Determining the Complexity of Parallel Circuits: A Proposal

Brian Lee

May 2017

Abstract

The goal of the experiment is to determine if Parallel Circuits are NP-Complete or P-Complete. For this purpose, parallel circuits will be augmented with switches, which will be modelled in a computer program's logic flow. To start, three parallel circuits are to be constructed, consisting of 1,2, and 3 resistors respectively. From there, induction will be utilized to attempt to prove that the number of resistors can go up to infinity. The experiment is a form of inquiry into the prospect of modelling circuits using computers, which has numerous potential applications. It will also attempt to speculate, depending on the results, if $P = NP$.

1 Introduction

Given that the experiment incorporates some higher-level concepts within Computer Science, it's appropriate to outline them here for a proper grasp of the experiment's goals/function. Please note, however, that the following is a low-level overview meant for a basic understanding. The final paper will approach Computer Science on a substantially more formal level.

1.1 Computational Complexity Theory

Computational Complexity Theory is a branch of Theoretical Computer Science that asks the question of *how hard* certain problems are (in terms of efficiency in solvency). There are two general classes of problems: the first consists of problems that *are* easily processed (by a computer), and has a set runtime of $O(n^k)$ (in other words, a static polynomial function). Problems here reside within the complexity class **P** (Polynomial Time), and are described as 'P-Complete.'[1]

For example, Consider a computer program that compares an input value with each number of an array of numbers.[1] It would be classified as $P - Complete$ as it has a set runtime of $O(n)$ (the program goes through all the numbers in the array). Almost all the software a typical person uses is composed of $P - Complete$ altorithims (e.g. Microsoft Office).

The other complexity class composes of problems that are *hard* for a computer to solve. There is no distinct $O(n^k)$. Rather, a computer can *check* solutions to the problem in P-time. Problems of this class are said to be NP-Complete, or 'Non-Deterministic Polynomial Time.'

A good example of a NP-Complete problem is Minesweeper. This conceptually fits: a computer won't be able to exactly model problems based on *random chance* (such as the mines' locations); there will always be scenarios where the algorithim is wrong.[2] This further implies that computers can't model *everything.*

1.2 Physics

The aformentioned Computer Science terminology/concepts will be utilized to examine one type of circuit commonly tested on the AP Physics 1 Exam: the parallel circuit.[3] Conceptually speaking, they separate I into smaller branches and, in its most fundamental variation, dedicates a single resistor to reach branch.[2]The following equations relating V, I, and R are derivable using Olm's Law ($V = IR$):

$$V_{net} = V_1 = ... = V_n$$

[1]Code is available in the appendix

[2]Circuit diagrams of parallel circuits are available in the appendix

$$I_{net} = \Sigma I_n$$
$$\frac{1}{R_{eq}} = \Sigma \frac{1}{R_n}$$

A very common variant of question on the AP Physics 1 Exam for parallel circuits asks for the effect changing a portion of the circuit has on the entire circuit, for a good reason. Parallel Circuits have unique properties regarding such changes (e.g. if a resistor is removed). For example, it was proven that adding R_n decreases R_{eq}, removing an R_n increases V_n for the other R_n, etc. These properties of change will become instrumental to the experiment, as they will enable Computer Science to effectively test parallel circuits.

2 Experimental Structure

2.1 Goals

The primary goal of the experiment is to determine if parallel circuits are P-Complete, NP-Complete, or some variant of the two. It will follow the path previous computer scientists have taken in proving problems to be P or NP[2], primarily via the usage of logic gates.

The biggest motivation for this experiment was the opportuntiy to utilize Theoretical Computer Science, but the experiment may also reasonably explain why parallel circuits (and other circuits) are so prevalent in electronics (the possibility of P-Completeness would explain this very well).

2.2 Procedure (Conceptual)

It is worth noting that a formalized procedure is yet to bee concieved of. The general course of the experimental design has been concieved of, but issues regarding implementation of the design must be resolved.

In any case, the general course of the experiment follows a few steps. First, concrete paralell circuits are constructed, with V, I, and R changes recorded. Switches will be added to the circuit to construt logical gatess and facilitate the rest of the math moving forward. This will serve as the basis for an eventual proof of induction to isolate some $O(n^k)$ out of the data (notably as the base case). If this is successfull, a computer program modelling the circuits will next be constructed. If this is not successfull, additional experimentation will be necessary to isolate that the procedure yielded NP-Complete, as opposed to a mere fluke.

3 Appendix

Listing 1: Sequential Search

```
def sequentialSearch(test, arr):
        output = -1
        count = 0
        run = True
        #the method ieterates through the function
        #demonstrates complexity O(n)
        while run and count < len(arr):
                if arr[count] == test:
                        output = count
                        run = False
                else:
                        count+=1
        return output
```

References

[1] Anil Maheshwari and Michiel Smid. *Introduction to Theory of Computation*. Carleton University, Ottawa, Canada, 2017.

[2] Richard Kaye. Minesweeper is np-complete. *The Mathematical Intelligencer*, 22(2):9–15, 2000.

[3] The College Board. *AP Physics 1: Algebra 1-Based and AP Physics 2: Algebra 1-Based - Course and Exam Description*. The College Board, New York, NY, 2014.