Definition 2.1 (Problem).

A problem P is described by giving:

* a general description of all its parameters;
* a statement of what properties the answer, or solution, is required to satisfy.

Definition 2.2 (TSP).

The Travelling Salesman Problem (TSP) asks the following question: given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Definition 2.3 (Algorithm).

An algorithm is a general, self-contained, step-by-step procedure for solving problems that, given an input, produces an output.

Definition 2.4 (Size of an instance).

The size of an instance I is the number of bits needed to encode I.

Definition 2.5 (Polynomial algorithm).

An algorithm is polynomial if it requires, in the worst case, a number of elementary operations f(n) = O(nd), where d is a constant and n = |I| is the size of the instance.

Observation 1.

If the number of elementary operations required is f(n) = O(nd \* logc n), the algorithm is still polynomial. (log < polynomial)

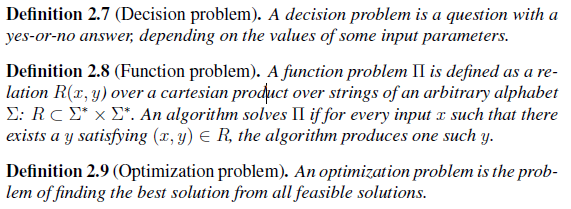
Definition 2.6 (Exponential algorithm).

An algorithm is exponential if it requires, in the worst case, a number of elementary operations f(n) = O(2n), where n = |I| is the size of the instance.

Observation 2.

In real life scenarios, polynomial algorithms with a big exponent, e.g., >= 5 may require a lot of time to be executed, becoming not usable.

Problem forms



Observation 3 (Problem formulation). Each problem can be formulated in each form: decision, function and optimization.

Complexity Classes

For decision problems-------------------

Definition 2.10 (P).

P is the set of all decision problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time.

Definition 2.11 (NP).

NP is the set of all decision problems such that, for each instance with yes answer, there exists a concise certificate, called proof, which allows to verify in polynomial time that the answer is yes.

Definition 2.12 (coNP).

A decision problem \_ is a member of coNP if and only if its complement \_ is in the complexity class NP.

For Functional Problems --------------

Definition 2.13 (FP).

The complexity class FP is the set of function problems which can be solved by a deterministic Turing machine in polynomial time.

A binary relation P(x; y) is in FP if and only if there is a deterministic polynomial time algorithm that, given x, can find some y such that P(x; y) holds.

Definition 2.14 (FNP).

A binary relation P(x; y), where y is at most polynomially longer than x, is in FNP if and only if there is a deterministic polynomial time algorithm that can determine whether P(x; y) holds given both x and y.

Definition 2.15 (coFNP).

A function problem P is a member of coFNP if and only if its complement P’ is in the complexity class FNP.

For Optimization problems --------------

Definition 2.16 (PO).

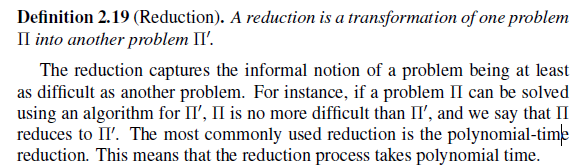
PO is the set of all optimization problems that can be solved by a deterministic Turing machine using a polynomial amount of computation time.

Definition 2.17 (NPO).

NPO is the set of all optimization problems such that, for each instance with yes answer, there exists a concise certificate, called proof, which allows to verify in polynomial time that the answer is yes.

Definition 2.18 (coNPO).

An optimization problem O is a member of coNPO if and only if its complement O’ is in the complexity class NPO.



Definition 2.20 (NP-hardness).

NP-hardness (non-deterministic polynomialtime hard) is the class of decision problems that are at least as hard as the hardest problems in NP.

More precisely, a problem Pi is NP-hard when every problem Pi’ in NP can be reduced in polynomial time to Pi. (Pi can be any hardness, just reducible to it will do)

Definition 2.21 (NP-completness).

A decision problem Pi is NP-complete if:

1. Pi is in NP;

2. every problem in NP is reducible to Pi in polynomial time.

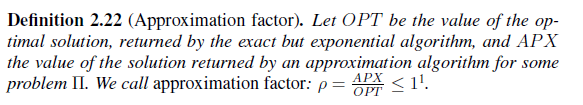
Thus, the class of NP-complete problems contains the most difficult problems

in NP, in the sense that they are the ones most likely not to be in P.

(NP-complete needs to be in NP)

Theorem 2.1. Proving that a problem is NP-hard does not imply proving

that the problem is in NP.

Approximability of a problem 

Definition 2.23 (APX).

APX is the complexity class of all the optimization problems Pi that are in NP and that admit polynomial approximation algorithms with an approximation factor bounded by a constant value: p\* = 1/c.

Definition 2.24 (APX-hardness).

A problem is said to be APX-hard if there is a polynomial-time reduction, within every multiplicative factor of the optimum other than 1, from every problem in APX to that problem.

Definition 2.25 (APX-completness).

A problem is said to be APX-complete if it is APX-hard and belongs to APX.

(TSP problem is APX-complete).

Definition 2.26 (log-APX).

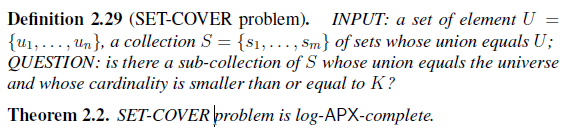
APX is the complexity class of all the optimization problems Pi that are in NP and that admit polynomial approximation algorithms with an approximation factor bounded by a logarithmic approximation factor: p\* = 1/ logc (n)

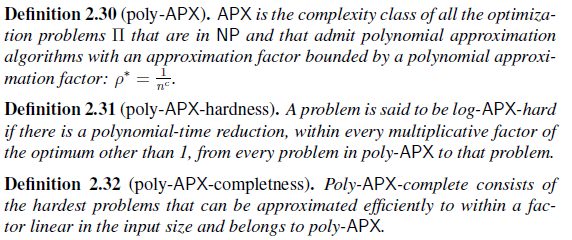
Definition 2.27 (log-APX-hardness).

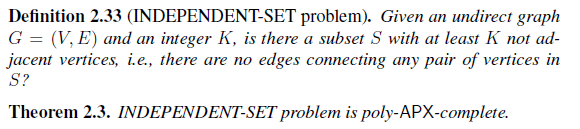
A problem is said to be log-APX-hard if there is a polynomial-time reduction, within every multiplicative factor of the optimum other than 1, from every problem in log-APX to that problem.

Definition 2.28 (log-APX-completness).

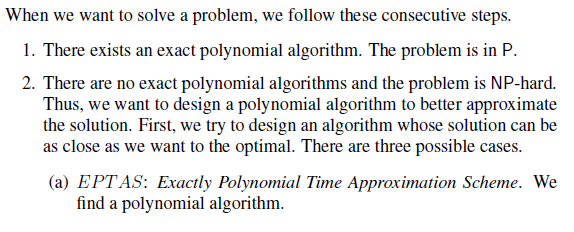
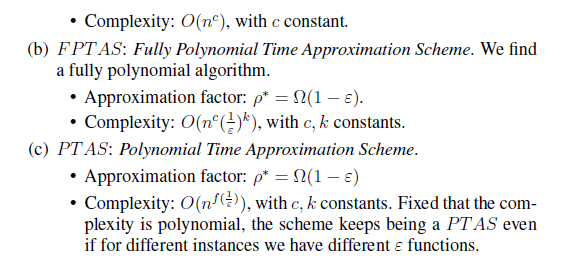
log-APX-complete consists of the hardest problems that can be approximated efficiently to within a factor logarithmic in the input size and belongs to log-APX.

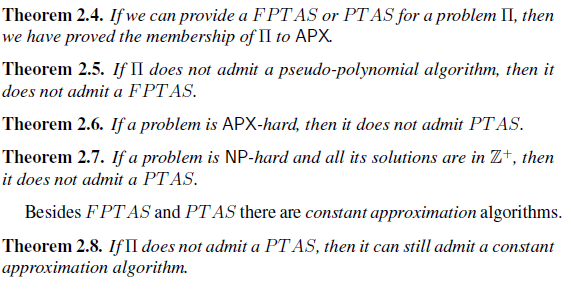






What is Independent Set Problem?? (can’t find in reference)



Game Theory

Definition 2.34 (Prisoner’s Dilemma).

Two members of a gang, say A, B get arrested and put in jail. Each prisoner is put in isolation, without any possibility of communicating with the other. Police lack evidence to convict the pair on the principal charge. Thus, they hope to get both sentenced to a year in prison on a lesser charge. In the meantime, the following offer is made to both prisoners.

Normal Form Games

Extensive Form Games

The main technique to solve extensive form games is called Backward Induction.

In general, there three main forms in which an extensive form game is put to be solved.

• Normal form: discussed above, it is the most known form for solving games. Unfortunately, the size of such game is exponential w.r.t. the size of the tree and so there is no gain in terms of how fast we can solve the game.

• Agent form: it consists of considering a different player for each node of the tree while adopting the utilities of the player whose node has been taken. This representation is a step ahead, being polynomial w.r.t. the size of the tree and the number of players.

• Sequence form: such form defines actions as sequences adding further constraints on the computation of the strategies. Such representation is linear w.r.t. the size of the tree.

Bayesian Games (Example- Battle of the Sexes)

Solution Concepts

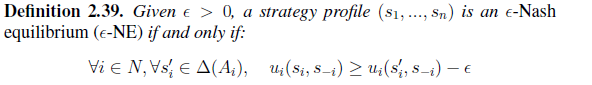
1. Nash Equilibrium

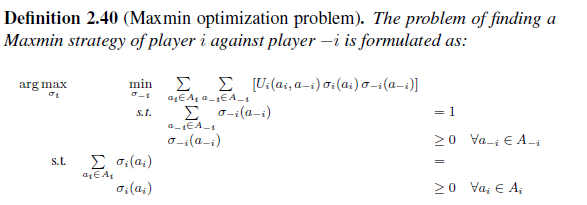
Theorem 2.10.

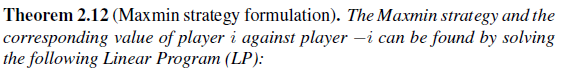
Every game played by a finite number of players, each with a finite number of actions, has at least a mixed strategies NE.

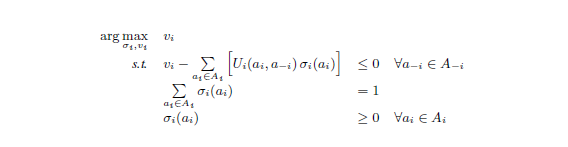
Corollary 2.11.

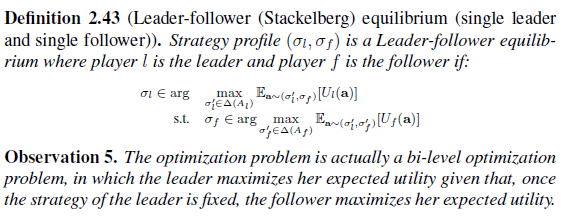
If a game admits at least an NE, then it admits an odd number of NE.











Theorem 2.15.

Every zero-sum finite game admits at least an NE in mixed strategies.

Theorem 2.16.

In a zero-sum game, NE, maxmin/minmax strategies and leader-follower equilibrium coincide.

Theorem 2.17.

In non zero-sum games, the utility obtained by the leader from a Leader-follower equilibrium is greater than or equal to the utility provided by an NE.

Online Learning

1. Prediction with Expert Advice
2. Follow the leader



1. Follow the perturbed leader



1. Multi Armed Bandit
   1. Upper Confidence Bound
   2. Thompson Sampling