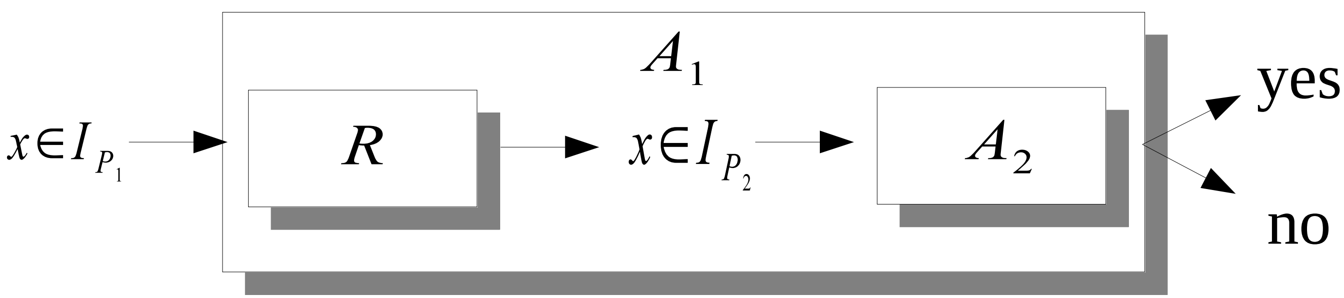
The Complexity of Optimization Problems

Summary Lecture 02

- Reducibility
  - Karp reducibility & Turing reducibility
  - NP-complete problems
- Complexity of optimization problems
  - Classes PO and NPO
  - NP-hard optimization problems

Karp reducibility

- A decision problem P_1 is *Karp reducible* to a decision problem P_2 (in short, $P_1 \leq_K P_2$) if there exists a polynomial-time computable function R such that, for any x, x is a YES-instance of P_1 if and only if $R(x)$ is a YES-instance of P_2



Karp reducibility

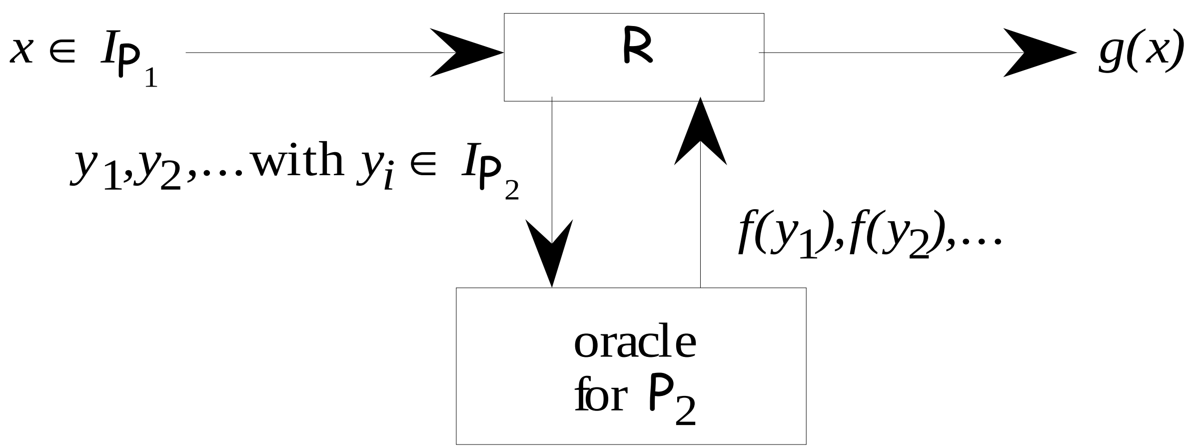
- A computational class C is *closed* with respect to *Karp reducible* if and only if, given two decision problem P_1, P_2

$$(P_1 \leq_K P_2) \wedge (P_2 \in C) \Rightarrow P_1 \in C$$

- $\leq_K$ is a reflexive, transitive, partial relation
- (C, $\leq_K$) is a partial preorder
  - In a partial preorder there can be maximum elements (unless of equivalence relation)
- $P_1 \leq_K P_2$ and P_2 is in P, then P_1 is in P

Turing reducibility

- Let P_1 a problem of computing a function $g: I_{P_1} \to S_{P_1}$ and P_2 a problem of computing a function $f: I_{P_1} \to S_{P_1}$
- P_1 is *Turing reducible* to P_2 if there exists a polynomial-time algorithm R solving P_1 such that R may access to an *oracle* algorithm solving P_2



Turing reducibility

- A Turing reducibility is denoted by $P_1 \leq_T P_2$
- A Karp-reducibility is just a particular case of Turin-reducibility: $P_1 \leq_K P_2 \Rightarrow P_1 \leq_T P_2$
  - problems P_1, P_2 are decision problem,
  - the oracle for P_2 can be queried just once,
  - and R returns the same value answered by oracle

Turing reducibility

- A computational class C is *closed* with respect to *Turing reducible* if and only if, given two decision problem P_1, P_2

$$(P_1 \leq_T P_2) \wedge (P_2 \in C) \Rightarrow P_1 \in C$$

- $\leq_T$ is a reflexive, transitive, partial relation
- $(C, \leq_T)$ is a partial preorder
  - In a partial preorder there can be maximum elements (unless of equivalence relation)

Complete problems

- For any complexity class C, a decision problem $P \in C$ is said to be *complete* in C (*C-complete)* with respect to a reducibility $\leq_r$ if, for any other decision problem $P_1 \in C$, $P_1 \leq_r P$
  - Two problems P_1, P_2 C-complete w.r.t. $\leq_r$ are equivalents $P_1 \equiv_r P_2$
  - Two classes C, C', closed, such that $C' \subset C$: a problem P_1 C-complete has to be in C-C'
  - The best approach to study if $C' \subseteq C$ are different is to study C-complete problems

NP-complete problems

- A decision problem P is *NP-complete* if $P \in$ NP and, for any decision problem $P_1 \in$ NP, $P_1 \leq_K P$
- If P is NP-complete and $P \in$ P, then P=NP
  - NP-complete problems are the hardest in NP
  - P versus NP question can be solved by focusing on an NP-complete problem

- Cook's Theorem: SAT is NP-complete

Optimization problem

- Optimization problem P characterized by
  - Set of instances *I*
  - Function SOL that associates to any instance the set of feasible solutions
  - Measure function *m* that, for any feasible solution of an instance, provides its positive integer value
  - Goal, that is, either MAX or MIN
- An optimal solution is a feasible solution y^* such that

$$m(x,y^*) = \text{Goal}\{m(x,y) \mid y \in \text{SOL}(x)\}$$

- For any instance x, $m^*(x)$ denotes optimal measure

MINIMUM VERTEX COVER

- INSTANCE: Graph $G=(V,E)$

- SOLUTION: A subset U of V such that, for any edge (u,v), either u is in U or v is in U

- MEASURE: Cardinality of U

- The goal of the problem is usually given by the name of problem

Three problems in one

- **Constructive problem (P_C)**: given an instance, compute an optimal solution and its value
  - We will study these problems
- **Evaluation problem (P_E)**: given an instance, compute the optimal value
- **Decision problem (P_D)**: given an instance and an integer *k*, decide whether the optimal value is at least (if Goal=MAX) or at most (if Goal=MIN) *k*

Class NPO

- Optimization problems such that
  - I is recognizable in polynomial time
  - Solutions are polynomially bounded and recognizable in polynomial time: $y \in \mathrm{SOL}(x) \Rightarrow |y| \leq q(|x|)$, $\forall y$ *s.t.* $|y| \leq q(|x|)$, it is decidable in polynomial time if $y \in \mathrm{SOL}(x)$
  - m is computable in polynomial time
- Example: MINIMUM VERTEX COVER
- **Theorem** : If P is in NPO, then the corresponding decision problem is in NP

Class PO

- NPO problems solvable in polynomial time.
  - There exists a polynomial-time computable algorithm A that, for any instance $x \in I_P$, returns an optimal solution $y \in$ SOL*(x), together with its value $m^*(x)$
- **Fact** : If P is in PO, then the corresponding decision problem is in P

MINIMUM PATH

- INSTANCE: Graph $G=(V,E)$, two nodes v_s, $v_t \in V$
- SOLUTION: A path $(v_s, v_{i1}, v_{i2}, \ldots, v_t)$ from v_s to v_t.
- MEASURE: The number of edges in the path

- The problem is solvable in polynomial time by a breadth-first search algorithm, that finds all minimum paths from all nodes to v_t

Classes NPO and PO

- PO $\subseteq$ NPO
- Practically all interesting optimization problems belong to the class NPO
  - Graphs problems (MINIMUM TRAVELLING SALESPERSON, MINIMUM GRAPH COLORING
  - Packing & scheduling problems
  - Integer & binary linear programming
- The question PO=NPO is strictly related to P=NP

NP-hard problem

- An optimization problem P is NP-hard if any decision problem in NP is Turing reducible to P:

$$\forall P_1 \in \mathrm{NP},\ P_1 \leq_T P$$

- **Theorem**: If the decision problem corresponding to a NPO problem P is NP-complete, then P is NP-hard
  - Example: MINIMUM VERTEX COVER

- **Corollary**: If P $\neq$ NP then PO $\neq$ NPO

Evaluating versus constructing

- Decision problem is Turing reducible to evaluation problem
- Evaluation problem is Turing reducible to constructive problem
- Evaluation problem is Turing reducible to decision problem
  - Binary search on space of possible measure values
- Is constructive problem Turing reducible to evaluation (decision) problem?

MAXIMUM SATISFIABILITY

- INSTANCE: CNF Boolean formula, that is, set C of clauses over set of variables V

- SOLUTION: A truth-assignment f to V

- MEASURE: Number of satisfied clauses

Evaluating versus constructing: MAX SAT

```
begin
        for each variable v
        begin
                k := MAX SAT_eval(x);
                x_TRUE := formula obtained by setting v to TRUE in x;
                x_FALSE := formula obtained by setting v to FALSE in x;
                if MAX SAT_eval(x_TRUE) = k then
                begin
                        f(v) := TRUE; x := x_TRUE
                end
                else
                begin
                        f(v) := FALSE; x := x_FALSE
                end;
        return f
end
```

Evaluating versus constructing

Theorem: if the decision problem is NP-complete,

then the constructive problem is Turing reducible to the decision problem

proof

Let P a maximization problem.

We derive a NPO problem P' s.t. $P_C \leq_T P'_{D}$*, since* P_D *is NP-complete,* $P'_D \leq_T P_D$*, we have the theorem.*

P' is the same of P except for the measure definition $m_{P'}$*.*

Evaluating versus constructing

Let p() a polynomial s.t. $y \in SOL_P(x) \Rightarrow |y| \leq p(|x|)$,

Let $\lambda(y)$ *the rank of y in the lexicographic order.*

For any instance $x \in I_{P'} = I_P$ *and for any* $y \in SOL_{P'}(x) = SOL_P(x)$

$$\text{let } m_{P'}(x, y) = 2^{p(|x|)+1} m_P(x,y) + \lambda(y)$$

Every solution y has a unique value $m_{P'}$.

Therefore there exists a unique optimal solution $y^*_{P'}(x)$ *in* $SOL^*_{P'}(x)$.

$y^*_{P'}(x) \in SOL^*_P(x)$ *too.*

$y^*_{P'}(x)$ *can be derived polynomial time by means of oracle for* P'_E*: the position of* $y^*_{P'}(x)$ *in the order can be derived by computing the remainder of division between* $m^*_{P'}(x)$ *and* $2^{p(|x|)+1}$.

Evaluating versus constructing

P'_D can be used to simulate P'_E in polynomial time.

Therefore the optimal solution of P can be derived in polynomial time using an oracle for P'_D.

Since $P'_D \in NP$, and P_D is NP-complete, an oracle for P_D can be used to simulate the oracle for P'_D

Evaluating versus constructing

Open question: is there a NPO problem whose constructive version is harder than the evaluation version?

A possible answer is in P. Crescenzi & R. Silvestri "*Relative complexity of evaluating the optimum cost and constructing the optimum for maximization problems*" *IPL 33, pag. 221-226 (1990)*

Exercise

1. Recall that a disjunctive normal formula is a collection of conjunctions and it is satisfied by a truth assignment if and only if at least one conjuction is satisfied. Show that the problem SAT of DNF is in co-NP.
2. Prove that VERTEX COVER is NP-complete.
3. Prove that 2-COLORING is in P.