Heuristic Algorithms

Master's Degree in Computer Science/Mathematics

Roberto Cordone

DI - Università degli Studi di Milano

Schedule:	Thursday 14.30 - 16.30 in classroom 201
	Friday 14.30 - 16.30 in classroom 100
Office hours:	on appointment
E-mail:	roberto.cordone@unimi.it
Web page:	https://homes.di.unimi.it/cordone/courses/2023-ae/2023-ae.html
Ariel site:	https://rcordoneha.ariel.ctu.unimi.it

Combinatorial Optimization

$$\begin{array}{l} \text{opt} \;\; f(x) \\ x \in X \end{array}$$

where $X \subseteq 2^B$ and B finite

We will survey a number of problem classes

- set problems
- logic function problems
- numerical matrix problems
- graph problems

Why a problem survey?

Reviewing several problems is useful because

- abstract ideas must be concretely applied to different algorithms for different problems
- the same idea can have different effectiveness on different problems
- some ideas only work on problems with a specific structure
- different problems could have nonapparent relations, which could be exploited to design algorithms

So, a good knowledge of several problems teaches how to

- apply abstract ideas to new problems
- find and exploit relations between known and new problems

Sure, the "Magical Number Seven" risk exists...

To control it, we will make some interludes devoted to general remarks

Weighted set problems: Knapsack Problem (KP)

Given

- a set E of elementary objects
- a function $v : E \rightarrow \mathbb{N}$ describing the volume of each object
- a number $V \in \mathbb{N}$ describing the capacity of a knapsack
- a function $\phi : E \rightarrow \mathbb{N}$ describing the value of each object

select a subset of objects of maximum value that respects the capacity

The ground set is trivially the set of the objects: $B = E$

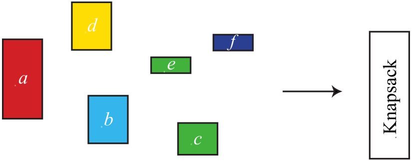
The feasible region includes all subsets of objects whose total volume does not exceed the capacity of the knapsack

$$X = \left\{ x \subseteq B : \sum_{j \in x} v_j \leq V \right\}$$

The objective is to maximise the total value of the chosen objects

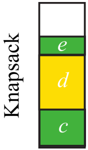
$$\max_{x \in X} f(x) = \sum_{j \in x} \phi_j$$

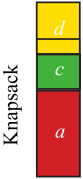
Example



E	a	b	c	d	e	f
ϕ	7	2	4	5	4	1
v	5	3	2	3	1	1

$V = 8$





$x' = \{c, d, e\} \in X$
$f(x') = 13$

$x'' = \{a, c, d\} \notin X$
$f(x'') = 16$

Set problems in metric spaces: *Maximum Diversity Problem* (*MDP*)

Given

- a set P of points
- a function $d : P \times P \rightarrow \mathbb{N}$ providing the distance between point pairs
- a number $k \in \{1, \ldots, |P|\}$ that is the number of points to select

select a subset of k points with the maximum total pairwise distance

The ground set is the set of points: $B = P$

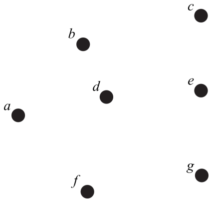
The feasible region includes all subsets of k points

$$X = \{x \subseteq B : |x| = k\}$$

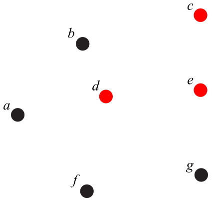
The objective is to maximise the sum of all pairwise distances between the selected points

$$\max_{x \in X} f(x) = \sum_{(i,j):i,j \in x} d_{ij}$$

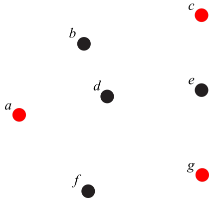
Example



$$k = 3$$



$$x' = \{C, D, E\} \in X$$
$$f(x') = 24$$



$$x'' = \{A, C, G\} \in X$$
$$f(x'') = 46$$

Interlude 1: the objective function

The objective function associates integer values to feasible subsets

$$f : X \to \mathbb{N}$$

Computing the objective function can be complex (even exhaustive)

We have seen two simple cases

- the *KP* has an additive objective function which sums values of an auxiliary function defined on the ground set

$$\phi : B \to \mathbb{N} \text{ induces } f(x) = \sum_{j \in x} \phi_j : X \to \mathbb{N}$$

- the *MDP* has a quadratic objective function

Both are defined not only on X, but on the whole of 2^B (*is this useful?*)

Both are easy to compute, but the additive functions $f(x)$ are also fast to recompute if subset x changes slightly: it is enough to

- sum ϕ_j for each element j added to x
- subtract ϕ_j for each element j removed from x

For quadratic functions, this seems more complex (*we will talk about it*)

Partitioning set problems: *Bin Packing Problem* (*BPP*)

Given

- a set E of elementary objects
- a function $v : E \to \mathbb{N}$ describing the volume of each object
- a set C of containers
- a number $V \in \mathbb{N}$ that is the volume of the containers

divide the objects into the minimum number of containers respecting the capacity

The ground set $B = E \times C$ includes all (object,container) pairs

The feasible region includes all partitions of the objects among the containers not exceeding the capacity of any container

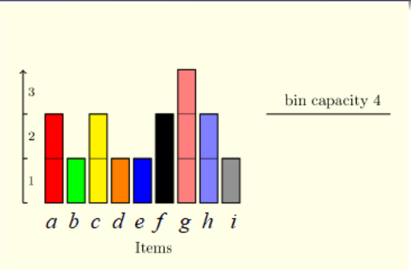
$$X = \left\{ x \subseteq B : |x \cap B_e| = 1 \ \forall e \in E, \sum_{(e,c) \in B^c} v_e \leq V \ \forall c \in C \right\}$$

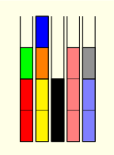
with $B_e = \{(i,j) \in B : i = e\}$ and $B^c = \{(i,j) \in B : j = c\}$

The objective is to minimise the number of containers used

$$\min_{x \in X} f(x) = |\{c \in C : x \cap B^c \neq \emptyset\}|$$

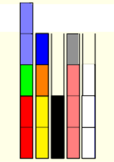
Example





$$x' = \{(a,1),(b,1),(c,2),(d,2),(e,2),(f,3),$$
$$(g,4),(h,5),(i,5)\} \in X$$

$$f(x') = 5$$



$$x'' = \{(a,1),(b,1),(c,2),(d,2),(e,2),(f,3),$$
$$(g,4),(h,1),(i,4)\} \notin X$$

$$f(x'') = 4$$

Partitioning set problems: *Parallel Machine Scheduling Problem* (*PMSP*)

Given

- a set T of tasks
- a function $d : T \to \mathbb{N}$ describing the time length of each task
- a set M of machines

divide the tasks among the machines with the minimum completion time

The ground set $B = T \times M$ includes all (task,machine) pairs

The feasible region includes all partitions of tasks among machines (*the order of the tasks is irrelevant!*)

$$X = \left\{ x \subseteq B : |x \cap B_t| = 1 \; \forall t \in T \right\}$$

The objective is to minimise the maximum sum of time lengths for each machine

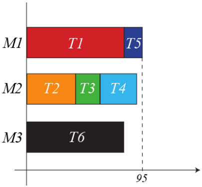
$$\min_{x \in X} f(x) = \max_{m \in M} \sum_{t:(t,m) \in x} d_t$$

Example

$$T = \{T1, T2, T3, T4, T5, T6\}$$

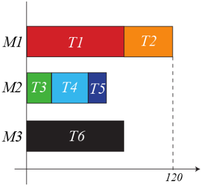
$$M = \{M1, M2, M3\}$$

task	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$
d	80	40	20	30	15	80



$$x' = \{(T1, M1), (T2, M2), (T3, M2), (T4, M2), (T5, M1), (T6, M3)\} \in X$$

$$f(x') = 95$$



$$x'' = \{(T1, M1), (T2, M1), (T3, M2), (T4, M2), (T5, M2), (T6, M3)\} \in X$$

$$f(x'') = 120$$

Interlude 2: the objective function again

The objective function of the *BPP* and the *PMSP*

- is not additive
- is not trivial to compute (*but not hard, as well*)

Small changes in the solution have a variable impact on the objective

- equal to the time length of the moved tasks (increase or decrease) (e.g., move $T5$ on $M1$ in x'')
- zero (e.g., move $T5$ on $M3$ in x'')
- intermediate (e.g., move $T2$ on $M2$ in x'')

In fact, the impact of a change to the solution depends

- both on the modified elements
- and on the unmodified elements (*contrary to Interlude 1*)

The objective function is "flat": several solutions have the same value

(*this is a problem when comparing different modifications*)

Logic function problems: *Max-SAT* problem

Given a *CNF*, assign truth values to its logical variables so as to satisfy the maximum weight subset of its logical clauses

- a set V of logical variables x_j with values in $\mathbb{B} = \{0, 1\}$ (*false*, *true*)
- a literal ℓ_j is a function consisting of an affirmed or negated variable

$$\ell_j(x) \in \{x_j, \bar{x}_j\}$$

- a logical clause is a disjunction or logical sum (*OR*) of literals

$$C_i(x) = \ell_{i,1} \vee \ldots \vee \ell_{i,n_i}$$

- a conjunctive normal form (*CNF*) is a conjunction or logical product (*AND*) of logical clauses

$$CNF(x) = C_1 \wedge \ldots \wedge C_n$$

- to satisfy a logical function means to make it assume value 1
- a function w provides the weights of the *CNF* clauses

Logic function problems: *Max-SAT* problem

The ground set is the set of all simple truth assignments

$$B = V \times \mathbb{B} = \{(x_1, 0), (x_1, 1), \ldots, (x_n, 0), (x_n, 1)\}$$

The feasible region includes all subsets of simple assignments that are

- complete, that is include at least a literal for each variable
- consistent, that is include at most a literal for each variable

$$X = \{x \subseteq B : |x \cap B_v| = 1 \ \forall v \in V\}$$

with $B_{x_j} = \{(x_j, 0), (x_j, 1)\}$

The objective is to maximise the total weight of the satisfied clauses

$$\max_{x \in X} f(x) = \sum_{i : C_i(x) = 1} w_i$$

Example

- Variables

$$V = \{x_1, x_2, x_3, x_4\}$$

- Literals

$$L = \{x_1, \bar{x}_1, x_2, \bar{x}_2, x_3, \bar{x}_3, x_4, \bar{x}_4\}$$

- Logical clauses

$$C_1 = \bar{x}_1 \vee x_2 \qquad \ldots \qquad C_7 = x_2$$

- Conjunctive normal form

$$CNF = (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_4) \wedge (\bar{x}_2 \vee \bar{x}_4) \wedge x_1 \wedge x_2$$

- Weight function (uniform):

$$w_i = 1 \qquad i = 1, \ldots, 7$$

$x = \{(x_1, 0), (x_2, 0), (x_3, 1), (x_4, 1)\}$ satisfies $f(x) = 5$ clauses out of 7

Complementing a variable does not always change $f(x)$ (x_1 does, x_4 not)

Numerical matrix problems: Set Covering (SCP)

Given

- a binary matrix $A \in \mathbb{B}^{m,n}$ with row set R and column set C
- column $j \in C$ covers row $i \in R$ when $a_{ij} = 1$
- a function $c : C \to \mathbb{N}$ provides the cost of each column

Select a subset of columns covering all rows at minimum cost

The ground set is the set of columns: $B = C$

The feasible region includes all subsets of columns that cover all rows

$$X = \left\{ x \subseteq B : \sum_{j \in x} a_{ij} \geq 1 \ \forall i \in R \right\}$$

The objective is to minimise the total cost of the selected columns

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Example

$$c \quad \boxed{\begin{array}{cccccc} 4 & 6 & 10 & 14 & 5 & 6 \end{array}}$$

$$A \quad \left[\begin{array}{cccccc} 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{array}\right]$$

$$A \quad \left[\begin{array}{cccccc} 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{array}\right] \begin{array}{c} 2 \\ 1 \\ 1 \\ 1 \\ 3 \end{array}$$

$$x' = \{c_1, c_3, c_5\} \in X$$

$$f(x') = 19$$

$$A \quad \left[\begin{array}{cccccc} 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{array}\right] \begin{array}{c} 1 \\ 0 \\ 2 \\ 2 \\ 2 \end{array}$$

$$x'' = \{c_1, c_5, c_6\} \notin X$$

$$f(x'') = 15$$

"Set Covering": covering a set (rows) with subsets (columns)

Interlude 3: the feasibility test

Heuristic algorithms often require to solve the following problem

Given a subset x, is x feasible or not? In short, $x \in X$?

It is a decision problem

The feasibility test requires to compute from the solution and test

- a single number: the total volume (*KP*), the cardinality (*MDP*)
- a single set of numbers: values assigned to each variable (*Max-SAT*), number of machines for each task (*PMSP*)
- several sets of numbers: number of containers for each object and total volume of each container (*BPP*)

The time required can be different if the test is performed

- from scratch on a generic subset x
- on a subset x' obtained slightly modifying a feasible solution x

Some modifications can be forbidden *a priori* to avoid infeasibility (insertions and removals for *MDP*, *PMSP*, *Max-SAT*), while others require an *a posteriori* test (exchanges)

Numerical matrix problems: Set Packing

Given

- a binary matrix $A \in \mathbb{B}^{m,n}$ with row set R and column set C
- columns j' e $j'' \in C$ conflict with each other when $a_{ij'} = a_{ij''} = 1$
- a function $\phi : C \to \mathbb{N}$ provides the value of each column

Select a subset of nonconflicting columns of maximum value

The ground set is the set of columns: $B = C$

The feasible region includes all subsets of nonconflicting columns

$$X = \left\{ x \subseteq B : \sum_{j \in x} a_{ij} \leq 1 \ \forall i \in R \right\}$$

The objective is to maximise the total value of the selected columns

$$\max_{x \in X} f(x) = \sum_{j \in x} \phi_j$$

Example

ϕ	4	6	10	14	5	6

$$A = \begin{array}{|cccccc|} 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

$$A = \begin{array}{|cccccc|c} 0 & \color{red}{1} & 0 & \color{red}{0} & 1 & 0 & \color{red}{1} \\ 0 & \color{red}{0} & 1 & \color{red}{1} & 0 & 0 & \color{red}{1} \\ 1 & \color{red}{0} & 0 & \color{red}{0} & 0 & 1 & \color{red}{0} \\ 0 & \color{red}{0} & 0 & \color{red}{1} & 1 & 1 & \color{red}{1} \\ 1 & \color{red}{1} & 1 & \color{red}{0} & 0 & 0 & \color{red}{1} \end{array}$$

$$x' = \{c_2, c_4\} \in X$$

$$f(x') = 20$$

$$A = \begin{array}{|cccccc|c} \color{red}{0} & 1 & 0 & 0 & \color{red}{1} & \color{red}{0} & \color{red}{1} \\ \color{red}{0} & 0 & 1 & 1 & \color{red}{0} & \color{red}{0} & \color{red}{0} \\ \color{red}{1} & 0 & 0 & 0 & \color{red}{0} & \color{red}{1} & \color{red}{2} \\ \color{red}{0} & 0 & 0 & 1 & \color{red}{1} & \color{red}{1} & \color{red}{2} \\ \color{red}{1} & 1 & 1 & 0 & \color{red}{0} & \color{red}{0} & \color{red}{1} \end{array}$$

$$x'' = \{c_1, c_5, c_6\} \notin X$$

$$f(x'') = 15$$

"Set Packing": packing disjoint subsets (columns) of a set (rows)

Numerical matrix problems: Set Partitioning (SPP)

Given

- a binary matrix $A \in \mathbb{B}^{m,n}$ with a set of rows R and a set of columns C
- a function $c : C \to \mathbb{N}$ that provides the cost of each column

select a minimum cost subset of nonconflicting columns covering all rows

The ground set is the set of columns: $B = C$

The feasible region includes all subsets of columns that cover all rows and are not conflicting

$$X = \left\{ x \subseteq B : \sum_{j \in x} a_{ij} = 1 \; \forall i \in R \right\}$$

The objective is to minimise the total cost of the selected columns

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Example

c	4	6	10	14	5	6

$$
A \quad \begin{array}{|cccccc|}
\hline
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 \\
\hline
\end{array}
$$

$$
A \quad \begin{array}{|cccccc|c}
\cline{1-6}
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 \\
\cline{1-6}
\end{array}
$$

$x' = \{c_2, c_4, c_6\} \in X$

$f(x') = 26$

$$
A \quad \begin{array}{|cccccc|c}
\cline{1-6}
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 2 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 \\
\cline{1-6}
\end{array}
$$

$x'' = \{c_1, c_5, c_6\} \notin X$

$f(x'') = 15$

"Set Partitioning": partition a set (rows) into subsets (columns)

Graph problems: Travelling Salesman Problem (*TSP*)

Given

- a directed graph $G = (N, A)$
- a function $c : A \to \mathbb{N}$ that provides the cost of each arc

select a circuit visiting all the nodes of the graph at minimum cost

The ground set is the arc set: $B = A$

The feasible region includes the circuits that visit all nodes in the graph (Hamiltonian circuits)

How to determine whether a subset is a feasible solution?

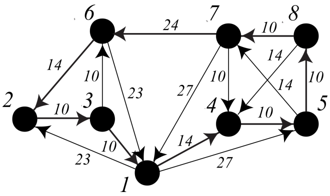
And a modification of a feasible solution?

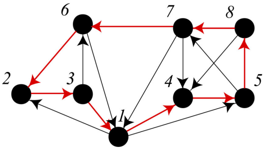
Can we rule out some modifications?

The objective is to minimise the total cost of the selected arcs

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

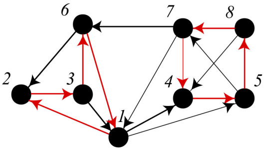
Example





$$x' = \{(1,4),(4,5),(5,8),(8,7),(7,6),(6,2),(2,3),(3,1)\} \in X$$

$$f(x') = 102$$



$$x'' = \{(4,5),(5,8),(8,7),(7,4),(1,2),(2,3),(3,6),(6,1)\} \notin X$$

$$f(x'') = 106$$

Interlude 4: the search for feasible solutions

Heuristic algorithms often require to solve the following problem

Find a feasible solution $x \in X$

It is a search problem

The search for a feasible solution is trivial or easy for some problems:

- some sets are always feasible, such as $x = \emptyset$ (*KP*, *Set Packing*) or $x = B$ (feasible instances of *SCP*)
- random subsets satisfying a constraint, such as $|x| = k$ (*MDP*)
- random subsets satisfying consistency constraints, such as assigning one task to each machine (*PMSP*), one value to each logic variable (*Max-SAT*), etc. . .

Interlude 4: the search for feasible solutions

But it is hard for other problems:

- in the *BPP* the problem is easy if the number of containers is large (e. g., one container for each object)
- in the *SPP* no polynomial algorithm is known to solve the problem
- in the *TSP* the problem is easy for dense graphs (e. g., complete)

One can apply a relaxation, i. e. enlarge the feasible region from X to X'

- the objective f must be extended from X to X' (*see Interlude 1*)
- but often $X' \setminus X$ includes better solutions (*. . . how about that?*)

Graph problems: Vertex Cover (VCP)

Given an undirected graph $G = (V, E)$, select a subset of vertices of minimum cardinality such that each edge of the graph is incident to it

The ground set is the vertex set: $B = V$

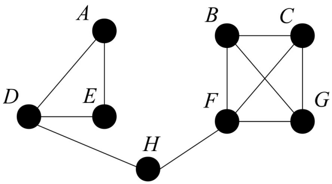
The feasible region includes all vertex subsets such that all the edges of the graph are incident to them

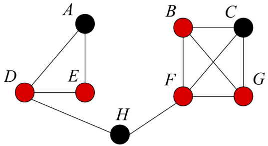
$$X = \left\{ x \subseteq V : x \cap (i,j) \neq \emptyset \, \forall (i,j) \in E \right\}$$

The objective is to minimise the number of selected vertices

$$\min_{x \in X} f(x) = |x|$$

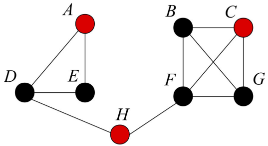
Example





$$x' = \{B, D, E, F, G\} \in X$$

$$f(x') = 5$$



$$x'' = \{A, C, H\} \notin X$$

$$f(x'') = 3$$

Given

- an undirected graph $G = (V, E)$
- a function $w : V \to \mathbb{N}$ that provides the weight of each vertex

select the subset of pairwise adjacent vertices of maximum weight

The ground set is the vertex set: $B = V$

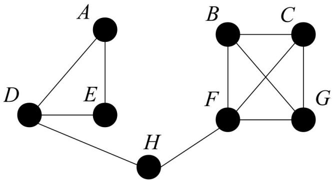
The feasible region includes all subsets of pairwise adjacent vertices

$$X = \left\{x \subseteq V : (i, j) \in E \ \forall i \in x, \forall j \in x \setminus \{i\}\right\}$$

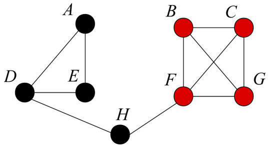
The objective is to maximise the weight of the selected vertices

$$\max_{x \in X} f(x) = \sum_{j \in x} w_j$$

Example

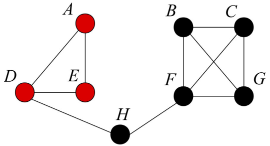


Uniform weights: $w_i = 1$ for each $i \in V$



$x' = \{B, C, F, G\} \in X$

$f(x') = 4$



$x'' = \{A, D, E\} \in X$

$f(x'') = 3$

Graph problems: Maximum Independent Set Problem

Given

- an undirected graph $G = (V, E)$
- a function $w : V \rightarrow \mathbb{N}$ that provides the weight of each vertex

select the subset of pairwise nonadjacent vertices of maximum weight

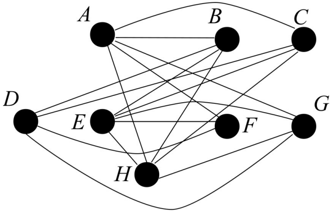
The ground set is the vertex set: $B = V$

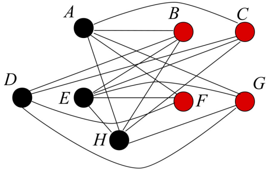
The feasible region includes the subsets of pairwise nonadjacent vertices

$$X = \{x \subseteq B : (i,j) \notin E \; \forall i \in x, \forall j \in x \setminus \{i\}\}$$

The objective is to maximise the weight of the selected vertices

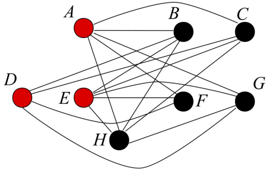
$$\max_{x \in X} f(x) = \sum_{j \in x} w_j$$





$x' = \{B, C, F, G\} \in X$

$f(x') = 4$



$x'' = \{A, D, E\} \in X$

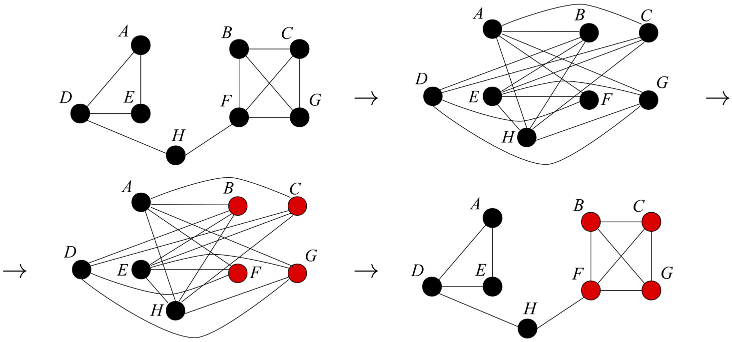
$f(x'') = 3$

Interlude 5: the relations between problems (1)

Each instance of the *MCP* is equivalent to an instance of the *MISP*

1. start from the *MCP* instance, that is graph $G = (V, E)$
2. build the complementary graph $\bar{G} = (V, (V \times V) \setminus E)$
3. find an optimal solution of the *MISP* on $\bar{G}$
4. the corresponding vertices give an optimal solution of the *MCP* on G

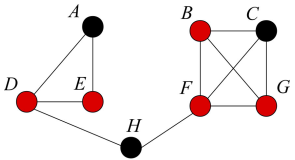
(*a heuristic MISP solution gives a heuristic MCP solution*)



The process can be applied also in the opposite direction

The *VCP* and the *SCP* are also related, but in a different way; each instance of the *VCP* can be transformed into an instance of the *SCP*:

- each edge i corresponds to a row of the covering matrix A
- each vertex j corresponds to a column of A
- if edge i touches vertex j, set $a_{ij} = 1$; otherwise $a_{ij} = 0$
- an optimal solution of the *SCP* gives an optimal solution of the *VCP* (*a heuristic SCP solution gives a heuristic VCP solution*)



	A	B	C	D	E	F	G	H
(A, D)	1	0	0	1	0	0	0	0
(A, E)	1	0	0	0	1	0	0	0
(B, C)	0	1	1	0	0	0	0	0
(B, F)	0	1	0	0	0	1	0	0
(B, G)	0	1	0	0	0	0	1	0
(C, F)	0	0	1	0	0	1	0	0
(C, G)	0	0	1	0	0	0	1	0
(D, E)	0	0	0	1	1	0	0	0
(D, H)	0	0	0	1	0	0	0	1
(F, G)	0	0	0	0	0	1	1	0
(F, H)	0	0	0	0	0	1	0	1

It is not simple to do the reverse

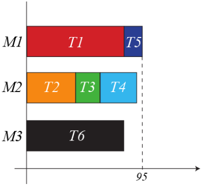
The *BPP* and the *PMSP* are equivalent, but in a more sophisticated way:

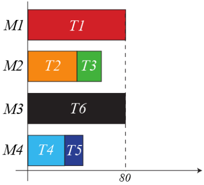
- the tasks correspond to the objects
- the machines correspond to the containers, but
    - *BPP*: minimise the number of containers, given the capacity
    - *PMSP*: given the number of machines, minimise the completion time

Start from a *BPP* instance

1. make an assumption on the optimal number of containers (e.g., 3)
2. build the corresponding *PMSP* instance
3. compute the optimal completion time (e.g., 95)
    - if it exceeds the capacity (e.g., 80), increase the assumption (4 or 5)
    - if it does not, decrease the assumption (2 or 1)

(*using heuristic PMSP solutions leads to a heuristic BPP solution*)





The reverse process is possible

The two problems are equivalent, but each one must be solved several times

Graph problems: Capacitated Min. Spanning Tree Problem

Given

- an undirected graph $G = (V, E)$ with a root vertex $r \in V$
- a function $c : E \to \mathbb{N}$ that provides the cost of each edge
- a function $w : V \to \mathbb{N}$ that provides the weight of each vertex
- a number $W \in \mathbb{N}$ that is the subtree appended to the root (branch)

select a spanning tree of minimum cost such that each branch respects the capacity

The ground set is the edge set: $B = E$

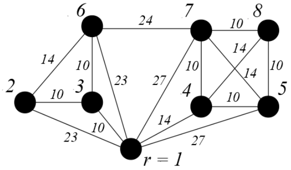
The feasible region includes all spanning trees such that the weight of the vertices spanned by each branch does not exceed W

The feasibility test requires to visit the subgraph

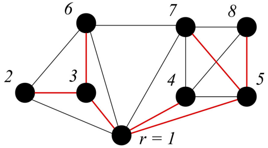
The objective is to minimise the total cost of the selected edges

$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Example

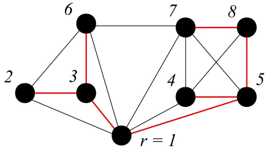


Uniform weight ($w_i = 1$ for each $i \in V$) and capacity: $W = 3$



$$x' = \{(r,3),(3,2),(3,6),(r,4),\\ (r,5),(5,7),(5,8)\} \in X$$

$$f(x') = 95$$



$$x'' = \{(r,3),(3,2),(3,6),(r,5),\\ (5,4),(5,8),(8,7)\} \notin X$$

$$f(x'') = 87$$

It is easy to evaluate the objective, less easy the feasibility

Cost of the main operations

The objective function is

- fast to evaluate: sum the edge costs
- fast to update: sum the added costs and subtract the removed ones

but it is easy to obtain subtrees that span vertices in a nonoptimal way

The feasibility test is

- not very fast to perform:
  - visit to check for connection and acyclicity
  - visit to compute the total weight of each subtree
- not very fast to update:
  - show that the removed edges break the loops introduced by the added ones
  - recompute the weights of the subtrees

This also holds when the graph is complete

What if we described the problem in terms of vertex subsets?

An alternative description

Define a set of branches T *(as the containers in the BPP)*

One for each vertex in $V \setminus \{r\}$: some can be empty

The ground set is the set of the (vertex,branch) pairs: $B = V \times T$

The feasible region includes all partitions of the vertices into connected subsets (*visit, trivial on complete graphs*) of weight $\leq W$ (*as in the BPP*)

$$X = \left\{ x \subseteq B : |x \cap B_v| = 1 \; \forall v \in V \setminus \{r\}, \sum_{(i,j) \in B^t} w_i \leq W \; \forall t \in T, \ldots \right\}$$

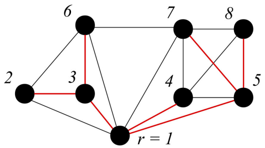
with $B_v = \{(i,j) \in B : i = v\}$, $B^t = \{(i,j) \in B : j = t\}$

The objective is to minimise the sum of the costs of the branches spanning each subset of vertices and appending it to the root

It is a combination of minimum spanning tree problems

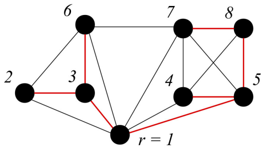
Example

The previously considered solutions now have a different representation



$$x' = \{(2, T1), (3, T1), (6, T1), (4, T2), (5, T3), (7, T3), (8, T3)\} \in X$$

$$f(x') = 95$$



$$x'' = \{(2, T1), (3, T1), (6, T1), (4, T2), (5, T2), (7, T2), (8, T2)\} \notin X$$

$$f(x'') = 87$$

The feasibility test only requires to sum the weights,
computing the objective requires to solve a MST problem

Cost of the main operations

The objective function is

- slow to evaluate: compute a MST for each subset
- slow to update: recompute the MST for each modified subset

but the subtrees are optimal by construction

If the graph is complete, the feasibility test is

- fast to perform:
  - sum the weights of the vertices for each subtree
- fast to update:
  - sum the added weights and subtract the removed ones

Advantages and disadvantages switched places

Graph problems: Vehicle Routing Problem (*VRP*)

Given

- a directed graph $G = (N, A)$ with a depot node $d \in N$
- a function $c : A \to \mathbb{N}$ that provides the cost of each arc
- a function $w : N \to \mathbb{N}$ that provides the weight of each node
- a number $W \in \mathbb{N}$ that is the capacity of each circuit

select a set of circuits of minimum cost such that each one visits the depot and respects the capacity

The ground set could be

- the arc set: $B = A$
- the set of all (node,circuit) pairs: $B = N \times C$

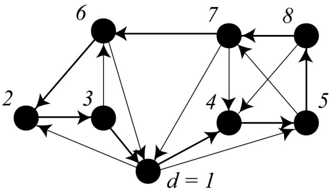
The feasible region could include

- all arc subsets that cover all nodes with circuits visiting the depot and whose weight does not exceed W (*again the visit of a graph*)
- all partitions of the nodes into subsets of weight non larger than W and admitting a spanning circuit ($\mathcal{NP}$-*hard problem!*)

The objective is to minimise the total cost of the selected arcs

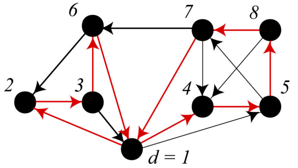
$$\min_{x \in X} f(x) = \sum_{j \in x} c_j$$

Example



Uniform weight ($w_i = 1$ for each $i \in N$) and capacity: $W = 4$

The solutions could be described as



- arc subsets
  $$x = \{(d,2),(2,3),(3,6),(6,d),(d,4),(4,5),(5,8),(8,7),(7,d)\} \in X$$
- node partitions
  $$x = \{(2,C1),(3,C1),(6,C1),(4,C2),(5,C2),(7,C2),(8,C2)\} \in X$$

$$f(x) = 137$$

Interlude 6: combining alternative representations

The *CMSTP* and the *VRP* share an interesting complication:
different definitions of the ground set B are possible and natural

- the description as a set of edges/arcs
  looks preferable to manage the objective
- the description as a set of pairs (vertex,tree)/(node/circuit) looks better to generate optimal solutions and to deal with feasibility

Which description should be adopted?

- the one that makes easier the most frequent operations
- both, if they are used much more frequently than updated, so that the burden of keeping them up-to-date and consistent is acceptable

Homework

Answer all the fundamental questions on all the considered problems

1. Objective function:
   a) What is the cost of computing $f(x)$ given x?
   b) Is $f(x)$ additive, quadratic, etc...?
   a) What is the cost of computing $f(x')$
   given $f(x)$ and a "small" transformation $x \to x'$?
   c) Is $f(x)$ "flat"?
2. Feasibility:
   a) What is the cost of testing whether subset x is a feasible solution?
   b) What is the cost of testing whether subset x' is a feasible solution
   given a feasible solution x and a "small" transformation $x \to x'$?
   c) Are some transformations intrinsically feasible (or unfeasible)?
   d) Is it easy to find a feasible solution?
   Is there a subset that is always feasible?
3. Relations between problems:
   a) Are there trasformations from/to the problem to/from other ones?
4. Ground sets:
   a) Are there alternative definitions of the ground set?
   b) What are their relative advantages and disadvantages?