A POLYNOMIAL-TIME REDUCTION OF THE 3-SAT TO THE QUADRATIC CONGRUENCE PROBLEM AND OTHER RELATED PROBLEMS

Renato Lui Geh
NUSP: 8536030

Computational Number Theory (MAC6927)
Prof. Sinai Robins
University of São Paulo

Abstract. In this term paper for MAC6927 — Computational Number Theory, we explore the history behind the quadratic congruence problem (QCP) and other related number theoric problems; show a polynomial-time reduction from the 3-SAT to the QCP quoting Adleman and Mander's 1978 theorem [MA78], implying that quadratic congruence is NP-complete; and show some solved and unsolved problems in Number Theory that are directly (or indirectly) related to the QCP problem and its membership in NP.

1. History

German mathematician David Hilbert published [Hil02] in 1902 a set of 23 unsolved problems in mathematics he deemed to be most important unanswered mathematical problems of the 20th century. Since then 9 of them have been solved (at the time the author is writing this line and as far as the author is aware), 9 are considered partially solved, three of them are unsolved and two of them are considered too vague. Unsolved problems include the infamous Riemann Hypothesis and an extension to the Kronecker-Weber Theorem. Amongst solved problems is the 10th Hilbert problem.

10th Hilbert Problem: Given a diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: *to devise a process according to which it can be determined by a finite number of operations whether the equation is solvable in rational integers.*

It was answered in 1970 by Matiyasevich [Mat70] to be impossible. The question now becomes, in which cases is there an algorithm for solvability and what is the complexity of such algorithms? In 1976, Adleman and Manders [MA76] partially answered these questions by proving that, for the quadratic cases, there exists an algorithm and it is NP-complete. In their proof, they also found that, through a slight modification in the final step of their proof, it was possible to answer the

quadratic congruence problem. A cleaner version of this proof was published in 1978 by Adleman and Manders [MA78], proof we try to explain in this paper.

In this paper we focus on the second result of Adleman and Mander's 1978 article, but also show the main result, namely that the set of quadratic diophantine equations with natural numbers solutions is NP-complete. The proof is done through a polynomial-time reduction from the 3-SAT problem. This reduction implies that both problems covered in Adleman and Mander's article are in NP-complete.

In 1971, American mathematician Stephen Cook published "The Complexity of Theorem-proving Procedures" [Coo71], and in the next year, his fellow countryman Richard Karp published "Reducibility Among Combinatorial Problems" [Kar72]. The two articles introduced the concepts of P and NP classes, yielding the duo a Turing Award. Interestingly in 1973, on the other side of the Iron Curtain, Ukrainian Leonard Levin published [Lev73] in the USSR equivalent results to Cook's and Karp's, but considering search problems instead of decision problems (an interesting remark is that Levin did not receive a Turing Award for his work, despite having achieved equivalent results). Both works resulted in the following statement: that any problem in NP can be reduced in polynomial time by a deterministic Turing machine to the problem of satisfiability of a Boolean formula, i.e. the SAT problem. Additionally, if there exists a deterministic polynomial time algorithm for solving SAT, then every NP problem can be solved by a deterministic polynomial time algorithm.

This independent, parallel work from opposite parts of the world, ideologically and geographically, gave rise to what is considered one of the most important open questions in theoretical computer science, the P vs NP problem.

- Satisfiability (SAT)
  - 0–1 integer programming
  - Clique
    - Set packing
    - Vertex cover
      - Set covering
      - Feedback node set
      - Feedback arc set
      - Directed Hamiltonian cycle
        - Undirected Hamiltonian cycle
  - Satisfiability with at most 3 literals per clause (3-SAT)
    - Chromatic number
      - Clique cover
      - Exact cover
        - Hitting set
        - Steiner set
        - 3-dimensional matching
        - Knapsack
          - Job sequencing
          - Partition
            - Max cut

In his 1972 paper, Karp also published a list of 21 NP-complete problems in which he showed reductions implying its membership in the NP-complete class. The list above is Karp's 21 NP-complete problems, where the nesting indicates the direction of the reduction. For instance, the exact cover problem was reduced to the knapsack problem, chromatic number was reduced to exact cover, 3-SAT was reduced to exact cover, and the SAT was reduced to the 3-SAT problem.

From this we know that the 3-SAT is NP-complete. Not only that, any problem that is reduces 3-SAT is also in NP. In the next sections we will provide a brief review on polynomial-time reductions, present proper definitions on the QCP and 3-SAT, and prove the reduction. This section provided a brief historical overview, but we do not restrict historical remarks to only this section. When relevant, we provide short historical anecdotes on the subject.

2. Brief Review on Complexity Theory

In this section we define decision problems, the P and NP complexity classes and all the tools we need to prove a polynomial-time reduction. We give a shallow definition of the 3-SAT problem and give other examples of NP-complete problems.

Before we prove reductions, we first need to properly define the concepts we are going to use. We shall define a Problem as a question that takes a set of objects as input and returns another set of objects as output. There are many types of problems: decision problems, where the output is restricted to yes or no (or true or false) answers; optimization problems, where the answer is given by a particular element of a set such that such an object optimizes certain criteria; and search problems, which is when the answer is an element of the set of output answers.

We can give a format for problems:

Problem: vector sorting

Input $n \in \mathbb{N}$ and a vector $A[1..n]$ with n integers
Output An ordered vector
Sorting the vector A in increasing order.

It is natural to conclude that problems have a certain complexity attached to them. Furthermore, we can provide several algorithms that take the same input and yield the same output. In this case in particular we know that sorting a vector is $\mathcal{O}(n \ln n)$ at best. Examples of algorithms for the above problem are InsertSort, MergeSort and BubbleSort, with each one having different worst case complexities. We next show a few examples of different types of problems:

Search problem: greatest common divisor

Input $a, b \neq 0 \in \mathbb{N}$
Output $d|a$, $d|b$, and for all $d'|a$, $d'|b$ we have $d' \leq d$
Finding the $\gcd(a, b)$

The Euclidean algorithm is an example of an algorithm that solves the gcd problem, with $\mathcal{O}(\ln(\max\{a, b\}))$.

Optimization problem: longest common subsequence

Input Two strings $X[1..m]$ and $Y[1..n]$

Output A string

Finding the longest common subsequence in X and Y.

This classic computer science problem turns out to be $\Theta(m \cdot n)$. Following Cook and Karp's, we treat only the case for decision problems. One might think this severly restricts the generality of complexity classes, but it is easy to see that one can treat optimization and search problems as special cases of decision problems. We do this by restricting these problems into their decision problem subsets:

Decision problem: greatest common divisor

Input $a, b \neq 0 \in \mathbb{N}$ and $k \in \mathbb{N}$

Output Boolean value

Is $\gcd(a, b) = k$ true?

Decision problem: longest common subsequence

Input Two strings $X[1..m]$ and $Y[1..n]$

Output Boolean value

Is there an LCS of X and Y such that its length $\geq k$?

We simply added a k restriction on the input and modified the question so that the answer is a yes or no question.

For the next part we assume Turing machines to be already defined, as we wish to keep this section brief, as the title says. We consider an algorithm a series of steps that can be performed by a Turing machine. A problem is solvable in polynomial time if there exists an algorithm that takes a polynomial number of steps on the size of the instance. We define an instance as a particular input of a problem. In the gcd decision problem, we could take the tuple $(253, 37, 1)$ as an instance of the input. Both the Euclidean algorithm and the optimal 2-dimension LCS algorithm run in polynomial time on the size of the instance. However, the general case of the LCS algorithm is $\mathcal{O}(2^{n_1})$, where n_i is the length of the i-th string, and thus is not polynomial.

Definition 2.1. *The P class is the set of all decision problems that can be solved by polynomial (on the size of the instance) time algorithms.*

Let Π be a Problem. We say that Π admits a polynomial verifier $\mathcal{V}$ for a YES answer if there exists a polynomial algorithm that takes an instance I of Π and an object $\mathcal{C}$ such that the size of $\mathcal{C}$ is polynomial and returns YES for some $\mathcal{C}$ if the answer $\Pi(I)$ is YES and NO for all $\mathcal{C}$ if $\Pi(I)$ is NO.

In other words, $\mathcal{V}$ takes an instance I and checks whether such an instance is a true answer to the problem. We call the object $\mathcal{C}$ a polynomial certificate of problem Π.

Analogally, a polynomial verifier for a NO answer takes a polynomial certificate $\mathcal{C}$ and returns NO if the answer $\Pi(I)$ is NO and YES for all $\mathcal{C}$ if $\Pi(I)$ is YES.

Problem: Hamiltonian cycle

Input A graph $G = (V, E)$

Output Boolean value

Is there a Hamiltonian cycle, i.e. a path $p = (e_1, e_2, \ldots, e_k)$ s.t. $\forall v \in V$, v is visited by p exactly once, and p is a cycle?

Consider the Hamiltonian cycle problem above. In the image below, the polynomial certificate is the red path, and the polynomial verifier is an algorithm that checks whether the instance given is an actual solution to the problem. It is easy to see that a polynomial verifier for the Hamiltonian cycle is an algorithm that checks whether the set of edges in $\mathcal{C}$ traverse all the vertices and that the edges form a cycle.

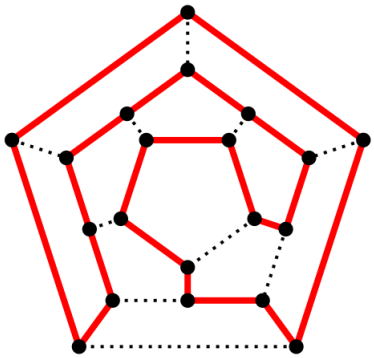

FIGURE 1. The red path is a polynomial certificate for the Hamiltonian cycle problem. Source: Wikipedia.

Definition 2.2. *The NP class is the set of all decision problems that admit a polynomial verifier for the YES answer.*

Definition 2.3. *The co-NP class is the set of all decision problems that admit a polynomial verifier for the NO answer.*

Note how every problem in P is already in NP and co-NP, since the algorithm that solves the problem could be used as a verifier for both the YES and NO answer.

Let Π and Π' be problems. A polynomial-time reduction from Π to Π' is an algorithm that solves Π using an algorithm that solves Π' as a subroutine, and that, excluding its subroutine, is polynomial on the size of its instance. We denote such a reduction by $\Pi \leq_p \Pi'$ if there exists a polynomial reduction, from Π to Π'.

Problem: SAT

Input A Boolean formula with n variables written in CNF, e.g. $(x_1 \vee x_2 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee x_5) \wedge \cdots$

Output Boolean value

Is there a valuation $\nu : \{x_1, x_2, \ldots, x_n\} \to \{\text{false}, \text{true}\}$ such that the input formula evaluates to true?

Problem: 3-SAT

Input A Boolean formula with n variables written in CNF with at most 3 literals in each clause, e.g. $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_1 \vee x_5) \wedge \cdots$

Output Boolean value

Is there a valuation $\nu : \{x_1, x_2, \ldots, x_n\} \to \{\text{false}, \text{true}\}$ such that the input formula evaluates to true?

We can prove that SAT $\leq_p$ 3-SAT by turning the clauses in the SAT problem into 3-SAT clauses [Kar72]. Recall Karp's 23 NP-complete list of problems. Each nesting means that the nested item Π' and its parent item Π obey $\Pi \leq_p \Pi'$. The SAT problem, can also be reduced to the Hamiltonian cycle problem by turning edges and vertices into SAT clauses. The vertex cover problem can also be reduced to the Hamiltonian cycle problem [Kar72].

Definition 2.4. *A problem* Π *in NP is NP-complete if* $\forall\ \Pi' \in NP$, $\Pi' \leq_p \Pi$.

From the Cook-Levin Theorem [Coo71; Lev73], we know that the SAT problem is NP-complete. So if $\Pi \leq_p \Pi'$, and Π is NP-complete, then Π' is also NP-complete. So showing that a problem Π' is NP-complete consists of the following steps:

(1) Take $\Pi' \in$ NP,
(2) Take Π NP-complete,
(3) Show that $\Pi \leq_p \Pi'$.

Definition 2.5. *A problem* Π *is NP-hard if the existence of a polynomial algorithm for* Π *implies on* $P = NP$.

So every NP-complete problem is NP-hard, but an NP-hard problem is not necessarily in NP.

We wish to show that the 3-SAT problem can be reduced to the quadratic congruence problem, proving that QCP is NP-complete.

3. The Reduction

References

[Coo71] Stephen Cook. "The Complexity of Theorem-proving Procedures". In: *STOC '71 Proceedings of the third annual ACM symposium on Theory of computing* (1971).

[Hil02] David Hilbert. “Mathematical problems”. In: *Bulletin of the New York Mathematical Society* 10 (1902).

[Kar72] Richard Karp. “Reducibility Among Combinatorial Problems”. In: *Complexity of Computer Computations* (1972).

[Lev73] Leonard Levin. “Universal Sequential Search Problems”. In: *Probl. Peredachi Inf.* 9 (1973).

[MA76] Kenneth Manders and Leonard Adleman. “NP-complete Decision Problems for Quadratic Polynomials”. In: *STOC ’76 Proceedings of the eighth annual ACM symposium on Theory of computing* (1976).

[MA78] Kenneth Manders and Leonard Adleman. “NP-Complete Decision Problems for Binary Quadratics”. In: *Journal of Computer and System Sciences* 16 (1978).

[Mat70] Yuri Matiyasevich. “Diofantovost’perechislimykh mnozhestv (Enumerable sets are diophantine)”. In: *Doklady Akademii Nauk SSSR* (1970).