NP-fullständighetsreduktioner- Rollbesättning

Laboration 4 - DD2350 – 2019

Gustav Röhss – Jonas Nylund

För att bevisa Rollbesättningsproblemet NP-fullständigt använder vi tillvägagångssättet som beskrivs i *ADK Övning 9; NP-fullständighetsbevis* av *Lisa Li* [1]. Vidare kommer ”Rollbesättningsproblemet” refereras till som *”RB”*, och dessa kan anses synonyma.

Till att börja med ska problemet visas tillhöra NP. Detta gjordes som teoriuppgift informellt inför Övning 9, men görs här formellt för att göra detta bevis komplett.

En lösning på RB är en *rolltilldelning*, i vilken varje roll har tilldelats en skådespelare att spela denna. För att formalisera detta hjälper det att formalisera indata till RB. Detta kommer även att hjälpa oss senare. Vi kommer här att avvika från originalnotationen så som den introduceras i problembeskrivningen för *Labb 4: NP-fullständighetsreduktioner - Rollbesättning* [2].

En instans av RB beskrivs av:

- **P**: En mängd skådespelare.
    - Specifikt kommer *divorna* som introduceras som p_1 och p_2 i denna text kallas p_a respektive p_b.
- **R**: En mängd roller. Varje roll kan i sin tur tolkas som en *delmängd* av **P**; detta är de skådespelare som kan spela en viss roll.
- **S**: En mängd scener. Varje scen kan i sin tur tolkas som en *delmängd* av **R**; detta är de roller som deltar i en viss scen.

Tillbaka till lösningen. En rolltilldelning kan beskrivas med en funktion.

$$f : \boldsymbol{R} \rightarrow \boldsymbol{P}$$

Funktionen tar alltså ett argument från *rollerna* och resultatet är en *skådespelare*, som alltså har blivit tilldelad rollen. Vi kan nu formalisera vad som krävs av en *ja-instans* av RB. Kraven beskrivs informellt senare.

1. $\exists f : \boldsymbol{R} \rightarrow \boldsymbol{P}$
2. $\forall s\ (s \in \boldsymbol{S} \rightarrow |s| > 1)$
3. $\exists x(f(x) = p_a)$
4. $\exists x(f(x) = p_b)$
5. $\forall s \forall \{r_a, r_b\}\ (\{r_a, r_b\} \subseteq s \wedge s \in \boldsymbol{S} \rightarrow \{f(r_a), f(r_b)\} \neq \{p_a, p_b\})$
6. $\forall s \forall \{r_a, r_b\}\ (\{r_a, r_b\} \subseteq s \wedge s \in \boldsymbol{S} \rightarrow f(r_a) \neq f(r_b))$
7. $f(x) = y \rightarrow y \in x$

Det första kravet är ett det existerar en rolltilldelning alls. Detta garanterar att alla roller har blivit tilldelade. Krav två garanterar att inga monologer förekommer. Krav tre och fyra garanterar att båda divorna innehar en roll. Krav fem och sex innebär att; för varje scen är

godtyckliga två deltagande roller i denna scen inte spelade av samma skådespelare, samt inte en vardera av divorna. Det sjunde kravet garanterar att varje roll spelas av en skådespelare som kan spela rollen.

Om det finns en verifikator som för godtycklig probleminstans (**P**, **R**, **S**) samt lösningsförslag *f* kan verifiera huruvida *f* är en korrekt lösning av probleminstansen i polynomisk tid relativt indatas storlek så gäller RB ∈ NP. Vi väljer för att demonstrera att detta är möjligt att designa en verifikator.

```
VerifyRB((P, R, S), f):
    for r in R:
        assert(f(r) in P)                                   [1]
    for s in S:
        assert(size(s) > 1)                                 [2]
    assert(p_a in map(R, f) && p_b in map(R, f))            [3, 4]
    for s in S:
        assert(size(s) = size(map(s, f)))                   [6]
        assert(not(p_a in map(s, f) and p_b in map(s, f)))  [5]
    for r in R:
        assert(f(r) in r)                                   [7]
    return true
```

Två saker bör tydliggöras här. För det första ses *assert* som att ta ett booleskt argument. Ifall argumentet är *sant*, sker ingenting. Om det är *falskt*, tvingas *VerifyRB* returnera falskt. Vidare ses funktionen *map* som den ”funktionella” map. Alltså;

$$map(\{a, b, c \dots x\}, f) = \{f(a), f(b), f(c) \dots f(x)\}$$

Då har vi alltså en verifikator. Att denna körs i polynomisk tid är enkelt att verifiera och lämnas som uppgift till läsaren. Vi ska istället bevisa dennas korrekthet. Markeringarna i koden hjälper med detta (inte att förvirras med källhänvisningar).

- [1] Garanterar att varje roll har en tilldelning.
- [2] Garanterar att ingen monolog förekommer.
- [3, 4] Garanterar att divorna har fått roller.
- [6] Garanterar att varje scen har en skådespelare per roll.
- [5] Garanterar att inte båda divorna deltar i en scen.
- [7] Garanterar att varje roll spelas av en skådespelare som kan spela rollen.

Saken är klar! Vi kan konstatera att RB är ett NP-problem, och vårt första steg i processen att visa problemet NP-fullständigt är färdigt.

Vi ska nu visa att RB är NP-svårt. Om RB är både NP-svårt och tillhör NP (som visat) så är RB NP-fullständigt. För att visa att RB är NP-svårt utför vi en *Karp-reduktion*. Genom att reducera ett problem som sedan tidigare är känt att vara NP-fullständigt visar vi att RB är NP-svårt. Problemet vi kommer att reducera är *Graffärgning (GF).* Definitionen av detta problem återfinns i beskrivningen för Labb 4 [2]. Vi kommer sedan bevisa att reduktionen är *polynomisk* i tidskomplexitet relativt indata från en godtycklig instans av GF, samt att reduktionen är korrekt. Reduktionen är korrekt då varje ja-instans av GF reduceras till en ja-instans av RB, och varje nej-instans av GF reduceras till en nej-instans av RB.

För att få en intuition på problemet överväger vi de *minsta möjliga ja-instanserna* av båda problemen. För RB ser den ut som följande:

- **P** = $\{p_a, p_b, p_1\}$
- **R** = $\{r_a, r_b, r_1\} = \{\{p_a\}, \{p_b\}, \{p_1\}\}$
- **S** = $\{s_a, s_b\} = \{\{r_a, r_1\}, \{r_b, r_1\}\}$

Lösningen kontrolleras enkelt. Noteringen tydliggör hur tilldelning av skådespelare till roller, och deltagande roller i scener sker.

Motsvarande för GF är ännu enklare.

- **G** = {{1}, {}}
- **m** = 1

Alltså en graf med endast ett hörn (då detta är ett definierat krav på indata), inga kanter, och med en färg att tillgå. Lösningen är att färglägga alla (inga) kanter med färg ett.

Ledtråden här är att divorna, konstanta som de är, egentligen inte gör mycket. Vid första anblick kan de tyckas obekväma begränsningar, men sådant är inte fallet. Faktum är att de kommer visa sig vara användbara verktyg.

Vi inser att ”utökningar” av grafen (fler hörn, kanter och/eller färger) torde innebära någon motsvarande utökning av RB-instansen (fler scener, roller, och/eller skådespelare). Vi väljer att försöka visualisera *hörn* som *scener*. Vi tänker oss *kanter* som *roller*, där en kant mellan två hörn motsvarar en *roll som förekommer i båda scenerna*. En *färgning* av en *kant* är då intuitivt en tilldelning av en *skådespelare* till en *roll*.

Vi är nu redo att utföra vår reduktion. Informellt kommer det att gå till på följande sätt.

- Vi kommer att utgå från vår ”minsta” instans ovan.
- För varje hörn kommer vi att lägga till en ny scen. Ifall att hörnet har valens noll, kommer scenen innefatta rollerna r_a (r_b hade gått lika bra) samt r_1. Har hörnet valens 1, kommer scenen (till att börja med) innefatta r_a.

- För varje kant kommer vi att lägga till en ny roll. Rollen kan spelas av vem som helst, förutom divorna. Rollen ingår i båda scenerna som introducerades för att motsvara dess hörn.
- För varje tal $i \in ([2, \boldsymbol{m}] \cap \boldsymbol{N})$ (heltalen 2, 3 … m) kommer vi att lägga till en skådespelare p_i. Intervallet [2, 1] är nollmängden. Intervallet [2, 2] är {2}.

En uppmärksam läsare insatt i originaldefinitionen i Labb 4 förstår nu varför divornas notation har manipulerats, med motsvarande ändring för rollerna de ”till att börja med” spelar, och scenerna dessa deltar i.

Följande är en beskrivning av reduktionen i pseudokod. Konvention här är att “*:=*” motsvarar *assignment*, som brukar motsvaras av ” ←”; pilen kommer i pseudokoden att representera att ett element läggs till i en mängd.

```
ReduceToRB((V, E), m)
    P := {pa, pb, p1}
    R := {ra := {pa}, rb := {pb}, r1 := {p1}}
    S := {sa := {ra, r1}, sb := {rb, r1}}
    for v in V:                                          [a]
        sv := {}
        if degree(v) = 0:
            sv ← ra, r1
        else if degree(v) = 1:
            sv ← ra
        S ← sv
    for n in [2, m]:                                     [b]
        P ← pn
    i := 2                                               [c]
    for {u, v} in E:                                     [d]
        ri := P \ {pa, pb}
        su ← ri
        sv ← ri
        i := i + 1
    return RB(P, R, S)
```

Denna reduktion menar vi alltså visar att RB är NP-svårt. Detaljerna som är kvar är att bevisa reduktionens tidskomplexitet polynomisk relativt indatas storlek, samt dess korrekthet. Vi börjar med tidskomplexiteten.

En överskådlig titt räcker för den med grundläggande erfarenhet till för att verifiera att reduktionen är polynomisk. Inte desto mindre formaliserar vi det till en passande grad här. Markeringarna i koden hjälper oss.

- Allt innan [a] är instantiering som oproblematisk görs i konstant tid.
- Slingan mellan [a] och [b] körs lika många gånger som **V** har element. I slingan sker konstanttidsoperationen, samt en beräkning av valensen för varje hörn. Denna kan beräknas exempelvis genom att kontrollera i hur många kanter i **E** ett visst hörn förekommer. Slingan i sin helhet tar tid $O(|\boldsymbol{V}||\boldsymbol{E}|)$ – polynomisk.
- Slingan mellan [b] och [c] körs som mest $\boldsymbol{m} - 1$ gånger. I slingan sker en konstanttidsoperation. Slingan tar polynomisk tid.
- [c] är markerad för en kommentar. Användning av variabel *i* framgår, men är inte särskilt beskrivande till namnet. Den benämner/räknar antalet tillagda nya roller.
- Slingan från och med [d] utför konstanttidsoperationer, samt skapandet av en ny mängd. Detta kan göras genom att iterera över mängdens element och utesluta divorna. Slingan är polynomisk.
- Att returnera RB(**P**, **R**, **S**) tar potentiellt polynomisk tid om P = NP, men det ska vi verkligen inte ge oss in på, och är inte heller relevant för uppgiften.

Vi konstaterar att reduktionen är polynomisk.

En enda sak kvarstår för att konstatera RB NP-fullständig (så spännande)! Vi måste bevisa att reduktionen är korrekt.

Det första fallet är då vi utgår ifrån att indatan beskriver en ja-instans av GF. Vi måste övertyga oss om att reduktionen producerar en ja-instans av RB.

- Varje scen som introducerades i samband med ett noll-valent hörn spelas av p_a och p_1 utan konflikt, i rollerna r_a och r_1. Båda divorna har dessutom var sin roll i någon scen, och de spelar aldrig mot varandra.
- Varje scen som introducerades i samband med ett ett-valent hörn spelas av r_a (p_a) och någon ytterligare roll som introducerades till scenen i [d].
- Vi har designat vår reduktion sådan att vi har lika många icke-divor till hands att skådespela som vi hade för att färga grafens kanter. Vi utnyttjar den graffärgning som vi vet existerar som lösning till GF som rolltilldelning. För varje färg som används i graffärgningen, tilldela en av skådespelarna (förutom divorna) denna färg. Tilldela varje roll som sådant: om graffärgningen färgade kant *e* med färg *f*, ge rollen som skapades i samband med kant *e* i [d] till den skådespelare du associerat med denna färg.
- Då vi vet att ingen *färg* finns på *två olika kanter* mot ett *hörn*, vet vi att ingen *skådespelare* tilldelats *två olika roller* till en *scen*.

Vi konstaterar att varje ja-instans av GF reduceras till en ja-instans av RB.

Nu ska vi visa att en nej-instans av GF reduceras till en nej-instans av RB.

- Vi matchar igen varje skådespelare (förutom divorna) med en färg.
- För varje ytterligare *hörn* har vi en ytterligare *scen*, för varje ytterligare *kant* en ytterligare *roll*, och för varje ytterligare *färg* en ytterligare *skådespelare*.
- Då ingen graffärgning existerar, finns det i något *hörn* tvunget *flera kanter* som måste ges *samma färg*.
- Det följer att det i någon *scen* tvunget krävs tilldelning av *flera roller* till *samma skådespelare*. Detta är förbjudet.

Vi konstaterar att varje nej-instans av GF reduceras till en nej-instans av RB.

Beviset är klart. RB är NP-fullständigt.

[1] https://kth.instructure.com/courses/12404/pages/lisas-ovningsmaterial?module_item_id=157169
11:00 2019-11-08
[2] https://kth.instructure.com/courses/12404/assignments/95377?module_item_id=151179
11:00 2019-11-08