Draft version February 11, 2019
Preprint typeset using LaTeX style emulateapj v. 01/23/15

COMPUTATION, COMPLEXITY, AND $\mathbf{P} \neq \mathbf{NP}$ PROOF

Hugh Wang[1]
McGill University
Draft version February 11, 2019

ABSTRACT

If we refer to a string for Turing machines as a guess and a rejectable substring a flaw, then all algorithms reject similarly flawed guesses flaw by flaw until they chance on an unflawed guess, settle with a flawed guess, or return the unflawed guesses. Deterministic algorithms therefore must identify all flaws before guessing flawlessly in the worst case. Time complexity is then bounded below by the order of the product of the least number of flaws to cover all flawed guesses and the least time to identify a flaw. Since there exists 3-SAT problems with an exponential number of flaws, 3-SAT is not in **P**, and therefore $\mathbf{P} \neq \mathbf{NP}$.

The radiating paths problem, illustrated in Figure 1, offers insight into the **P** versus **NP** problem. Given a natural number n, I lay 2^n paths of length n^2 radiating from where you stand. Then, I place a stone at the end of each path, and may or may not bury some fruits under half of the stones. At your luckiest, you need to walk only a polynomial n^2 to be fruitful. At your unluckiest, you need to walk an exponential $(2^{n-1}+1)\cdot n^2$ to leave no stone unturned. That is, overlooking walking back to square one. Therefore, deciding if some fruits are within reach is in **NP** but not in **P**, hinting that $\mathbf{P} \neq \mathbf{NP}$.

We later shape this problem into an emblem of classical computation by analogizing walking down one of the 2^n paths of length m to deciding a truth assignment of a 3-SAT problem with n variables and m clauses. Suppose the analogy holds, then if the length of a path or the number of clauses decreases from n^2 to 1, you need to remember only 1 clause but walk an exponential $(2^{n-1}+1)\cdot 1$ to leave no stone unturned, and auxiliary space complexity decreases from **PSPACE** to **REG**, but not the time complexity. We see that the exponential number of unfruitful paths alone excludes the radiating paths problem from **P**.

On the other hand, the radiating directions problem, shown in Figure 2, is in **P** despite its exponential number of unfruitful paths. It differs from the previous problem in that, after I may or may not bury the fruits, I devise n^2 directions, not necessarily disjoint, that cover the unfruitful paths. Then, an unfruitful path implies that the paths in any of its directions are unfruitful, and at your unluckiest, you need to walk at most a polynomial $(n^2+1)\cdot n^2$ to leave no stone unturned. This problem is then in **P**, hinting that the number of directions to cover the unfruitful paths matters to time complexity.

Suddenly, the length of a path matters to the time complexity. If the length of the paths or the number of clauses is 2^n, then you need to walk an exponential $(n^2+1)\cdot 2^n$ and remember at most 2^n clauses to leave no stone unturned, and this problem is beyond **P** and maybe **PSPACE** too. Since you can but leave no stone unturned to decide this problem, the time complexity is the product of the least number of directions to cover all unfruitful paths and the time to decide a direction unfruitful, or the length of the path. Is this the number of clauses and therefore the auxiliary space complexity if the analogy holds?

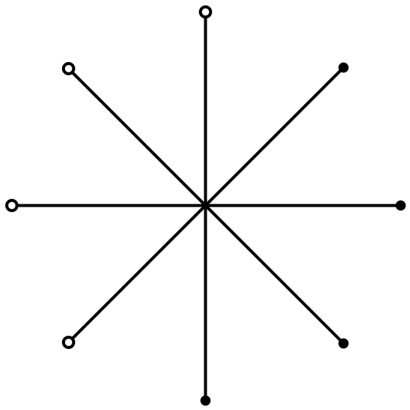

Figure 1. A radiating paths problem with $n = 3$ and therefore 2^3 paths of length 3^2.

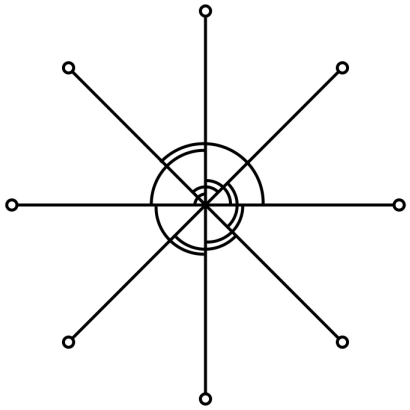

Figure 2. A radiating directions problem with $n = 3$ and therefore 3^2 directions.

[1] B.S. Joint Honours Mathematics and Computer Science, Interdisciplinary Life Sciences Minor, McGill University, Canada

We now support the analogy between walking down one of the 2^n paths of length m in a radiating directions problem and deciding one of the 2^n assignments of a 3-SAT problem with n variables and m clauses. Suppose you walk a unit length each time a classical computer transitions through the necessary states to decide a clause, the stone turning unfolds according to the decision of the final state, and the implications of an unfruitful path correspond to the implications of a false partial assignment. Then, deciding the radiating directions analogue of a 3-SAT problem is deciding the problem itself.

This analogy holds because walking and classical computation abide by the same physical laws. You cannot foresee what lies ahead of a path or be at two places at once, just as a classical computer cannot foresee what lies ahead of a computation or be in two states at once. Then at your unluckiest in 3-SAT, you must reject all unsatisfying partial assignments before accepting. Time complexity is then bounded below by the order of the product of the least number of flaws to cover all flawed guesses and the least time to identify a flaw. Is this the number of clauses and therefore the auxiliary space complexity?

If we refer to the analogue of an assignment in a general problem as a guess, that of a false partial assignment a flaw, then for Turing machines, a string is a guess, a rejectable substring a flaw. Also, all algorithms reject similarly flawed guesses flaw by flaw until they chance on an unflawed guess, settle with a flawed guess, or return the unflawed guesses. More, deterministic algorithms in the worst case identify all flaws before guessing flawlessly, so their time complexity is bounded below by the order of the product of the least number of flaws to cover all flawed guesses and the least time to identify a flaw.

We now consider an algorithm from each paradigm in Algorithm Design[1] as examples. In breadth-first search for connectivity, a guess is a path from s, a flaw an end not in t. In the cashier's algorithm, a guess is a sequence of coins, a flaw a coin not of largest value less than the unpaid amount. In mergesort, a guess is a permutation, a flaw a value succeeded by one smaller. In weighted interval scheduling, a guess is a subset of the intervals, a flaw a wrong inclusion. In Ford-Fulkerson, a guess is a residual network, a flaw a path not augmented. In global min cut, a guess is a cut, a flaw an edge probably wrong to cut.

We do the same for Artificial Intelligence: A Modern Approach[2]. In genetic programming, a guess is an individual, a flaw low fitness. In minimax, a guess is a move, a flaw trouble down the line. In backtracking, a guess is an assignment, a flaw an unsatisfied constraint. In forward chaining, a guess is a proof, a flaw a dead end. In bayesian inference, a guess is a probability, a flaw a wrong decimal. In supervised learning, a guess is a function, a flaw nonconformity to examples. In unsupervised learning, a guess is a set cover, a flaw an inclusion too different. In reinforcement learning, a guess is a strategy, a flaw bad experiences.

We hereinafter refer to the shortest runtime of a k-SAT problem as T, smallest auxiliary space S, the length of a guess G, the least number of flaws to cover all flawed guesses N, and the length of the logic optimization of the flaws in disjunctive normal form L. Since an inverted boolean circuit implementing the logic optimization of the flawed guesses in disjunctive normal form with a guessing circuit is the smallest computer to solve this problem, and the flawed guesses and flaws are equivalent in disjunctive normal form, $O(S) = O(L) \leq O(N \cdot G)$. Space complexity is then $O(G + L) \leq O(G + N \cdot G) = O(N \cdot G)$.

It is also clear that $O(T) \geq O(N)$ as each flaw requires time to identify, and an exponential N implies an exponential T. More, the shortest time to identify any flaw is the shortest time to decide a guess, which is at most $O(L)$. Then, $O(T) \leq O(N \cdot L) \leq O(N^2 \cdot L)$, and an exponential T implies an exponential N. Thus, the time complexity of a k-SAT problem is polynomial if and only if the least number of flaws to cover all flawed guesses is polynomial. More, $O(T) = O(N \cdot L) = O(N \cdot S) = O(N) \cdot O(S)$ for algorithms that do not prune, then their auxiliary space complexity is a dimension of their time complexity.

In a 2-SAT problem with n variables and m clauses, we can convert the m clauses into at most $2m$ implications and further into at most $2m$ implication chains that are at most $2m+1$ literals long. In an implication chain, once a literal is true, all that follow are true. A flaw is thus a break in the chain, and any deterministic algorithm in the worst case must identify at most $O(m^2)$ of them, leaving an $O(m^2)$ possible flaws left as each of the $2m$ chains has at most $2m + 1$ dissimilarly flawed guesses. Since the number of flaws to cover all flawed guesses is polynomial, so is the time complexity, and 2-SAT is in $\mathbf{P}$.

Finally, we construct a 3-SAT problem by converting $(a_i)_{i \in \mathbb{N}} = \sum_{j=2}^{i+1} j$ to binary and generating from a_i the ith flawed guess by assigning the kth digit to the kth variable. Since arithmetic sequences grow only quadratically, the number of flawed guesses grows exponentially with respect to the number of variables and therefore to the number of possible clauses too. More, the flawed guesses are by construction dissimilarly flawed, and therefore almost all flaws in themselves. Since the least number of flaws to cover all flawed guesses is exponential, so is its time complexity. Therefore, 3-SAT is not in $\mathbf{P}$, and $\mathbf{P} \neq \mathbf{NP}$.

REFERENCES

Jon Kleinberg and Eva Tardos. 1982, Algorithm Design. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Stuart Russell and Peter Norvig. 2009. Artificial Intelligence: A Modern Approach (3rd ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.