**Ausarbeitung im Kurs der Ferienakademie 2014 „Moderne Suchmethoden der Informatik“ zum Thema:**

*Counting Number of Knapsack Solutions*

von Julius Kobold

Seminarleiter: Prof. Wanka, Prof. Mayr, Manuel Schmitt

**Thema:**

Es werden zwei Approximationsalgorithmen vorgestellt, um die Zahl der möglichen Lösungen beim Rucksackproblem zu zählen. Dabei werden die Ansätze von M. Dyer [1] und P. Gopalan [2] vorgestellt. Die Beweise werden skizziert, aber es sollen nicht die Beweise aus den Original-Papers ersetzt werden.

**Einführung:**

Gegeben seinen Gewichte und eine Kapazität . Sei und die Potenzmenge von W.

Gesucht ist .

In der Anschauung sagt man, dass eine Teilmenge von Gewichten aus eine gültige Bepackung für einen Rucksack mit der Kapazität ist, wenn die Summe der Gewichte aus nicht größer ist als . Gesucht ist also die Zahl der gültigen Bepackungen für den Rucksack mit der Kapazität .

Dieses Problem ist NP-schwer. Deshalb untersuchen wir zwei Approximationsalgorithmen. Zuerst betrachten wir einen randomisierten Algorithmus von M. Dyer und im Anschluss einen deterministischen Algorithmus von P. Gopalan. Bei beiden Algorithmen handelt es sich um streng polynomielle Approximationsschemata, das heißt, die Komplexität ist polynomiell in der Eingabe und polynomiell in , wobei der relative Fehler der approximativen Lösung kleiner ist als und einen Parameter für den Algorithmus darstellt. Der randomisierte Algorithmus von Dyer hat eine Komplexität von und der deterministische Algorithmus von Gopalan hat eine Komplexität von .

**Randomisiertes polynomielles Approximationsschema:**

Wir betrachten zunächst die Idee, die hinter dem Algorithmus steckt. Sei . Wir definieren die Funktion .

Dabei ist die exakte Lösung für das Problem, das der Algorithmus approximativ lösen soll. lässt sich mithilfe dynamischer Programmierung berechnen, da sich die Funktion wiefolgt rekursiv darstellen lässt:

Der Rekursionsanfang stimmt deshalb, weil genau zwei Teilmengen hat, die leere Menge und die Menge die genau enthält. In unserer Interpretation bedeutet das, dass wenn ist, dass es dann zwei gültige Bepackungen mit Gewichten aus für einen Rucksack der Größe c gibt und ansonsten nur die leere Menge eine gültige Bepackung ist. Die Gültigkeit der rekursiven Formel lässt sich wie folgt erklären: Die Menge der gültigen Bepackungen einen Rucksacks der Kapazität mit Gewichten aus setzt sich zusammen aus den gültigen Bepackungen, die nicht enthalten (also alle gültigen Bepackungen mit Gewichten aus ) und den gültigen Bepackungen, die und weitere Gewichte aus enthalten (wobei diese weiteren Gewichte in einen Rucksack passen müssen, der eine Kapazität von hat). Formal bedeutet das:

Um mit einem dynamischen Programm zu berechnen, werden tabellarisch alle Funktionswerte berechnet, mit und . Dazu sind Tabelleneinträge notwendig. Dieser exakte Algorithmus ist also pseudopolynomiell, da die Zahl C in binär codiert exponentiell groß zur Eingabe sein kann.

Um die Mächtigkeit von , die Menge aller gültigen Bepackungen, zu approximieren, wollen wir das Monte-Carlo-Verfahren verwenden. Sei , mit modifizierten Gewichten und sei . lässt sich wie oben gezeigt mit einem dynamischen Programm berechnen, mit Tabelleneinträgen, bei konstantem Zeitaufwand für die Berechnung eines Tabelleneintrags. Sei und mithin . Wir zeigen im Folgenden, dass und und dass in polynomieller Zeit in n gleichverteilt ein Element aus erzeugt werden kann und geprüft werden kann, ob dieses in ist. Damit kann dann das Monte-Carlo-Verfahren genutzt werden.

Es gilt , denn für alle gilt:

Sei . Dann ist , weil für alle gilt und deshalb . Außerdem sei eine Funktion, die wie folgt definiert ist:

Die Funktion ist wohl definiert, weil wenn , dann und dann gibt es mindestens ein , sodass . Sei nun , dann gilt:

Dann ist . Daraus folgt: . Die Funktion f ist also surjektiv. Es gibt für jedes Element y aus maximal Elemente aus , die darauf abbilden, da jedes solche , mit , sich von y entweder gar nicht unterscheidet oder genau ein Gewicht mehr beinhaltet und dafür gibt es Möglichkeiten. Das bedeutet wiederum, dass .

Wir zeigen nun, dass in polynomieller Zeit ein zufälliges Element aus berechnet werden kann, wobei jedes Element aus dabei mit gleicher Wahrscheinlichkeit berechnet wird. Seien und .

Mit Wahrscheinlichkeit setze und mit Wahrscheinlichkeit setze . Für : mit Wahrscheinlichkeit setze und mit Wahrscheinlichkeit setze . Das geht in polynomieller Zeit zu berechnen, da nur einmal die Funktion F mit dynamischer Programmierung als Tabelle berechnet werden muss. Die Wahrscheinlichkeiten ergeben sich dadurch, dass es genau gültige Bepackungen gibt, welche die Gewichte enthalten und es genau gültige Bepackungen gibt welche die Gewichte und nicht das Gewicht enthalten und es genau gültige Bepackungen gibt, welche die Gewichte enthalten.

Zu jeder Stichprobe lässt sich offensichtlich prüfen, ob . Sei die charakteristische Funktion, ob . Ziehe Stichproben . Berechne . Z ist die Ausgabe für den Algorithmus. Das ist das Ergebnis der Monte-Carlo-Methode. Für gilt nach Estimator-Theorem der Monte-Carlo-Methode:

**Deterministisches polynomielles Approximationsschema:**

Wir definieren wieder ein exaktes dynamisches Programm zur Lösung des Problems, das allerdings wieder nur in pseudopolynomieller Zeit berechnet werden kann. Die Idee ist diesmal nicht wie im vorherigen Algorithmus, die Lösung auf ein einfacheres Problem zurückzuführen, wofür sich das dynamische Programm berechnen lässt, sondern nun soll die Tabelle des dynamischen Programms nur stichprobenhaft und approximativ berechnet werden.

Sei . Dabei ist das kleinste , sodass mindestens unterschiedliche Lösungen für das Rucksack-Problem mit Gewichten und Kapazität existieren. Wir werden zeigen:

Außerdem definieren wir:

Dabei ist . Um zu berechnen, also um zu berechnen, ab welcher Kapazität es mindestens Lösungen mit den gegebenen Gewichten gibt, ist es hilfreich, zu unterscheiden, wieviel Prozent der mindestens a Lösungen enthalten und wie viel Prozent nicht. In der Rekursionsgleichung wird das minimale solche gewählt, also von allen Aufteilungen in und Prozent diejenige, bei der die benötigte Rucksackkapazität am kleinsten ist.

Dabei gibt einem die minimale Rucksackkapazität zurück, für die es irgendwelche unterschiedliche Lösungen gibt, von denen jede nur eine Teilmenge an Gewichten aus enthält. Während einem die minimale Rucksackkapazität zurückgibt, für die es irgendwelche unterschiedliche Lösungen gibt, von denen jede das Gewicht und eine Teilmenge aus enthält. Das Maximum von und ist eine Rucksackkapazität, zu der es Lösungen ohne und Lösungen mit und somit insgesamt unterschiedliche Bepackungen gibt, die reinpassen.

Also ist .

Da für alle gilt: , gilt auch und und mithin . Deshalb stimmt die rekursive Gleichung.

Da für eine Berechnung von mit dynamischer Programmierung die benötigte Tabelle zu groß würde, soll eine neue Funktion definiert werden, für die dieselbe rekursive Beziehung wie für gelten soll, die aber nur für eine eingeschränkte Menge an Werten definiert sein soll, anstatt für alle . Wir wollen zeigen, dass approximativ angibt, was das kleinste ist, sodass mindestens unterschiedliche Lösungen für das Rucksack-Problem mit Gewichten und Kapazität existieren. Sei und . Jedes steht repräsentativ für einen Wert , sodass den Wert approximieren soll. Wenn nun eine minimale Rucksackkapazität gesucht ist, für die Lösungen mit den Gewichten existieren, also der Wert von , soll dieser durch approximiert werden. ist dabei rekursiv definiert. Sei und . Die rekursive Definition von greift dabei anstatt auf irgendwelche Approximationen von und , was der rekursiven Gleichung für entsprechen würde, auf und zurück, wobei und größtmöglich gewählt sind, sodass und definiert sind. Genau genommen ist wie folgt definiert:

Dabei werden die sinnvollerweise für den Algorithmus zu betrachtenden Größen des Lösungsraumes von reduziert auf Potenzen . Es kann dann für alle und in polynomieller Zeit berechnet werden und damit und es kann als Ergebnis des Algorithmus ausgegeben werden. Wir zeigen nun, dass das eine Approximation für das Problem ist.

Wir zeigen induktiv:

Der Induktionsanfang stimmt per Definition. Nach Induktionshypothese gilt:

und

Daraus folgt:

=

Außerdem gilt analog:

und

Daraus folgt:

=

Damit ist der Induktionsschritt vollständig.

Sei die exakte Lösung für unser Problem. Wir zeigen, dass das von unserem Algorithmus berechnete eine (1+)-Approximation von ist. Nach Definition von und der eben gezeigten Abschätzung gilt:

Das bedeutet, dass der kleinste Rucksack , für den es verschiedene gültige Bepackungen gibt, größer ist als C, also dass . Das bedeutet . Wir zeigen nun :

Dazu stellen wir fest, dass ebenfalls nach Definition von und unserer Abschätzung gilt:

Das bedeutet, dass der kleinste Rucksack, für den es verschiedene gültige Bepackungen gibt, nicht größer ist als C und dass es somit auf jeden Fall gültige Bepackungen für einen Rucksack mit der Kapazität C gibt. Damit erhalten wir und mithin:

**Quellen:**

[1]: M. Dyer. *Approximate counting by dynamic programming.* in: Proc. 35th ACM Symp. on Theory of Computing (STOC), pp. 693-699, 2003

[2]: P. Gopalan, A. Klivans, R. Meka, D. Stefankovic, S. Vempala, E. Vigoda. *An FPTAS for #Knapsack and Related Counting Problems*. in: Proc. 52nd IEEE Symp. on Foundations of Computer Science (FOCS), pp. 817-826, 2011