UNIVERSITÀ DEGLI
STUDI DI TRENTO

ICT DEPARTMENT

On the computational complexity of enumerating certificates of NP problems

Marco Rospocher
PhD Student
International Doctorate School in ICT
Department of Information and Communication Technology
marco.rospocher@unitn.it

ITC Seminars
Oct 18, 2006 - Trento, Italy

Summary

- Introduction.
- A structural complexity theory for listing problems associated to NP relations.
- Listing solutions of a broad class of combinatorial optimization problems.
- Listing satisfying truth assignments of some peculiar classes of boolean formulas (XOR and 2SAT).
- Conclusions.

Historical Introduction

- Historically, computational problems have been considered from a decision perspective.
- Problems associated with binary relations (instance, solution):
  - Decision problem: given x, is there y such that (x,y) belongs to the relation?
  - Search problem: given x, return y (if any) such that (x,y) belongs to the relation.
  - Counting problem: given x, count the number of y such that (x,y) belongs to the relation.
  - Listing problem: given x, return all y (if any) such that (x,y) belongs to the relation.

NP Relations

- A binary relation R is **polynomially balanced** if xRy implies that the length of y is polynomially bounded by the length of x.

- A binary relation R is **polynomially decidable** if there is a polynomial time algorithm which decides whether xRy for each couple (x,y).

- A binary relation R is an **NP relation** if R is both polynomially balanced and polynomially decidable.

NP Relations: an example

- Consider the following relation:

R_{SAT} :={(φ,T) : φ is a CNF boolean formula, T is a satisfying truth assignment for φ}.

- R_{SAT} is an NP relation.

1. R_{SAT} is polynomially balanced: the length of a truth assignment is bounded by the length of the formula;
2. R_{SAT} is polynomially decidable: given a truth assignment T, we can decide in polynomial time if T satisfies φ.

NP Relations

- The language L(R) associated to a relation R is the set of strings x such that there exists a string y with xRy.
- A language L belongs to NP if and only if there exists an NP relation R such that L=L(R).
- The strings y such that xRy are called **certificates** or **witnesses** of yes instance x.
- Hence, we are investigating the complexity of listing, with respect to an NP relation R, all the certificates of a string x of the language L(R).

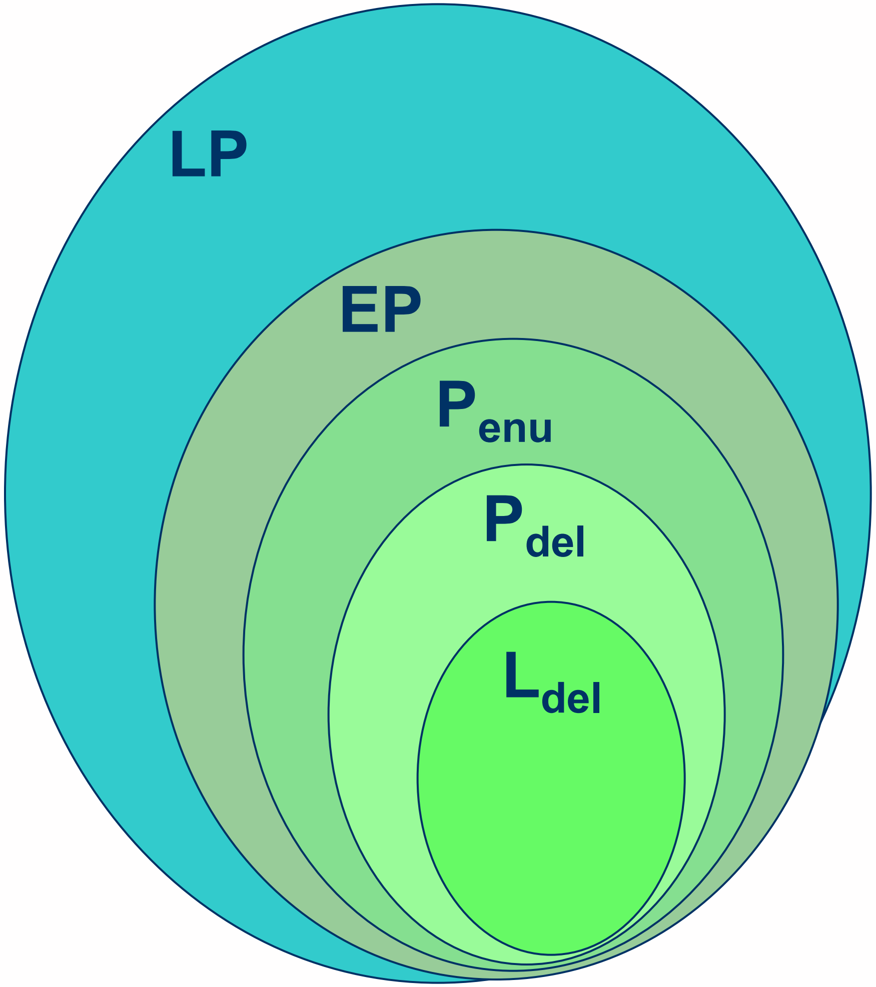
Listing Algorithms

- Return all solutions without duplicates.
- How do we measure efficiency of listing algorithms?
  - **Polynomial total time**: time complexity polynomial in the input size and the output size (the number of solutions);
  - **P-enumerability**: polynomial in the input size and linear in the output size; **strong P-enumerability**, if space used is polynomial in the input size only;
  - **Polynomial (Linear) Delay**: first solution outputted in polynomial time in the input size; delay between two consecutive outputs polynomial (linear) in the input size.

Listing Problems and NP relations

- LP is the class of listing problems associated with NP relations.

- We define some subclasses of LP according to the various notions of efficiency for listing algorithm EP, P_{enu}, P_{del}, L_{del}.



LP-completeness

- We say that a listing problem L is **LP-complete** if:
    1. the listing problem belongs to class LP;
    2. If there exists a polynomial total time algorithm for problem L, then there exists a polynomial total time algorithm for any problem in LP.

- We define **LPC** as the class of LP-complete problems.

Levin Reductions

- Given two relations R_1 and R_2, a **Levin reduction** from R_1 to R_2 is a triplet (f,g,h) of polynomial time computable functions such that:
  1. $x \in L(R_1)$ if and only if $f(x) \in L(R_2)$;
  2. If $(x,y) \in R_1$, then $(f(x),g(x,y)) \in R_2$;
  3. If $(f(x),z) \in R_2$, then $(x,h(x,z)) \in R_1$.

- A Levin reduction implies a Karp reduction.

An LP-complete problem

- Consider relation R_{BH} defined as:

R_{BH}:={((M,x,1^t),y) : M is a deterministic Turing machine which accepts (x,y) within t steps}.

- R_{BH} is an NP relation.
- The decision problem associated to relation R_{BH} is called **Bounded Halting**, and it is an NP-complete problem.

An LP-complete problem

- Lem: There exists a Levin reduction from the generic NP relation R to R_{BH} which preserves the certificates (ie, $(x,y) \in R$ iff $(f(x),y) \in R_{BH}$).

  – R is an NP relation: hence, there exists:

  1. For each $(x,y) \in R$, $|y| \leq p(x)$, where p() is a polynomial;
  2. A Turing machine M_R which decides R in time $q(|x|+|y|)$, where q() is a polynomial.

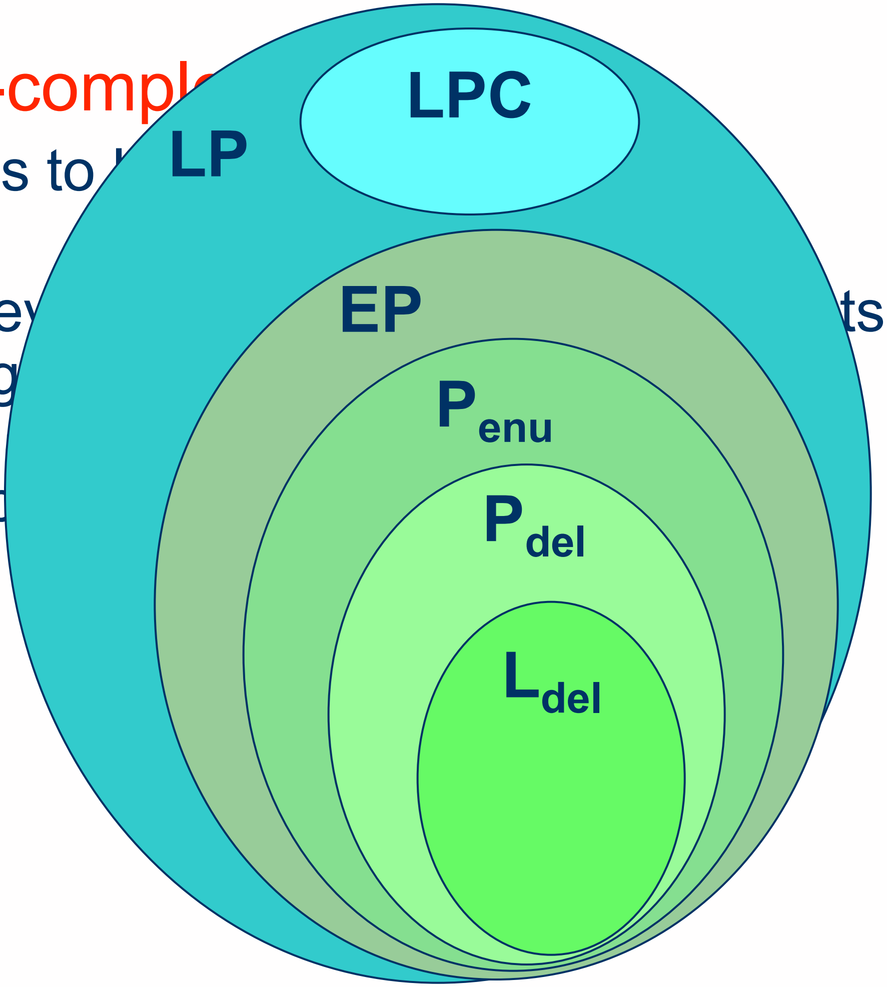
  – We define f,g,h as follows:

  1. $f(x):=(M_R,x,1^{q(|x|+p(x))})$;
  2. $g(x,y):=y$;
  3. $h(x,z):=z$.

$(x,y) \in R$ iff $((M_R,x,1^{q(|x|+p(x))}),y) \in R_{BH}$

An LP-complete problem

- LBounded Halting is LP-compl
  - LBounded Halting belongs to L relation;
  - By the Levin reduction prev ts a polynomial total time alg Halting, then there exists for the generic listing prob



- LPC is not empty!
- LBounded Halting is a strong member of LPC.

More LP-complete problems

- **One-to-one certificates reduction** from R_1 to R_2: is a Levin reduction (f,g,h) from R_1 to R_2 which is:
  - Parsimonious: the number of solutions of instance x of R_1 is the same as the number of solutions of instance f(x) of R_2;
  - the function h which retrieves a certificate y for yes instance x from a certificate z for yes instance f(x) is injective in z.

More LP-complete problems

- Let R_A and R_B be two NP relations. Let LA and LB be the listing problems associated to R_A and R_B respectively. Assume that LA is LP-complete. If there exists a one-to-one certificates reduction (f,g,h) from R_A to R_B, then LB is LP-complete.

- Examples of LP-complete problems due to one-to-one certificates reductions: LSat, LCircuit Sat, LHS, LIP,…

Question

Does the LP-completeness of the listing problem associated to an NP relation imply the NP-completeness of the decision problem associated to the relation?

Easy to decide, hard to list

- A **monotone boolean formula** does not contain any negation symbol.
- An **implicant** of a boolean formula is a subset of variables such that setting these variables to 1, the formula is satisfied whatever value is assigned to the remaining variables.
- A **prime implicant** is a minimal inclusionwise implicant.
- Example of monotone boolean formula: $(a \vee b) \wedge (a \vee c) \wedge (b) \wedge (d \vee b)$.
- {a,b,d} is an implicant, {b,c} is a prime implicant.

Easy to decide, hard to list

- Consider relation R_{PI} defined as:

R_{PI} :={(φ,I) : φ is a monotone boolean formula, I is a prime implicant for φ}.

- R_{PI} is an NP relation.
- OBS1: every monotone boolean formula admits a prime implicant (ie, the decision problem associated to R_{PI} is polynomial time solvable).
- OBS2: a prime implicant of a monotone boolean formula can be obtained in polynomial time applying a greedy strategy (ie, the search problem associated to R_{PI} is polynomial time solvable).

Easy to decide, hard to list

- TEO: **LPrime Implicants**, the problem of listing all prime implicants of a monotone boolean formula is LP-complete.
- There exists no polynomial total time algorithm for listing all prime implicants of a monotone boolean formula unless P=NP (Goldberg 1991).
- TEO: There exists no polynomial total time algorithm for an LP-complete listing problem unless P=NP.

End of first part…

- Introduction.
- A structural complexity theory for listing problems associated to NP relations.
- Listing solutions of a broad class of combinatorial optimization problems.
- Listing satisfying truth assignments of some peculiar classes of boolean formulas (XOR and 2SAT).
- Conclusions.

Combinatorial Ensembles

- A **combinatorial ensemble** is a family of couples (S,F), where S, called the ground set, is a finite set of elements, and F is a collection of subsets of S (feasible solutions). We assume F to be given implicitly by a compact representation.
- Examples:
  - Matching ensemble: S is the set of edges of a graph G, F is the family of matchings of G, G is a compact representation of F;
  - Truth assignments ensemble: S is the set of variables of a boolean formula φ, F is the family of satisfying truth assignments of φ, φ is a compact representation of F.

Combinatorial Ensembles

- Problems investigated on a combinatorial ensemble:
  - Decision problem: does a feasible solutions exist?
  - Search problem: find a feasible solution if any.
  - Listing problem: find all feasible solutions in F.

- To every instance of a combinatorial ensemble we can associate a 0/1-polytope $P_{(S,F)}$ in R^S whose vertices are in one-to-one and onto correspondence with feasible solutions of the instance.

$S=\{s_1,s_2,s_3,s_4,s_5\}, \quad \{s_1,s_4,s_5\} \in F \quad \leftrightarrow \quad (1,0,0,1,1)$

Combinatorial Ensembles

1. We say that a combinatorial ensemble has a **compact description** if the description of $P_{(S,F)}$ in terms of inequalities can be obtained in time polynomial in the size of the instance.
2. We say that a combinatorial ensemble is **separable** if we have a **separation algorithm** for $P_{(S,F)}$, that is, a polynomial time algorithm that, given a rational vector z in R^S, tests if z belongs to $P_{(S,F)}$ or, if not, returns a rational vector c in R^S such that cx < cz for each x in $P_{(S,F)}$.

- Clearly, (1) implies (2), but (2) does not imply (1).

Combinatorial Ensembles: Results

- Bussieck and Lübbecke (1997) showed that if a combinatorial ensemble has a compact description, then there exists a polynomial space polynomial delay algorithm for listing all feasible solutions for any given instance.

- We show that if a combinatorial ensemble is separable, then there exists a polynomial space polynomial delay algorithm for listing all feasible solutions for any given instance.

Combinatorial Optimization Problem

- To each element e of the ground set is assigned a weight w_e.
- The value of a feasible solution is defined as the sum of the weights of the elements in it.
- In the combinatorial optimization problem associated to a combinatorial ensemble, the goal is to find a minimum value feasible solution.

Separable Combinatorial Ensembles

- If a combinatorial ensemble is separable, then there exists a polynomial space polynomial delay algorithm that, for any instance (S,F):
  1. lists all feasible solutions;
  2. lists all minimum/maximum cardinality feasible solutions;
  3. for any weight vector w in R^S, lists all minimum/maximum value feasible solutions;
  4. for any weight vector w in R^S, lists all minimum/maximum value minimum/maximum cardinality feasible solutions.
- Applies to: matching ensemble, t-join ensemble, spanning tree ensemble,…

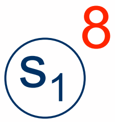
Separable Combinatorial Ensembles

- We just need to solve one of the variants of case 3.

- TEO: a combinatorial ensemble is separable if and only if the associated combinatorial optimization problem is polynomial time solvable for any weight vector (Consequence of the equivalence between optimization and separation by Grötschel, Lovasz, Schrijver).

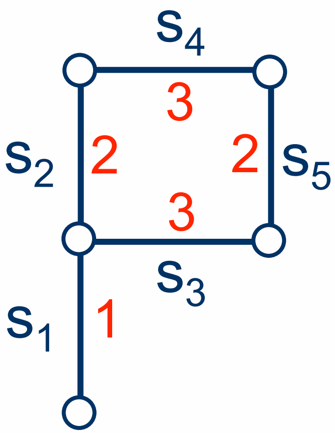
- We assume to have a polynomial time algorithm MIN(S,F,w) which returns the minimum value.

MWST Simulation



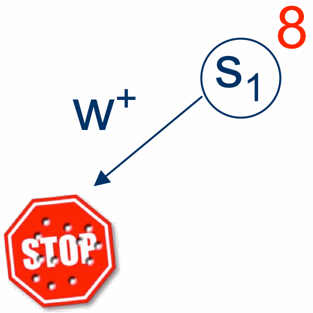
$\text{Min}(S,F,w^+) = \text{Min}(S,F,w)$

$\text{Min}(S,F,w^-) = \text{Min}(S,F,w)-1$



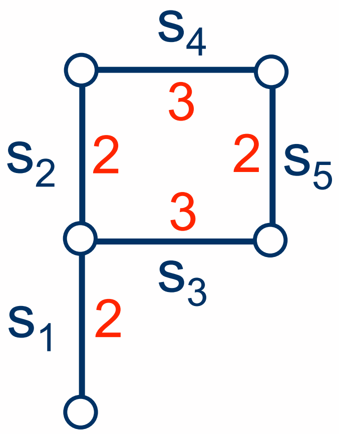
Min = 8

MWST Simulation



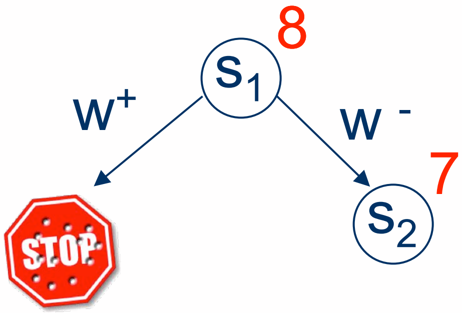
$\mathrm{Min}(S,F,w^+) = \mathrm{Min}(S,F,w)$

$\mathrm{Min}(S,F,w^-) = \mathrm{Min}(S,F,w)-1$



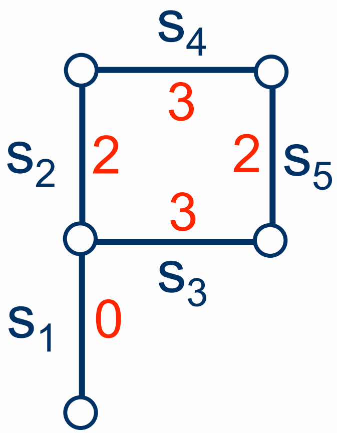
Min = 9

MWST Simulation



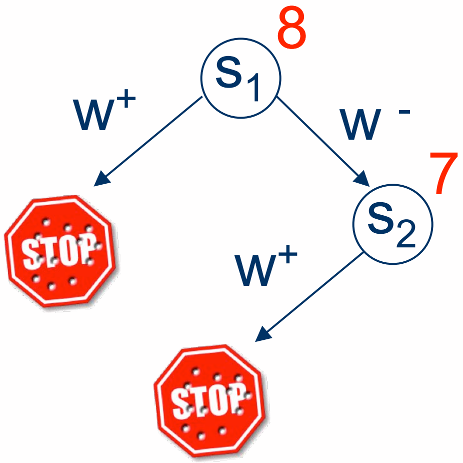
$\mathrm{Min}(S,F,w^+) = \mathrm{Min}(S,F,w)$

$\mathrm{Min}(S,F,w^-) = \mathrm{Min}(S,F,w)-1$



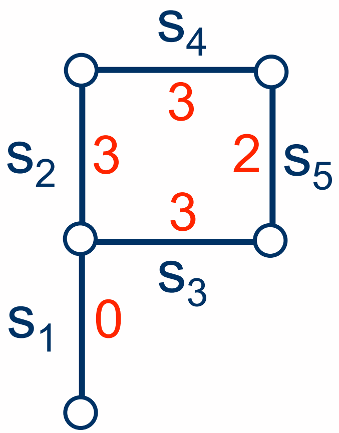
Min = 7

MWST Simulation



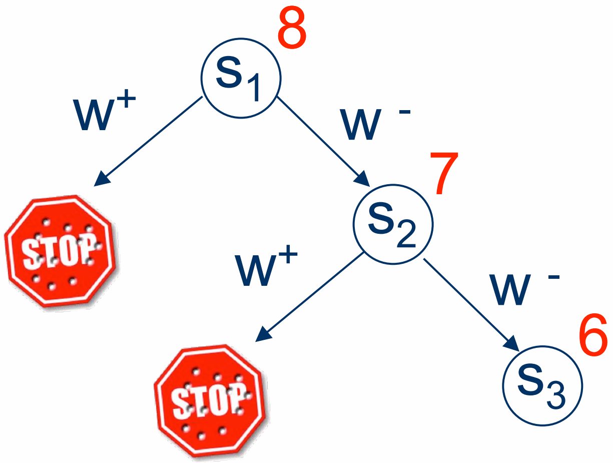
$\mathrm{Min}(S,F,w^+) = \mathrm{Min}(S,F,w)$

$\mathrm{Min}(S,F,w^-) = \mathrm{Min}(S,F,w)-1$



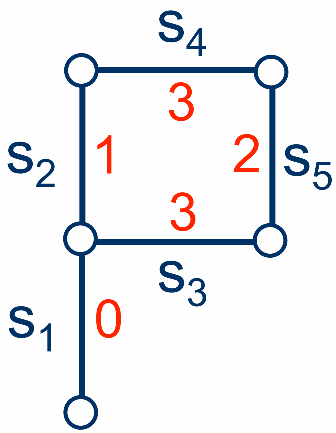
Min = 8

MWST Simulation



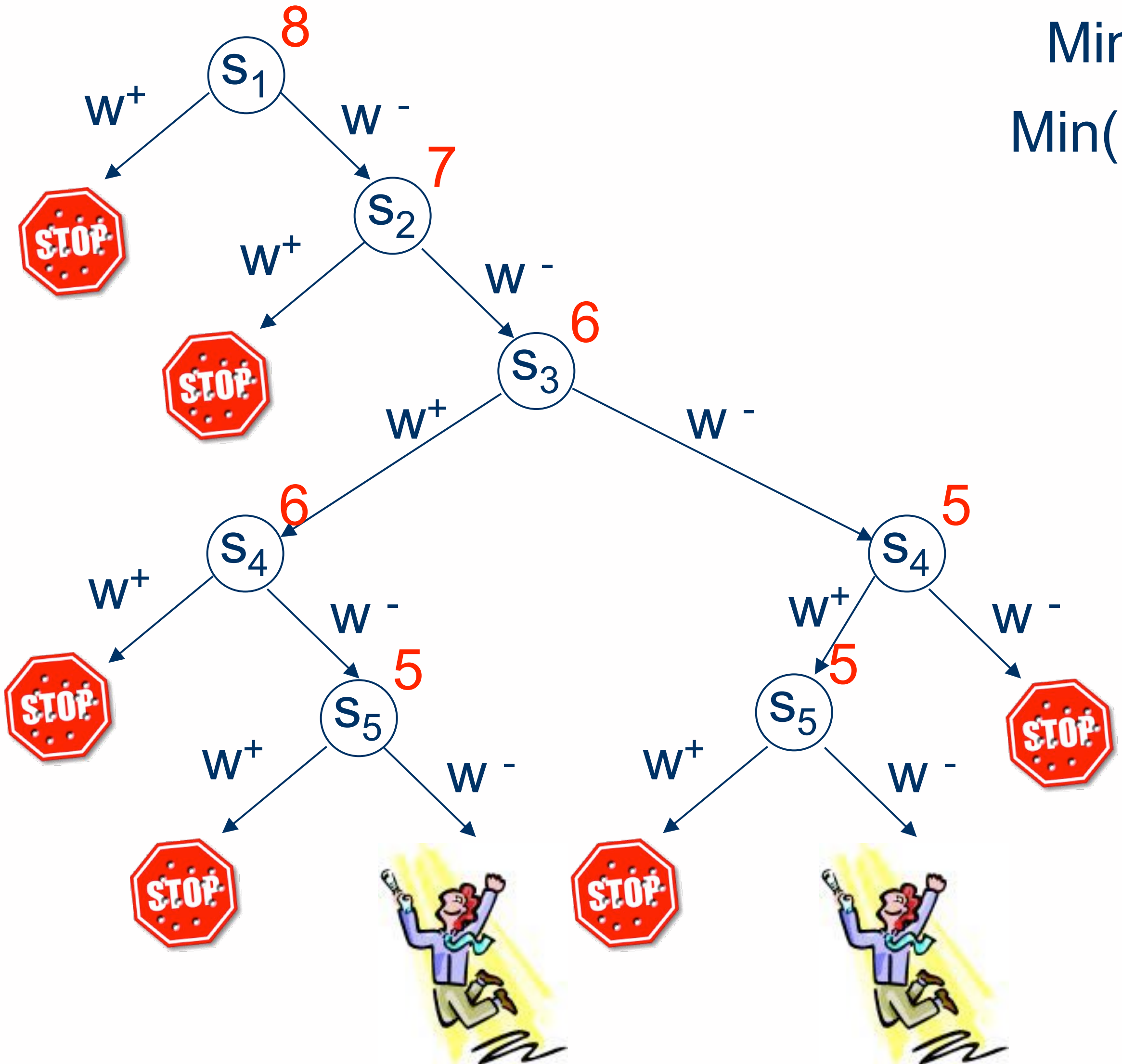
$$\mathrm{Min}(S,F,w^+) = \mathrm{Min}(S,F,w)$$

$$\mathrm{Min}(S,F,w^-) = \mathrm{Min}(S,F,w)-1$$



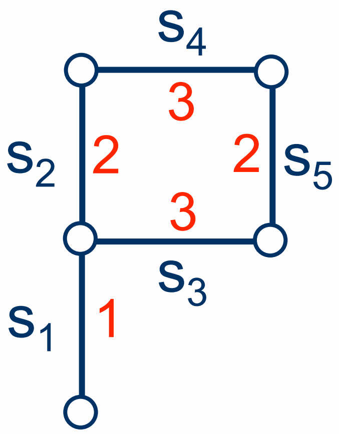
Min = 6

MWST Simulation

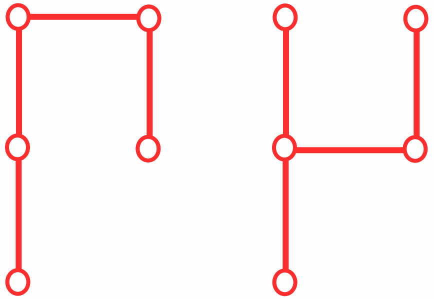


Min(S,F,w⁺) = Min(S,F,w)

Min(S,F,w ⁻) = Min(S,F,w)-1



Min = 8



Question

Consider a combinatorial ensemble for which the associated combinatorial optimization problem is polynomial time solvable only for nonnegative weights (e.g. min-cut problem): can we efficiently list all (optimal) feasible solutions?

Negative example

- Weighted 0Valid Sat:
  - Input: 0Valid CNF-formula φ of n variables, a non-negative weight w_i associated to each variable x_i.
  - Output: a minimum value satisfying truth assignment of φ, where the value of a truth assignment T is $\sum_{i=1..n} w_i T(x_i)$.
- Weighted 0Valid Sat is polynomially solvable: $T=0^n$ is always a minimum value truth assignment!
- LWeighted 0Valid Sat is LP-hard: setting all weights equal to 0, we get L0Valid Sat, an LP-complete problem.

A partial compensation…

- If the combinatorial optimization problem associated to a combinatorial ensemble is polynomial time solvable for nonnegative weights, then there exists a polynomial space polynomial delay algorithm that:
  1. lists all minimum cardinality feasible solutions;
  2. for any weight vector w in $R_{>0}^{S}$, lists all minimum value feasible solutions;
  3. for any weight vector w in $R_{>0}^{S}$, lists all minimum value minimum cardinality feasible solutions.

- Slight modification of the previous algorithm.

End of second part…

- Introduction.
- A structural complexity theory for listing problems associated to NP relations.
- Listing solutions of a broad class of combinatorial optimization problems.
- Listing satisfying truth assignments of some peculiar classes of boolean formulas (XOR and 2SAT).
- Conclusions.

XOR-formulas

- An XOR-formula is a CNF boolean formula where disjunction is replaced by exclusive disjunction.
- Example: $(a \oplus b) \wedge (\mathrm{not}(a) \oplus c \oplus \mathrm{not}(b)) \wedge (b)$.
- The problem of deciding whether an XOR-formula is satisfiable or not is in P (Gaussian elimination is polynomial time computable – Edmonds ‘67).
- There exists a polynomial time algorithm to count the number of satisfying truth assignments of an XOR formula (Creignou, Hermann ‘96).
- There exists a polynomial space polynomial delay algorithm for listing all satisfying truth assignments of an XOR formula (Creignou, Hébrard ‘97).

2SAT formulas

- A 2SAT formula is a CNF boolean formula with at most two literals per clause.
- Example: (aVb) Λ (not(a)Vc) Λ (b).
- There exist several linear time algorithms to decide whether a 2SAT formula is satisfiable [Even, Itai, Shamir '76 – Del Val '01].
- The counting problem is #P-complete (Valiant '79).
- There exists a polynomial space polynomial delay algorithm for listing all satisfying truth assignments of a 2SAT formula (Creignou, Hébrard '97).

XOR-formulas & 2SAT formulas

- We propose a polynomial space linear delay algorithm for listing all satisfying truth assignments of XOR formulas and a polynomial space linear delay algorithm for listing all satisfying truth assignments of 2SAT formulas.

- Differently from the previously proposed approaches, our algorithms achieve this improved delay-time bound by exploiting the underlying structure of the considered problems.

Conclusions: new contributions

- A new structural computational complexity theory for listing problems associated to NP relations.
- Listing (optimal) feasible solutions of combinatorial optimization problems.
- New polynomial space linear delay algorithms for listing satisfying truth assignments of XOR formulas and 2SAT formulas.

Conclusions: an open problem

- There exists a polynomial delay algorithm to list all maximal independent sets of a graph (Johnson, Papadimitriou, Yannakakis ‘88).
- It is unknown whether even a polynomial total time algorithm exists for hypergraphs.
- Given an hypergraph and a collection of maximal independent sets for it, there exists a quasi-polynomial time ($m^{\log(m)}$) algorithm which finds a new maximal independent set or concludes that the given collection is complete (Fredman and Khachiyan ‘96).

Thank you!

Questions ?