9. Complexity Theory : The Frontier

Sai Divya Enni

9.1 Introduction

- Complexity theory is the most active area of research in the theory of Computation.
- This chapter deals above the practical applications of Complexity theory.
- It deals with basically 2 issues :
  - The complexity of single instances (characterizing the complexity of a single instance into a worst case behavior or average case behavior).
  - The second issue is can we develop complexity classes and completeness results based on average case.
- Much of the complexity theory is about modeling computation in order to understand it we would naturally want to study these new devices, develop models for their mode of computation and compare the results with current models.

- Parallel computing has been applied intensively to various applications like Optical computing, DNA computing, Quantum computing.
- Parallelism helps us to make many problem tractable but it does not alas helps us to solve NP-hard problems in Polynomial time.
- The most important development in complexity theory has been in “Proof theory”.
  - The researchers have focused in 2 distinct models they are:
    - One where the prover and checker interact for as many rounds as the prover needs to convince the checker.
    - Second one where the prover simply writes down the argument as a single, non interactive communication to the checker.
- Complexity theory has its own side which deals with internal questions like P vs NP this is called “ Structural Theory”

- Some of what we have covered in the previous chapters comes under structure theory like polynomial hierarchy.
- Some of the blossoming areas of research are Complexity Learning Theory.
- This research into fine structure of NP has also lead to research in fine structure in P this is called Fixed-Parameter Tractability.
- All these above mentioned researches have been of theoretical interests only.

9.2 The Complexity of Specific Instances

- Complexity core: Complexity core for a problem is an infinite set of instances all but a finite number of which are hard by hard meaning we shall look at complexity cores of P and define hard problems as the problems that are not solvable in polynomial time.
- Theorem 9.1: if a set is not in P then it posses an infinite subset $x \subseteq S$, such that any decision algorithms must take more than a polynomial number of steps almost everywhere on X

Proof: our proof proceeds by diagonalizing over all the TMs. Denote the *i*th turing machine in the enumeration by Mi and the output that it produces when it runs with string x is Mi(x). Let pi be the sequence of polynomials

$$p_i = \sum_{j=0}^{i} x^i$$

- Note that this sequence has following 2 properties:
  - For any value n>0, i>j→Pi(n)>Pj(n).
  - Given any polynomial p there exists an I such that Pi(n) >Pj(n) holds for all n>0.
- We construct a sequence of elements of s such that the nth elements cannot be accpted in Pn|Xn| time by any of the first n turing machines in the enumeration.
- Denote by the Xs the characteristic function of s that is we have xЄs→Xs(s)=1 and the converse implies that Xs(s)=0. we construct each element by element s follows:
  - Let y be the empty string and let the stage number n be 1.
  - For each i 1≤ i ≤n such that I is not yet cancelled run machine Mi on the string y Pn(|y|) until it terminates . If Mi terminates but does not solve instance y correctly we need not consider Mi again since it cannot decide membership in S.

- For each I not yet cancelled determine if it passed step 2 because machine Mi did not stop in time. If so let Xn=y and proceed to step 4 if not so replace y y the next string in lexicographic order and return to step 4.
- After replacing increase n by 1 and return to step 2.

- If stage n does not terminate it loops back between step 3 and 2 to produce infinite long strings y.
- This can happen only if there exists an un canceled i such that Mi terminates in no more than P|y| steps. But then we have Mi(y)=Xs(y) since I is not cancelled thus we have a polynomial time decision procedure for our problem.
- Hence we have proved that for all but a finite number of instances of X machine Mi must run in super polynomial time.

Theorem 9.2:

- Every NP complete problem has complexity cores of Super polynomial density.

Definition 9.1:

- A hard instance is one that can be solved efficiently only through table lookup.

Definition 9.2:

- Let Iy be the yes instances of some problems ∏, x an arbitrary instance of the problem and t() some time bound.
  - The t bound instance complexity of x w.r.t to ∏, IC^t (x|∏), is defined as the size of the smallest turing machine that solves ∏ and runs in time bounded by t|x| on the instance x, if no such machine exists then the instance complexity is infinite.

- The descriptional complexity of a string x, K(x) is the size of the smallest Turing machine that produces x when started with the empty string.
- We write K'(x) if we also require that the turing machine halt in no more than t|x| steps.

- Proposition 9.1:

For every problem $\prod$, there exists a contact C_π such that

$$IC^t(x \mid \pi) \leq K^t(x) + C\pi$$

Holds for any time bound t() and instance x.

- Definition 9.3:

Given constant c and time bound t(), an instance x is (t,c) hard for the problem $\prod$ if

$$IC^t(x \mid \pi) \geq K(x) - c \quad holds.$$

Questions ????