Föreläsning 27–28 i ADK

NP-fullständighetsreduktioner

1. **Minimal dominerande mängd**
   Optimeringsproblemet Minimal dominerande mängd definieras på följande sätt:

   > Inmatning: En oriktad graf $G = \langle V, E \rangle$.
   > Lösning: En delmängd $S \subseteq V$ av hörnen som dominerar V, det vill säga för varje $v \in V$ är antingen $v \in S$ eller finns det ett $u \in S$ så att $(u, v) \in E$.
   > Målfunktion: Antal hörn i S.
   > Problem: Minimera målfunktionen.

   Formulera minimeringsproblemet som ett beslutsproblem och visa att det är NP-fullständigt.

   Lösning:
   Inför ett mål M i indata:

   > Inmatning: En oriktad graf $G = \langle V, E \rangle$, ett positivt heltal M.
   > Fråga: Finns det en delmängd $S \subseteq V$ med högst M hörn som dominerar V, det vill säga för varje $v \in V$ är antingen $v \in S$ eller finns det ett $u \in S$ så att $(u, v) \in E$?

   Om vi får en lösning (S) given så kan vi kontrollera att den är giltig genom att kontrollera att $|S| \leq M$ och att S verkligen dominerar hela V. Detta tar bara linjär tid i grafens storlek. Därmed ligger problemet i NP.

   För att vi ska kunna visa att problemet är NP-svårt reducerar vi det känt NP-fullständiga problemet *Hörntäckning* till vårt problem. I hörntäckningsproblemet är indata en oriktad graf $G' = \langle V', E' \rangle$ och ett positivt heltal M'. Frågan är ifall det finns M' hörn i G' som tillsammans täcker alla kanter i G', det vill säga så att varje kant har minst en av ändpunkterna bland dom M' hörnen.

   Givet grafen G' så konstruerar vi en ny graf G som består av hela G' (utom eventuella isolerade hörn) utvidgad med ett extra hörn och två extra kanter för varje kant i E'. För varje kant $(u, w) \in E'$ skapar vi nämligen ett nytt hörn v_{uw} som är förbundet till både u och w med två kanter. Vi låter $M = M'$. Reduktionen går att göra i linjär tid i antalet kanter.

   Låt oss först visa att om det finns en hörntäckning av storlek M' i G' så kan man skapa en dominerande mängd av storlek M' i G. För varje kant (u, w) i G' är antingen u eller w med i hörntäckningen, vilket betyder att hörntäckningen dominerar både u, w och v_{uw} i G. Hörnen i hörntäckningen bildar alltså en dominerande mängd av rätt storlek.

   Nu återstår bara att visa att om det finns en dominerande mängd av storlek $M = M'$ i G så finns det en hörntäckning av storlek M' i G'. Ta den dominerande mängden och modifiera den så att inga hörn av typen v_{uw} finns med i den dominerande mängden. Om ett hörn v_{uw} är med kan vi alltid byta ut det mot antingen u eller w, för det kan inte dominera något annat hörn än u och w, och både u och w dominerar både u och w. Nu är den modifierade hörnmängden en hörntäckning av storlek M'. Anledningen är att alla v_{uw}-hörn måste vara dominerade, vilket betyder att antingen u eller w är med, varför kanten (u, w) är täckt.

2. **Konstruktion av dominerande mängd**
   Turingreducera det konstruktiva optimeringsproblemet till optimeringsproblemet, det vill säga konstruera en minimal dominerande mängd genom att anropa en algoritm för optimeringsproblemet (högst ett polynomiskt antal gånger).

Lösning:
Anta att algoritmen $MDM(G)$ returnerar storleken på den minimala dominerande mängden i G. Idén är att man tar ett hörn v i taget i grafen och lägger till en kant mellan detta hörn och ett nyskapat hörn. Om storleken på den minimala dominerande mängden då inte ökar vet man att v kan vara med i den minimala dominerande mängden. Vi låter det nya hörnet sitta kvar så att vi är säkra på att v är med i den dominerande mängd som vi håller på att konstruera. Om storleken däremot ökar så vet vi att hörnet inte kan vara med i den minimala dominerande mängden.

Konstruktivdominerandemängd($G = \langle V, E \rangle$)=
    $m \leftarrow MDM(G)$
    $D \leftarrow \emptyset$
    for $v \in V$ **do**
        Låt G' vara G utvidgad med ett nytt hörn x_v och en kant (x_v, v).
        if $MDM(G') = m$ **then** $G \leftarrow G'$; $D \leftarrow D \cup \{v\}$
    return D

$|V| + 1$ anrop av MDM görs.

3. **Satisfierbarhet av formel med begränsat antal variabelförekomster**
Ett sätt att begränsa indata till problemet Cnf-Sat är att begränsa antalet literaler i varje klausul till exakt 3. Då får vi 3-Cnf-Sat. I denna uppgift ska vi studera ett annat sätt att begränsa indata, nämligen till formler där varje variabel förekommer högst 3 gånger. Låt oss kalla satisfierbarhetsproblemet för denna typ av formler för Cnf-Sat(3)

Ett exempel på indata till Cnf-Sat(3) är

$$(x_1 \vee x_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_5) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3 \vee \overline{x}_4 \vee \overline{x}_5) \wedge (\overline{x}_5 \vee \overline{x}_6).$$

Visa att Cnf-Sat(3) är NP-svårt!

Bevis:
Låt oss reducera Cnf-Sat till Cnf-Sat(3). Anta att variabeln x_i förekommer $m \geq 3$ gånger i CNF-formeln. Ersätt dessa m förekomster med nya variabler $y_{i,1}$, $y_{i,2}$, …, $y_{i,m}$. För att dessa variabler ska tvingas ha samma värde lägger vi till uttrycket $(y_{i,1} \Rightarrow y_{i,2}) \wedge (y_{i,2} \Rightarrow y_{i,3}) \wedge \cdots \wedge (y_{i,m-1} \Rightarrow y_{i,m}) \wedge (y_{i,m} \Rightarrow y_{i,1})$ som också kan skrivas på CNF som $(\overline{y}_{i,1} \vee y_{i,2}) \wedge (\overline{y}_{i,2} \vee y_{i,3}) \wedge \cdots \wedge (\overline{y}_{i,m-1} \vee y_{i,m}) \wedge (\overline{y}_{i,m} \vee y_{i,1})$. Varje y-variabel förekommer nu exakt tre gånger. Byt på samma sätt ut alla x-variabler i formeln. Detta tar polynomisk tid.

4. **Konsultoptimering**
Du har en programutvecklingsfirma men inga anställda så du är tvungen att anlita konsulter. Du har n stycken små programmeringsjobb på gång, och dom tar i tur och ordning $a_1, a_2, \ldots, a_n$ timmar att utföra. Du har möjlighet att hyra in konsulter från konsultfirman Individor. Individor erbjuder k konsulter, och dom kräver $s_1, s_2, \ldots, s_k$ kronor i lön per dag (respektive). Varje arbetsdag är 8 timmar lång, och varje programmeringsjobb måste utföras av en och samma konsult under en och samma dag. Två konsulter kan alltså inte dela på ett jobb, och en konsult kan inte dela upp ett jobb på flera dagar. Din totala budget för konsultarvoden är B kronor. Alla tal i indata är ickenegativa rationella tal.

Problemet är att avgöra vilka konsulter som ska anlitas vilka dagar och vilken konsult som ska göra vilka programmeringsjobb under vilken dag. Målet är att alla programmeringsjobben ska klaras av på så få dagar som möjligt. Du måste förstås hålla löneutgifterna inom budgeten B.

Formulera detta optimeringsproblem matematiskt som ett beslutsproblem. Visa sedan att problemet är NP-fullständigt. Du får reducera vilket problem som helst som visats NP-fullständigt i kursen.

Lösning:
Ett beslutsproblem får vi om vi inför ett mål D för antalet dagar. Matematisk problemformulering:

> INMATNING: Ickenegativa rationella tal $a_1, a_2, \ldots, a_n$, $s_1, s_2, \ldots, s_k$. Ickenegativa heltal B och D.
> FRÅGA: Finns det någon $k \times D$-matris M som består av delmängder av talen $[1..n]$ så att nedanstående fyra villkor är uppfyllda. (M anger vilka jobb som ska utföras under vilken dag och av vilken konsult.)
>
> - Varje tal i intervallet $[1..n]$ finns med i någon delmängd $M_{i,j}$. (Varje jobb blir gjort.)
> - För alla i, j, x, y gäller att $M_{i,j}$ och $M_{x,y}$ är disjunkta. (Inget jobb görs av flera konsulter eller under flera dagar.)
> - För alla i, j gäller att $\sum_{x \in M_{i,j}} a_x \leq D$. (Ingen arbetsdag är längre än 8 timmar.)
> - $\sum_i s_i c_i \leq B$, där c_i anger hur många element på rad i i matrisen M som inte är tomma mängden. (Konsulternas sammanlagda löner överskrider inte budgeten.)

Konsultproblemet ligger i NP eftersom vi kan gissa en lösningsmatris M och sedan kolla dom fyra villkoren ovan i polynomisk tid.

Vi visar att konsultproblemet är NP-svårt genom att reducera lådpackningsproblemet. I lådpackningsproblemet är frågan ifall n ickenegativa och rationella vikter $w_1, w_2, \ldots, w_n$ kan packas i K stycken lådor som var och en högst kan innehålla vikten 1. Vi anlitar en enda konsult som jobbar gratis och med budgeten 0. Varje låda låter vi motsvara en dag och varje vikt låter vi motsvara ett jobb. Eftersom en låda rymmer 1 kilo och en dag 8 timmar så måste varje vikt multipliceras med 8.

Lådpackning($w_1, w_2, \ldots, w_n, K$)=
  for $i \leftarrow 1$ **to** n **do**
    $a_i \leftarrow 8w_i$
  $s_1 \leftarrow 0$
  $B \leftarrow 0$
  $D \leftarrow K$
  return Konsult($a_1, a_2, \ldots, a_n$,s_1, B, D)

Detta är en riktig reduktion för det finns en packning av $w_1, w_2, \ldots, w_n$ i K stycken 1-kiloslådor om och endast om det finns en schemaläggning av jobben $a_1, a_2, \ldots, a_n$ på D dagar med en enda gratis konsult som jobbar högst 8 timmar per dag. Kopplingarna mellan lådor och dagar samt mellan vikter och jobb gör detta uppenbart i båda riktningarna. Reduktionen tar polynomisk tid, så konsultproblemet är NP-svårt.

Eftersom konsultproblemet ligger i NP och är NP-svårt så är det NP-fullständigt.

5. **MAX 2-CNF-SAT**
Satisfierbarhetsproblemet för booleska CNF-formler där varje klausul består av högst två literaler kallas 2-CNF-SAT. Till skillnad från 3-CNF-SAT som är NP-fullständigt så är faktiskt 2-CNF-SAT lösbart i polynomisk tid. MAX 2-CNF-SAT är ett optimeringsproblem som generaliserar 2-CNF-SAT. Där kräver man inte att alla klausuler ska vara satisfierade utan vill istället veta hur många klausuler som som mest kan satisfieras samtidigt. Motsvarande beslutsproblem frågar, givet en 2-CNF-formel och ett mål K, om det finns någon variabeltilldelning som satisfierar minst K klausuler.

Betrakta följande tio klausuler:

$$(x),\ (y),\ (z),\ (w),$$
$$(\neg x \vee \neg y),\ (\neg y \vee \neg z),\ (\neg z \vee \neg x),$$
$$(x \vee \neg w),\ (y \vee \neg w),\ (z \vee \neg w).$$

Räkna ut hur många av dessa klausuler som kan satisfieras samtidigt och använd sedan detta för att visa att Max 2-Cnf-Sat är NP-fullständigt.

Lösning (Engelsk läsövning)
We notice that the clauses are symmetric with respect to x, y and z (but not w). So, assume that all three of x, y and z are true. Then the second row is lost, and we can get all the rest by setting w to true. If just two of x, y and z are true, then we lose a clause from the first row, and one clause from the second row. Then we have a choice: If we set w to true, we get one extra clause from the first row; if we set w to false, we get one from the third. So, we can again satisfy seven clauses, and no more. If only one of x, y and z is true, then we have one clause from the first row and the whole second row. For the third row, we can satisfy all three clauses by setting w to false, but then we lose (w). The maximum is again seven. However, suppose that all three are false. Then it is easy to see that we can satisfy at most six clauses: The second and third row.

Thus we have found that any truth assignment that satisfies $(x \vee y \vee z)$ can be extended to satisfy seven of the clauses and no more, while the remaining truth assignment can be extended to satisfy only six of them. Now we can construct a reduction of 3-Cnf-Sat to Max 2-Cnf-Sat. For every clause $(\alpha \vee \beta \vee \gamma)$ we add the ten clauses above, with α, β and γ replacing x, y and z; w is replaced by a new variable w, particular to the clause. We call these ten clauses a *group*. If the 3-Cnf-Sat problem instance has m clauses, then the constructed Max 2-Cnf-Sat instance has m groups, consisting of $10m$ clauses. The goal is chosen as $K = 7m$.

A satisfying assignment for the 3-Cnf-Sat problem instance gives us an assignment of the Max 2-Cnf-Sat instance in which exactly seven clauses in each group are satisfied (by above we can always obtain this by choosing the value of the w variables in the right way). Conversely, suppose that $7m$ clauses can be satisfied. Since we know that in each group we can satisfy at most seven clauses, and there are m groups, seven clauses must be satisfied in each group. However, such an assignment would satisfy all clauses in the 3-Cnf-Sat problem instance.

Finally, it is easy to check that Max 2-Cnf-Sat is in NP, and that the reduction can be carried out in polynomial time.