UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Algorithmic Methods for Mathematical Models (AMMM)

Greedy Algorithms (for Combinatorial Optimization)

Luis Velasco
(lvelasco @ ac.upc.edu)
Campus Nord D6-107

1

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Combinatorial Optimization

- A combinatorial optimization problem is defined by:
  - N: finite **ground set** of elements, index *i*
  - F: set of **feasible solutions** of *N*
  - c_i: **cost** of the element *i*

$$\min_{S \subseteq N} \sum_{i \in S} c_i$$
$$s.t. \quad S \in F$$

- Combinatorial problems can be modeled using binary variables $x_i \in \{0,1\}$, one per element.

2

Example: The Travelling Salesman Problem (TSP)

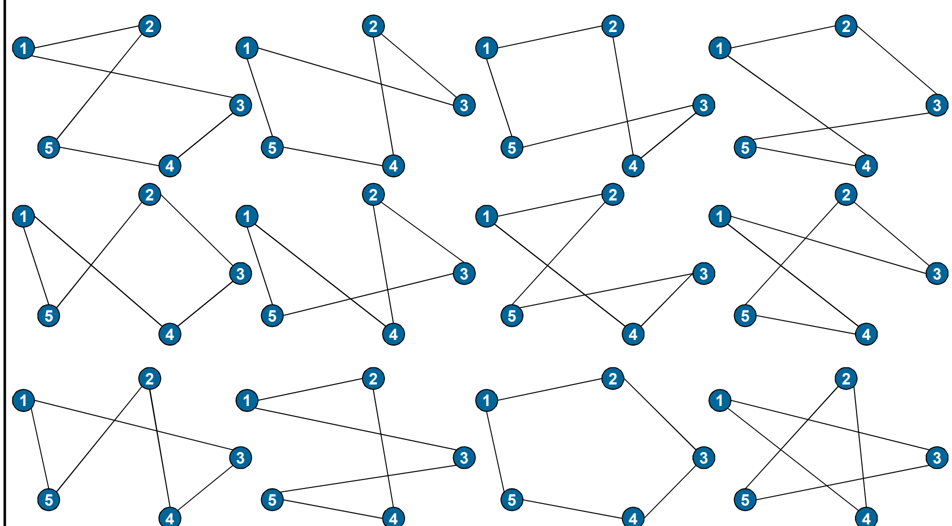
- Given a graph G(*V*,*E*) with a set of cities (*V*) and their pairwise distances
- The task is to find a **shortest** possible **tour** that visits each city exactly once (Hamiltonian cycle).
- TSP is a **combinatorial** problem. Its search space is (n-1)!/2
  - the ground set is that of all edges connecting the cities to be visited,
  - *F* is formed by all edge subsets that determine a Hamiltonian cycle.
  - *f*(*S*) is the sum of the distances of all edges in each Hamiltonian cycle
- What factorial means?

n	search space
3	1
4	3
5	12
6	60
7	360
8	2,520
9	20,160
10	181,440

n	search space
20	6.08E+16
30	4.42E+30
40	1.02E+46
50	3.04E+62
60	6.93E+79
70	8.56E+97
80	4.47E+116
90	8.25E+135
100	4.67E+155

Information on the largest TSP instances solved to date can be found in:
http://www.math.uwaterloo.ca/tsp/optimal/index.html

The (n-1)!/2 combinations (n=5)



Greedy algorithm

- A greedy algorithm builds the solution in an iterative manner.
  - At each iteration, **the best element** from a **candidate list** is added to the **partial solution**
- In general they have **five pillars**:
  - A **candidate set** C, from which a solution is created
  - A **selection function**, which chooses the best candidate to be added
  - A **feasibility function**, to determine if a candidate can be used
  - An **objective function** $f(S)$, which assigns a value to a solution, or a partial one
  - A **solution function**, which indicate when we have a complete solution

Greedy Algorithm for Combinatorial Problems

C: Candidate set, index c
$S \subseteq C$: (partial) solution
$q(c)$: quality of element c (*greedy function*). **Added value** of c for the partial solution S. If c makes S not feasible, then $q(c)$=INFINITE.

solution function
selection function
feasibility function

Initialize C
$S \leftarrow \{\}$
while $\boxed{S \text{ is not a solution}}$ **do**
  evaluate $q(c)\ \forall c \in C$
  $c_{best} \leftarrow \boxed{\text{argmax}\{q(c) \mid c \in C\}}$
  $S \leftarrow S \cup \{c_{best}\}$
  update C, e.g., $C \leftarrow C \setminus \{c_{best}\}$ ($\boxed{\text{you might want to exclude infeasible } c}$)
return $<f(S), S>$

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Example: TSP

The nearest neighbor (NN) algorithm

Given graph G(V,E)

Partial Solution → Tour ← {v}, where v is an arbitrary node (starting point) from V

while Tour ≠ V **do** *V is the candidate set*

  $C \leftarrow$ set of feasible links w.r.t. $Tour \subseteq E$

  $e_{best} \leftarrow (u \in Tour,\ v \notin Tour) \leftarrow \mathrm{argmin}\{d(Tour, e) \mid e \text{ in } C\}$

  $Tour \leftarrow Tour \cup \{v\}$

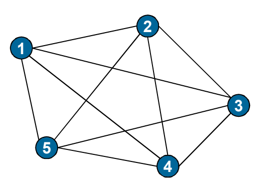
return Tour

For |*V*| cities randomly distributed on a plane, the algorithm on average yields length = 1.25 * shortest (optimal) length.

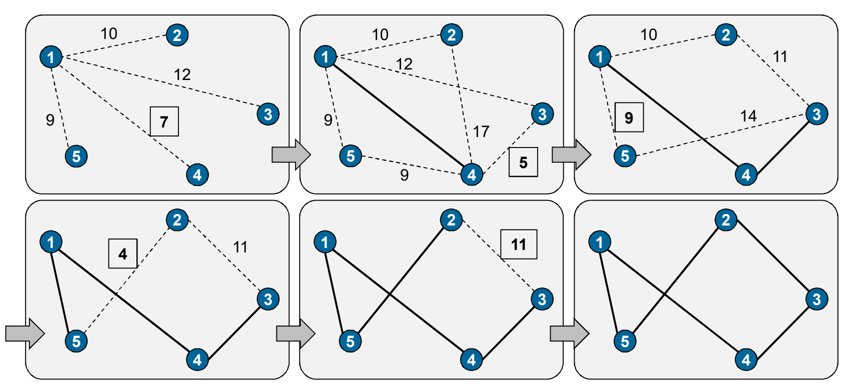
UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Example: TSP



	1	2	3	4	5
1	-	10	12	7	9
2		-	11	17	4
3			-	5	14
4				-	9
5					-

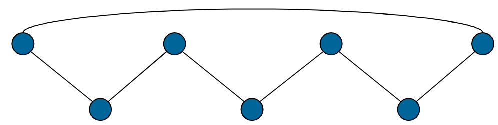


UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Example: TSP

- A greedy algorithm suffers from myopia.
  - It looks for the best candidate at each iteration.



UNIVERSITAT POLITÈCNICA DE CATALUNYA

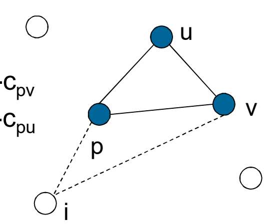
DAC

Other algorithms for the TSP

- **Nearest Insertion (greedy)**: From a small cycle, the algorithm expands the cycle by adding the nearest vertex.

$c_{ip}+c_{iv}-c_{pv}$

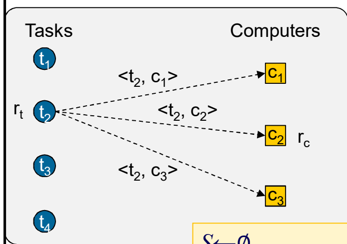
$c_{ip}+c_{iu}-c_{pu}$



- **Christofides algorithm**: Produces solutions within 3/2 of an optimal solution, i.e., $Z_{heuristic} \leq 3/2\ Z^*$.

  - Create the minimum spanning tree MST *T* of G.
  - Denote *O* the set of vertices with odd degree in *T*
  - Find a perfect matching *M* with minimal weight in the complete graph over the vertices from *O*.
  - Combine the edges of *M* and *T* to form a multigraph *H*.
  - Form an Eulerian path in *H* (*H* is Eulerian because it is connected, with only even-degree vertices).
  - Transform the path found in last step to be Hamiltonian by skipping visited nodes (*shortcutting*).

Example 1: Assign tasks to computers (lab session 2)



$$q(<t,c>) = max \begin{Bmatrix} 1 - \dfrac{(residualCapaciy(c) - r_t)}{r_c} \\ 1 - \dfrac{residualCapaciy(c')}{r_{c'}} \mid c' in\ C, c' \neq c \end{Bmatrix}$$

$S \leftarrow \emptyset$
$sortedT \leftarrow$ sort(T, r_t, $DESC$)
for each c in C **do** $residualCapacity(c) = r_c$
for each t in T **do**
    $C(t) \leftarrow \emptyset$
    for each c in C **do**
        if $r_t \leq residualCapacity(c)$ **then** $C(t) \leftarrow C(t) \cup \{c\}$
    if $|C(t)|=0$ **then return** INFEASIBLE
    $c_{best} \leftarrow$ argmin$\{q(<t,c>) \mid c$ in $C(t)\}$
    $residualCapacity(c_{best}) \leftarrow residualCapacity(c_{best})$ - r_t
    $S \leftarrow S \cup \{<t, c_{best}>\}$
return S

11

Assignment Tasks to computers: Iterative execution

Computers	c1	c2	c3	
rc	505.67	503.68	701.78	
Tasks	t1	t2	t3	t4
rt	261.27	560.89	310.51	105.8

sortedTasks	t2	t3	t1	t4

Computers	c1	c2	c3
residualCap	505.67	503.68	701.78

#1

task: t2	560.89
C(t2)	c3
cbest	c3

Computers	c1	c2	c3
residualCap	505.67	503.68	**140.89**
load	0	0	**0.799**
S	**{<t2,c3>}**		

#2

task: t3	310.51	
C(t2)	c1	c2
Load if assignment		
c1	0.6141	
c2	0.6165	
cbest	c1	

Computers	c1	c2	c3
residualCap	**195.16**	503.68	140.89
load	**0.6141**	0	0.799
S	{<t2,c3>,**<t3,c1>**}		

#3

task: t1	261.27
C(t1)	c2
Load if assignment	
c2	0.5187
cbest	c2

Computers	c1	c2	c3
residualCap	195.16	**242.41**	140.89
load	0.6141	**0.5187**	0.799
S	{<t2,c3>,<t3,c1>,**<t1,c2>**}		

#4

task: t4	105.8		
C(t4)	c1	c2	c3
Load if assignment			
c1	0.8233		
c2	0.7288		
c3	0.95		
cbest	c2		

Computers	c1	c2	c3
residualCap	195.16	**136.61**	140.89
load	0.6141	**0.7288**	0.799
S	{<t2,c3>,<t3,c1>,<t1,c2>,**<t4,c2>**}		

Solution

S	**{<t2,c3>,<t3,c1>,<t1,c2>,<t4,c2>}**
f(S)	**0.799**

12

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Set Covering

- Let M={1, 2, … , m} be the universe of elements to be covered.
- Let $P=\{p_j\}_{j\in N}$, be a family of subsets p_j, N={1, 2, … , n}
- Let c_j be the cost associated with p_j, e.g. its cardinality ($|p_j|$).
- The set covering problem consists on finding the sub-family of elements $\{p_j\}_{j\in N^*}$, $N^* \le N$, with minimum cost such that $\cup p_j = M$, i.e., covering M.

M/P	p1	p2	p3	p4	p5	p6	p7	p8
1						X		
2			X	X			X	
3	X	X		X	X		X	
4	X			X	X	X		X
5					X	X		
cost	2	1	1	3	3	3	2	1

Optimal solutions (cost 5)
S={p6, p7}
S={p2, p3, p6}

Other feasible solutions
S={p1, p3, p6} (cost=6)
S={p4, p6} (cost=6)

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Example: Greedy for set covering

Let S the solution sub-family
Let R the set of covered elements
Greedy function:

$q(p_j)=|p_j \cap (M\backslash R)| = |p_j \backslash (R\cap p_j)|$ → Number of additional elements of p_j

If every p_j has its own associated cost c_j, the greedy function would be:

$q(p_j) = c_j / |p_j \cap (M\backslash R)|$

S={}
R={}

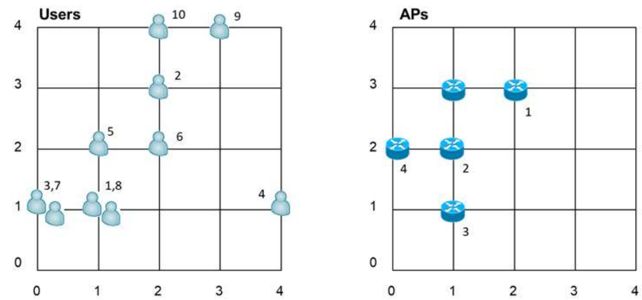
compute q(pj) ∀ $p_j \in P\backslash S$	q(p1) = 2	q(p5) = 3
Select the best element: p4	q(p2) = 1	q(p6) = 3
S={p4}	q(p3) = 1	q(p7) = 2
R={2, 3, 4}	q(p4) = 3	q(p8) = 1

compute q(pj) ∀ $p_j \in P\backslash S$	q(p1) = 0	q(p5) = 1
Select the best element: p6	q(p2) = 0	q(p6) = 2
S={p4,p6}	q(p3) = 0	q(p7) = 1
R=M		q(p8) = 0

Cost: 6

Example 2: Network planning

- A set of users U needs to be connected to the Internet. For that purpose, we have a set of access point locations A where we could install routers (one per access point at the most).
  - For each user u, the amount cr_u of capacity units it consumes from the router it is connected to is given.



15

Example 2: Network planning

- We have a set M of router models.
  - Each model m with its fixed cost f_m, capacity k_m, and reach d_m.
  - A router m can only connect users that are within a distance d_m from the access point.
- We assume Euclidean distances, so for each user u and each access point a, we know its Cartesian coordinates (x, y).
- We have to decide:
  - which model of router, if any, should be installed in each access point,
  - which access point each user should be connected to.
  - The goal is to minimize the total cost, computed as the summation of the cost of all the installed routers.

16

Network planning: Greedy algorithm

$$q(u) = \min\{q(u,a)\}$$

Infeasible either because of the reach or the load

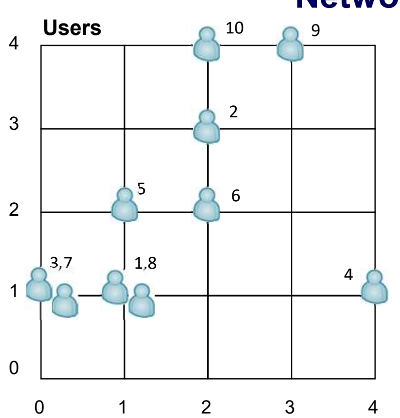
$$q(u,a) = \begin{cases} \infty & if \quad \boxed{d(u,a) > \max\{d_m\} \vee cr_u > \left(\max\{k_m\} - \sum_{u' \in U(a)} cr_{u'}\right)} \\ 0 & if \quad \boxed{d(u,a) \le d_a \wedge cr_u \le \left(k_a - \sum_{u' \in U(a)} cr_{u'}\right)} \\ f_m - f_a & if \quad \boxed{d(u,a) > d_a \vee cr_u > \left(k_a - \sum_{u' \in U(a)} cr_{u'}\right), d(u,a) \le d_m \wedge cr_u \le \left(k_m - \sum_{u' \in U(a)} cr_{u'}\right)} \end{cases}$$

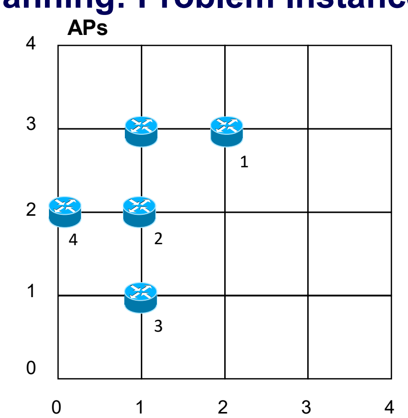
User *u* can be served with the router currently installed in location *a*

Router currently installed in location *a* needs to be upgraded because of the reach or the load to serve user *u*

$S \leftarrow \emptyset$, $C \leftarrow U$
Evaluate the incremental costs $q(u)$ for all $u \in C$
while $C \neq \emptyset$ **do**
    $u^{min} \leftarrow \text{argmin } \{q(u) \mid u \in C\}$
    $S \leftarrow S \cup \{u^{min}\}$
    $C \leftarrow C \setminus \{u^{min}\}$
    Reevaluate the incremental costs $q(u)$ for all $u \in C$
return S

Network planning: Problem Instance





	R1	R2	R3
f	100	140	180
k	6	8	10
d	2	3	4

d(u,a)		u	1	2	3	4	5	6	7	8	9	10
		x	1	2	0	4	1	2	0	1	3	2
		y	1	3	1	1	2	2	1	1	4	4
1	2	3	2.2	0.0	2.8	2.8	1.4	1.0	2.8	2.2	1.4	1.0
2	1	2	1.0	1.4	1.4	3.2	0.0	1.0	1.4	1.0	2.8	2.2
3	1	1	0.0	2.2	1.0	3.0	1.0	1.4	1.0	0.0	3.6	3.2
4	0	2	1.4	2.2	1.0	4.1	1.0	2.0	1.0	1.4	3.6	2.8
5	1	3	2.0	1.0	2.2	3.6	1.0	1.4	2.2	2.0	2.2	1.4
a	x	y										

Network planning: Iterative execution (1/5)

#1

u	1	2	3	4	5	6	7	8	9	10
cr	2	3	4	1	2	2	1	2	3	4
q(u)	100	100	100	140	100	100	100	100	100	100
a	3	1	3	1	2	1	3	3	1	1
d(u,a)	0.0	0.0	1.0	2.8	0.0	1.0	1.0	0.0	1.4	1.0

a	1	2	3	4	5
m			R1		
U(a)			{1}		
km-cr			4		

#2

u	1	2	3	4	5	6	7	8	9	10
cr	2	3	4	1	2	2	1	2	3	4
q(u)		40	0	40	0	0	0	0	80	80
a	3	3	3	3	3	3	3	3	3	3
d(u,a)	0.0	2.2	1.0	3.0	1.0	1.4	1.0	0.0	3.6	3.2

a	1	2	3	4	5
m			R1		
U(a)			{1,8}		
km-cr			2		

Network planning: Iterative execution (2/5)

#3

u	1	2	3	4	5	6	7	8	9	10
cr	2	3	4	1	2	2	1	2	3	4
q(u)		40	40	0	0	0	0		80	80
a	3	3	3	3	3	3	3	3	3	3
d(u,a)	0.0	2.2	1.0	3.0	1.0	1.4	1.0	0.0	3.6	3.2

a	1	2	3	4	5
m			R1		
U(a)			{1,5,8}		
km-cr			0		

#4

u	1	2	3	4	5	6	7	8	9	10
cr	2	3	4	1	2	2	1	2	3	4
q(u)		80	80	40		40	40		80	80
a	3	3	3	3	3	3	3	3	3	3
d(u,a)	0.0	2.2	1.0	3.0	1.0	1.4	1.0	0.0	3.6	3.2

a	1	2	3	4	5
m			R2		
U(a)			{1,5,7,8}		
km-cr			1		

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Network planning: Iterative execution (3/5)

#5

u	1	2	3	4	5	6	7	8	9	10
cr	2	3	4	1	2	2	1	2	3	4
q(u)		100	100	0		40			100	100
a	3	1	4	3	3	3	3	3	1	1
d(u,a)	0.0	0.0	1.0	3.0	1.0	1.4	1.0	0.0	1.4	1.0

a	1	2	3	4	5
m			R2		
U(a)			{1,4,5,7,8}		
km-cr			0		

#6

u	1	2	3	4	5	6	7	8	9	10
cr	2	3	4	1	2	2	1	2	3	4
q(u)		100	100			40			100	100
a	3	1	4	3	3	3	3	3	1	1
d(u,a)	0.0	0.0	1.0	3.0	1.0	1.4	1.0	0.0	1.4	1.0

a	1	2	3	4	5
m			R3		
U(a)			{1,4,5,6,7,8}		
km-cr			0		

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DAC

Network planning: Iterative execution (4/5)

#7

u	1	2	3	4	5	6	7	8	9	10
cr	4	3	4	1	2	2	1	2	3	4
q(u)		100	100						100	100
a	3	1	4	3	3	3	3	3	1	1
d(u,a)	0.0	0.0	1.0	3.0	1.0	1.4	1.0	0.0	1.4	1.0

a	1	2	3	4	5
m	R1		R3		
U(a)	{2}		{1,4,5,6,7,8}		
km-cr	3		0		

#8

u	1	2	3	4	5	6	7	8	9	10
cr	4	3	4	1	2	2	1	2	3	4
q(u)			40						0	40
a	3	1	1	3	3	3	3	3	1	1
d(u,a)	0.0	0.0	2.8	3.0	1.0	1.0	1.0	0.0	1.4	1.0

a	1	2	3	4	5
m	R1		R3		
U(a)	{2,9}		{1,4,5,6,7,8}		
km-cr	0		0		

Network planning: Iterative execution (5/5)

#9

u	1	2	3	4	5	6	7	8	9	10
cr	4	3	4	1	2	2	1	2	3	4
q(u)			80							80
a	3	1	1	3	3	3	3	3	1	1
d(u,a)	0.0	0.0	2.8	3.0	1.0	1.0	1.0	0.0	1.4	1.0

a	1	2	3	4	5
m	R3		R3		
U(a)	{2,9,10}		{1,4,5,6,7,8}		
km-cr	0		0		

#10

u	1	2	3	4	5	6	7	8	9	10
cr	4	3	4	1	2	2	1	2	3	4
q(u)			100							
a	3	1	4	3	3	3	3	3	1	1
d(u,a)	0.0	0.0	1.0	3.0	1.0	1.0	1.0	0.0	1.4	1.0

Solution Cost=460

a	1	2	3	4	5
m	R3		R3	R1	
U(a)	{2,9,10}		{1,4,5,6,7,8}	{3}	
km-cr	0		0	2	

23

Algorithmic Methods for Mathematical Models (AMMM)

Greedy Algorithms

Luis Velasco
(lvelasco @ ac.upc.edu)
Campus Nord D6-107

24