LSPACE VS NP IS AS HARD AS P VS NP

FRANK VEGA

Abstract. The P versus NP problem is a major unsolved problem in computer science. This consists in knowing the answer of the following question: Is P equal to NP? Another major complexity classes are LSPACE, PSPACE and EXP. Whether LSPACE = P is a fundamental question that it is as important as it is unresolved. We show if P = NP, then LSPACE = NP. Consequently, if LSPACE is not equal to NP, then P is not equal to NP. According to Lance Fortnow, it seems that LSPACE versus NP is easier to be proven. However, with this proof we show this problem is as hard as P versus NP. Moreover, we prove the complexity class P is not equal to PSPACE as a direct consequence of this result. Furthermore, we demonstrate if PSPACE is not equal to EXP, then P is not equal to NP.

1. Introduction

In complexity theory, a function problem is a computational problem where a single output is expected for every input, but the output is more complex than that of a decision problem [6].

A functional problem F is defined as a binary relation $(x, y) \in R$ over strings of an arbitrary alphabet Σ:

$$R \subset \Sigma^* \times \Sigma^*.$$

A Turing machine M solves F if for every input x such that there exists a y satisfying $(x, y) \in R$, M produces one such y, that is $M(x) = y$ [6].

The verification problem for a functional problem F based on a binary relation R is the problem of deciding the elements of language

$$\{x; y : (x, y) \in R\}.$$

We say that R is polynomially decidable if the verification problem for which is defined its functional problem can be decided by a deterministic Turing machine in polynomial time within the length of x [5]. R is polynomially balanced if $(x, y) \in R$ implies $|y| \leq |x|^k$ for some positive constant $k \geq 1$ where $|\dots|$ is the string length function [5].

Let $L \subseteq \Sigma^*$ be a language. $L \in NP$ if and only if there is a polynomially decidable and polynomially balanced relation R_L, such that

$$L = \{x : (x, y) \in R_L \textit{ for some } y\}.$$

If the language $L \in NP$ can be decided in polynomial time by a deterministic Turing machine, then $L \in P$ [5].

The function problem associated with L, denoted FL, is the following computational problem:

2000 *Mathematics Subject Classification.* Primary 68Q15, Secondary 68Q17.

Key words and phrases. Function Problem, Verification Problem, Turing Machine, Logarithmic Space, Polynomial Time.

Given x, find a string y such that $(x, y) \in R_L$ if such a string exists; if no such string exists, return "no" when there is a string y with $(x, y) \in R_L$ if and only if $x \in L$ [5].

The class of all function problems associated as above with languages in NP is called FNP [5]. FP is the class resulting if we only consider the function problems in FNP that can be solved in polynomial time [5].

In complexity theory, $LSPACE$ (also known as L) is the complexity class containing the decision problems that can be decided by a deterministic Turing machine in logarithmic space [1]. Formally, the Turing machine for $LSPACE$ has two special tapes and the work tapes, one of which encodes the input and can only be read, the second one of which encodes the output and can only be write, whereas the other work tapes have logarithmic size but can be read as well as written [1].

Definition 1.1. *FPL is the class of all function problems in FNP whose verification problems can be decided in logarithmic space through a string function G computable in logarithmic space. That is, for a binary relation R_L, for which is defined a functional problem in $FL \in FNP$, the language*

$$\{x; y : (x, G(y)) \in R_L\}$$

can be decided by a deterministic Turing machine in logarithmic space within the length of x where G is a string function computable in logarithmic space: We say the function problem FL is in FPL through the string function G.

We show $FP \subseteq FPL$, but $FPL \subsetneq FNP$ under the assumption of the statement $LSPACE \neq P$. In this way, we obtain $FP \neq FNP$ when $LSPACE \neq P$. If $FP \neq FNP$ then $P \neq NP$ [5]. To sum up, $LSPACE \neq P \Rightarrow P \neq NP$ and by contraposition $P = NP \Rightarrow LSPACE = NP$ is proven. In complexity theory, $PSPACE$ is the class of problems that can be decided in polynomial space [5]. As a consequence of this result, we show $P \neq PSPACE$. The complexity class EXP contains those problems which can be decided in running time $2^{O(n^k)}$ for a positive constant k [5]. This result also implies $EXP \neq PSPACE \Rightarrow P \neq NP$.

2. Results

Theorem 2.1. *$FP \subseteq FPL$.*

Proof. We define a CNF Boolean formula using the following terms: A literal in a Boolean formula is an occurrence of a variable or its negation and a Boolean formula is in conjunctive normal form, or CNF, if it is expressed as an AND of clauses, each of which is the OR of one or more literals [2]. An instance of WEIGHTED CNF SATISFIABILITY, or $WSAT$ for short, is a Boolean formula in CNF with a positive integer weight on each clause [7]. A feasible solution is an assignment (of 0 or 1) to all the variables [7]. The measure of a feasible solution is the sum of the weights of the satisfied clauses [7]. The neighborhood of a feasible solution contains all solutions obtained by flipping the value of one variable [7]. This is a maximization problem since an optimal solution is the one which has a maximum sum of weights of the satisfied clauses in relation to the neighborhood solutions [7].

An instance of WEIGHTED kCNF SATISFIABILITY, or $kWSAT$ for short, is an instance of $WSAT$ in which every disjunctive clause contains at most k literals. Solutions, measures, and neighborhoods are defined for $kWSAT$ in the same way as $WSAT$ [7]. $WSAT$ is in FPL through the identity string function [4]. Certainly,

the identity function can be computable in logarithmic space. Indeed, the verification problem of $WSAT$ is in $LSPACE$ [4]. This implies the problem $kWSAT$ is in FPL through the identity string function as well [4]. In the unweighted versions of $kWSAT$ and $WSAT$ each clause has exactly as a weight the positive integer 1 [7].

We say that a function problem A reduces to a function problem B if the following holds: There are string functions Q and S, both computable in logarithmic space, such that for any strings x and z the following holds: If x is an instance of A, then $Q(x)$ is an instance of B [5]. Furthermore, if z is a correct output of $Q(x)$, then $S(z)$ is a correct output of x [5]. Q produces an instance $Q(x)$ of the function problem B such that we can construct an output $S(z)$ for x from any correct output z of $Q(x)$ [5]. We say that a function problem A is complete for a class FC of function problems if it is in FC, and all the problems in that class reduce to A [5].

The unweighted version of $2WSAT$ is in *FP–complete* [7], [3]. Hence, there is a *FP–complete* in FPL through the identity string function [7], [4], [3]. Furthermore, since every single truth assignment of a formula in $2CNF$ is bounded by the formula itself, then the verification problem only depends on the space of the instances of the unweighted version of $2WSAT$ (ignoring the truth assignment string length), that is in the logarithmic space over the Boolean formulas in $2CNF$. Since both string functions in the function problem reductions are computable in logarithmic space, thus if there is some *FP–complete* problem which is in FPL through the identity string function, then all the problems in FP belong to FPL. Indeed, every problem in FP can be reduced in logarithmic space to the unweighted version of $2WSAT$ by the function problem reductions and the unweighted version $2WSAT$ is in FPL through the identity string function [7], [4], [3].

Certainly, for every instance x of some function problem $FL \in FP$, we can reduce it to an instance $Q(x)$ of $2WSAT$ using some string function Q that is computable in logarithmic space. If z is the correct output for $Q(x)$ in $2WSAT$, then there is a Turing machine M which decides $M(Q(x), z)$ in logarithmic space because $(Q(x), G(z)) = (Q(x), z)$ and $(Q(x), z) \in R_{2WSAT}$ because of the string function G is the identity function and R_{2WSAT} is the binary relation for which is based on the function problem $2WSAT$. M can be simulated by another Turing machine M' such that $M'(x, z)$ can be decided in logarithmic space within the length of x. We just need to reduce in logarithmic space the input $x; z$ in M' to $Q(x); z$ and simulate M on input $Q(x); z$. This is feasible since Q is computable in logarithmic space and the composition reduction is closed under logarithmic space [5]. However, we know $(x, S(z)) \in R_{FL}$ by definition of the function problem reduction where R_{FL} is the binary relation for which is based on the function problem FL. Therefore, we can affirm FL is in FPL through the string function $S = G$ which is computable in logarithmic space. In conclusion, we obtain $FP \subseteq FPL$. □

Theorem 2.2. *If $LSPACE \neq P$, then $FPL \subsetneq FNP$.*

Proof. Given a Boolean circuit C, the problem *CIRCUIT–SAT* consists in deciding whether there is some input such that C accepts [5]. The function problem associated with *CIRCUIT–SAT*, denoted *FCIRCUIT–SAT*, is in *FNP–complete* [5]. The verification problem of *FCIRCUIT–SAT* is precisely the *P–complete* problem *CIRCUIT–VALUE* [5]. However we know that any problem in *P–complete* cannot be decided in logarithmic space by a deterministic Turing machine under the assumption of $LSPACE \neq P$ [5]. Therefore, the identity string function cannot

be used in this case for the Definition 1.1. Moreover, this is not possible through any string function G computable in logarithmic space for the Definition 1.1. This is because the resulting language for the verification problem will be equivalent to the problem *CIRCUIT–VALUE* under the logarithmic computable function G. Certainly, the language associated to the function problem will be the same, that is the problem *CIRCUIT–SAT*. The only changes will be in the form of the certificate for the elements of *CIRCUIT–SAT*, therefore the resulting language for the verification problem will be a *P–complete* problem as well. Indeed, the functionality of the new certificate will be the same, that is the evaluation of a Boolean circuit for an input value which can be obtained transforming the new certificate in a truth assignment through the logarithmic computable function G. Consequently, we obtain if $LSPACE \neq P$, then $FPL \subsetneq FNP$ since under the assumption of $LSPACE \neq P$, the function problem *FCIRCUIT–SAT* is not in FPL through any string function G computable in logarithmic space. □

Theorem 2.3. *If $LSPACE \neq P$, then $FP \neq FNP$.*

Proof. Due to the Theorems 2.1, and 2.2, we obtain $FP \subseteq FPL \subsetneq FNP$. As result, we have $FP \neq FNP$. □

Theorem 2.4. *If $LSPACE \neq P$, then $P \neq NP$. Consequently, when $P = NP$ then $LSPACE = NP$.*

Proof. If $FP \neq FNP$, then $P \neq NP$ [5]. The proof is completed as a consequence of the Theorem 2.3. □

Theorem 2.5. *$P \neq PSPACE$.*

Proof. As result of Theorem 2.4, if $P = PSPACE$ then $LSPACE = PSPACE$. However, we know that $LSPACE = PSPACE$ is not possible due to the Hierarchy Theorem [5]. Thus, $P = PSPACE$ is not possible either. □

Theorem 2.6. *If $EXP \neq PSPACE$, then $P \neq NP$.*

Proof. It is a known result if $EXP \neq PSPACE$, then $LSPACE \neq P$ [5]. Hence, this is a direct consequence of Theorem 2.4. □

References

1. Sanjeev Arora and Boaz Barak, *Computational complexity: a modern approach*, Cambridge University Press, 2009.
2. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein, *Introduction to algorithms*, MIT press, 2009.
3. Raymond Greenlaw, H James Hoover, and Walter L Ruzzo, *Limits to Parallel Computation: P-completeness Theory*, Oxford University Press, 1995.
4. Mark W Krentel, *On finding locally optimal solutions*, [1989] Proceedings. Structure in Complexity Theory Fourth Annual Conference, IEEE, 1989, pp. 132–137.
5. Christos H Papadimitriou, *Computational complexity*, John Wiley and Sons Ltd., 2003.
6. Elaine Rich, *Automata, computability and complexity: theory and applications*, Pearson Prentice Hall Upper Saddle River, 2008.
7. Alejandro A Schäffer and Mihalis Yannakakis, *Simple Local Search Problems That are Hard to Solve*, SIAM journal on Computing **20** (1991), no. 1, 56–87.

Joysonic, Belgrade, Serbia
E-mail address: vega.frank@gmail.com