Parametrisierte Algorithmen
Wintersemester 2022/2023
Thomas Bläsius & Marcus Wilhelm

Karlsruher Institut für Technologie

Lösung zum Übungsblatt 2

Aufgabe 1: 3-Hitting Set

3-Hitting Set lässt sich mithilfe eines Suchbaumes mit einem dominierenden Verzweigungsvektor von (1,2,2,2,2) lösen. Hierfür definieren wir folgende Reduktionsregeln und Verzweigungsregeln.

I. Reduktionsregel: falls eine einelementige Menge existiert, nehme ihr Element in die Lösung H auf.

II. Reduktionsregel: falls eine Menge existiert, die keine Elemente mit einer anderen Menge gemeinsam hat, nehme ein beliebiges Element dieser Menge in H auf.

III. Falls eine zweielementige Menge existiert, wähle eines der beiden Elemente. Dies resultiert in einem Verzweigungsvektor von (1,1).

IV. Falls zwei Mengen S_a, S_b mit $|S_a \cap S_b| = 2$ existieren, wähle entweder eines der beiden gemeinsamen Elemente oder die beiden verschiedenen Elemente. Dies resultiert in einem Verzweigungsvektor von (1,1,2).

V. Falls zwei Mengen S_a, S_b mit $|S_a \cap S_b| = 1$ existieren, wähle entweder das gemeinsame Element oder eine Kombination der jeweils zwei verschiedenen Elemente. Dies resultiert in einem Verzweigungsvektor von (1,2,2,2,2).

Nach Anwendung der ersten drei Regeln sind nur noch dreielementige Mengen übrig, die sich Elemente Teilen. Die letzten zwei Regeln behandeln dann die verbleibenden Fälle. Der Verzweigungsvektor (1,2,2,2,2) resultiert in einer Baumgröße von $2{,}562^k$. Also lässt sich 3-Hitting Set in $2{,}562^k \cdot n^{O(1)}$ entscheiden.

Aufgabe 2: d-Hitting Set

Wir definieren d-Hitting Set Compression und Disjoint d-Hitting Set analog wie in der Vorlesung gezeigt. Das heißt, bei Disjoint d-Hitting Set ist eine Familie an Mengen $\{S_1, \dots, S_n\}$ mit $|S_i| \leq d$ über einer Grundmenge U sowie ein Hitting Set H der Größe $k+1$ gegeben. Es soll entschieden werden, ob es ein Hitting Set X der Größe k gibt, sodass $X \cap H = \emptyset$. Ähnlich ist bei d-Hitting Set Compression eine d-Hitting Set Instanz mit einer Lösung H der Größe $k+1$ gegeben und es soll entschieden werden, ob es eine Lösung X der Größe k gibt.

Wir stellen fest, dass wir das Kompressionsproblem lösen können, indem wir das Disjoint Problem auf jeder möglichen Teilmenge der gegebenen Lösung H ausführen. Außerdem läst sich das Grundproblem lösen, indem die Instanz auf eine k-elementige Teilmenge des Universums eingeschränkt

wird. Die Instanz kann dann iteratv um ein Element erweitert werden, das einer Lösung der Größe k hinzugefügt wird woraufhin das Kompressionsproblem gelöst werden muss.

Wir zeigen nun per Induktion über d, dass d-Hitting Set in $O((d-0.658)^k \cdot n^{c+d})$ Zeit gelöst werden kann. Als Induktionsanfang betrachten wir $d = 2$. Hier besteht jede Menge aus zwei Elementen, also ist das Problem äquivalent zu Vertex Cover (Mengen sind Kanten, die Grundmenge ist die Knotenmenge). Mit dem Algorithmus aus der Vorlesung können wir 2-Hitting Set in $O(1.342^k \cdot n^c)$ lösen.

Wir zeigen nun, dass auch $(d+1)$-Hitting Set in $O((d+1-0{,}658)^k \cdot n^{c+d+1})$ Zeit lösbar ist, falls wir d-Hitting Set in $O((d-0.658)^k \cdot n^{c+d})$ Zeit lösen können. Wir betrachten hierfür das Disjoint Problem, d.h. wir nehmen an eine Lösung H der Größe $k+1$ zu erhalten und eine disjunkte Lösung X der Größe k zu suchen. H schneidet alle Mengen in mindestenz einem Element, welches nicht mehr gewählt werden darf. Das heißt, dass für jede Menge nur noch $d-1$ Elemente gewählt werden dürfen. Somit ist Disjoint $d+1$-Hitting Set äquivalent zu d-Hitting Set, welches nach Induktionsvoraussetzung in $O((d-0.658)^k \cdot n^{c+d})$ Zeit gelöst werden kann. Nach Vorlesung ergibt das eine Laufzeit von $O((k+1)(d+1-0{,}658)^k \cdot n^{c+d})$ für $(d+1)$-Hitting Set. Wenn wir $(k+1)$ als n abschätzen, dann ergibt sich daraus $O((d+1-0{,}658)^k \cdot n^{c+d+1})$. Da d als konstant angenommen wird, ist das die Laufzeit, die zu zeigen war.

Aufgabe 3: Minimum-Maximal Matching

Teilaufgabe (a) Pro Kante im Matching gibt es zwei Knoten, also hat $V(M)$ Größe $2k$. Angenommen es gäbe ein Kante $\{v,w\} \in E$, die von $V(M)$ nicht abgedeckt wird. Dann ist weder v noch w in $V(M)$, also hätte die Kante zum Matching M hinzugefügt werden können, was der inklusions-maximalität widerspricht. $V(M)$ ist also ein Vertex Cover der Größe $2k$.

Teilaufgabe (b) $M_1 \cup M_2$ ist inklusions-maximal, da M_2 größtmöglich in $G - V(M_1)$ ist und somit jede Kante inzident zu einem Knoten in $V(M_1 \cup M_2)$ ist.

Es verbleibt $|M_1 \cup M_2| \leq |M|$ zu zeigen. Seien hierfür M_X die Kanten von M bei denen beide Endpunkte in X liegen. Dann gilt

$$|M| = |M_X| + |M \setminus M_X|.$$

Da X ein Vertex Cover ist und somit alle Kanten abdeckt, haben Kanten in $M \setminus M_X$ genau einen Endpunkt in X. Es gilt also $|M \setminus M_X| = |X| - 2|M_X|$ und wir erhalten

$$|M| = |M_X| + (|X| - 2|M_X|) = |X| - |M_X|.$$

Da M_1 ein größtmögliches Matching von $G[X]$ ist gilt $|M_1| \geq |M_X|$. Außerdem gilt $|M_2| \leq |X| - 2|M_1|$, da Kanten in M_2 genau einen Endpunkt in X haben, von denen aber jede Kante in M_1 zwei benutzt. Wir erhalten also

$$|M_1 \cup M_2| \leq |X| - |M_1| \leq |X| - |M_X| = |M|.$$

Teilaufgabe (c) Die Idee ist, alle inklusions-minimalen Vertex Cover X der Größe höchstens $2k$ in G aufzuzählen und jeweils Teilaufgabe (b) anzuwenden. Falls G ein inklusions-maximales Matching M mit höchstens k Kanten enthält, dann können wir für ein inklusionsminimales Vertex Cover $X \subseteq M$ von G in Polynomialzeit ein größtmögliches Matching M_1 von $G[X]$ und ein größtmögliches Matching M_2 von $G - V(M_1)$ berechnen, sodass $M_1 \cup M_2$ ebenfalls höchstens k Kanten enthält. Aus Teilaufgabe (a) ist klar, dass $|X| \leq 2k$.

Wir können alle Vertex Cover der Größe $2k$ aufzählen, indem wir den einfachen Suchbaum für Vertex Cover, bei dem auf den Endpunkten einer Kante gebranched wird anpassen. Anstatt das erste gefundene Vertex Cover auszugeben, wird der gesamte Suchbaum bis zur Tiefe $2k$ traversiert und alle inklusionsminimalen Vertex Cover ausgegeben. Der Suchbaum hat dann $2^{2k} = 4^k$ Blätter. Da pro gefundenem inklusionsminimalen Vertex Cover nur polynomieller Aufwand nötig ist ergibt sich ein Algorithmus mit Laufzeit $4^k n^{O(1)}$.