Literature Review of PPA

Sawyer Maloney & Drew Bevington

1 Introduction

This paper will talk about ideas of “complete” problems in economics as well as explain to the reader how this completeness is proved. In particular, we will anaylze and discuss the findings from ”The Complexity of Computing a Nash Equilibrium”.

2 TFNP & sub-classes

In “On the complexity of the parity argument and other inefficient proofs of existence“, Papadimitriou showed that, while there are no problems which are complete for TFNP, there are (at least) four subclasses of TFNP that have complete problems.

TFNP is the total functional non-deterministic polynomial time class. More specifically, it is the collection of functional problems in NP that always have a solution. It is because of this requirement that TFNP does ont have any complete problems for the entire complexity class. TFNP is a semantic class, meaning that its requirement cannot be enforced mechanically or algorithmically; instead, it is a description of how the machine (in our case, a deterministic turing machine) interacts with the problem at hand. The subgroups that will eventually be defined under TFNP will instead have verifiable and enforceable requirements for problems to be in their respective subgroups, which will mean that they are syntactic, and will have complete problems.

This requirement of having a solution is enforced through a number of exponentially non-constructive combinatorial lemma. The term ’exponentially non-constructive’ means a lemma that does not explicitly give the structure, but only shows that the structure exists, and that trying to construct such a structure would be exponentially difficult.

The most important of these lemmas are defined by Papadimitriou from the above paper:

- Parity argument: if a graph has a node of odd degree, then there must be another odd degree node. This argument creates the PPA subclass of TFNP.

- Parity for directed graphs: If a directed graph has an unbalanced node (a vertex with different in and out degrees), then there must be another unbalanced node. This creates the PPAD subclass.
- Polynomial Local Search: Every directed acyclic graph must have a sink. This argment creates the PLS subclass.
- Pigeonhole Principle: If a function maps n elements to n-1 spots, then there must be a collision. This argument creates the PPP subclass.