHz			2.4 GHz			I5 2.67 GHz			Xe 3.07G			

Contents

1. Giant-tour representations and the VRP
2. Bellman-based Split algorithm
3. Linear-time Split algorithm
   - Properties of the shortest-path graph
   - Unlimited fleet
   - Limited fleet
   - Soft capacity constraints
   - Computational experiments
4. Application: VRP with intermediate facilities
   - Problem Statement
   - Methodology
   - Computational experiments
5. Perspectives and Conclusions

Conclusions

- Introduced a simple linear-time Split algorithm
  - Simple to implement, efficient in practice
  - Large speedups when run on problem instances with long routes
  - Possible limited fleet, soft capacity constraints, etc...

- Opportunity of applications to problem classes with intermediate facilities, multiple trips, or recharging stations
  - Allows to deal with the decision subset related to intermediate-facilities visits via tailored solution evaluation procedures rather than tailored moves
  - Preliminary results on the VRP-IF (with a short termination criterion) look OK.

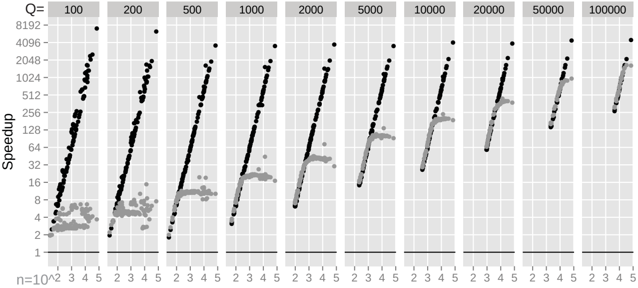
Conclusions

- Many other opportunities related to Split in the VRP:
  - More intensive search in the space of giant tours
  - Improvements for other forms of split algorithms, e.g., HVRP, LRP, etc...
  - Many results that we know on Split have connections with results on other enumerative neighborhoods in local searches...

- Aiming for a paradigm shift — we assume too fast that the classical neighborhoods and their complexities are established
  - When an improvement occurs, large potential gains
  - Wide scope of application
  - Average case $O(n \log n)$ exploration procedures are also known for several other problems and neighborhoods... (Bentley and Friedman, 1978; Bentley, 1992)

Thank You I

Thank you for your attention !



… and a happy optimized birthday !!

Thank You II

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

Bein, W., P. Brucker, L.L. Larmore, J.K. Park. 2005. The algebraic Monge property and path problems. *Discrete Applied Mathematics* **145**(3) 455–464.

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

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

Burkard, R.E., B. Klinz, R. Rudolf. 1996. Perspectives of Monge properties in optimization. *Discrete Applied Mathematics* **70**(2) 95–161.

Crevier, B., J.-F. Cordeau, G. Laporte. 2007. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research* **176**(2) 756–773.

Hemmelmayr, V, K F Doerner, R F Hartl, S Rath. 2013. A heuristic solution method for node routing based solid waste collection problems. *Journal of Heuristics* **19** 129–156.

Nagata, Y., O. Bräysy. 2009. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks* **54**(4) 205–215.

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

Thank You III

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., P. Lacomme, C. Prodhon. 2014. Order-first split-second methods for vehicle routing problems: A review. *Transportation Research Part C: Emerging Technologies* **40** 179–200.

Schneider, Michael, Andreas Stenger, Julian Hof. 2015. An Adaptive VNS Algorithm for Vehicle Routing Problems with Intermediate Stops. *OR Spectrum* **37** 353–387.

Tarantilis, Christos D., Emmanouil E. Zachariadis, Chris T. Kiranoudis. 2008. A hybrid guided local search for the vehicle-routing problem with intermediate replenishment facilities. *INFORMS Journal on Computing* **20**(1) 154–168.

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. 2014. A unified solution framework for multi-attribute vehicle routing problems. *European Journal of Operational Research* **234**(3) 658–673.