Programming Exercise 2
Construction Heuristic

Meta-Heuristic Approach

Soeren Nickel
Borna Feldsar

Approach

1. Start out with initial solution (Construction heuristic)
2. Variable Neighbourhood Descent
3. (Generalized) Variable Neighbourhood Search

Used Ressources

- Provided paper (The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations; Schneider, Stenger, Goeke) and slides
- Python 3.x
- Provided Verifier
- Graphviz/Dot for visualization

Computer

id:	`memory`
description:	System memory
physical id:	`0`
size:	3878MiB

id:	`cpu`
product:	Intel(R) Pentium(R) 3556U @ 1.70GHz
vendor:	Intel Corp.
physical id:	`1`
bus info:	`cpu@0`
size:	1700MHz
width:	64 bits

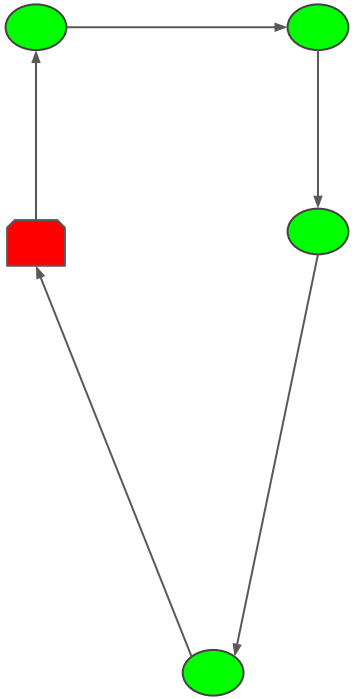
VND

- pick customer with biggest distance to weight point of the route
- 3 Neighbourhood Structures:
  - Choose customer which is farthest from the weight point for each route and insert in every valid place in its route
  - Choose customer which is farthest from the weight point for each route and insert in random place in any route
  - Choose customer which is farthest from the weight point for each route swap with each selected customer of any route
- find best solution in generated neighbourhood
- accept if cost is smaller
- terminate if no improvement on any level

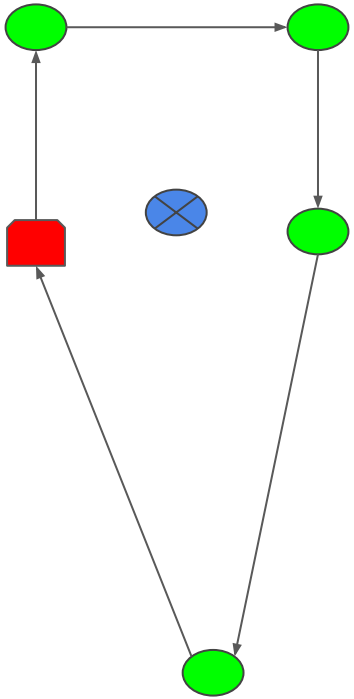
procedure VND (x)

begin
  $l \leftarrow 1$;
  repeat:
    find a $x' \in \mathcal{N}_l(x)$ with $f(x') \leq f(x'')$, $\forall x'' \in \mathcal{N}_l(x)$;
    if $f(x') < f(x)$ **then**
      $x \leftarrow x'$;
      $l \leftarrow 1$;
    else
      $l \leftarrow l + 1$;
  until $l > l_{max}$;
  return x;
end

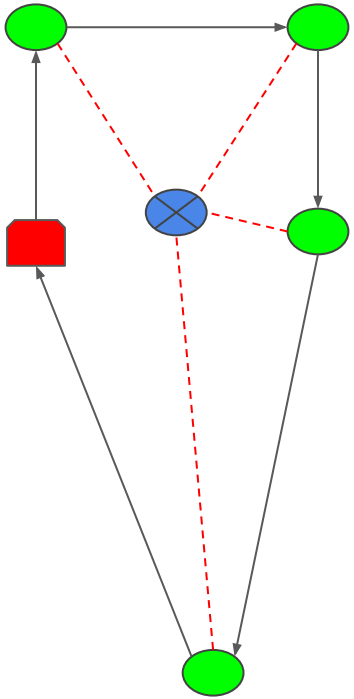
Weight Point



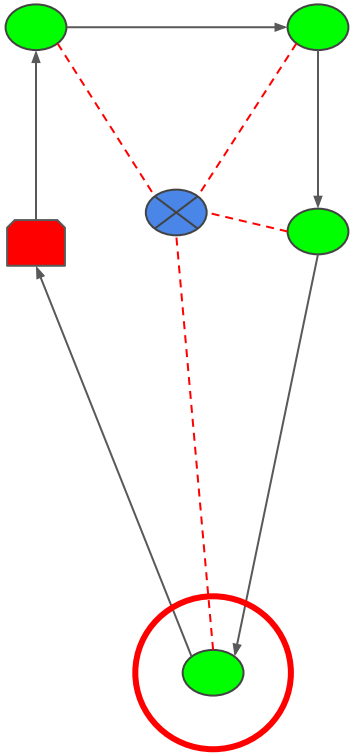
Weight Point



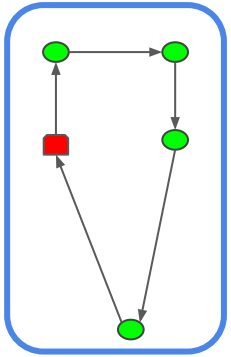
Weight Point



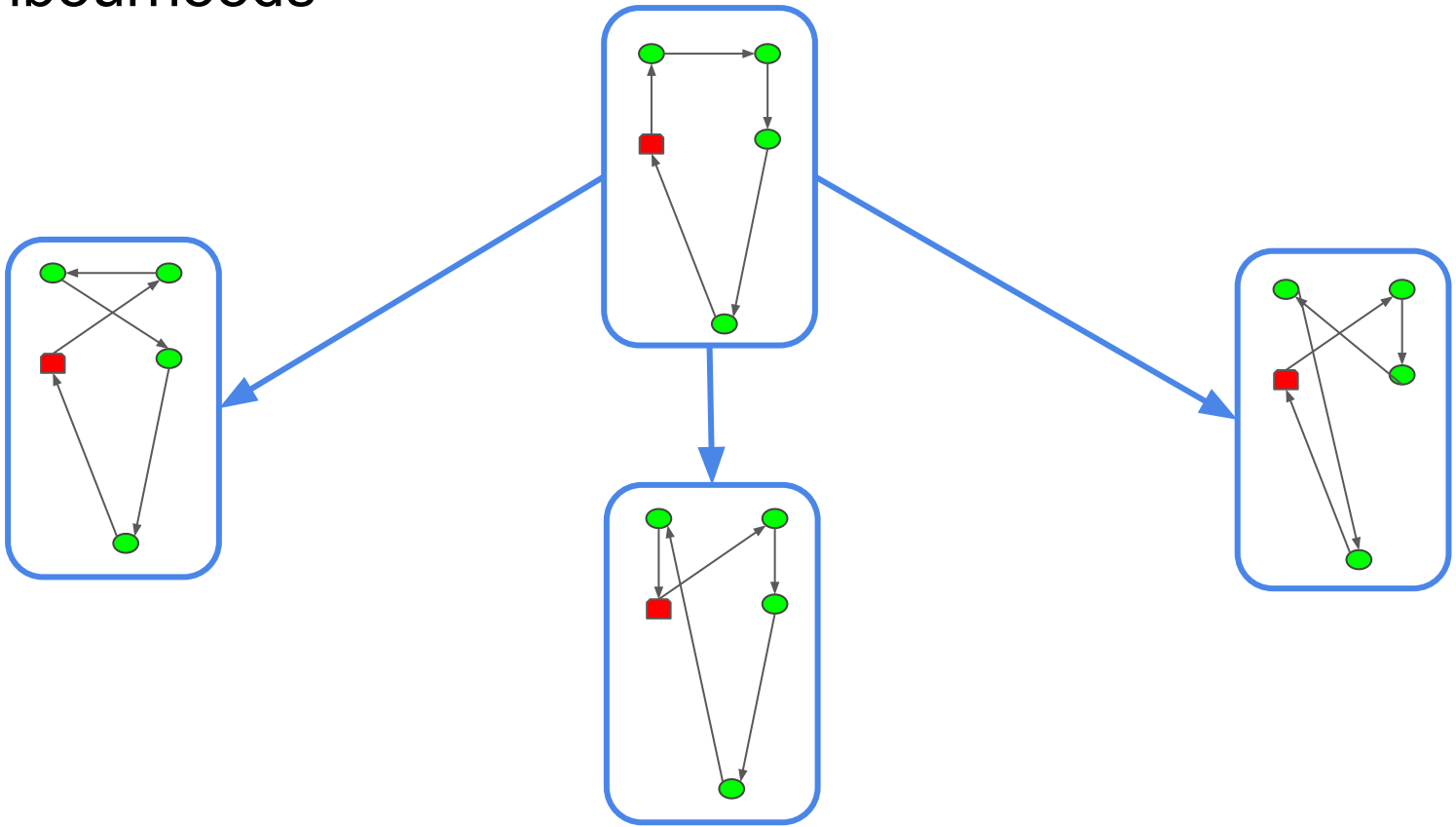
Weight Point



Neighbourhoods



Neighbourhoods



Remark

While generating new route, every Charger is removed and the route is made feasible (if possible) by inserting Charger at necessary Positions

Only Customers are swapped in Neighbourhood creation

GVNS

- pick customer with biggest waiting time of the route
- 2 Neighbourhood Structures:
  - Choose customer with biggest waiting time for each route and insert in random valid place in other routes
  - Choose customer with biggest waiting time for each route swap with each selected customer of any route
- find random solution in generated neighbourhood
- run VND on found solution
- terminate after set number of shaking procedures

procedure General VNS (x)

begin
  repeat:
    $k \leftarrow 1$;
    repeat:
      Shaking: generate random $x' \in \mathcal{N}_k(x)$;
      $x' \leftarrow$ ***local search with VND*** (x');
      if $f(x') < f(x)$ **then**
        $x \leftarrow x'$;
        $k \leftarrow 1$;
      else
        $k \leftarrow k + 1$;
    until $k > k_{max}$;
  until termination criterion met;
  return x;
end

Insights

- Savings creates reasonable solutions

- VNS finds better solution but increases are only around 7%

- Main improvement: find better neighbourhood structures

Instance	Initial Cost	Final Cost	Improvement in %	Running Time
c103_21.txt	1730.76	1638.17	5.35	355.75
c105_21.txt	2004.05	1837.62	8.30	404.30
c204_21.txt	915.10	853.60	6.72	108.88
r102_21.txt	1,936.42	1,865.84	3.64	698.12
r107_21.txt	1,559.54	1,508.55	3.27	454.86
r205_21.txt	1,237.12	1,224.59	1.01	167.89
r211_21.txt	905.72	887.14	2.05	112.88
rc101_21.txt	3,117.66	2,894.39	7.16	1,477.43
rc106_21.txt	2,487.89	2,279.24	8.39	688.47
rc203_21.txt	1,081.22	1,078.84	0.22	79.10