11. Übung zur Vorlesung "Algorithmische Massenspektrometrie"

Wintersemester 2020/2021

Sebastian Böcker, Kai Dührkop

Ausgabe: 29. Januar 2021, Abgabe: 04. Februar 2021

1. **Dünnbesetzte Dynamische Programmierung** Der Speicherverbrauch für das Maximale Farbenfrohe Teilbaum Problem ist exponentiell in der Anzahl der Farben. Allerdings kann man mittels einer dünnbesetzten dynamischen Programmierung Speicher und Laufzeit sparen. Die Idee dahinter ist, nur jene Einträge in die DP Tabelle abzuspeichern, die Teil der optimalen Lösung sein könnten.

   Beobachtung: Wenn $D[u, S_1] \geq D[u, S_2]$ mit $S_1 \subset S_2$, dann gibt es immer eine exakte Lösung, die über $D[u, S_1]$ statt $D[u, S_2]$ geht. Wir müssen also nicht **alle** 2^k vielen Farbmengen in der DP pro Knoten abspeichern, sondern nur jene, deren Score echt größer ist als der ihrer Teilmengen.

   Für einen Knoten u müssen wir also nur jene Farbmengen abspeichern, die von u aus über einen Pfad erreichbar sind, und für die es keine andere Lösung gibt, die mit weniger Farben auskommt und dabei einen ebenso guten Score erzielt.

   Gegeben sei ein Fragmentierungsgraph mit einem Knoten u und dessen Kindknoten v, w, x, y. Für die Kindknoten wurden die optimalen Lösungen/DP-Tabellen bereits berechnet (siehe Figure 1). Es wurden nur die notwendigen Einträge abgespeichert. Die **F**arben **B**lau, **G**elb, **R**ot, **S**chwarz und **V**iolett sind in den Tabellen über ihren Anfangsbuchstaben abgekürzt.

   (a) Berechnen und schreiben sie für den Knoten u mit Farbe violett alle Einträge $D[u, S]$, die Teil einer optimalen Lösung sein könnten.

   (b) Gibt es eine Farbe die, innerhalb des von u induzierten Teilgraphen, niemals Teil der optimalen Lösung ist und damit *wegfällt*?

   (c) Angenommen u wäre die Wurzel des Fragmentierungsgraphen, was wäre dann der optimale Score des Maximalen Farbenfrohen Teilbaums? Wie viele Knoten und Kanten hätte dieser Baum?

   Tipp: es empfiehlt sich, zuerst den ersten Fall der Rekurrenz zu berechnen, also alle Einträge aus den Kindknoten „hochzuziehen" (das nennt sich auch „Bottom-Up" oder „Push"-DP). Danach kann man den zweiten Teil der Rekurrenz berechnen, also alle Paare von Mengen, die man in $D[u, \cdot]$ abgespeichert hat und deren Schnittmenge gleich $\{c(u)\}$ ist, zu rekombinieren.

   (8 Punkte)

2. **Integer Lineare Programmierung** Angenommen eine Fabrik produziert zwei Produkte: A und B. Es werden drei Maschinen (M_1, M_2 und M_3) benötigt, um diese Produkte herzustellen. Produkt A benötigt 6 Minuten Verarbeitungszeit in M_1, 4 Minuten in M_2 und 3 Minuten in M_3. Produkt B braucht 4 Minuten in M_1, 8 Minuten in M_2 und 3 Minuten in M_3. Das Produkt A kann zum Stückpreis von 9 €verkauft werden, Produkt B für 13 €. Wie würden Sie die Produktion von A und B takten (also wie viele Produkte von A und B sollten jeweils in einer Stunde hergestellt werden), so dass der Profit der Fabrik maximimiert wird?

(a) Stellen Sie ein ganzzahliges lineares Programm auf, das dieses Problem löst. Geben Sie die Zielfunktion und die Nebenbedingungen an.

(b) Was wäre die optimale Lösung des Problems?
Tipp: Auf der Webseite `https://online-optimizer.appspot.com` können Sie ein solches ILP interaktiv definieren und lösen. Denken Sie daran, an die Variablendefinition ein *integer* anzuhängen, da sonst nur ein lineares Problem gelöst wird.

(5 Punkte)

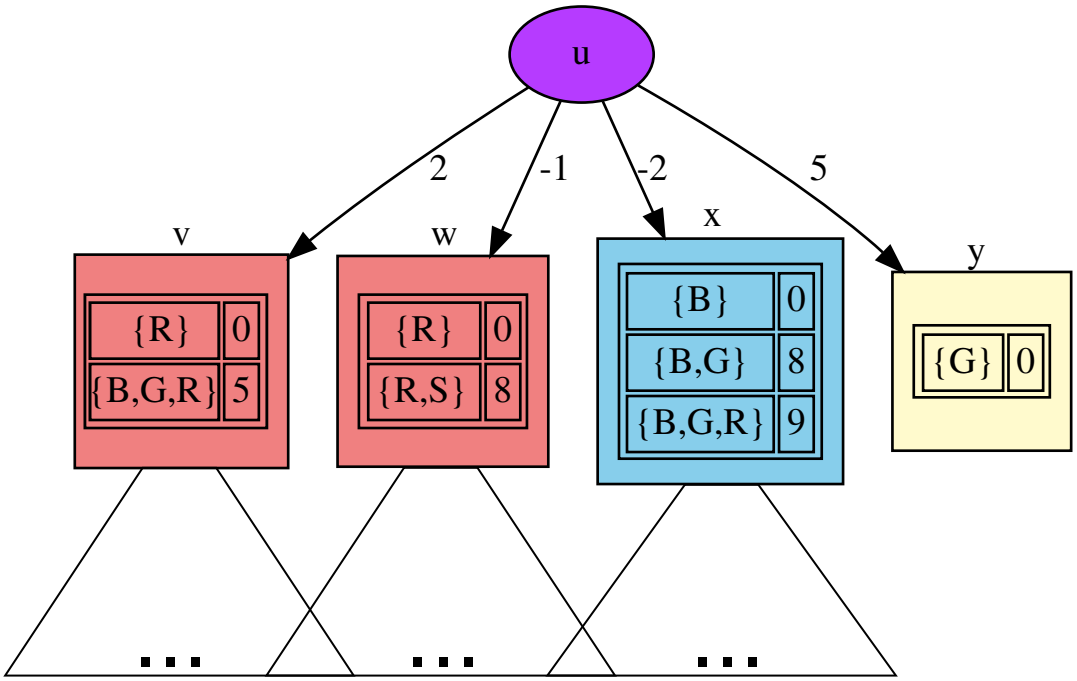

Abbildung 1: Ausschnitt eines Fragmentierungsgraphen. Die Knoten v, w, x, y sind Kindknoten von u. Diese können selbst wiederum andere Kindknoten haben. Mögliche fehlenden Kanten von u zu Kindknoten von v, w, x, y wurden in einer Vorverarbeitung entfernt, weil sie zu schlechte Scores hatten. Zu jedem Kindknoten ist eine Tabelle mit den Scores für die jeweilige Farbmenge angegeben.