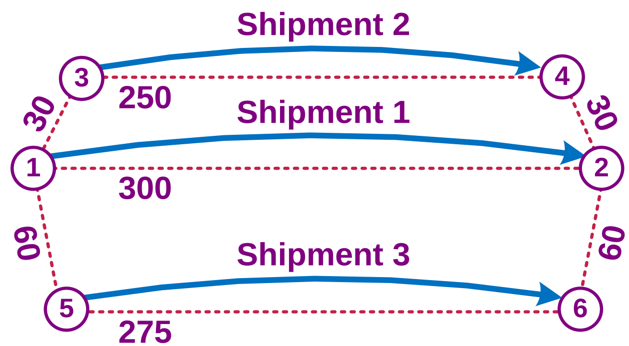
Multi-Stop Routing

- Each shipment might have a different origin and/or destination $\Rightarrow$ node/location sequence not adequate



$L = (y_1, \ldots, y_n) = (1, 2, 3)$ n-element shipment sequence

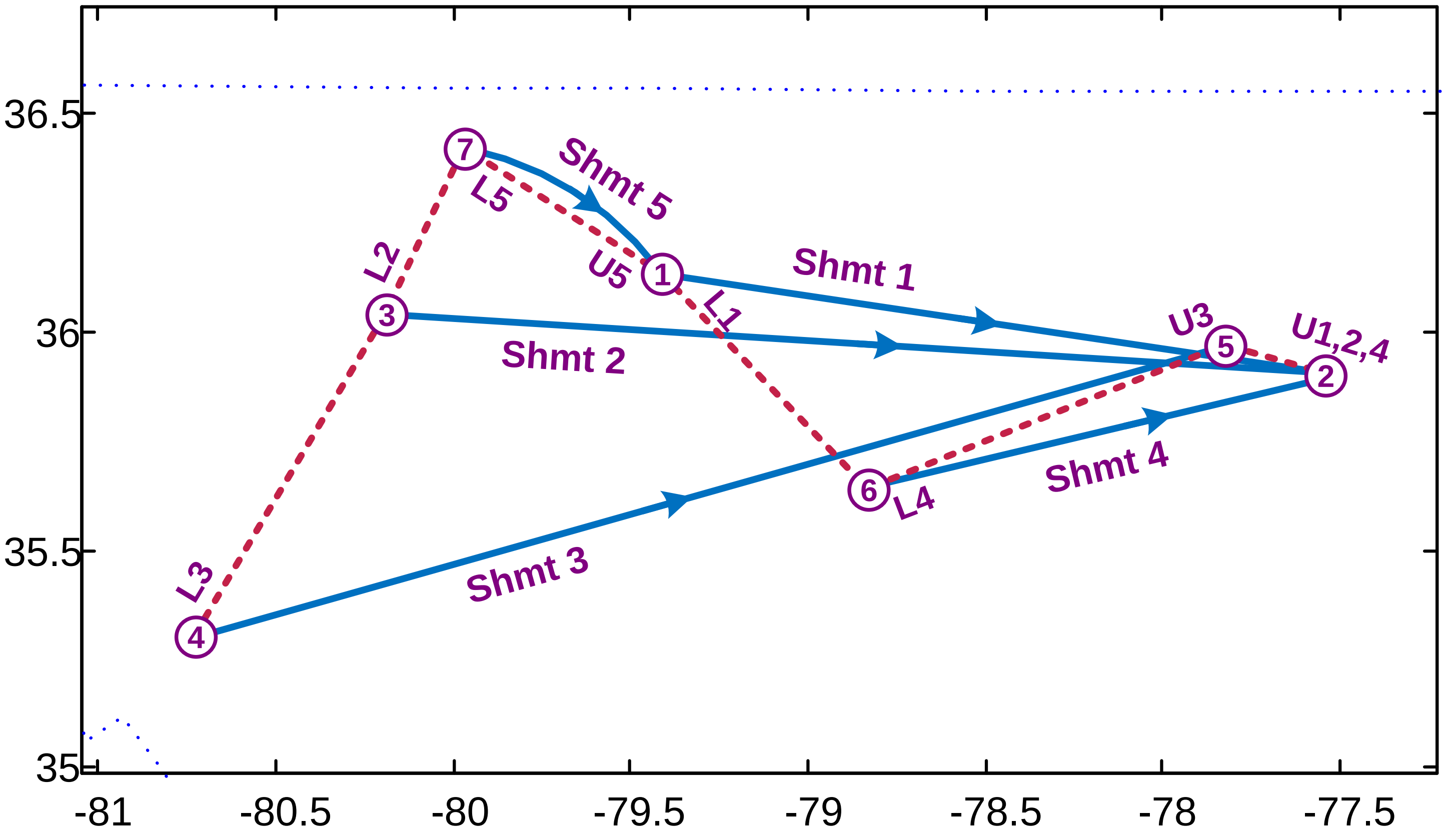
$R = (z_1, \ldots, z_{2n}) = (3, 1, 2, 2, 1, 3)$ $2n$-element route sequence

$X = (x_1, \ldots, x_{2n}) = (5, 1, 3, 4, 2, 6)$ $2n$-element location (node) sequence

c_{ij} = cost between locations i and j

$$c(R) = \sum_{i=1}^{2n-1} c_{x_i, x_{i+1}} = 60 + 30 + 250 + 30 + 60 = 430, \quad \text{total cost of route } R$$

5-Shipment Example



Route sequence: $R = (3, 2, 5, 5, 1, 4, 3, 1, 2, 4)$

Location sequence: $X = (4, 3, 7, 1, 1, 6, 5, 2, 2, 2)$

Route Sequencing Procedures

- Online procedure: add a shipment to an existing route as it becomes available
  - Insert and Improve: for each shipment,
    insert where it has the least increase in cost for route and then improve (`mincostinsert` → `twoopt`)
- Offline procedure: consider all shipments to decide order in which each added to route
  - Savings and Improve: using all shipments,
    determine insert ordering based on “savings,” then improve final route (`savings` → `twoopt`)

Min Cost Insert

		1			1			
1		•			•	2	2	×
2	2	•			•	2		c_2
3		•	2	2	•			c_3^*
4		•		2	•	2		c_4
5	2	•	2		•			c_5

		1			2		2		1	
1	3	•	3		•		•		•	
2	3	•			•	3	•		•	
3	3	•			•		•	3	•	
4	3	•			•		•		•	3
5		•	3	3	•		•		•	
6		•		3	•	3	•			
7		•		3	•		•	3		
⋮		⋮			⋮		⋮		⋮	

Insert and Improve Online Procedure

- To route each shipment added to load:
  - Minimum Cost Insertion
  - Two-opt improvement
- Different shipment sequences L can result in different routes
  - Order shipment joins load important

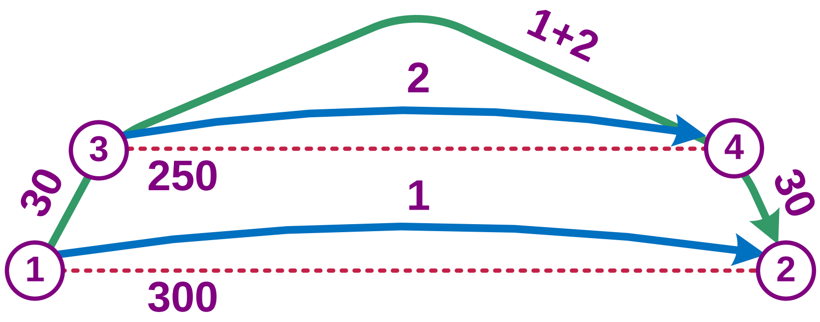
procedure *insertImprove*$(y_i \in L)$
$R = (y_1, y_1)$
for $i = 2, \ldots, |L|$
$\quad R = \text{minCostInsert}(y_i, R)$
$\quad R = \text{twoOpt}(R)$
endfor
return R

subprocedure *minCostInsert*$(y, z_i \in R)$
$c_R = c(R)$
for $i = 1, \ldots, |R|+1$, **for** $j = 1, \ldots, |R|+1$
$\quad R' = (z_1, \ldots, z_{i-1}, y, z_i, \ldots, z_{j-1}, y, z_j, \ldots, z_{|R|})$
$\quad$ **if** $c(R') < c_R$, $c_R = c(R')$, $R = R'$, **endif**
endfor, **endfor**
return R

subprocedure *twoOpt*$(z_i \in R)$
$c_R = c(R)$
repeat
$\quad$ $done = \text{true}$, $i = 1$, $j = 2$
$\quad$ **while** $done$ and $i < |R|$
$\quad\quad$ **while** $done$ and $j < |R|+1$
$\quad\quad\quad$ $R' = (z_1, \ldots, z_{i-1}, \text{reverseSequence}(z_i, \ldots, z_j), z_{j+1}, \ldots, z_{|R|})$
$\quad\quad\quad$ **if** $c(R') < c_R$
$\quad\quad\quad\quad$ $c_R = c(R')$, $R = R'$, $done = \text{false}$
$\quad\quad\quad$ **endif**
$\quad\quad\quad$ $j = j + 1$
$\quad\quad$ **endwhile**
$\quad\quad$ $i = i + 1$, $j = i + 1$
$\quad$ **endwhile**
until $done = \text{true}$
return R

First improvement (cf. steepest descent)

Pairwise Savings



$$s_{ij} = \text{pairwise savings between shipments } i \text{ and } j$$

$$= c_i + c_j - c_{ij} > 0$$

$$s_{1,2} = 300 + 250 - 310$$

$$= 240$$

Clark-Wright (Offline) Savings Procedure

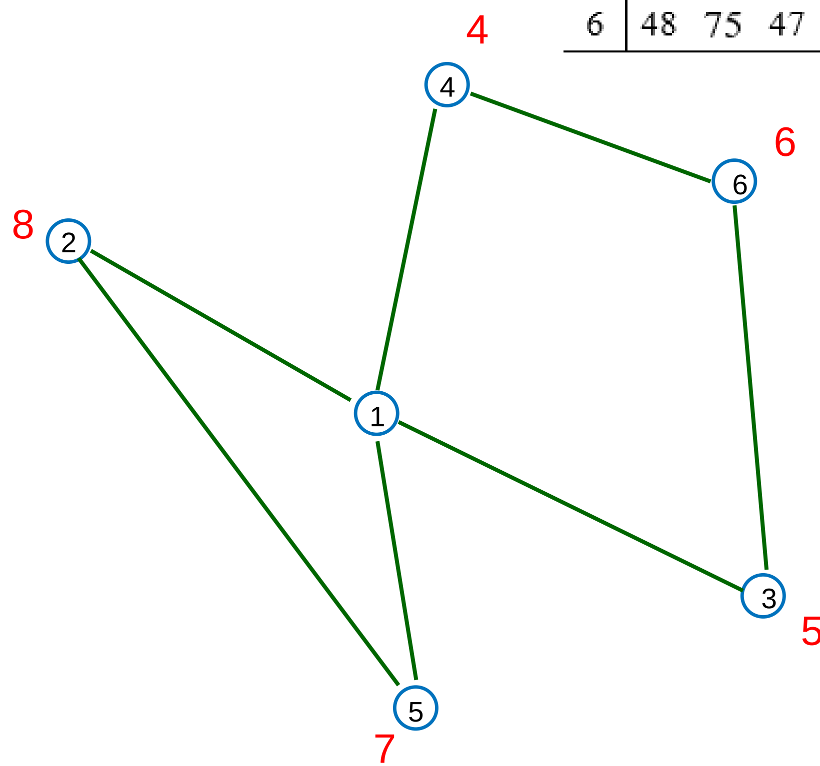
- First (1964), and still best, offline routing procedure if only have vehicle capacity constraints (`vrpsavings`)
- Pairs of shipments ordered in terms of their decreasing (positive) pairwise savings
- Given savings pair i-j, without exceeding capacity constraint, either:
  1. Create new route if i and j not in any existing route
  2. Add i to route only if j at beginning or end of route
  3. Combine routes only if i and j are endpoints of each route

Ex 17: Clark-Wright Savings Procedure

- Node 1 is depot, nodes 2-6 customers
- Customer demands 8, 3, 4, 7, 6, resp.
- Vehicle capacity is 15
- Symmetric costs

	1	2	3	4	5	6
1	0	40	48	38	33	48
2	40	0	87	46	65	75
3	48	87	0	67	41	47
4	38	46	67	0	70	34
5	33	65	41	70	0	69
6	48	75	47	34	69	0

i	j	s_{ij}
2	3	$40+48-87=1$
~~2~~	~~4~~	$40+38-46=32$
$\boxed{2}$	$\boxed{5}$	8
~~2~~	~~6~~	13
~~3~~	~~4~~	19
~~3~~	~~5~~	40
$\boxed{3}$	$\boxed{6}$	49
4	5	1
$\boxed{4}$	$\boxed{6}$	52
~~5~~	~~6~~	12



Multi-Stop (Offline) Savings Procedure

- Pairs of shipments ordered by their decreasing pairwise savings to create **i** and **j** (`pairwisesavings`)
- Creates set of multi-shipment routes (`savings`)

$$R = \{R_1, \ldots, R_m\}$$

  - Shipments with no pairwise savings are not included (use `sh2rte` to add)
- Clark-Wright only adds to beginning or end of a route
  - Multi-stop savings considers adding anywhere in route via min cost insert
  - More computation required, but can include sequence-dependent constraints like time windows (capacity not sequence dependent)

procedure $savings(\mathbf{i}, \mathbf{j})$

$R \leftarrow \{\,\}$

for $k = \{1, \ldots, |\mathbf{i}|\}$

 if $i_k \notin R$ and $j_k \notin R$ **1. Form new route**

  $R \leftarrow R \cup minCostInsert(i_k, j_k)$

 elseif $(i_k \notin R$ and $j_k \in R)$ or $(i_k \in R$ and $j_k \notin R)$

  if $j_k \notin R$ **2. Add shipment to route**

   $temp \leftarrow i_k, \quad i_k \leftarrow j_k, \quad j_k \leftarrow temp$

  endif

  $h \leftarrow \arg\{R_l : j_k \in R_l\}$

  $R' \leftarrow minCostInsert(i_k, R_h)$

  if $c(R') < c(i_k) + c(R_h)$

   $R_h \leftarrow R'$

  endif

 else   **3. Combine two routes**

  $g \leftarrow \arg\{R_l : i_k \in R_l\}, \quad h \leftarrow \arg\{R_l : j_k \in R_l\}$

  if $g \neq h$

   $R' \leftarrow minCostInsert(R_g, R_h)$

   if $c(R') < c(R_g) + c(R_h)$

    $R_g \leftarrow \{\,\}, \quad R_h \leftarrow R'$

   endif

  endif

 endif

endfor

return R