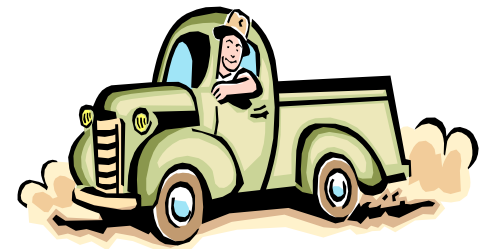


Vehicle Routing and Scheduling



Martin Savelsbergh
The Logistics Institute
Georgia Institute of Technology

Vehicle Routing and Scheduling

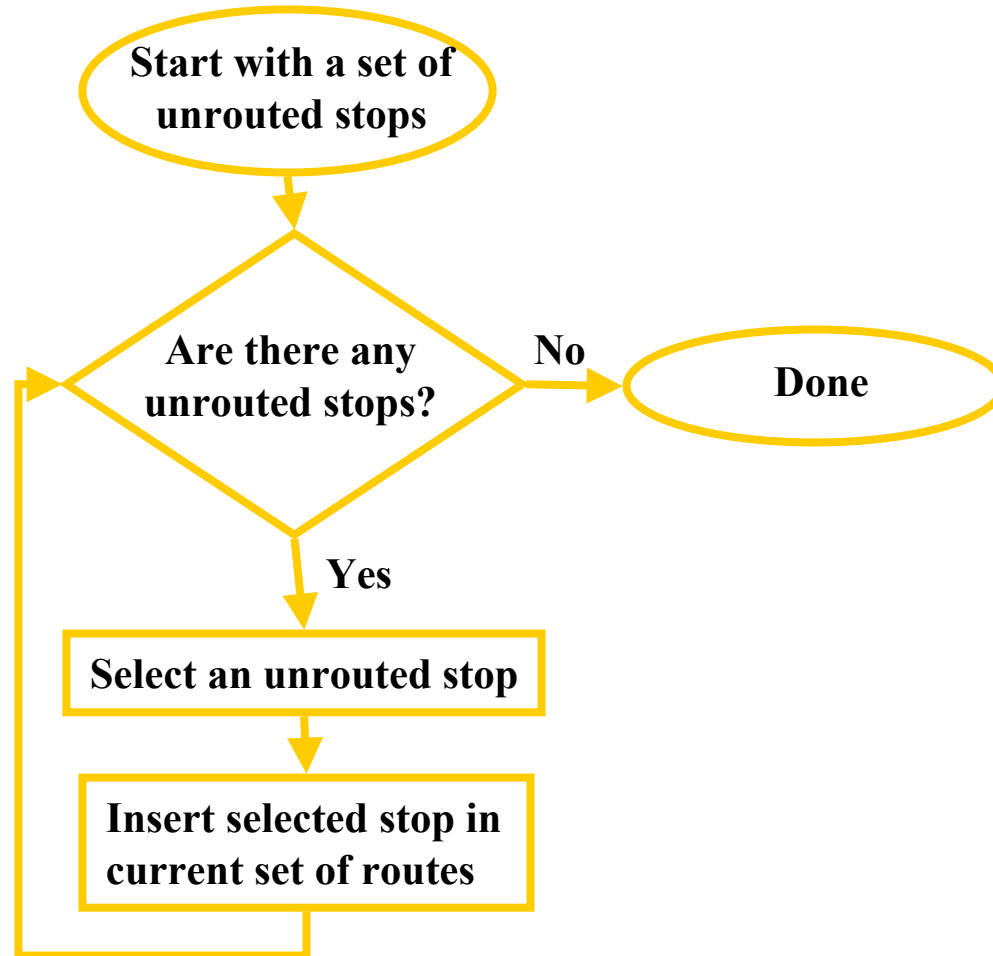


Part II: Algorithmic Enhancements

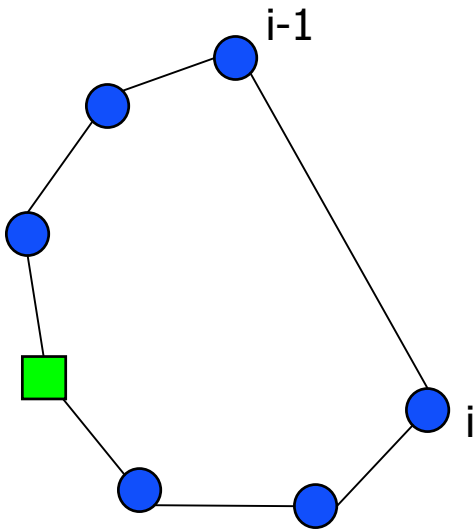


Handling Practical Complexities In Construction Heuristics

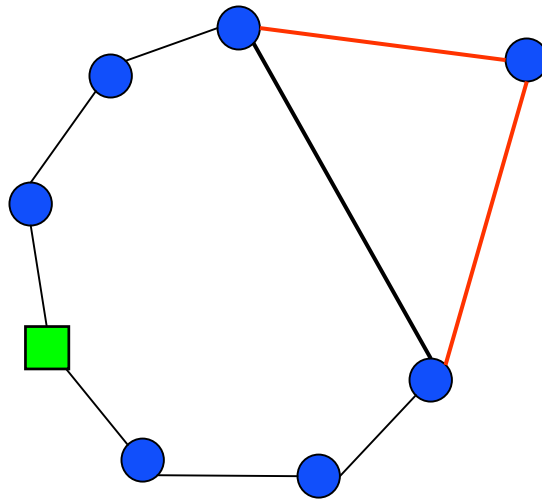
Insertion Heuristics



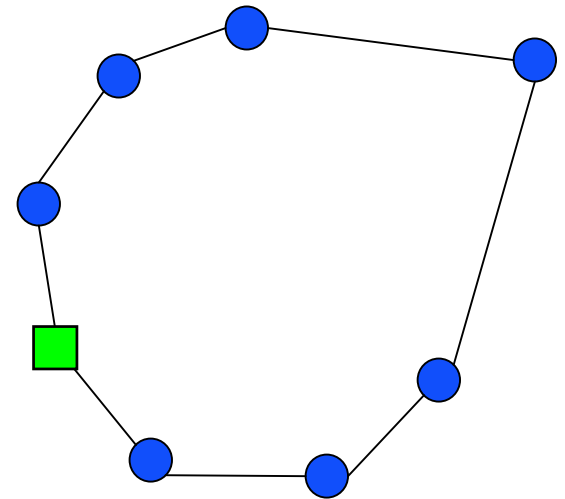
Insertion



Initial



Intermediate



Final

Insertion Heuristics



- Efficient (fast)
- Effective (good quality solutions)
- Easy to implement
- Easy to extend
 - time windows
 - route duration
 - variable delivery quantities
 - ...

Insertion Heuristic

$N = \{\text{unassigned customers}\}$

$R = \{\text{set of routes}\}$

WHILE $N \neq \emptyset$ DO

$p^* = -\infty$

 FOR $j \in N$ DO

 FOR $r \in R$ AND $(i-1, i) \in r$ DO

 IF **Feasible**(i, j) AND **Profit**(i, j) $> p^*$ THEN

$r^* = r, i^* = i, j^* = j, p^* = \text{Profit}(i, j)$

 Insert(i^*, j^*)

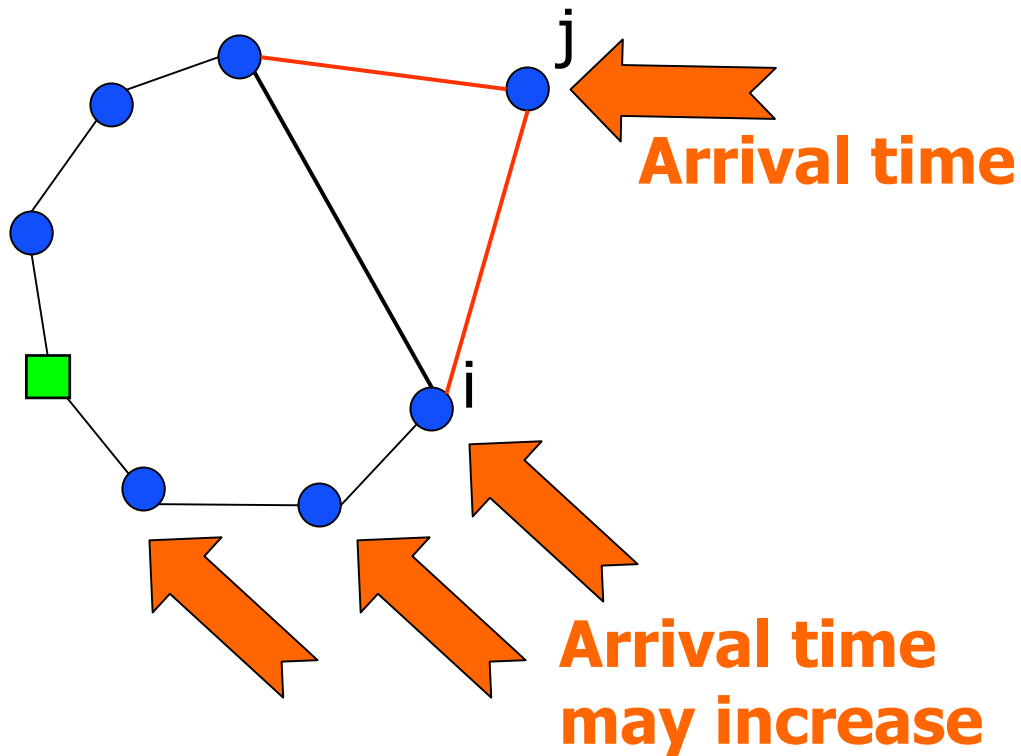
Update(r^*)

$N = N \setminus \{j^*\}$

Insertion Heuristic

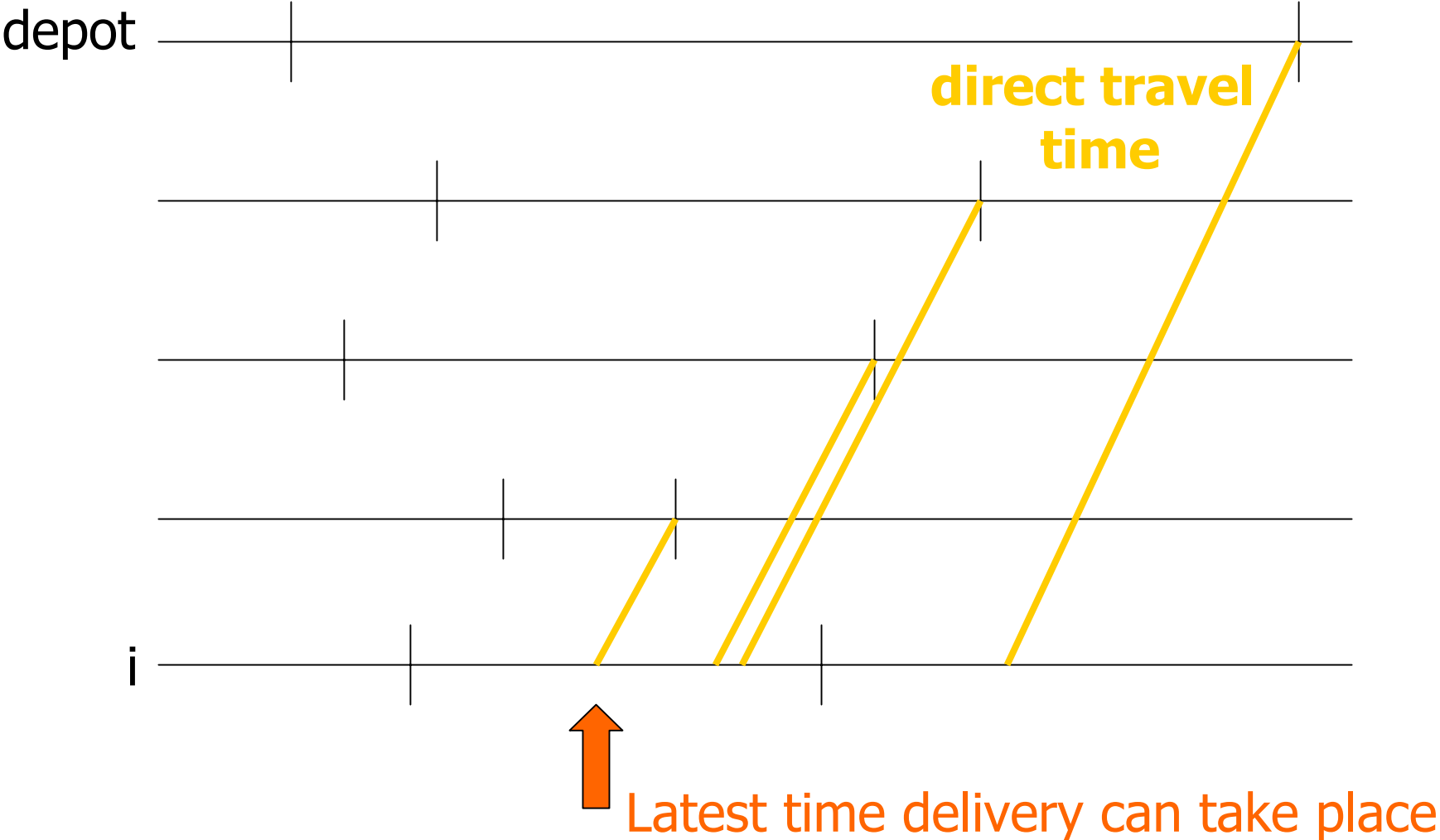
- Complexity is $O(n^3)$ if:
 - **Feasible()** is $O(1)$
 - **Profit()** is $O(1)$
 - **Update()** is at most $O(n^2)$
- Vehicle Routing Problem
 - Feasible: $d_j < Q - q_r$
 - Update: $q_r = q_r + d_j$

Time Windows



How to quickly
check feasibility?

Latest delivery time



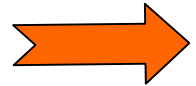
Time windows



- Notation:
 - $[E_k, L_k]$ time window on start of delivery at k
- Auxiliary information:
 - e_k earliest time delivery can take place at k
 - l_k latest time delivery can take place at k

Time windows (cont.)

- Feasible()
 - $d_j < Q - q_r$
 - $e_j = \max\{E_j, e_{i-1} + t_{i-1,j}\}$ & $l_j = \min\{L_j, l_i - t_{j,i}\}$
 - $e_j < l_j$



- Update()
 - for $k=i-1$ to 0 : $l_k = \min\{l_k, l_{k+1} - t_{k,k+1}\}$
 - for $k=i$ to $n+1$: $e_k = \max\{e_k, e_{k-1} + t_{k-1,k}\}$

Route duration

- Notation:

- T planning horizon
- L route duration limit

- Auxiliary information:

- e_0 earliest start time route

$$e_0 = \max\{0, e_{n+1} - L\}$$

- l_{n+1} latest completion time route

$$l_{n+1} = \min\{l_0 + L, T\}$$

Route duration (cont.)



- Auxiliary information
 - f_i total travel time from i until the end
 - b_i total travel time from the beginning until i

Route duration (cont.)

■ Feasible()

- $e_{n+1} = \max\{e_{n+1}, e_j + t_{j,i} + f_i\}$, $e_0 = \max\{e_0, e_{n+1} - L\}$
- $l_0 = \min\{l_0, l_j - t_{i-1,j} - b_{i-1}\}$, $l_{n+1} = \min\{l_{n+1}, l_0 + L\}$
- $d_j < Q - q_r$
- $e_j < l_j$
- $e_0 < l_0$ & $l_{n+1} > e_{n+1}$

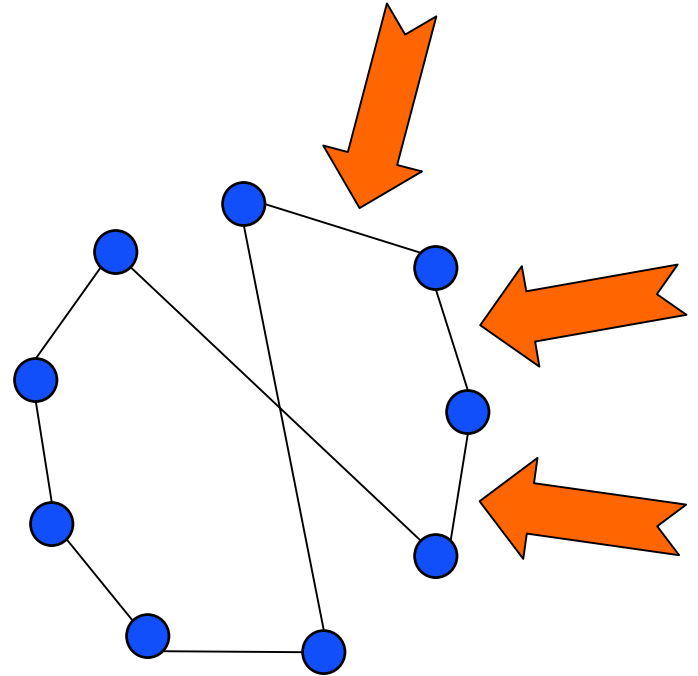
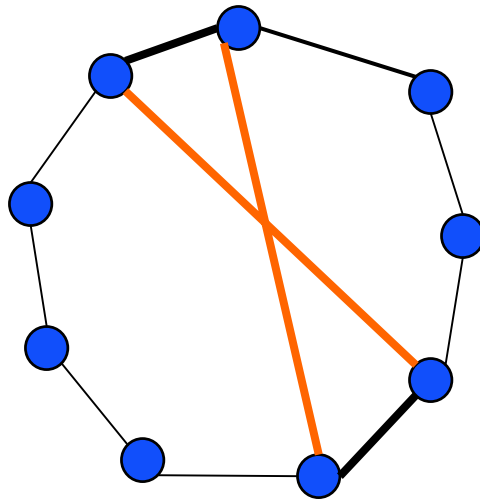
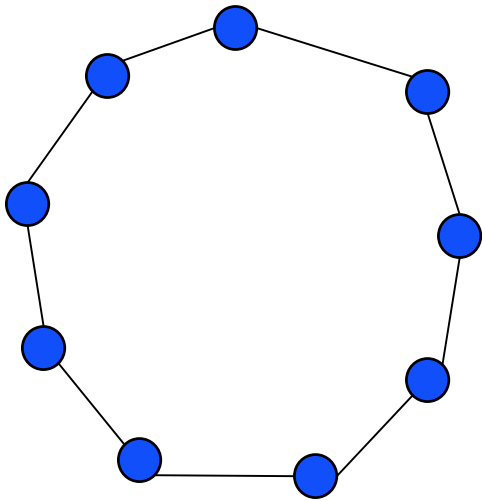
Route duration (cont.)

- Update()
 - for $k=i-1$ to 0 : $l_k = \min\{l_k, l_{k+1} - t_{k,k+1}\}$
 - for $k=i$ to $n+1$: $e_k = \max\{e_k, e_{k-1} + t_{k-1,k}\}$
 - if e_0 updated
 - for $k=1$ to $i-1$: $e_k = \max\{e_k, e_{k-1} + t_{k-1,k}\}$
 - If l_{n+1} updated
 - for $k=n+1$ to i : $l_k = \min\{l_k, l_{k+1} - t_{k,k+1}\}$



Handling Practical Complexities In Improvement Heuristics

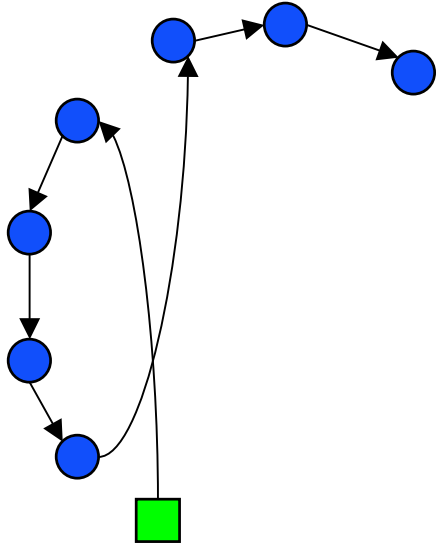
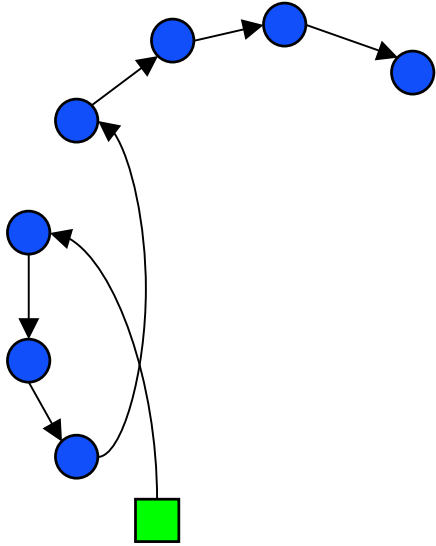
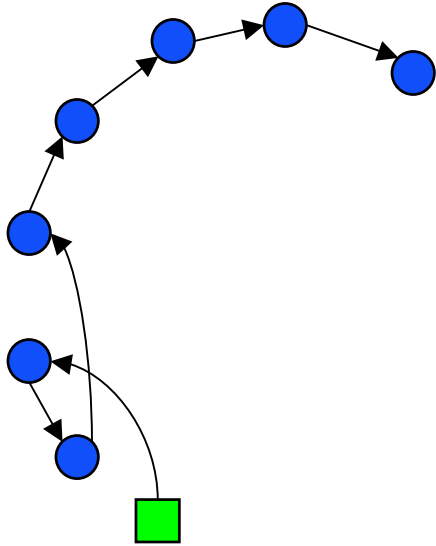
2-change



**Orientation of traversal
is reversed!**

Lexicographic search

2-changes



Global variables

- Total travel time $T(u_1, \dots, u_k)$
 $\text{sum}(i=1, \dots, k-1) t(u_i, u_{i+1})$
- Earliest delivery time $E(u_1, \dots, u_k)$ at u_k
assuming u_1 is left at the opening of its
time window
 $\text{max}(i=1, \dots, k) \{E_i + T(u_i, \dots, u_k)\}$
- Latest delivery time $L(u_1, \dots, u_k)$ at u_1 such
that the path remains feasible
 $\text{min}(i=1, \dots, k) \{L_i - T(u_1, \dots, u_i)\}$

Path concatenation

- $T(u_1, \dots, u_k, v_1, \dots, v_l) =$
 $T(u_1, \dots, u_k) + t(u_k, v_1) + T(v_1, \dots, v_l)$
- $E(u_1, \dots, u_k, v_1, \dots, v_l) =$
 $\max\{E(u_1, \dots, u_k) + t(u_k, v_1) + T(v_1, \dots, v_l), E(v_1, \dots, v_l)\}$
- $L(u_1, \dots, u_k, v_1, \dots, v_l) =$
 $\min\{L(u_1, \dots, u_k), L(v_1, \dots, v_l) - T(v_1, \dots, v_l) - t(u_k, v_1)\}$

Path concatenation



Earliest delivery at v_1 : $E(u_1, \dots, u_k) + t(u_k, v_1)$

Earliest delivery at v_l : $E(u_1, \dots, u_k) + t(u_k, v_1) + T(v_1, \dots, v_l)$

Path concatenation



Latest delivery at u_k : $L(v_1, \dots, v_l) - t(u_k, v_1)$

Latest delivery at u_1 : $L(v_1, \dots, v_l) - t(u_k, v_1) - T(u_1, \dots, u_k)$

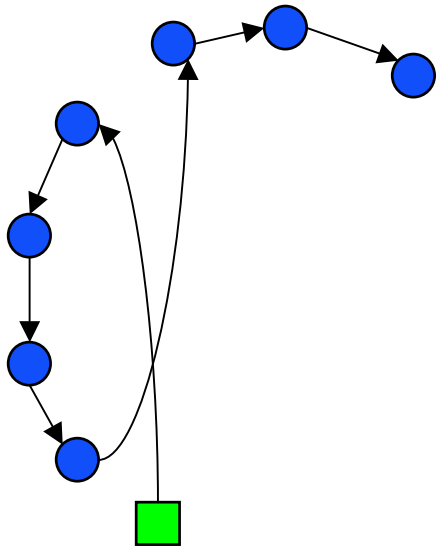
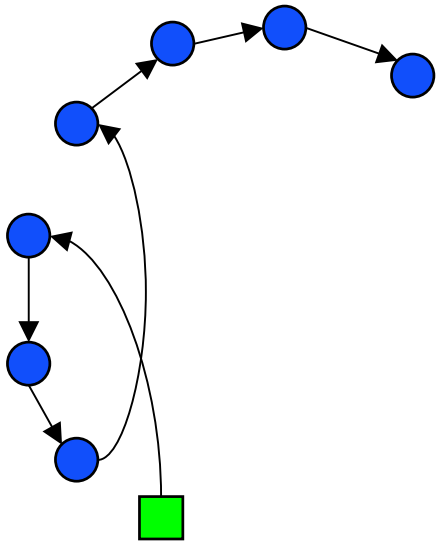
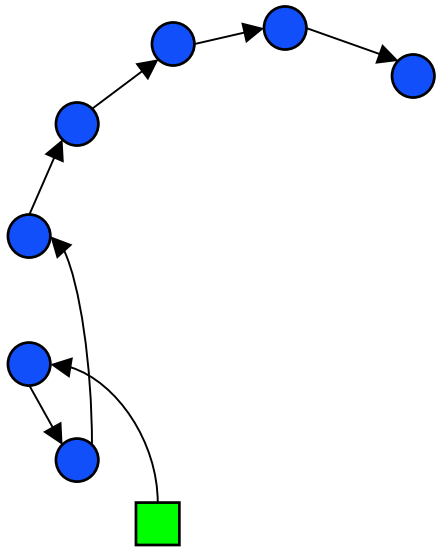
Observations



- The set of global variables makes it possible to test feasibility of an exchange in constant time
- The lexicographic search strategy makes it possible to maintain the correct values for the set of global variables in constant time

Lexicographic search

2-changes





The Dial-a-Ride Problem

Dial-a-Ride Problem



- Dispatch vehicles to pickup person/package at one location (origin) and deliver the person/package at another location (destination)
- Service related constraints
 - pickup window / delivery window
 - maximum wait time
 - maximum ride time

Service related constraints



- Pickup and delivery window
- Maximum wait time
 - Limit waiting time at a stop before departing
- Maximum ride time
 - Limit time between pickup and delivery

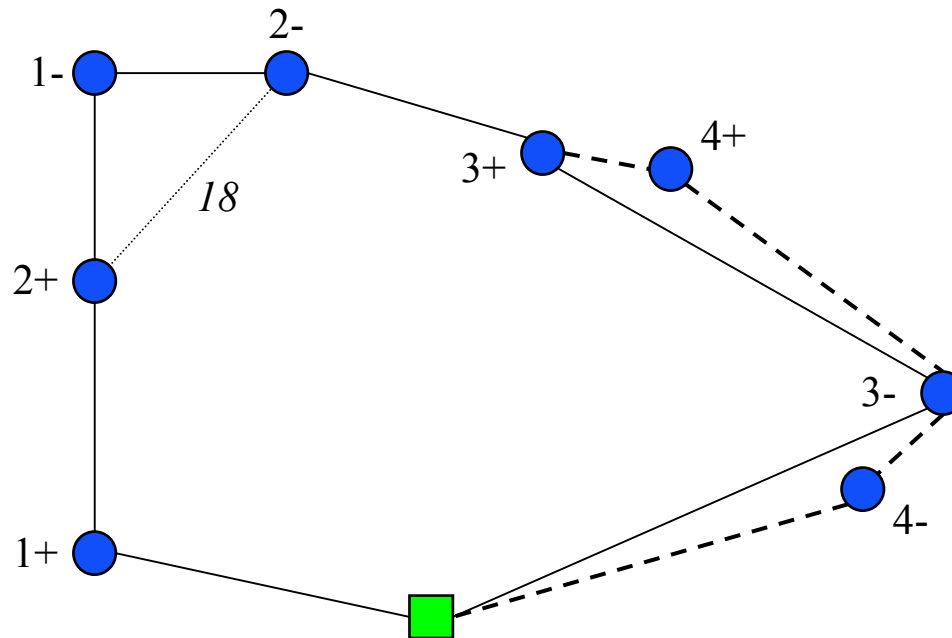
Feasibility testing



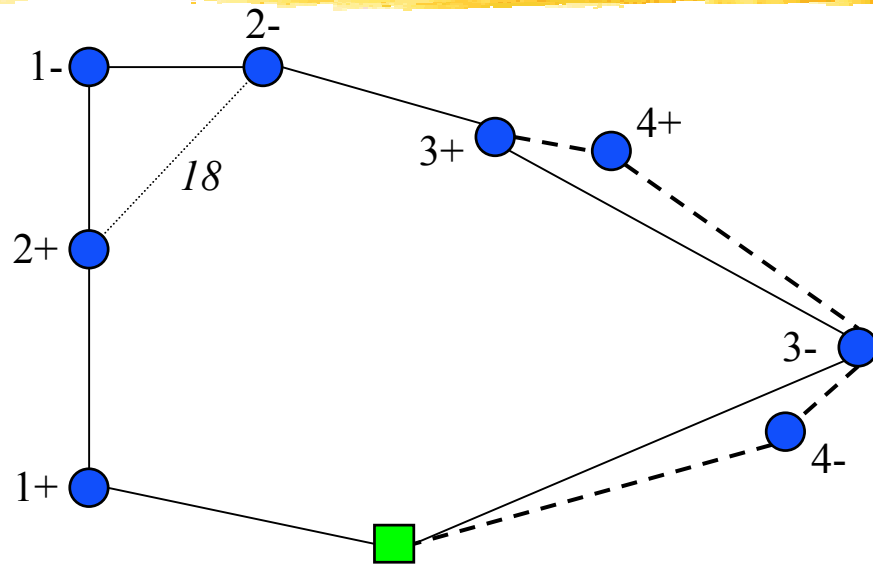
- Given a sequence of pickups and deliveries does there exist a feasible schedule satisfying pickup and delivery windows, maximum wait time, and maximum ride time constraints?

Example

- Waiting time limit: 10
- Ride time limit: $1.5 * \text{ride time}$

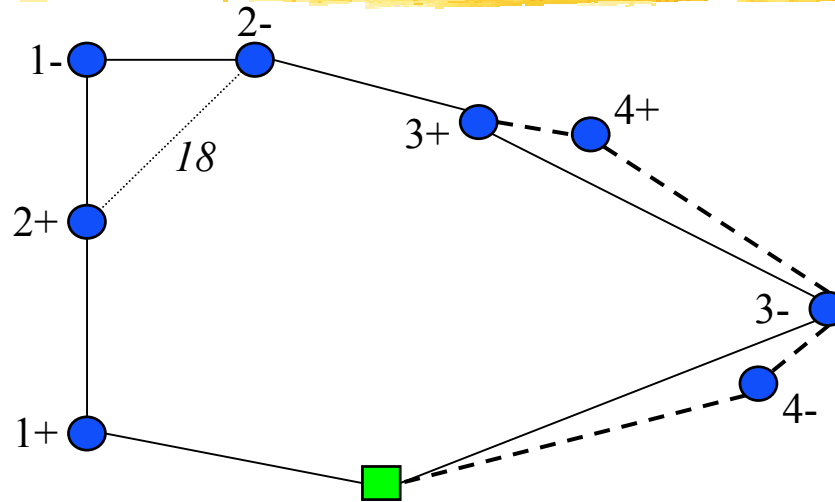


Example



Stop	Early	Late	Arrival	Departure	Waiting	Travel
1+	10.15	10.30	10.15	10.15	0	20
2+	10.45	11.00	10.35	10.45	10	15
1-	11.00	11.15	11.00	11.00	0	10
2-	11.20	11.40	11.10	11.20	10	15
3+	11.40	12.00	11.35	11.40	5	50
3-	12.30	13.00	12.30	12.30	0	

Example



Stop	Early	Late	Arrival	Departure	Waiting	Travel
1+	10.15	10.30	10.15	10.15	0	20
2+	10.45	11.00	10.35	10.45	10	15
1-	11.00	11.15	11.00	11.00	0	10
2-	11.20	11.40	11.10	11.20	10	15
3+	11.40	12.00	11.35	11.40	5	10
4+	12.10	12.30	11.50	12.10	20	40
3-	12.30	13.00	12.50	12.50	0	20
4-	13.00	13.45	13.10	13.10	0	

Example

Stop	Early	Late	Arrival	Departure	Waiting	Travel
1+	10.15	10.30	10.15	10.15	0	20
2+	10.45	11.00	10.35	10.45	10	15
1-	11.00	11.15	11.00	11.00	0	10
2-	11.20	11.40	11.10	11.20	10	15
3+	11.40	12.00	11.35	11.40	5	10
4+	12.10	12.30	11.50	12.10	20	40
3-	12.30	13.00	12.50	12.50	0	20
4-	13.00	13.45	13.10	13.10	0	

Stop	Early	Late	Arrival	Departure	Waiting	Travel
1+	10.15	10.30	10.20	10.20	0	20
2+	10.45	11.00	10.40	10.50	10	15
1-	11.00	11.15	11.05	11.05	0	10
2-	11.20	11.40	11.15	11.25	10	15
3+	11.40	12.00	11.40	11.50	10	10
4+	12.10	12.30	12.00	12.10	10	40
3-	12.30	13.00	12.50	12.50	0	20
4-	13.00	13.45	13.10	13.10	0	

Feasibility testing



- Can be done in linear time ...
- Invariant property:
 - No feasible schedule can have an arrival or departure time earlier than the computed arrival and departure times

Feasibility checking



- Notation:
 - $[e_j, l_j]$ time window
 - ω waiting time limit
 - $\alpha t_{i+,i-}$ ride time limit
 - A_j arrival time
 - D_j departure time
 - L_j latest feasible departure time

Pass 1 - Forward



- Account for pickup and delivery windows and maximum waiting time constraints
- Normal updates
 - $A_j = D_{j-1} + t_{j-1,j}$
 - $D_j = \max\{e_j, A_j\}$
 - $L_j = \min\{l_j, L_{j-1} + t_{j-1,j} + \omega\}$

Pass 1 (cont.)



- Infeasibility
 - $A_j > l_j$
 - $L_j < e_j$
- Special update when $A_j + \omega < e_j$
 - $A_j = e_j - \omega$
 - $D_j = e_j$

Pass 2 - Backward

- Update arrival and departure times and “check” ride time constraints
 - Waiting time from j until the end of route W
- Normal updates
 - $D_j = A_{j+1} - t_{j-1,j}$
 - $A_j = \max\{A_j, D_j - \omega\}$
 - $W = W + (D_j - A_j)$

Pass 2 (cont.)



- Infeasibility (pickup point $j = i+$)
 - $\Delta = (D_{i-} - D_{i+}) - \alpha t_{i+,i-}$
 - $D_{i+} + \Delta > L_{i+}$
 - $\Delta > W$
- Special update when $\Delta > W$
 - $D_j = D_j + \Delta$
 - $A_j = \max\{A_j, D_j - \omega\}$
 - $W = W - \Delta$

Pass 3 - Forward



- Finalize arrival and departure times and check ride time constraints
- Normal updates
 - $A_j = D_{j-1} + t_{j-1,j}$
 - $D_j = \max\{A_j, D_j\}$

Pass 3 (cont.)



- Infeasibility (drop-off point $j = i-$)
 - $\Delta = (D_{i-} - D_{i+}) - \alpha t_{i+,i-}$
 - $D_{i-} > I_{i-}$
 - $\Delta > 0$

Discussion




- People transportation
 - delivery window $[0, l_j]$
 - no pickup window
 - waiting time at pickup only
 - different ride time limits
 - consecutive stops at same location (waiting time per location rather than stop)
- Package transportation
 - no waiting time limits



Greedy Randomized Adaptive Search Procedure

Construction + Improvement



Greedily create
feasible set of routes



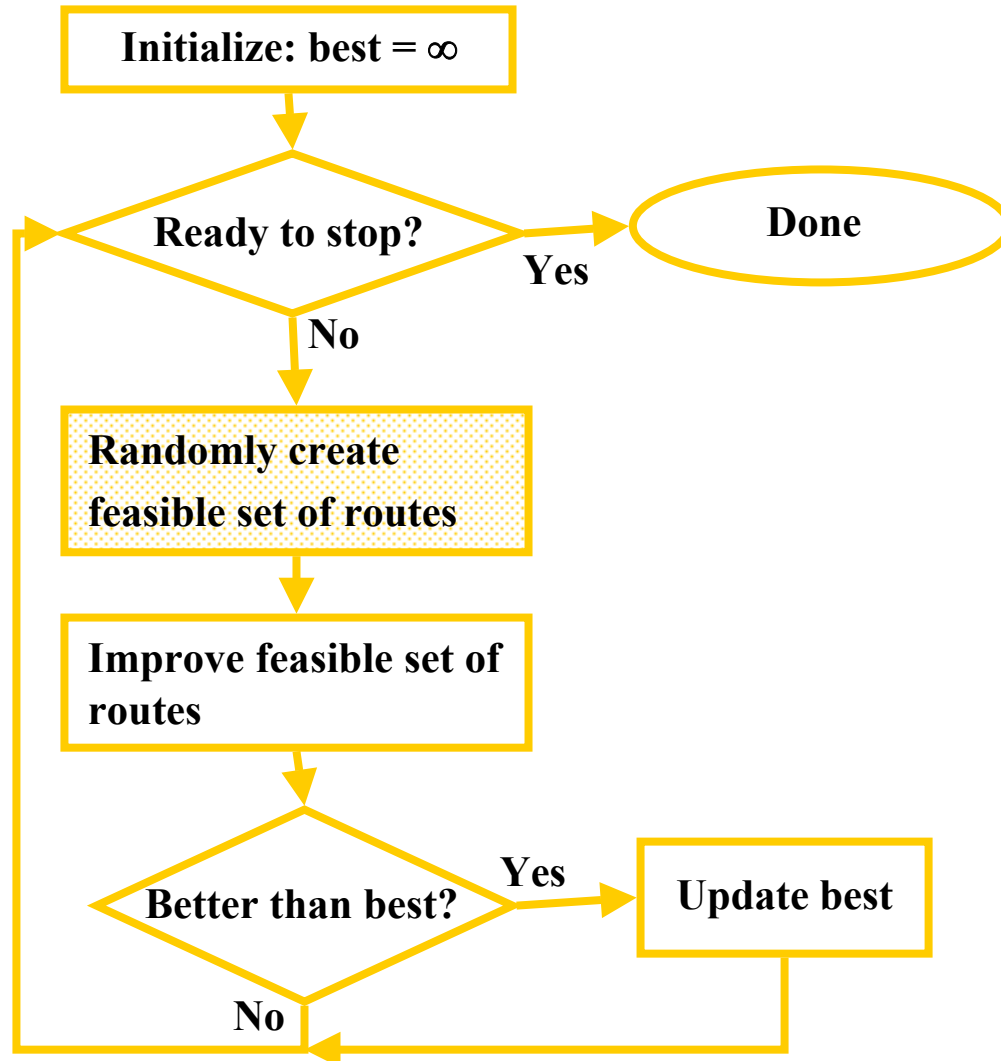
Improve feasible
Set of routes

Simple enhancement



- Multi-start neighborhood search:
 - independent neighborhood searches
 - random starting solutions

Multi-start neighborhood search

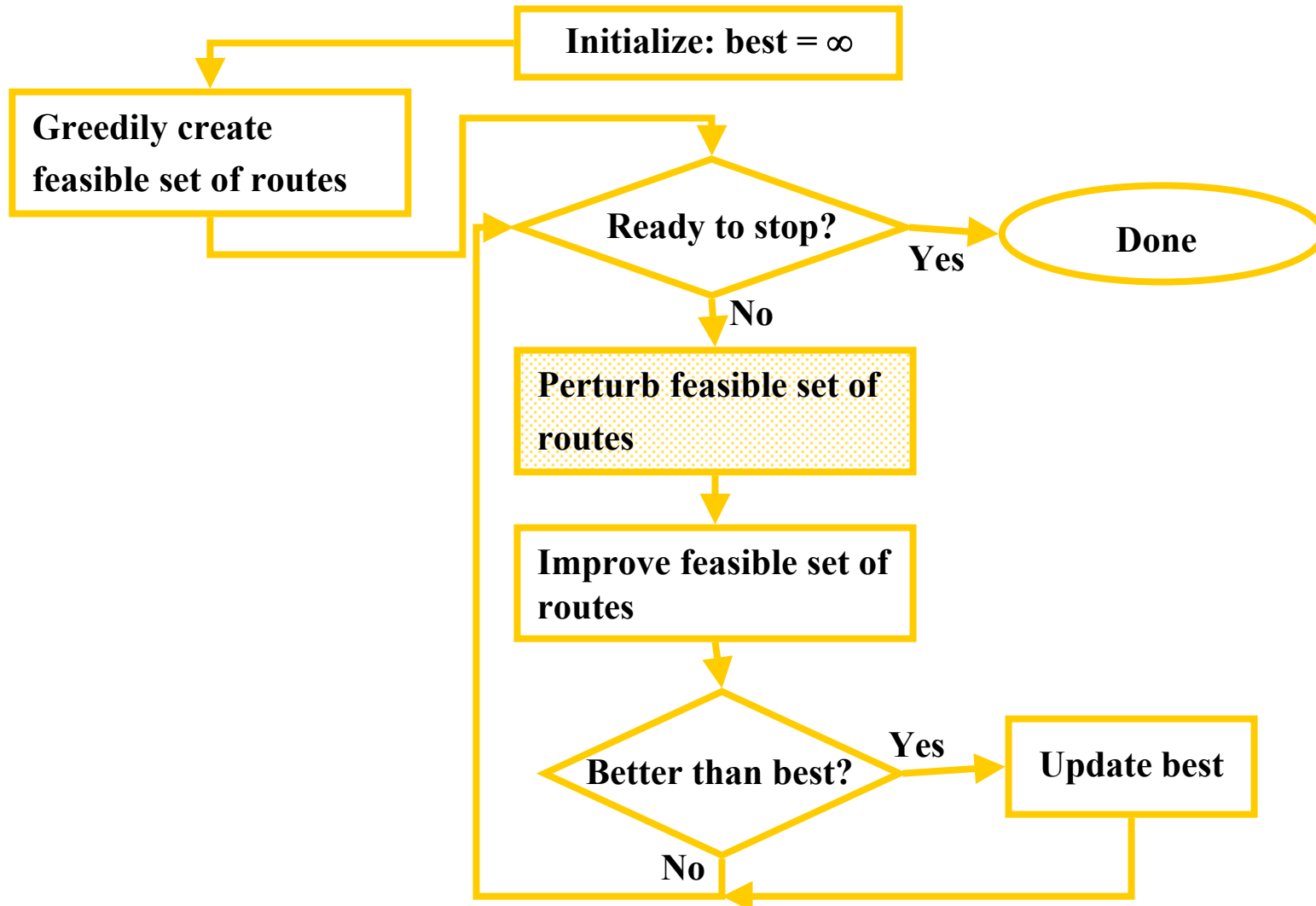


Simple enhancement



- Iterated neighborhood search:
 - independent neighborhood searches
 - starting solutions obtained from a previous local optimum by a suitable perturbation method

Iterated neighborhood search



Greedy algorithm



- Constructs a solution one element at a time:
 - Define candidate elements
 - Apply greedy function to each candidate element
 - Rank candidate elements according to greedy function value
 - Add best ranked element to solution

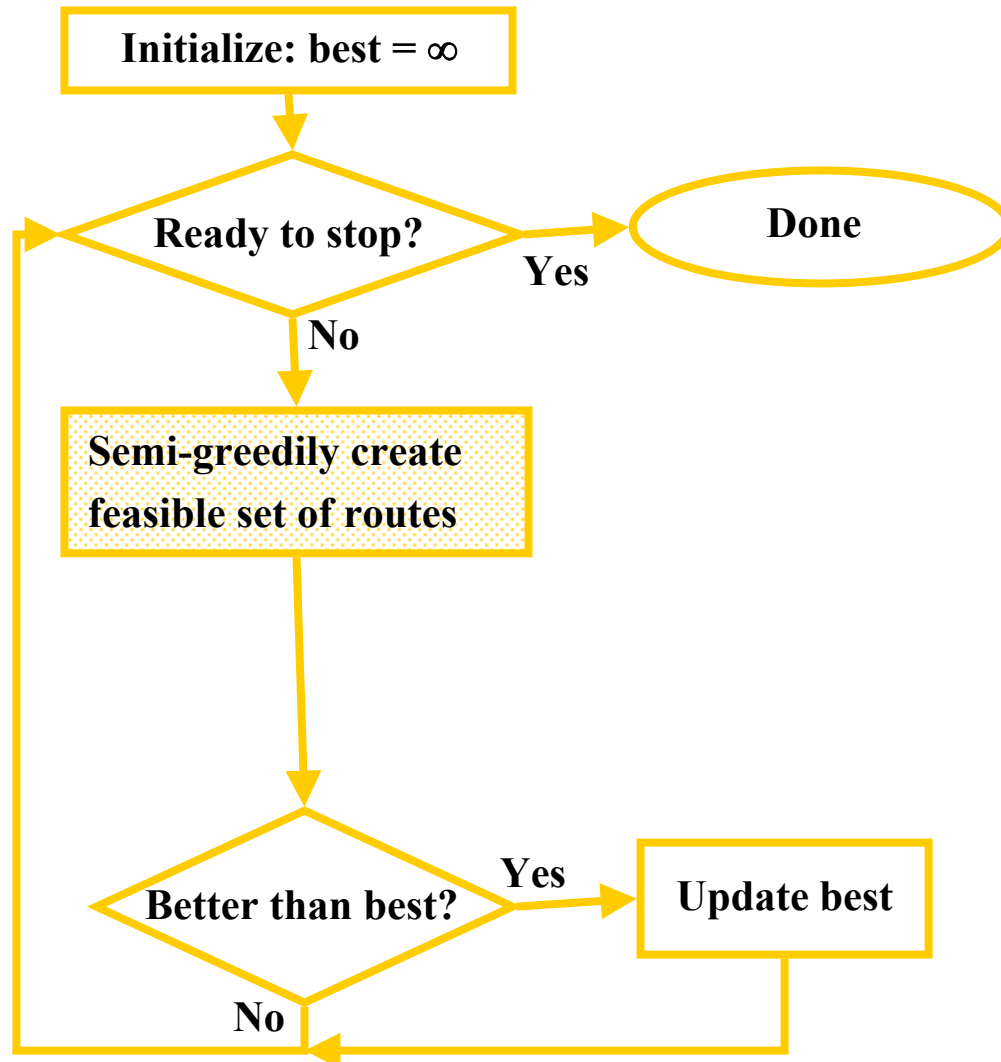
Semi-greedy algorithm

- Constructs a solution one element at a time:
 - Define candidate elements
 - Apply greedy function to each candidate element
 - Rank candidate elements according to greedy function value
 - Place well-ranked elements in a ***restricted candidate list*** (RCL)
 - Select an element from the RCL ***at random*** and add it to the solution

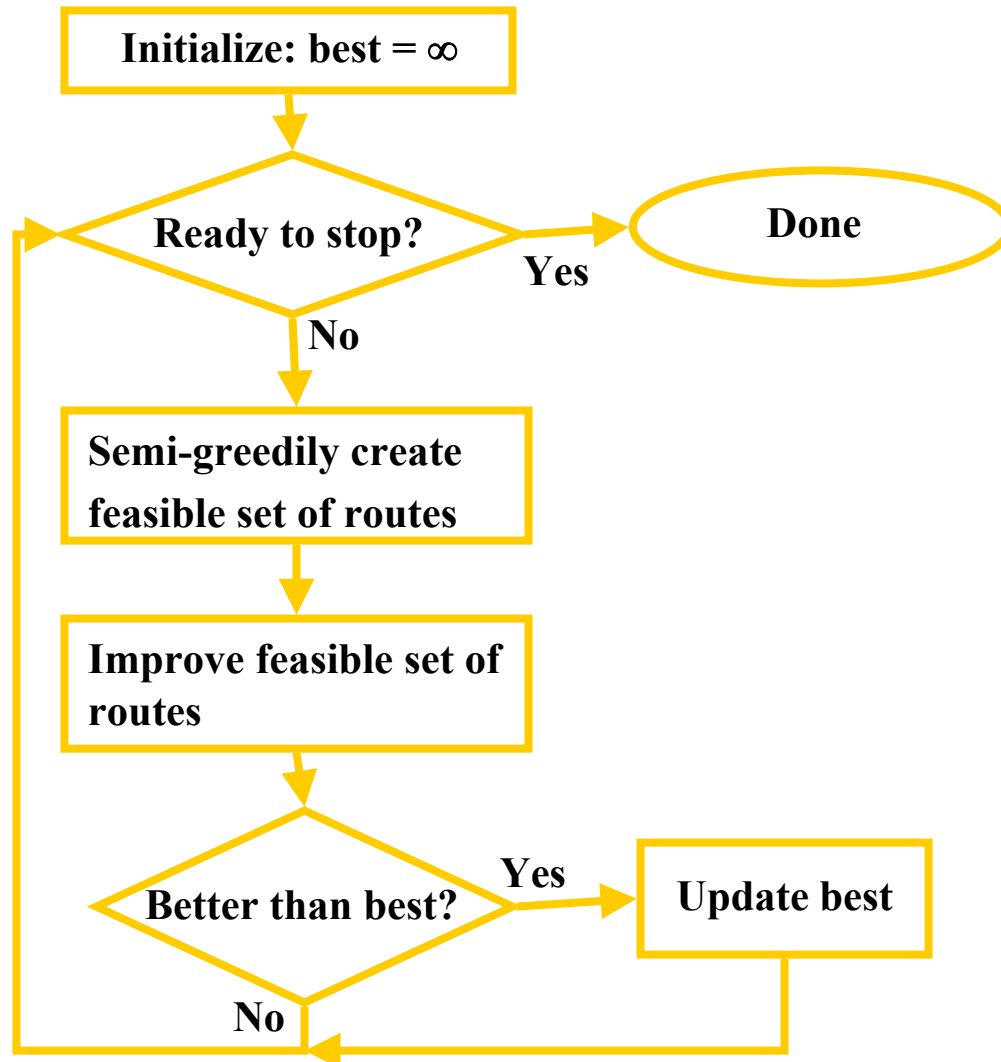
Restricted Candidate List

- Cardinality based:
 - Place k best candidates in RCL
- Value based I:
 - Place all candidates having greedy value better than $\alpha * \text{max value}$ in RCL (with $0 \leq \alpha \leq 1$)
- Value based II:
 - Place all candidates having greedy value better than $\text{min value} + \alpha * (\text{max value} - \text{min value})$ in RCL (with $0 \leq \alpha \leq 1$)

Semi-greedy



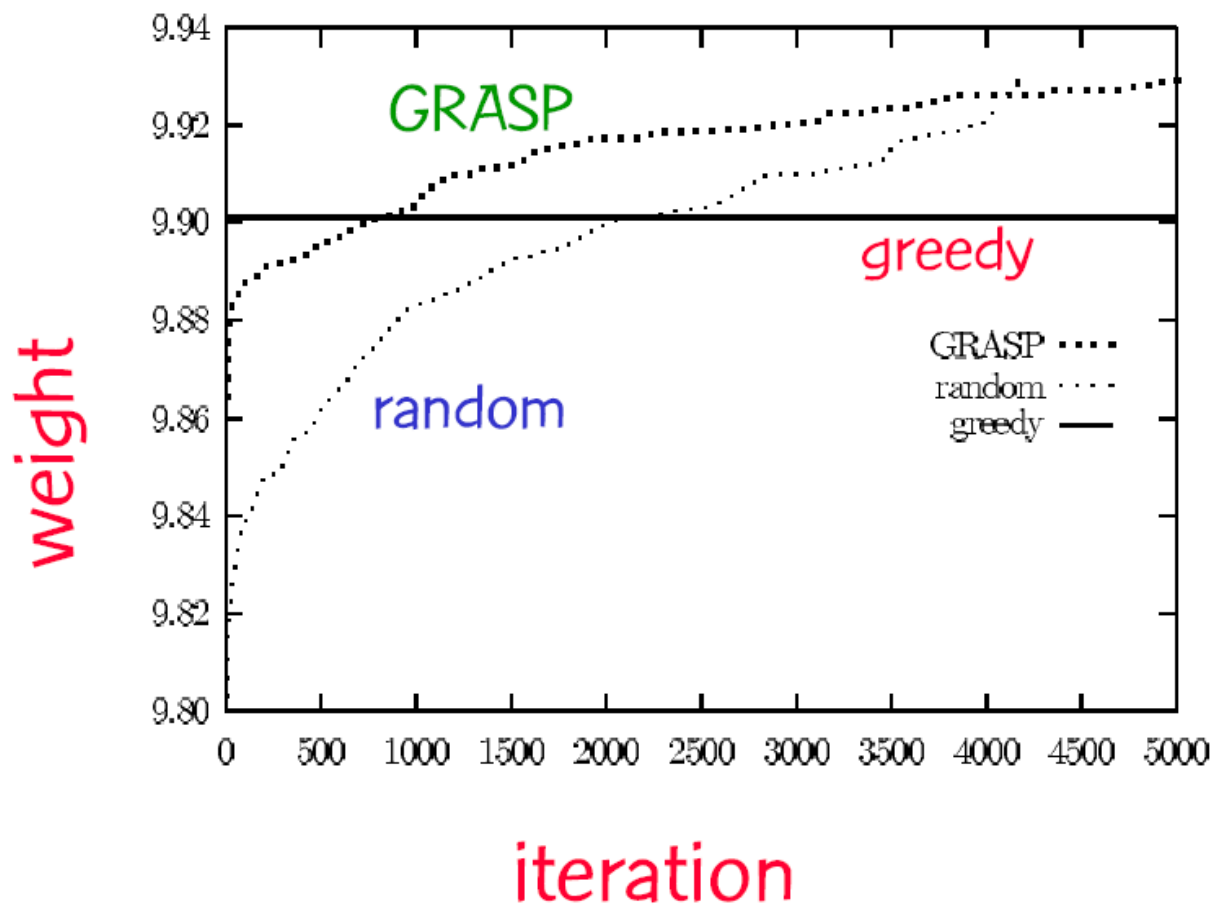
Greedy Randomized Adaptive Search Procedure



GRASP



- GRASP tries to capture good features of greedy & random constructions
- Iteratively
 - samples solution space using a greedy probabilistic bias to construct a feasible solution
 - applies local search to attempt to improve upon the constructed solution





Advanced Neighborhood Search

Neighborhood search - observations



- Weakness:
 - looks only one step ahead, and may get trapped in a bad local optimum
- Strength:
 - Fast and easy to implement

Neighborhood search - sophisticated enhancements

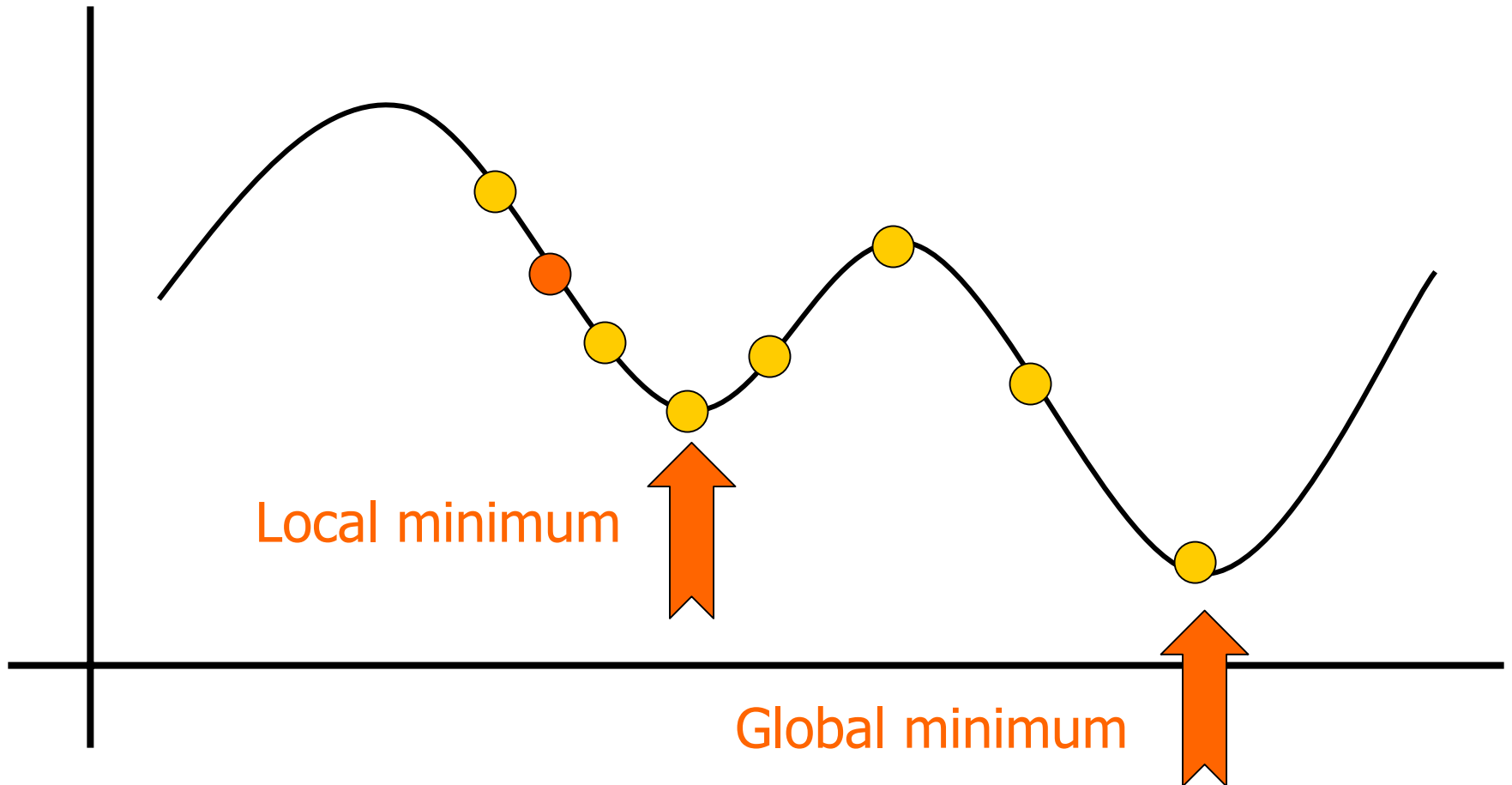


- Goal:
 - Increase quality of solution
 - Do not increase time to find solution too much
- Tabu Search
- Large Scale Neighborhood Search



Tabu Search

Tabu Search



Tabu Search



- Strategy to escape from a local optimum and continue the search
- Implementation
 - Best move is always performed
 - Avoid cycling using short-term memory
 - Attributes of recent solutions stored in tabu list
 - Moves involving attributes in tabu list are discarded (tabu)

Short-term Memory



- Tabu list
 - Tabu list size - maximum number of attributes stored in the list (FIFO)
 - Tabu list tenure - maximum number of iterations attribute remains in the list

Short-term Memory



- Recency-based
 - Last t moves
- Frequency-based
 - Number of times a specific move is performed
 - Penalize moves with higher frequency

Intensification and Diversification



- Intensification
 - Intensify the search in promising regions
- Diversification
 - Diversify the search across contrasting regions
- Examples
 - Varying the tabu list size
 - Adjusting the cost structure

Observations



- Tabu search can be highly effective
- Tabu search can be prohibitively time consuming
- *Remedy: speed up neighborhood search*

Granular Tabu Search



- Reduce the number of moves evaluated at each iteration
- Routing and scheduling problems:
 - *Long connections are unlikely to be part of an optimal solution*

Granular neighborhoods



- Restriction of ordinary neighborhoods
 - Consider only connections whose cost is below a threshold
 - Consider only moves involving promising connections
 - Threshold: $v * (UB / n)$
 - v sparsification parameter
 - UB/n average cost of connection in solution

Granular neighborhoods



- Intensification/diversification tool
 - small v -> intensification
 - large v -> diversification

Vehicle routing problem



- Set of connections:
 - connections of the current and best solution
 - connections involving the depot
 - connections with costs less than threshold
- Connections used as move generators

Vehicle routing problem



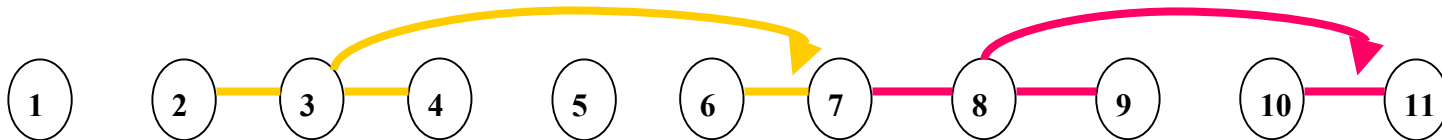
- Savings heuristic
- 1-relocate, 2-relocate, swap, 2-change
- Tabu tenure: random in [5,10]
- Granularity based intensification/diversification
 - intensification: v in [1,2]
 - diversification: no improvement in $15*n$ iterations, then $v = 5$ for n iterations



Large-Scale Neighborhood Search

Compounded 1-relocate

- Given the TSP tour $T = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)$



- The new TSP tour $T' = (1, 2, 4, 5, 6, 3, 7, 9, 10, 8, 11)$

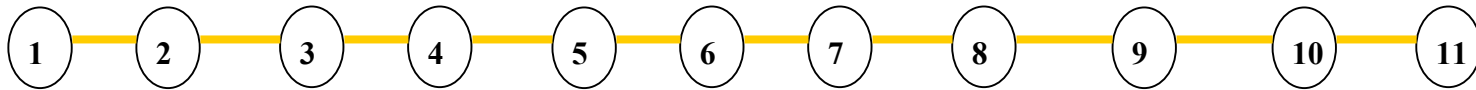
Compounded 1-relocate

- The size of the 1-relocate neighborhood is $O(n^2)$
- The size of the compounded independent 1-relocate neighborhood is $\Theta(1.7549^n)$
(Proof is by solving a recursion for the number of paths from 1 to $n+1$)

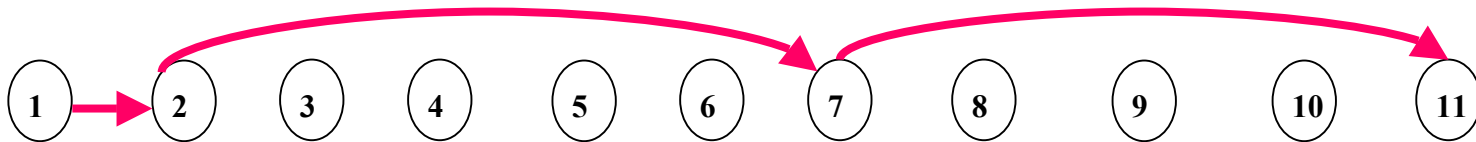
Compounded 1-relocate

Improvement Graph

$T = (1,2,3,4,5,6,7,8,9,10,1)$:



Construct improvement graph



$$c_{1,2} = 0$$

$$c_{2,7} = -(d_{2,3} + d_{3,4} + d_{6,7}) + (d_{2,4} + d_{6,3} + d_{3,7})$$

$$c_{7,11} = -(d_{7,8} + d_{8,9} + d_{10,1}) + (d_{7,9} + d_{10,8} + d_{8,1})$$

Improvement Graph

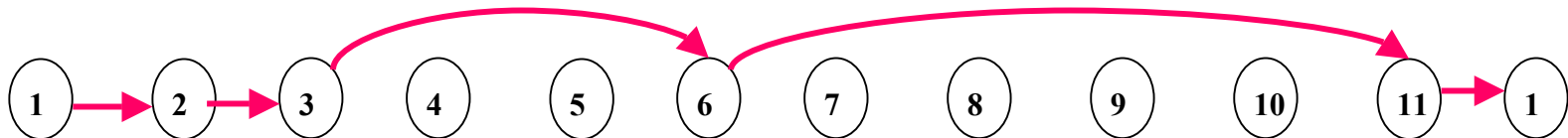


- Only forward arcs are allowed
- Node 1 is always kept fixed
- Find shortest path from 1 to $n+1$ in $O(n^2)$ time
- Negative cost shortest path implies an improving move

Compounded swap

- $T = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1)$

Compounded swap neighborhood

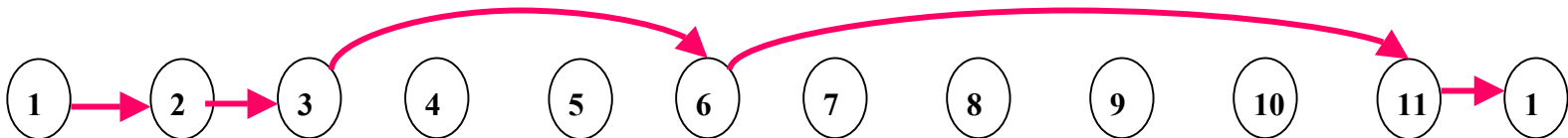


- $T' = (1, 2, 3, 5, 4, 6, 10, 8, 9, 7, 11, 1)$

Compounded 2-change

- $T = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1)$

Compounded 2-change neighborhood

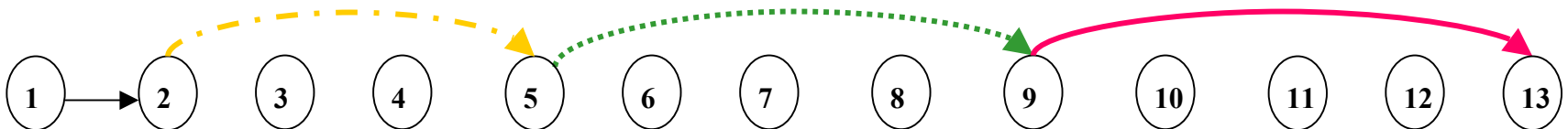


- $T = (1, 2, 3, 5, 4, 6, 10, 9, 8, 7, 11, 1)$

Moreover ...

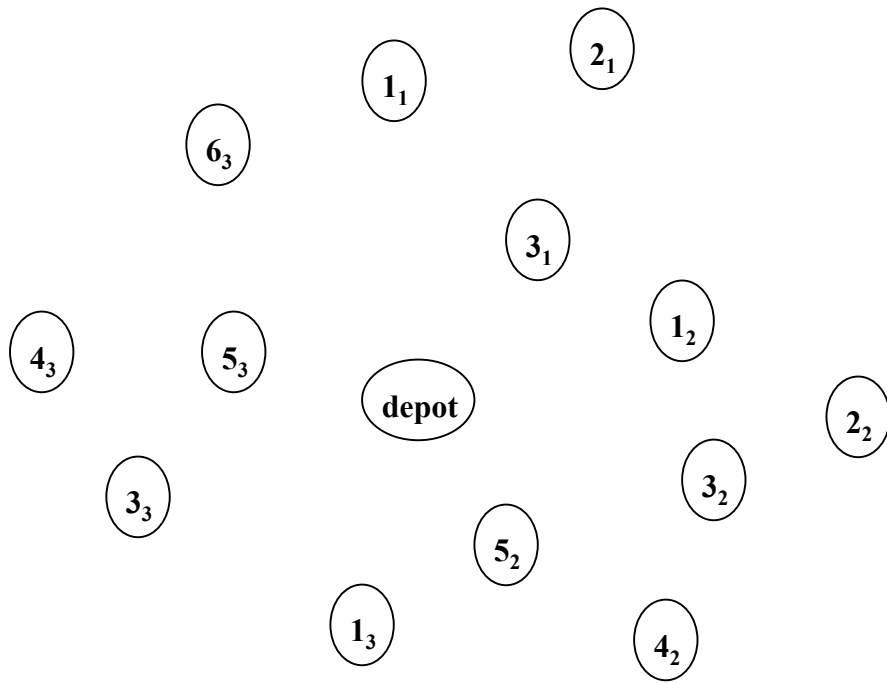
- $T = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1)$

Let  represent 2-change move
 represent swap move
 represent 1-relocate move

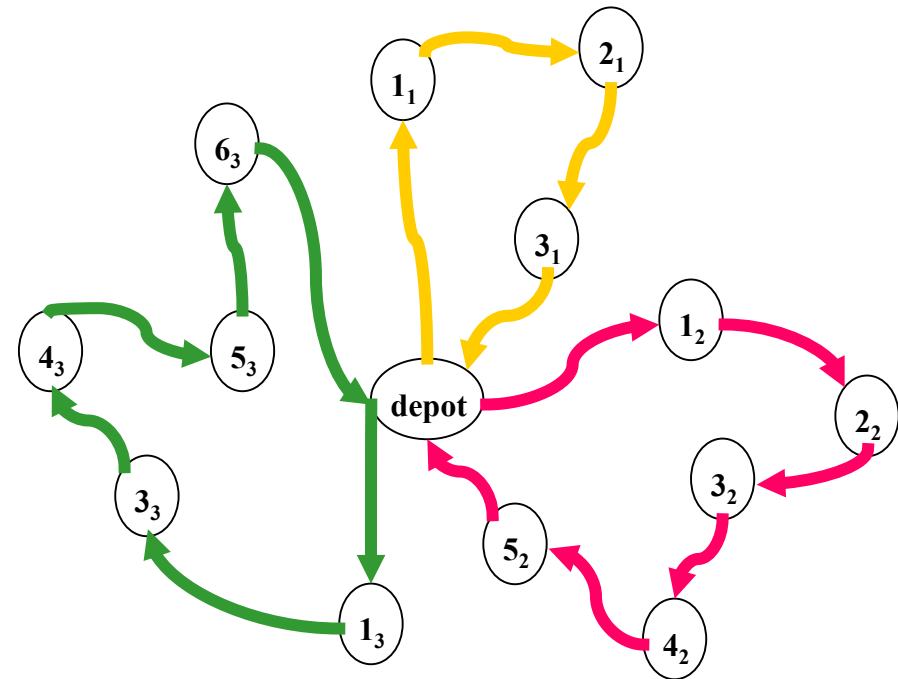


- $T = (1, 2, 4, 3, 5, 7, 8, 6, 9, 12, 11, 10, 1)$

Vehicle routing problem

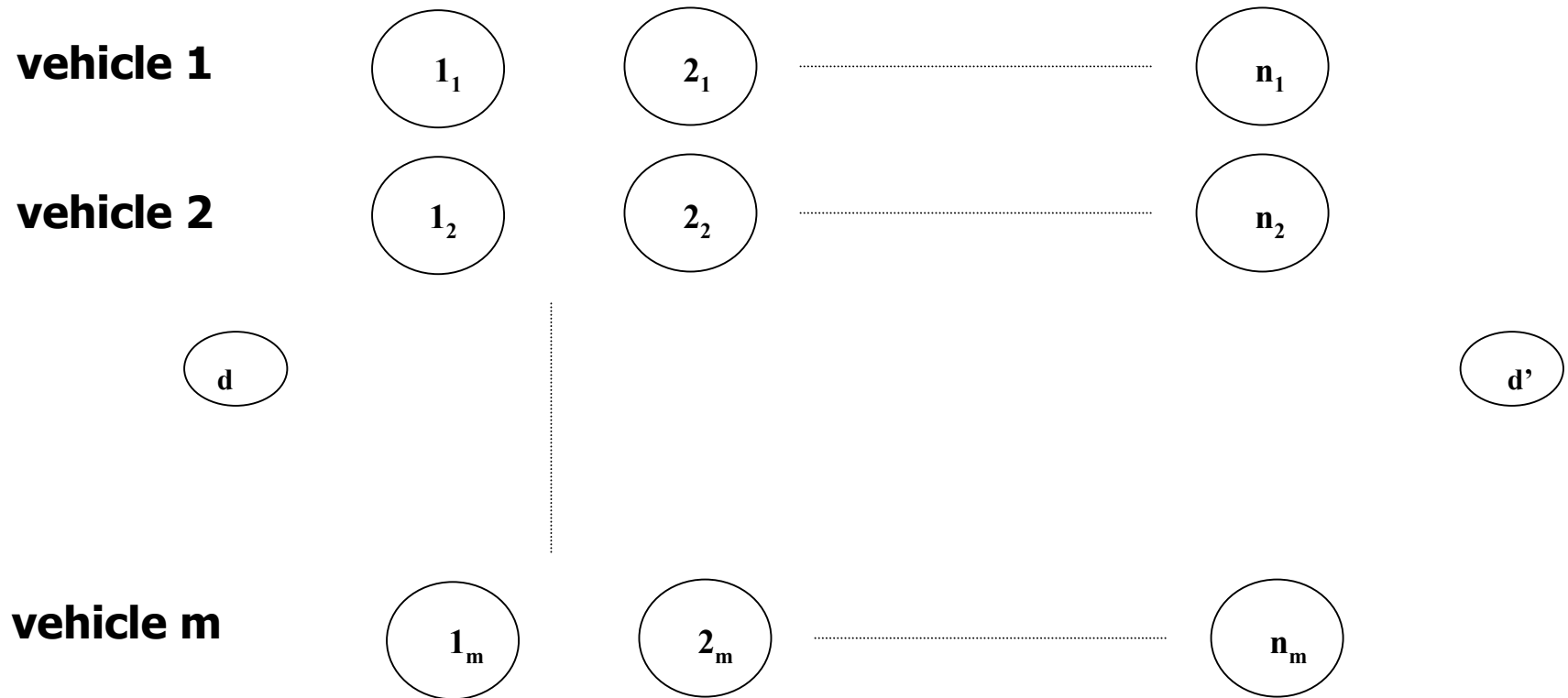


Instance



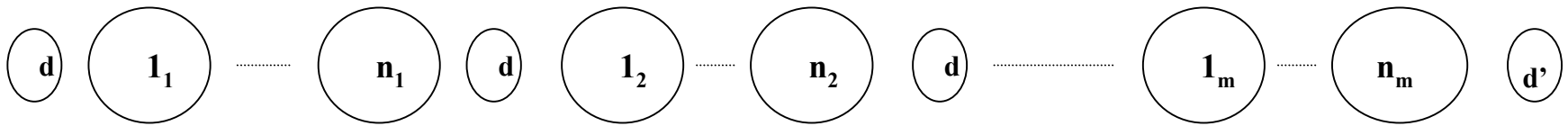
Solution

Two representations



Multi tour representation

Two representations



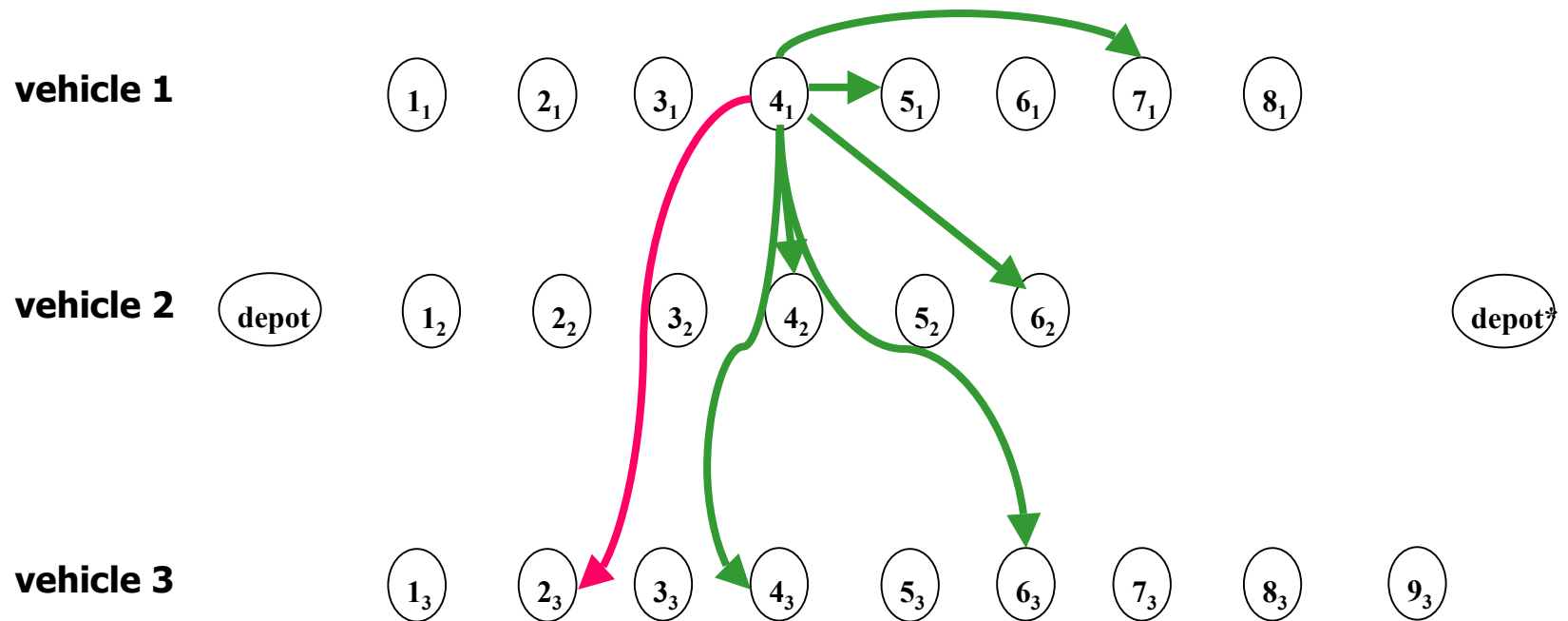
Single tour representation

Single tour representation



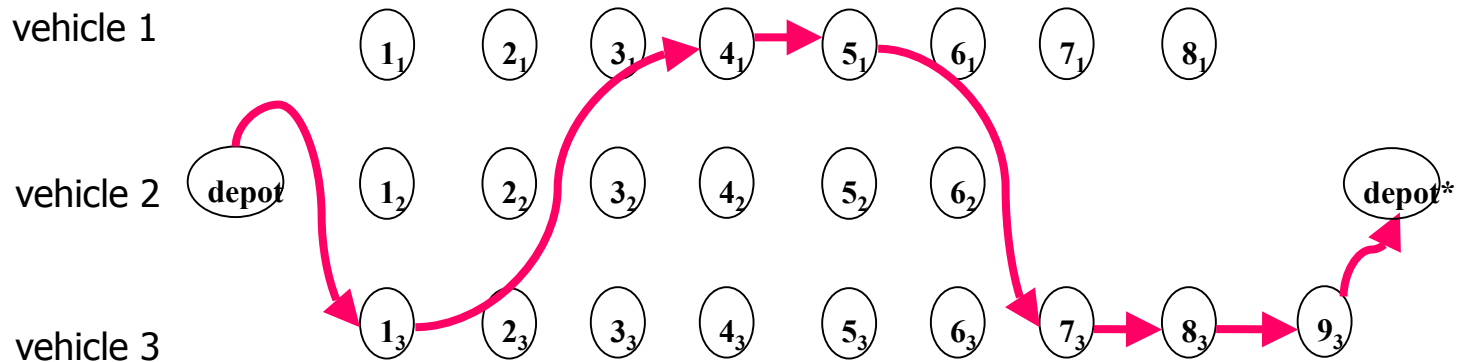
- Improvement graph is analogous to the TSP improvement graph
- For every ordering of vehicle one obtains a different neighborhood

Multi tour representation

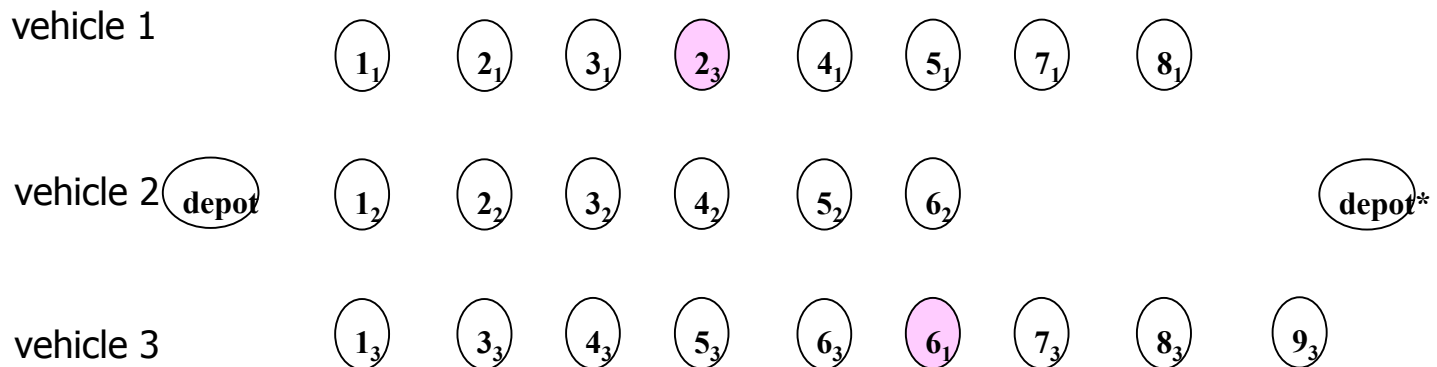


- The cost structure is not well defined for arc $(4_1, 2_3)$
- Establish an alignment scheme to define and allow only forward arcs

Multi tour improvement graph



After applying the exchanges implied by the shortest path:



MT neighborhood



- Additional flexibility:
 - shortest path from left to right
 - shortest path from right to left
- Additional complexity:
 - moves no longer independent due to capacity and distance restriction
- Constructing improvement graph and finding shortest path take $O(n^2)$ time

Handling capacity constraints

- for each node keep a **working capacity label** of each vehicle as well as a distance label
- $\text{available capacity}[k]$ = available capacity of vehicle k in current solution
- $\text{working capacity}[i,k]$ = $\text{available capacity}[k]$ + effects of changes corresponding to shortest path to i
- **allow only feasible arcs** with respect to working capacity in shortest path

Searching ST and MT neighborhoods



- Complexity of the search is $O(n^2 + nm)$
 - $O(n^2)$ for creating the improvement graph and running the shortest path algorithm
 - $O(nm)$ for updating the labels at each node once after all the incoming arcs to the node is considered