# Proof of Parallels on Computational Problem Resolution

Mirzakhmet Syzdykov
Independent researcher, Astana, Kazakhstan

**ABSTRACT**

Since our exhibition of the equivalence of complexity classes, we have met that they are still not giving correct and coherent solution for problems known as unsolvable or NP-complete by Cook-Levin theorem "P versus NP", in contrary, in this work we present the proof of nonequivalent relation between polynomial and non-polynomial classes and show that NP-complete problem cannot be approximated or solved on a hypothetical computational device within the proof by contradiction.

**Keywords:** Cook-Levin theorem, P versus NP, relation, proof, contradiction.

## INTRODUCTION

The theorem whether the problem can be solved as dependent polynomial function as it can be verified is known as "P versus NP" [1] and was tried to be proved for the past time [2], however, it led to the resulting statement till the present time.
The sample model used in our proof resembles to the parallel computing strategy [3] as parallel computations is a way of giving the more improved solution to algorithm or method [4, 5], however, this is only an analogy to our modeled virtual machine which hypothetically can solve any NP-complete problem.

## VIRTUAL MACHINE

Imagine that any query to the oracle in the given Virtual Machine (VM) can be processed in time at least *O(1)* as it will take some operation and time to check and verify the query towards global optimal solution. The oracle itself compares the result with the given one in the same time and totally in a separate thread in time *O(1)* for overall queue of incoming queries.
We then can construct the universal problem solver for any NP-complete problem based upon the number of parallel threads operating on this hypothetical VM, which will run in polynomial time, thus giving the following statement:

$$\lim_{N \to O(NP)} \frac{NP}{N} = P,$$

where in above statement *NP* is an NP-complete problem complexity, *P* is an oracle polynomial verification complexity and *N* is the number of threads operating on VM.

## PROOF

With the general assumption we can state that if *N* converges to infinity the whole solution process in observable time in above VM still takes polynomial time, thus we can assume the following:

$$\lim_{N \to \infty} \frac{NP}{N} = P.$$

However, we can also assume that the whole complexity with the infinite number of working threads will take the tame of at least *O(N)* for the given NP-complete problem *NP*, thus, to show our proof, we will assume that *P = NP*, then we have the following:

$$\lim_{N \to \infty} \frac{1}{N} = 1 \Rightarrow 0 = 1 \Rightarrow P \neq NP.$$

The above resolution is a contradiction, thus, $P \neq NP$ on VM even if the number of threads is infinitely large to check and verify each of the solution from the problem NP.

## DISCUSSION

We have gone through the long way of the complexity theory and its outcomes, however, the presented result and the following outlet demonstrate that more than practice is needed in order to overcome the existing barriers lying in the proof of $P = NP$, which supposedly can give a way to approach 'the almost unsolvable problem', however, the experiment and its result tell the different answer.
This dilemma can be seen as promising, but hard to present, since solutions often are described as algorithmic steps, which we avoid in order to rise our experiment.

## CONCLUSION

We have shown on our model that polynomial oracle $P$ for non-polynomial problem $NP$ will be never equal due to the contradiction on universal solver like VM for NP-complete problems – this is due to the fact that when infinite number of threads are performed on our abstract model and classes are equal it will take zero time in order to find the optimal and correct solution which contradicts with the 'at least' time $O(1)$ for polynomial method since it will take at least single step in order to produce the final result, which generally always exist and its minimal dimension can be two like 'true' or 'false', thus classes $P$ and $NP$ aren't equa, or $P \neq NP$, with this contradiction followed from assumption that these classes can be equal, considering the fact that VM can solve any problem in $NP$ as well as in $P$.
From that and on, it follows that classes are not equal, which can be discussed further, for this purpose we have used a hypothetical approach in order to simulate the best scenario, however, even with abstract and fully theoretical approach, it still remains clear that classes will never coincide, which may give a general proof of "P versus NP" theorem since our model is abstract and follows the assumption of the solution of any NP-complete problem, against which we have made a theoretical experiment and devised the exposed result.
Still we stake that the proposed method is a general case, since the given method uses the approximation to the absolute parity using infinite number of threads on abstract and hypothetical VM, at least it can be partial, but the given scenario is a way of abstracting from that 'partiality'.

## ACKNOWLEDGMENTS

We are very grateful to the many others who considers this problem actual up to the present day.

**REFERENCES**


1. Cook, S. (2003). The importance of the P versus NP question. *Journal of the ACM (JACM)*, *50*(1), 27-29.
2. Sipser, M. (1992, July). The history and status of the P versus NP question. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing* (pp. 603-618).
3. Kaur, P., Singh, D., Singh, G., & Singh, N. (2011). Analysis, comparison and performance evaluation of BNP scheduling algorithms in parallel processing. *International Journal of Information Technology and Knowledge Management*, *4*(1), 279-284.
4. Blelloch, G. E., & Maggs, B. M. (1996). Parallel algorithms. *ACM Computing Surveys (CSUR)*, *28*(1), 51-54.
5. Skillicorn, D. B., & Talia, D. (1998). Models and languages for parallel computation. *Acm Computing Surveys (Csur)*, *30*(2), 123-169.