Proof of Useful Work: Eine Nützliche Variante von Proof of Work

Jean-Marc Hendrikse

Abstract

Diese Dokumentation enthält eine sortierte Liste der wichtigsten LaTeX–Befehle. Die einzelnen Listeneinträge sind untereinander durch viele Querverweise verkettet, die ein Auffinden inhaltlich zusammengehöriger Informationen erheblich erleichtern.

1 Einführung

Durch das Aufstreben von Kryptowährungen und

While these sound attractive in theory, we can't discuss blockchain and climate in the same sentence without addressing the elephant in the room: proof-of-work.

The underlying protocol that drives many blockchain applications (including Bitcoin) has already caused significant environmental impact because of the amount of energy it takes to operate.

Many of us in the blockchain community (myself included) have turned a blind eye to this issue, in hopes that alternative consensus will come around sooner or later and perfectly erase the damage. But at Hack4Climate, I decided it was appropriate to find out the actual environmental impact of Bitcoin's proof-of-work.[https://blockchainatberkeley.blog/a-bitcoin-miner-and-an-environmentalist-walk-into-a-bar-bbdfb06a5232]

Collaborating with Carlos Càceres from Technical University of Berlin, the specific question we wanted to answer was "How much carbon dioxide is emitted for every Bitcoin that is mined?"

We started with the real-time Bitcoin hash rate data shown below. We used this to approximate the total tera hashes ever performed by the Bitcoin network.

Hauptaugenmerk auf Bitcoin. Bekannteste Kryptowährung. Hinter dem Konzept von Bitcoin steckt das Mining Verfahren, ein Verfahren, das dazu verwendet wird, um nachzuweisen, dass man eine gewisse Arbeit geleistet hat bevor man etwas in die Blockchain schreibt. Das ganze geschieht mich Rechenleistung. Sehr viel Rechenleistungen sogar. Aktuell (Stand 2017) benötigt das Minen von Bitcoins weltweit ..... . Wird die Blockchain größer dauert auch das Miningverfahren länger, sodass längere Zeit Energie aufgewendet werden muss. Aus ökologischer Sicht ist das höchst bedenklich. Manche sprechen hierbei sogar auch von einem "Umweltdistaster"[An13]. Doch wieso kann man dann nicht einfach auf das Protokoll verzichten, dass so viel Energie verschwendet? PoW ist ein sehr wichtiger Bestandteil von Bitcoin und ein wichtiges Verfahren für die Blockchain. Dabei führt ein Miner keine sinnvolle Berechnung durch, sodass dieser oder andere daraus keinen sinnvollen Nutzen ziehen können. Doch die Vorteile von Proof of Work liegen klar auf der Hand: es ist robust und erfolgreich über längere Zeit. Der nachteil des hohen Energieverbrauchs stößt allerdings zum Umdenken an, da es so nicht weitergehen kann.

In Kapitel [XYZ] gehe ich auf die Technologie, die dahintersteckt genauer ein.

1.1 Stand der Forschung

Mit dem Thema des Proof of Useful Work haben sich auch Ball et al. in ihrem Paper "Proof of Useful Work" beschäftigt. Dabei stellen die Autoren ein Framework vor, das als Grundlage für verschiedene Proof of Useful Work-Verfahren dienen sollen. Diese Arbeit nutze ich auch vor allem als Grundlage für meine Arbeit. Weitere Arbeiten gibt es auch in [1][2][3][4]. Bei [1] ... Bei [2] ... Ich werde vor allem

1.2 Ziele dieser Arbeit

Das Ziel dieser Arbeit ist das Finden eines Proof of Work Schemas, das zu einem gesellschaftlichen Nutzen führt und dessen Anwendung auf praktischen Nutzen prüfen. Dabei stelle ich in Kapitel 2 die Grundlagen vor. - Definition von Nutzen. Philosophische Frage - Blockchain und Bitcoin (Technologie) - Konsens-Mechanismen - Proof of Work - PrimeCOin - PermaCoin - Theorethische Informatik

In Kapitel 3 Gehe ich auf das Framework von Ball et al. ein und Definiere anhand dessen Proof of Useful Work - Proof of Useful Work - OV - Mögliche Kandidaten - Blockchain-Variante - Evaluierung

Abschließend in Kapitel 4 - Offene Fragen - Evaluierung der Konzepte - Praxistauglichkeit

Kapitel 5 Zusammenfassung

2 Grundlagen

2.1 Definition von Nutzen

Unterscheidung in gesellschaftlichen und privaten Nutzen. Unter Nutzen (Usefulness) verstehe ich im folgenden den Benefit, den man aus dem Gebrauch einer Sache ziehen kann[https://de.wiktionary.org/wi
Dadruch, dass es sich bei der Blockchain um ein verteiltest System in einer Community handelt, betrachte ich in meiner Arbeit die Sichtweise des gesellschaftlichen Nutzens. Mehrere Akteure handeln auf der Blockchain. Wieder unterscheide zwischen Sozialem oder wirtschaftlichen Nutzen. Die Betrachtung des gesellschaftlichen Nutzens führt sehr schnell zu einer Philosophischen Frage, auf die ich hier nicht genauer eingehende werde, sondern nur eine Definition ablegen, wie Nutzen im Rahmen meiner Arbeit gewählt wird.

2.2 Blockchain und Bitcoin

Grob gesprochen handelt es sich bei der Blockchain um ein dezentrales öffentliches Hauptbuch (engl. public ledger). Um bislang Transaktionen durchführen zu können, vertrauten „vertrauenswürdigen" Instanzen, beispielsweise Banken, denen wir unser Geld oder Daten zur Verwaltung übergaben. Diese dritten Instanzen besitzen einen permanenten Zugriff auf private Besitztümer und können jeder Zeit auf diese zugreifen. Mithilfe der Blockchain Technologie wird eine neue Möglichkeit offenbart, vom zentralen Gedanken der Systeme auf ein dezentrales System umzusteigen. Im wesentlichen handelt es sich bei der Blockchain um eine dezentrale Datenbank in einem Peer-to-Peer Netzwerk von Computersystemen. Was der Inhalt der Daten auf dieser Datenbank ist spielt dabei keine Rolle. Es kann sich um Transaktionsdaten einer Überweisung, digitale Ereignisse oder um Grundbucheinträge handeln. Anders als es bei herkömmlichen Datenbanken jedoch der Fall ist, wird die Blockchain nicht auf einem einzelnen zentralen Server, sondern verteilt auf mehreren Rechnern eines großen Netzwerks,

den Netzwerkknoten, gehalten. Jeder Teilnehmer dieses Netzwerks erhält bei Eintritt in die Blockchain eine lokale Kopie von allen Einträgen. Der Entwurf der Blockchain sieht wie folgt aus: Jede Transaktion wird zu einem Block zusammengefasst und mittels kryptographischer Berechnungen mit anderen Transaktionsblöcken zu einer Kette, der „Block-Chain“, verbunden. Ein einzelner Block besteht sowohl aus einem Zeitstempel, Transaktionsdaten sowie aus einem kryptographischen Hash des vorhergehenden gültigen Blocks. Bei der Neuaufnahme eines neuen Blockes in die Blockchain muss per Konsens die Mehrheit aller Teilnehmer des Systems verifizieren, dass dieser Block gültig ist, also dass diese Transaktion tatsächlich stattgefunden hat. Nur und nur dann kann dieser Block aufgenommen werden. Dadurch dass jeder Teilnehmer eine Kopie der aktuellen Blockchain-Historie enthält, kann er diese mit der neuen abgleichen und verifizieren. So kann eine mehrheitliche Einigung zwischen den Knoten geschaffen werden. In der Blockchain wird immer die gesamte Historie an Transaktionen, die jemals durchgeführt wurden, chronologisch linear erweitert. Die kryptographische Verkettung von Transaktionen bildet das Fundament vieler Kryptowährungen. Das wohl bekannteste Beispiel ist die digitale Währung Bitcoin, die im November 2008 mit dem Titel „Bitcoin: A Peer-to-Peer Electronic Cash System”1 von einer unbekannten Einzelperson oder Gruppe mit dem Pseudonym Satoshi Nakamoto entwickelt wurde. Viele bis dahin vorangehende Digitale Währungen scheiterten daran, dass sie durch eine zentrale Instanz verifiziert werden mussten, um das Double Spending Problem zu lösen. Mithilfe der Blockchain-Technologie konnte Nakamoto einen Ansatz beschreiben, der eben dieses Problem löste.

2.2.1 Konsensmechanismus

In zentralisierten Organisationen werden Entscheidungen meistens getroffen, indem eine Instanz die Entscheidung trifft. Da in der Blockchain keine zentrale Entschiedungsinstanz existiert bzw. auf diese bewusst verzichtet werden soll, müssen Entscheidungen anders getroffen werden. Im Fall von Bitcoin wird ein mehrheitlicher Konsens getroffen, indem ein Konsensmechanismus eingeführt wird. Ein Konsensmechanismus bezeichnet einen Algorithmus, der eine Einigung über den Status des Netzwerks zwischen den beteiligten (meistens anonymen) Netzwerkknoten schafft. Dies ist die Basis der Blockchain Technologie. Dabei wird sichergestellt, dass alle Teilnehmer eine identische Kopie der Blockchain besitzen. Wie bereits im vorangegangenen Kapitel erwähnt erschwert ein Konsensmechanismus die Manipulation von Daten erheblich. Der Hauptgedanke der dahinter steckt ist, dass durch einen dezentralen Konsens-Mechanismus ein zentrale Kontrollinstanz zur Integritätsbestätigung obsolet wird.

2.2.2 Proof of Work

Das Grundprinzip von Proof-of-Work basiert auf der Idee, dass Miner im Netzwerk nachweisen müssen, dass sie einen gewissen Aufwand in Form von Arbeit erbracht haben. Dabei wird Arbeit in Form von Energieverbrauch gemessen. Durch einen konkurrierenden Wettstreit untereinander versuchen Miner ein extrem schweres kryptographisches Puzzle zu lösen. Der erste Miner, der dieses Puzzle löst gewinnt und kann den neuen Block in die Blockchain anfügen. Damit Miner einen Anreiz bekommen an diesem Wettstreit teilzunehmen, bekommt jeder Gewinner einen Reward, im Fall von Bitcoin sind es 12.5 neu erstellte Bitcoins und eine kleine Transaktionsgebühr. [https://www.coindesk.com/short-guide-blockchain-consensus-protocols/]

Doch Proof of Work ist keine Neuerfindung von Bitcoin sondern wurde auch bereits für den E-Mail-Versand als Mechanismus gegen Spaming eingesetzt. Um eine E-Mail zu versenden, musste der Sender vereinfacht gesagt zusätzliche Berechnungen durchführen. Da die Maschinen in ihrer Rechenkapazität beschränkt waren, ist es notwendig gewesen, hohe Investitionen

tätigen, um die Berechnungen durchzuführen und letztlich E-Mails zu versenden. Dadurch sollten die Kosten zum Versenden von E-Mails hoch genug sein, dass sich der Versand für Spammer nicht mehr lohnt. Denn der Empfänger einer E-Mail konnte auf einfache Art prüfen, ob die Berechnungen korrekt durchgeführt wurden, während die Berechnungen für den Sender mit einem gewissen Aufwand verbunden waren.

Definition 1. *Ball et al. definieren drei wesentliche Algorithmen, die POW ausmachen:*

1. `Gen(`1^n`)`*, randomisierter Algorithmus, der eine challenge c erzeugt.*
2. `Solve(c)`*, Algorithmus, das die Challenge c entgegen nimmt und eine Lösunge s dafür findet.*
3. `Verify(c,s)`*, verifiziert die Lösung s für die challenge c.*

Dabei ist die Bedinung, dass `Gen` und `Verify` eine schnelle lineare Laufzeit als obere Schranke besitzen und für `Solve` soll eine Hardness1FUSSNOTE ICH BELASSE HIER DIE ENGLISCHE NOTATION (Schwere), also das schwere Finden einer Lösung, gelten. Dabei wird eine Zeit $t(n)$ vorgegeben, die im Schnitt die Arbeitszeit eines Algorithmus vorgibt. Somit soll `Solve` eine Lösung berechnen, die von `Verify` akzeptiert wird, andererseits soll `Verify` mit hoher Wahrscheinlichkeit alles ablehnen, das die Zeit von $t(n)^1 - \epsilon$ für jedes $\epsilon > 0$ unterschritten hat. Diese Bedingungen erschweren allerdings die suche nach nützlichen Problemen. IM Fall von Bitcoin beispielsweise wird soll als Challenge c ein SHA256-Hash berechnet werden, der eine gewisse Anzahl an führenden Nullen enthält.

Die Idee hinter der Blockchain ist, ein dezentrales Transaktionsbuch zu speichern. Daraus ergibt sich ein Vertrauensproblem, da viele Teilnehmer des Netzwerks in der Regel auf eine zentrale „trusted party“ vertrauen müssen. Um diesen unerwünschten Nebeneffekt zu umgehen, nutzt man Konsens-Algorithmen, die es ermöglichen, dass jeder Teilnehmer prüfen kann, ob seine Blockchain den Regeln entspricht.

2.2.3 Difficulty

Die Difficulty (genauer behandelt im Thema “Mining”) ergibt sich aus der Schwierigkeit, den gewünschten Output der Hashfunktion zu finden. Im Fall von Bitcoin wird bspw. vorgegeben, wie viele Nullen der Output am Anfang des Strings besitzen muss. Je mehr Nullen gefordert sind, desto schwieriger wird es schließlich, den Output zu finden. Das lässt sich leicht mit einem Lottospiel veranschaulichen: Zwei Richtige zu treffen ist wesentlich einfacher, als Sechs Richtige der vorgegebenen Zahlen zu treffen.

2.2.4 Sicherheit

PoW's security relies on the principle that no entity should gather more than 50% of the processing power because such an entity can effectively control the system by sustaining the longest chain. We now briefly outline known attacks on existing PoW-based blockchains. First, an adversary can attempt to double-spend by using the same coin(s) to issue two (or more) transactions—thus effectively spending more coins than he possesses. Recent studies have shown that accepting transactions without requiring blockchain confirmations is insecure [23]. The more confirmations a transaction obtains, the less likely this transaction will be reversed in the future. Second, miners might attempt to perform selfish mining [15] attacks in order to increase their relative mining share in the blockchain, by selectively withholding mined blocks and only gradually publishing them [15, 32]. Recent studies show that, as a result of these

attacks, a selfish miner equipped with originally 33% mining power can effectively earn 50% of the mining power. Double-spending attacks and selfish mining can be alleviated if all nodes in the blockchain system are tightly synchronised. Note that, in addition to network latency, synchronisation delays can be aggravated due to eclipse attacks [18,19] where an adversary creates a logical partition in the network, i.e., provides contradicting block and transaction information to different blockchain network nodes.

Erwähne unbedingt double spending!

2.2.5 The Byzantine Fault Tolerance

TODO Als byzantinische Fehler bezeichnet man in der Informationstechnik Fehler, bei denen sich ein System beliebig falsch verhalten kann. Beispielsweise schickt ein Server gelegentlich falsche Antworten und erreicht gelegentlich falsche Systemzustände. Ein byzantinischer Fehler beschreibt im Allgemeinen ein schwierig zu erfassendes Fehlermodell.

In Mehrprozessor-Systemen bezeichnet der byzantinische Fehler eine Fehlerklasse. Falls eine Komponente an verschiedene Prozessoren unterschiedliche (protokollkonforme) Ergebnisse liefert, spricht man von einem byzantinischen Fehler. Bei der Planung wird davon ausgegangen, dass x Prozessoren bösartig arbeiten und das System maximal stören wollen.

2.3 Altcoins

2.3.1 Primecoin

Der größte Kritikpunkt der immer wieder im Zusammenhang mit Bitcoin und dessen Proof of Work Protokoll unter Experten auftaucht ist, dass die Berechnung zum Minen von Blöcken, also das Lösen einer Challenge, keinen nachhaltigen Wert besitzt ■BELEGE■>. Wie schon bereits erwähnt hat das Minen keinen anderen Sinn als zu beweisen, dass Energieaufgewendet wurde und man sich so dafür qualifiziert Blöcke zur Blockchain hinzuzufügen. Trotzdem ist das Proof of Work Protokoll das Herzstück von Bitcoins Sicherheit. Denn würde es Proof of Work nicht geben, so könnte ein kompromitierter Angreifer sich als Millionen von Bitcoin Knoten zur selben Zeit ausgeben und somit ernsthaft Bitcoins Transaktionsblöcke angreifen oder austauschen. Allerdings werden lediglich SHA256 berechnet, wodurch eine große Menge an Energie verschwendet wird. Primecoin ist die erste Kryptowährung, die auf dem Proof of Work Ansatz basiert und eine effiziente Lösungen für Aufgaben generiert, die weiter verwendet werden können und somit einen Nutzen liefern [Primecoin paper]. Statt also einer nutzlosen Berechnung eines SHA256 Hashes1 müssen Miner beim Proof of Work Protokoll von Primecoin lange Primzahlenketten finden. Es gibt drei Arten von Primzahlenketten, die sich für das Proof of Work Protokoll eignen: Cunningham Chain (CC) 1. Art, CC 2. Art und Bi-twin chain.

Cunningham Chain 1. Art. Bei der Cunninghamchain muss jede Primzahl der Kette um ein größer als das doppelte der vorhergehenden Primzahl sein:

Beispiel Kette der Länge 5: 1531, 3061, 6121, 12241, 24481

Cunningham Chain 2. Art. Die Cunningham Chain 2. Art gibt vor, dass jede Primzahl um eins verringert als das doppelte des Vorgängers ist.

Beispiel-Kette der Länge 5: 2, 5, 11, 23, 47

Bi-Twin Chain. In der Bi-Twin chain werden Paare gehalten, deren Differenz 2 ergibt.

Beispiel-Kette: 211049, 211051, 422099, 422101, 844199, 844201

Nun könnte man sich hier die Frage stellen wo der Nutzen in Berechnungen von Primzahlen liegt oder ob es lediglich eine ebenso sinnlose Berechnung wie die Berechnung von SHA256 Hashes ist. Doch es gibt viele Anwendungsgebiete aus der Mathematik, die sich mit der Suche nach großen Ketten von Primzahlen beschäftigt. Unter anderem die Suche nach Mersenne Primzahlen oder das Primzahlen Theorem2. Nachfolgend seine Liste der University of Tennesseee erwähnt, die Gründe für die Suche nach Primzahlen liefert: http://primes.utm.edu/notes/faq/why.html

1 Dabei wird mit nutzlos im Sinne für Weiterverwendung gemeint 2 Fußnote

2.3.2 Permacoin

Einen ganz anderen Ansatz zur nützlichen Alternative zu Bitcoin im Vergleich zu Primecoin liefert Microsoft in Zusammenarbeit mit der University of Maryland mit Permacoin, bei dem die Blockchain als verteilter Speicher von Archivdaten dienen soll. Miner sollen nicht mehr nur reine Rechenleistung aufwenden sondern vielmehr ungenutzten Speicherplatz zur Verfügung stellen, weshalb er vielmehr dem Protokoll Proof of Retrievabilty ähnelt.

Der Grundgedanke, der hinter Permacoin steckt ist, dass statt Rechenleistung Speicherplatz für zum Minen von neuen Block aufgewendet wird. Im POR Protokoll wird bewiesen, dass ein Knoten des Netzwerks Speicherressourcen für das Abspeichern von Dateien oder Dateifragmenten aufwendet. Die Autoren von [Permacoin] schlagen ein Verteiltes Daten-Archiv vor, das so vor Datenverlusten geschützt werden soll.

2.4 NP-schwere Probleme

Wir greifen den Gedanken von nützlichen Berechnungen auf und wollen die nutzlose SHA256-Hash-Berechnung ersetzen gegen etwas, das weiter verwendet werden kann. Wie bereits im Abschnitt zum Proof of Work Protokoll erwähnt gibt es gewisse Probleme in der Informatik, die NP-schwer sind, das heißt Probleme, die entweder gar nicht lösbar sind oder dass es sich um eines der Schwierigsten Probleme der Klasse NP handelt. Die Komplexitätsklassen vieler Probleme dieser Seminararbeit behandelt eben solche Probleme.

Definition 2. *Ein Entscheidungsproblem E nennt sich NP-schwer wenn es für jedes Problem P aus NP eine polynomielle Reduktion von P nach E existiert.*

2.4.1 k-Orthogonale Vekotr Problem

Ein beliebtes NP-schweres Problem aus der theoretischen Informatik ist das k-Orthogonale Vekotren Problem.

2.4.2 3SUM

The 3sum problem is defined as follows: given a set S of n integers, do there exist three elements a, b, c S such that a + b + c = 0? This is a linear satisfiability problem. 3sum can be solved using a simple algorithm with (n 2) runtime (sort S, then test 3-tuples intelligently) and it is widely believed that (n 2) is the lower bound for the worst-case runtime of any solution to the 3sum problem.

2.4.3 All Pairs Shortest Path (APSP)

2.4.4 Eigenschaften von Graphenproblemen

3 Proofs of Useful Work

Ein Proof of Useful Work (uPoW) muss wie ein generelles PoW der Hardness-Anforderung gerecht werden, damit gewährleistet ist, dass der Prover tatsächlich Abreit aufgewendet hat. Darüberhinaus muss uPoW aber auch einen Nutzen (engl. Usefulness) erzeugen, der schnell verifizierbar und einfach rekonstruierbar ist.

Zwei wichtige Ziele müssen für ein Proof of Useful Work gelten:

- Hardness: Challenges können so verteilt werden, dass eine Lösung für die Challenge Aufwand in Form von Arbeit oder Energie garantiert.
- Usefulness: Aufgaben können als Challenge so delegiert werden, dass die Lösung der Worker schnell und effizient verifizierbar rekonstruiert werden kann.

Beide Ziele müssen immer gelten und zusammen betrachtet werden, da ohne Hardness kein Proof of Work möglich ist und ohne Usefulness wir keinen Proof of Useful Work erreichen. Wenn beispielsweise nur einfache Challenges generiert werden, die ein Worker ohne intensive Rechenleistung löst, bricht die Bedingung für den Nachweis der Arbeit (Hardness). Das ist allerdings die Grundvoraussetzung für Proof of Work. Wird dagegen Hardness gewährleistet aber Challenges zufällig geneiert werden, sodass sie nicht mehr rekonstruiert werden können, geht der Nutzen verloren ein bestimmtes Problem zu lösen und man verstößt gegen die Usefulness. Bei Bitcoin's Proof of Work werden Challenges mit dem Algorithmus `Gen()` zufällig generiert, was wiederum das Mapping von x auf $f(x)$ erschwert. In einem Proof of Useful Work Protokoll benötigen wir allerdings zusätzlich noch einen Algorithmus der die Verbindung zwischen einer Lösung und Challenge wiederherstellt und ein $f(x)$ rekonstruiert. [BRSV17] definieren dafür einen Algorithmus `Recon(`c_x`, s)` zur Rekonstruktion von $f(x)$. Dieser nimmt eine Challenge c_x entgegen und rekonstruiert basierend auf der dieser Challenge $f(x)$. Für die Nachfolgende Definition 3 ergeben sich 4 Algorithmen:

- `Gen(x)`
- `Solve(`c_x`)`
- `Verify(`c_x`, s)`
- `Recon(`c_x`, s)`

Definition 3 (Proof of Useful Work)**.** *Für die vier Algorithmen* `Gen`*,* `Solve`*,* `Verify` *und* `Recon` *eines gültigen Proof of Useful Work (uPoW) soll gelten:*

- *Efficiency (dt. Effizienz:* `Gen(x)` *läuft für*
- *Completeness (Vollständigkeit): Pr[Verify(c,s) = accept] = 1*
- *Soundness (Korrektheit): Pr[Verify(c,s) = accept] = 1*
- *Hardness:*
- *Usefulness:*

Ohne Hardness und Usefulness beschreibt die Definition 3 ein Proof-System wie es auch [Will16] beschreibt. Für die gewährleistung eines Proof of Useful Work Frameworks müssen Hardness und Usefulness immer enthalten sein. Ein einfaches Proof of Work würde ohne dem `Recon`-Algorithmus und ohne der Bedingung von Soundness auskommen. Bei einem PoW sind wir nicht daran interessiert, ob eine Lösung korrekt ist, sondern nur ob das FInden der Lösung Arbeit gekostet hat, das wiederum durch Hardness gegeben ist. Umso wichtiger wird diese Bedingung für einen uPoW-Schema, bei dem es uns daran gelegen ist aus einer Berechnung einen Mehrwert zu ziehen und nicht Fake-Berechnungen zuzulassen, die wir gar nicht erhalten wollten. Ball et al. beweisen Proofs of Useful Work für das k-Orthogonale Vektoren Problem, 3SUM und APSP. Darüberhinaus lassen sich weitere NP-schwere Probleme auf diese Probleme reduzieren. Besonders Eigenschaften aus der Graphentheorie passen auf das kOV-Problem

3.1 Herausforderungen

3.2 OV-Problem

Besonders Graphenprobleme können sehr gut auf das OV-Problem reduziert werden.

Definition 4 (k-Orthogonale Vektoren)**.** *Gegeben sind k Mengen $(U_1, ..., U_k)$, für ein $k \geq 2$ mit $k \in$, von n Vektoren der Dimension d $\{0,1\}^{d(n)}$. Dann heißt eine solche Menge von Vektoren k-orthogonal, wenn $u^s \in U_s$ für $s \in [k]$, sodass sich*

$$\sum_{l \in [d(n)]} u_l^1, ..., u_l^k = 0$$

ergibt.

Da das k-OV-Problem allerdings worst-case hart ist und es keine Aussagen zum Average-Case gibt, reduziert [BRSV17] das Problem auf ein verwandtes Problem, das sowohl im Worst-Case als auch im Average-Case hart ist. Dafür definiert [BRSV17] ein Polynom $gOV^k_{n,d,p}$: $F_p^k nd \longrightarrow F_p$, wobei p jede Primzahl sein kann. Alle weiteren Parameter sind zum k-OV-Problem äquivalent. Für eine Berechnung von gOV gilt dann:

$$gOV^k_{n,d,p}(U_1, ..., U_k) = \sum_{i_1,...,i_k \in [n]} \prod_{l \in [d]} (1 - u^1_{i_1 l} \cdots u^k_{i_k l})$$

3.3 Mögliche Kandidaten

Mögliche Kandidaten sidn:

- OV
- 33sum
- APSP
- Proteinfolding
- Aliensuche

3.4 Blockchain-Schema

3.5 Evaluierung der Kandidaten

4 Evaluierung und Diskussion

5 Zusammenfassung und Ausblick

Literatur

[BRSV17] Michael Ball, Alon Rosen, Manuel Sabin und Prashant Nalini Vasudevan. *Proofs of Useful Work*. Columbia University. 2017.