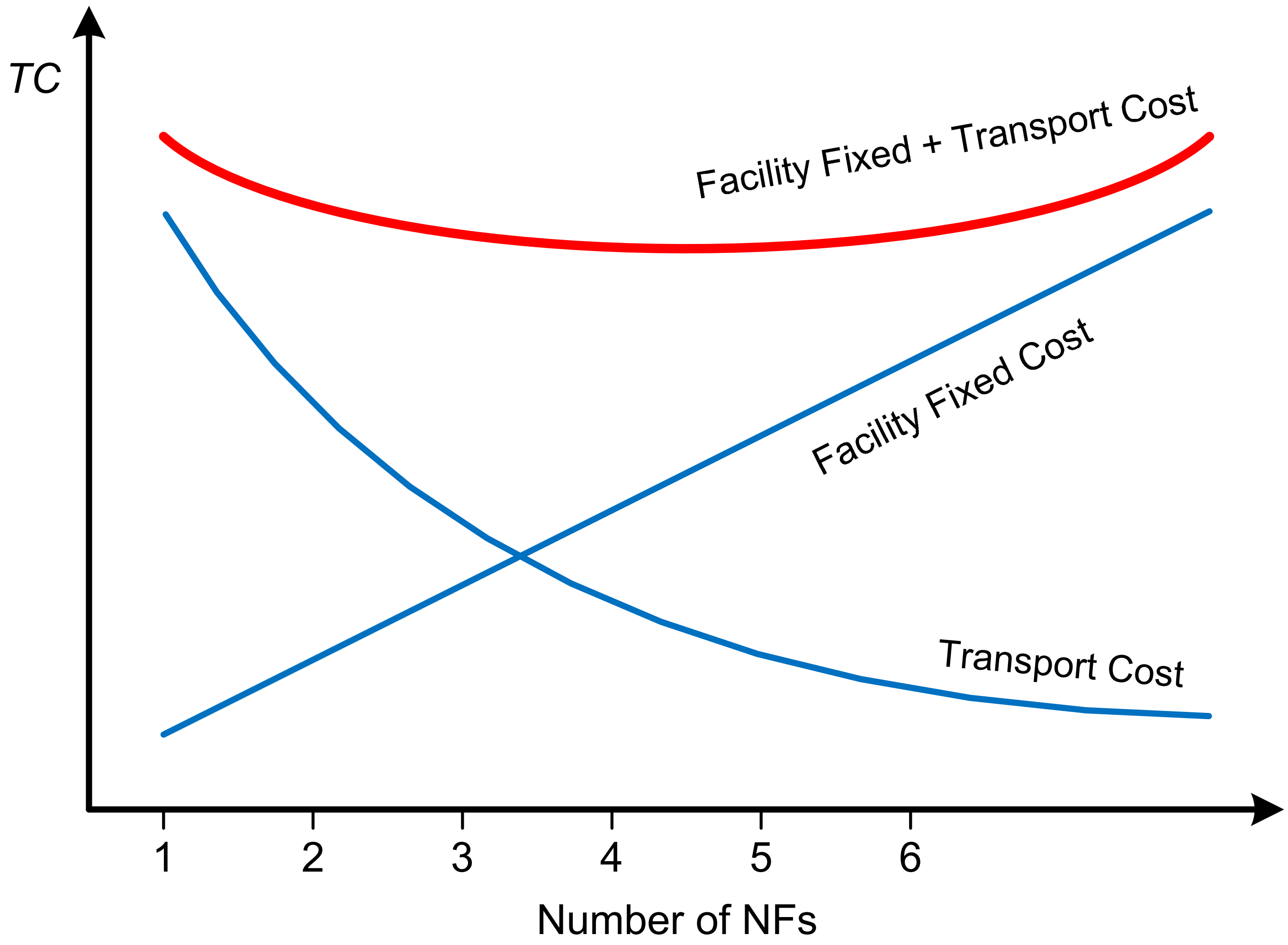
Location 5: UFL Heuristics

- Unlike the ALA, the *Uncapacitated Facility Location* (UFL) problem determines both the number of NFs and their locations
- Heuristics for the UFL usually provide a solution that is within ± 3% of the optimal
  - Optimal determined using MILP formulation of the UFL
  - Since data used for instances has significant errors, heuristics effectively solve the problem
  - Similar to ALA, MILP formulation is only needed if additional constraints are added to the problem

Optimal Number of NFs



Uncapacitated Facility Location (UFL)

- NFs can only be located at discrete set of sites
  - Allows inclusion of fixed cost of locating NF at site $\Rightarrow$ opt number NFs
  - Variable costs are usually transport cost from NF to/from EF
  - Total of $2^n - 1$ potential solutions (all nonempty subsets of sites)

$$M = \{1, \ldots, m\}, \quad \text{existing facilites (EFs)}$$

$$N = \{1, \ldots, n\}, \quad \text{sites available to locate NFs}$$

$$M_i \subseteq M, \quad \text{set of EFs served by NF at site } i$$

$$c_{ij} = \text{variable cost to serve EF } j \text{ from NF at site } i$$

$$k_i = \text{fixed cost of locating NF at site } i$$

$$Y \subseteq N, \quad \text{sites at which NFs are located}$$

$$Y^* = \arg\min_Y \left\{ \sum_{i \in Y} k_i + \sum_{i \in Y} \sum_{j \in M_i} c_{ij} : \bigcup_{i \in Y} M_i = M \right\}$$

$$= \text{min cost set of sites where NFs located}$$

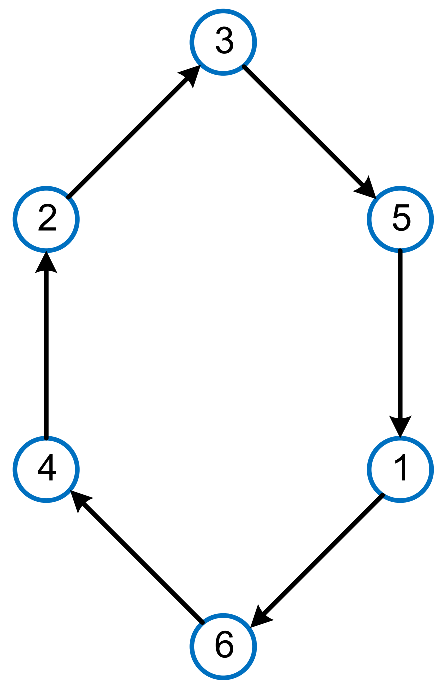
$$\left| Y^* \right| = \text{number of NFs located}$$

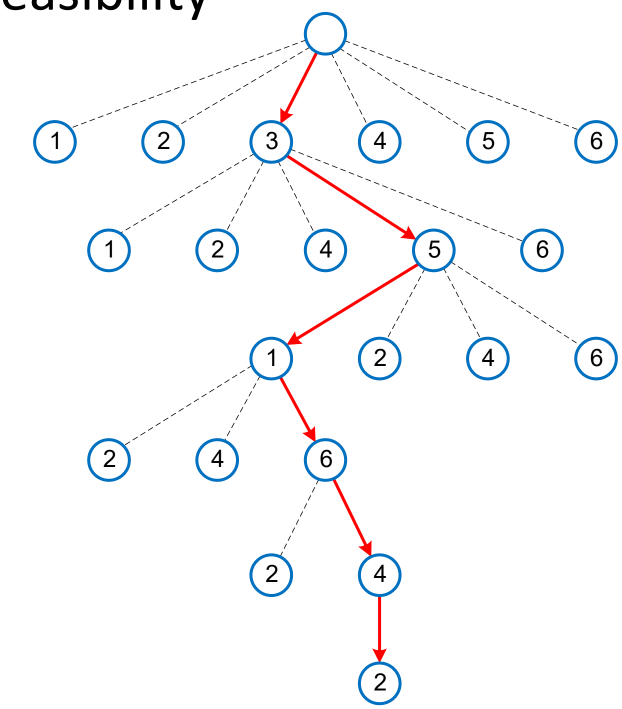
Heuristic Solutions

- Most problems in logistics engineering don't admit optimal solutions, only
  - Within some bound of optimal (provable bound, opt. gap)
  - Best known solution (stop when need to have solution)
- Heuristics - computational effort split between
  - Construction: construct a feasible solution
  - Improvement: find a better feasible solution
- Easy construction:
  - any random point or permutation is feasible
  - can then be improved $\Rightarrow$ *construct-then-improve* multiple times
- Hard construction:
  - almost no chance of generating a random feasible solution due to constraints on what is a feasible solution
  - need to include randomness at decision points as solution is generated in order to construct multiple different solutions (which "might" then be able to be improved)

Heuristic Construction Procedures

- Easy construction:
  - any random permutation is feasible and can then be improved
- Hard construction:
  - almost no chance of generating a random solution in a single step that is feasible, need to include randomness at decision points as a solution is constructed
  - each decision step checked for feasibility

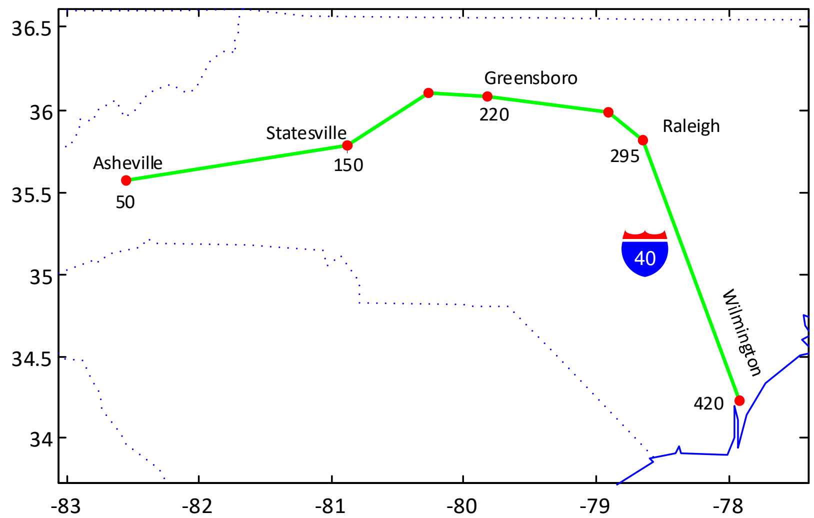




UFL Solution Techniques

- Being uncapacitated allows simple heuristics to be used to solve
  - ADD construction: add one NF at a time
  - DROP construction: drop one NF at a time
  - XCHG improvement: move one NF at a time to unoccupied sites
  - HYBRID algorithm combination of ADD and DROP construction with XCHG improvement, repeating until no change in Y
    - Use as default heuristic for UFL
    - See Daskin [2013] for more details
- UFL can be solved as a MILP
  - Easy MILP, LP relaxation usually optimal (for strong formulation)
  - MILP formulation allows constraints to easily be added
    - e.g., capacitated facility location, fixed number of NFs, some NF at fixed location
  - Will model UFL as MILP mainly to introduce MILP, will use UFL HYBRID algorithm to solve most problems

Ex: UFL ADD



$$k = [150 \quad 200 \quad 150 \quad 150 \quad 200]$$

$$c_{ij} = w_j d_{ij} = f_j r d_{ij} = (1)(1) d_{ij} = d_{ij}$$

	c_{ij}	1	2	3	4	5
Asheville:	1	0	100	170	245	370
Statesville:	2	100	0	70	145	270
Greensboro:	3	170	70	0	75	200
Raleigh:	4	245	145	75	0	125
Wilmington:	5	370	270	200	125	0

$Y = \{\ \}$

Y	1	2	3	4	5	c_{Yj}	k_Y	$c_{Yj} + k_Y$
1	0	100	170	245	370	885	150	1,035
2	100	0	70	145	270	585	200	785
3	170	70	0	75	200	515	150	665
4	245	145	75	0	125	590	150	740
5	370	270	200	125	0	965	200	1,165

$Y = \{3\}$

Y	1	2	3	4	5	c_{Yj}	k_Y	$c_{Yj} + k_Y$
3,1	0	70	0	75	200	345	300	645
3,2	100	0	0	75	200	375	350	725
3,4	170	70	0	0	125	365	300	665
3,5	170	70	0	75	0	315	350	665

$Y = \{3,1\}$

Y	1	2	3	4	5	c_{Yj}	k_Y	$c_{Yj} + k_Y$
3,1,2	0	0	0	75	200	275	500	775
3,1,4	0	70	0	0	125	195	450	645
3,1,5	0	70	0	75	0	145	500	645

$Y^* = \{3,1\}$

UFLADD: Add Construction Procedure

procedure $ufladd(\mathbf{k}, \mathbf{C})$
$Y \leftarrow \{\}$
$TC \leftarrow \infty$, $done \leftarrow$ false
repeat
    $TC' \leftarrow \infty$
    for $i' \in \{1, \ldots, n\} \setminus Y$
        $TC'' \leftarrow \sum_{h \in Y \cup i'} k_h + \sum_{j=1}^{m} \min_{h \in Y \cup i'} c_{hj}$
        if $TC'' < TC'$
            $TC' \leftarrow TC''$, $i \leftarrow i'$
        endif
    endfor
    if $TC' < TC$
        $TC \leftarrow TC'$, $Y \leftarrow Y \cup i$
    else
        $done \leftarrow$ true
    endif
until $done =$ true
return Y, TC

```
function ufladd(k, C)
    fTC(y) = sum(k[y]) + sum(minimum(C[y, :], dims=1))
    y = Int[]
    TC°, done = Inf, false
    while !done
        TC, i = Inf, nothing                      # Stops if y = all NF
        for i′ = setdiff(1:size(C, 1), y)  # since i′ = []
            TC′ = fTC(vcat(y, i′))
            if TC′ < TC
                TC, i = TC′, i′
            end
        end
        if TC < TC°                             # TC = Inf if y = all NF
            TC°, y = TC, push!(y, i)
        else
            done = true
        end
    end
    return y, TC°
end
```

UFLXCHG: Exchange Improvement Procedure

procedure $uflxchg(\mathbf{k}, \mathbf{C}, Y)$

$TC \leftarrow \sum_{i \in Y} k_i + \sum_{j=1}^{m} \min_{i \in Y} c_{ij}$

$TC' \leftarrow \infty$, $done \leftarrow$ false

while $|y| > 1$ and $done =$ false

  for $i' \in y$

    for $j' \in \{1,\ldots,n\} \setminus Y$

      $Y' \leftarrow Y \setminus i' \cup j'$

      $TC'' \leftarrow \sum_{i \in Y'} k_i + \sum_{j=1}^{m} \min_{i \in Y'} c_{ij}$

      if $TC'' < TC'$

        $TC' \leftarrow TC''$, $i \leftarrow i'$, $j \leftarrow j'$

      endif

    endfor

  endfor

  if $TC' < TC$

    $TC \leftarrow TC', Y \leftarrow Y \setminus i \cup j$

  else

    $done \leftarrow$ true

  endif

endwhile

return Y, TC

```
function uflxchg(k, C, y::Vector{Int})
    if k isa Number
        k = fill(k, size(C, 1))
    end
    fTC(y) = sum(k[y]) + sum(minimum(C[y, :], dims=1))
    N = 1:size(C, 1)
    TC° = fTC(y)
    done = false
    while length(y) > 1 && !done      # No exchange if 1 NF
        TC, i, j = Inf, nothing, nothing
        for i′ in  y
            for j′ in setdiff(N, y)
                swap!(y, i′, j′)      # Swap i′ in y with j′
                TC′ = fTC(y)
                if TC′ < TC
                    TC, i, j = TC′, i′, j′
                end
                revert!(y, i′, j′)   # Restore original y
            end
        end
        if TC < TC°
            TC° = TC
            swap!(y, i, j)
        else
            done = true
        end
    end
    return y, TC°
end
```

Modified UFLADD

procedure $ufladd(\mathbf{k}, \mathbf{C}, \boxed{Y, p})$

$\cancel{Y \leftarrow \{\}}$

$TC \leftarrow \infty$, $done \leftarrow$ false

repeat

  $TC' \leftarrow \infty$

  for $i' \in \{1, \ldots, n\} \setminus Y$

    $$TC'' \leftarrow \sum_{h \in Y \cup i'} k_h + \sum_{j=1}^{m} \min_{h \in Y \cup i'} c_{hj}$$

    if $TC'' < TC'$

      $TC' \leftarrow TC'', i \leftarrow i'$

    endif

  endfor

  if $\boxed{(p = \{\} \text{ and } TC' < TC) \text{ or } (p \neq \{\} \text{ and } |Y| < p)}$

    $TC \leftarrow TC', Y \leftarrow Y \cup i$

  else

    $done \leftarrow$ true

  endif

until $done =$ true

return Y, TC

- *Y* input can be used to start UFLADD with *Y* NFs
  - Used in hybrid heuristic
- *p* input can be used to keep adding until number of NFs = *p*
  - Used in p-median heuristic

UFL: Hybrid Algorithm

procedure $ufl(\mathbf{k},\mathbf{C})$
$Y', TC' \leftarrow ufladd(\mathbf{k},\mathbf{C})$
$done \leftarrow$ false
repeat
  $Y, TC \leftarrow uflxchg(\mathbf{k},\mathbf{C},Y')$
  if $Y \neq Y'$
    $Y', TC' \leftarrow ufladd(\mathbf{k},\mathbf{C},Y)$
    $Y'', TC'' \leftarrow ufldrop(\mathbf{k},\mathbf{C},Y)$
    if $TC'' < TC'$
      $TC' \leftarrow TC'', Y' \leftarrow Y''$
    endif
    if $TC' \geq TC$
      $done \leftarrow$ true
    endif
  else
    $done \leftarrow$ true
  endif
until $done =$ true
return Y, TC

```
function ufl(k, C)
    y′, TC′ = ufladd(k, C)
    y, TC = y′, TC′
    done = false
    while !done
        y, TC = uflxchg(k, C, y′)
        if Set(y) !== Set(y′)
            y′, TC′ = ufladd(k, C, y)
            y′′, TC′′ = ufldrop(k, C, y)
            if TC′′ < TC′
                y′, TC′ = y′′, TC′′
            end
            if TC′ >= TC
                done = true
            end
        else
            done = true
        end
    end
    return y, TC
end
```

Note: Drop starts from *y*

P-Median Location Problem

- Similar to UFL, except
  - Number of NF has to equal *p* (discrete version of ALA)
  - No fixed costs

$$p = \text{number of NFs}$$

$$Y^* = \arg\min_Y \left\{ \sum_{i \in Y} \sum_{j \in M_i} c_{ij} : \bigcup_{i \in Y} M_i = M, |Y| = p \right\}$$

  - Since usually *n* » *p*, only UFLADD used to construct the solution
    - Starting from *n* NFs, UFLDROP would take too long
    - Same is true for UFL Hybrid

```
function pmedian(p, C)
    y = ufladd(0, C, p = p)[1]
    return uflxchg(0, C, y)
end
```