Graf planiranja

Seminar iz kolegija *Umjetna inteligencija*

Goran Flegar

Prirodoslovno-matematički fakultet
Sveučilište u Zagrebu

19. siječnja 2015.

Sadržaj

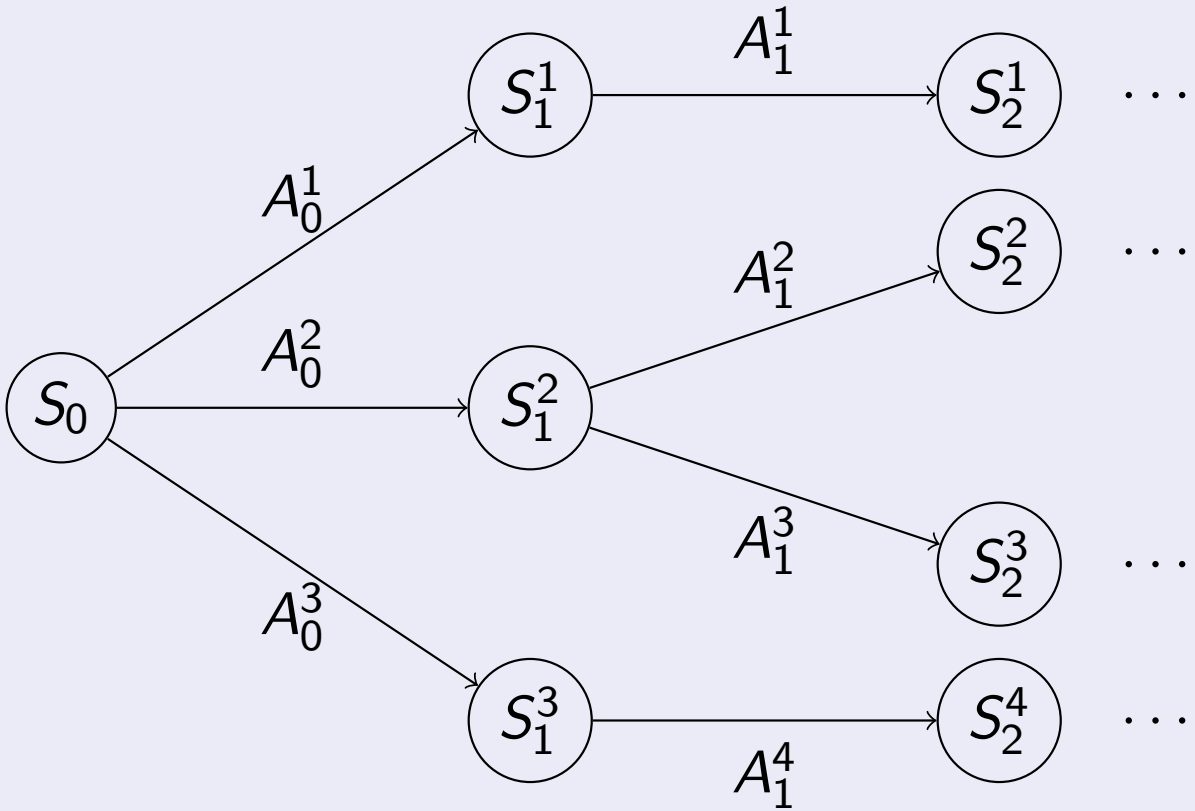
Sadržaj

Ideja grafa planiranja

- Cilj problema planiranja je pronaći plan akcija koji nas dovodi iz početnog u ciljno stanje
- Pretpostavimo da smo konstruirali stablo sa sljedećim svojstvima:
  - Čvorovi grafa su stanja
  - U korijenu se nalazi početno stanje
  - Djeca svakog čvora S su stanja u koja se dolazi nekom dopustivom akcijom iz S
- Nadalje, pretpostavimo da smo to stablo indeksirali tako da možemo u razumnom vremenu pronaći ciljno stanje
- Tada možemo pronaći plan do ciljnog stanja prateći bridove od čvora do roditelja, počevši od ciljnog stanja dok ne dođemo do korijena
- Problem
  - Broj čvorova svake razine stabla raste eksponencijalno

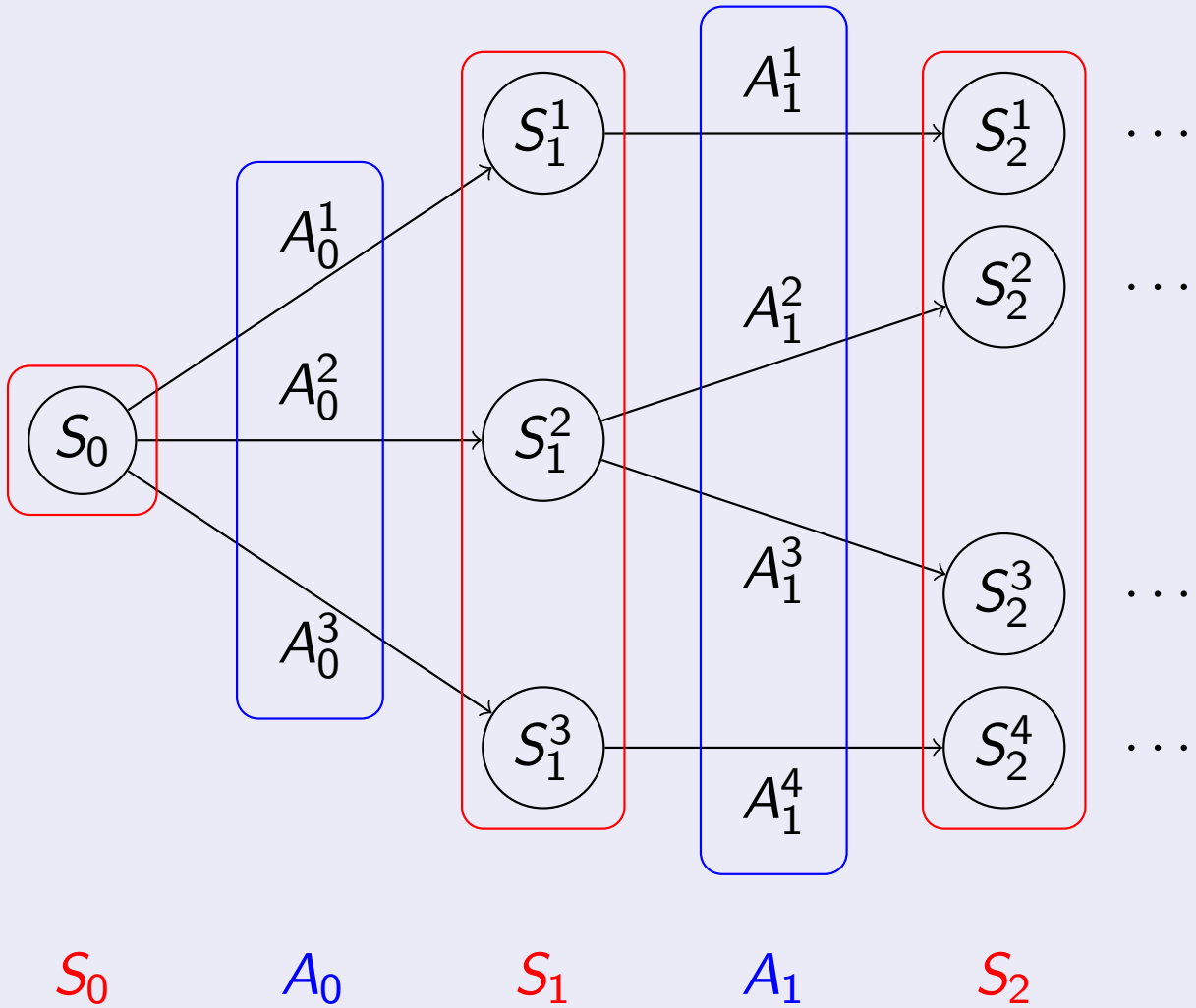
Primjer stabla

Stablo planiranja



Primjer stabla

Stablo planiranja



Ideja grafa planiranja (2)

- Graf planiranja je polinomijalna aproksimacija tog stabla
  - Okvirno, ne biramo jednu akciju koju ćemo napraviti na nekoj razini, nego napravimo sve moguće akcije
- Sastoji se od alternirajućih razina:

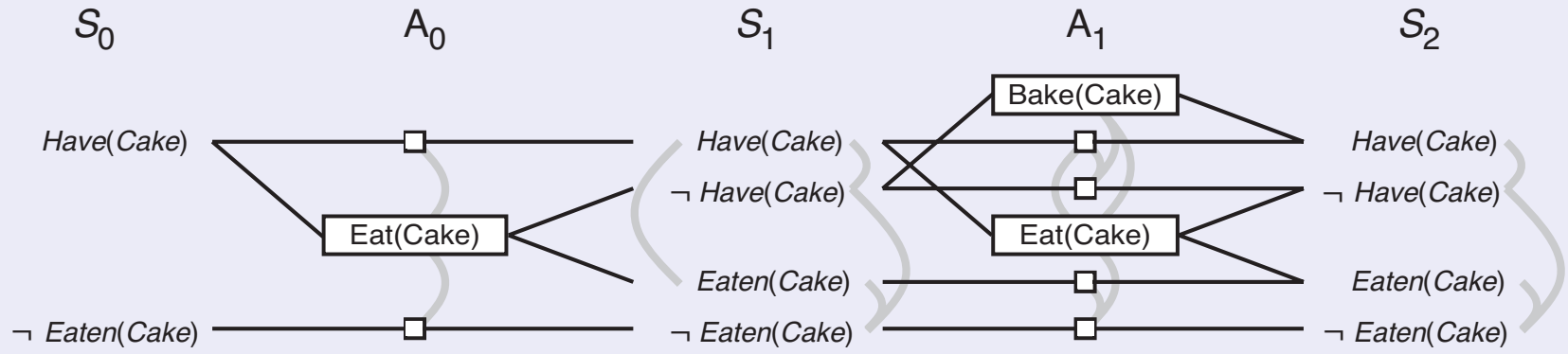
  S_i – čvorovi su svi literali koji bi mogli vrijediti nakon i akcija

  A_i – čvorovi su sve akcije koje bi mogle biti dopustive nakon i akcija

  - Bridovi povezuju akcije iz razine A_i s njihovim preduvjetima na razini S_i i njihovim posljedicama na razini S_{i+1}
  - tzv. *mutex* (*mutualy exclusive*) bridovi povezuju dva literala ili akcije na istoj razini ako su oni na neki način kontradiktorni

Primjer grafa

Graf planiranja



Problem "*Eat cake and have cake*"

$Init(Have(Cake))$

$Goal(Have(Cake) \wedge Eaten(Cake))$

$Action(Eat(Cake),$

$\quad \textsc{Precond} : Have(Cake)$

$\quad \textsc{Effect} : \neg Have(Cake) \wedge Eaten(Cake))$

$Action(Bake(Cake),$

$\quad \textsc{Precond} : \neg Have(Cake)$

$\quad \textsc{Effect} : Have(Cake))$

Konstrukcija grafa planiranja

- Razina S_0 se sastoji od svih literala početnog stanja (nema *mutex* bridova između literala)
- Razina A_i se sastoji od svih akcija koje kao preduvjete imaju samo literale koji se nalaze u S_i, te niti jedan par preduvjeta nije kontradiktoran (nema *mutex* brid)
  - Za svaki literal l dodajemo i *praznu* operaciju koja kao preduvjet i kao rezultat ima l
  - Između akcija u A_i dodajemo *mutex* bridove po pravilima opisanim na sljedećem slajdu
- Razina S_i se sastoji od svih literala koji su rezultati neke od akcija u A_{i-1}
  - Između literala u S_i dodajemo *mutex* bridove po pravilima na sljedećem slajdu
- Stajemo s konstrukcijom grafa kada su dvije uzastopne S razine jednake
  - Tada će i sljedeće S i A razine biti jednake prethodnim

Konstrukcija *mutex* bridova

- Dvije akcije u A_i povezuje *mutex* brid ako vrijedi jedno od pravila:

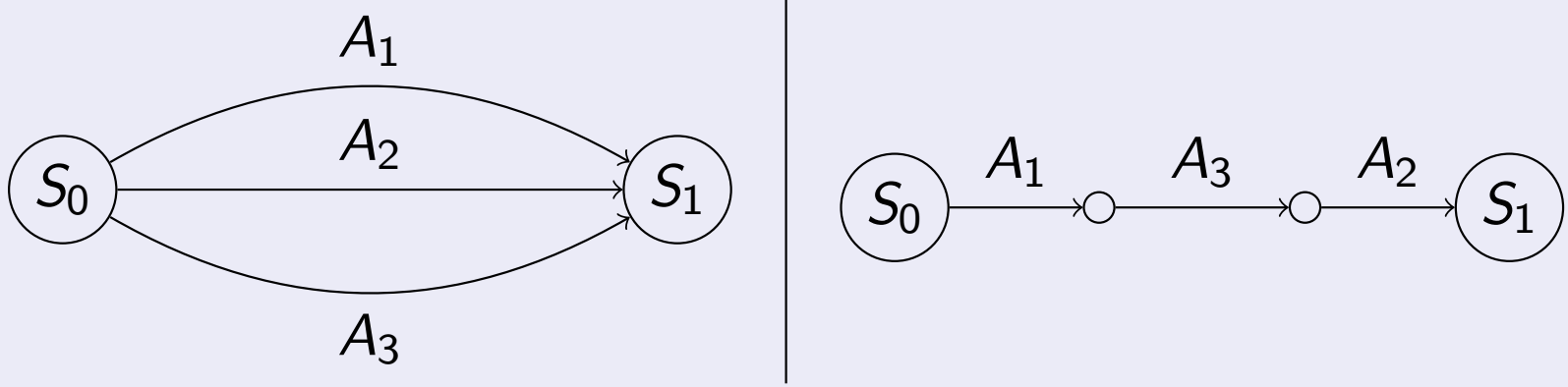
Nekonzistentni efekti	rezultat jedne akcije je negacija rezultata druge
Interferencija	rezultata jedne akcije je negacija preduvjeta druge
Kontradiktorne potrebe	preduvjet jedne akcije je kontradiktoran s preduvjetom druge

  (literale povezuje *mutex* brid)
- Dva literala u S_i povezuje *mutex* brid ako su svake dvije operacije iz A_{i-1} čiji su oni rezultati kontradiktorne (*nekonzistentna potpora*)

Ideja ovakve definicije *mutex* bridova

- Nekoliko istovremenih nekontradiktornih akcija daje isti rezultat kao da smo te operacije obavili jednu iza druge, u proizvoljnom poretku

Efekti nekontradiktornih akcija



Složenost grafa planiranja

- Neka je l broj literala i a broj akcija problema planiranja
  - Svaka S razina u grafu planiranja ima najviše l čvorova i $\frac{l(l-1)}{2}$ *mutex* bridova
  - Svaka A razina ima najviše $a + l$ akcija, $\frac{(a+l)(a+l-1)}{2}$ mutex bridova i $2(al + l)$ bridova između akcija i literala na susjednim S razinama
- Dakle, svaki par razina S i A je reda veličine $O((a + l)^2)$

Sadržaj

Heuristike dobivene iz grafa planiranja

Definicija

Neka su zadani početno stanje p i cilj $g = g_1 \wedge \ldots \wedge g_l$, gdje su g_i literali. Pretpostavimo da smo konstruirali graf planiranja $G(p)$ iz početnog stanja s. Definiramo ***cijenu razine*** *literala g_i, $l(g_i)$ kao indeks prve S razine u G u kojoj se pojavljuje literal g_i, tj. $l_{G(p)}(g_i) = \min\{k | g_i \in S_k, S_k \in S(G(p))\}$, gdje $S(G(p))$ označava skup svih S razina u G.*

Nadalje, definiramo

max-razinu *cilja g kao $m_{G(p)}(g) = \max_i l_{G(p)}(g_i)$,*

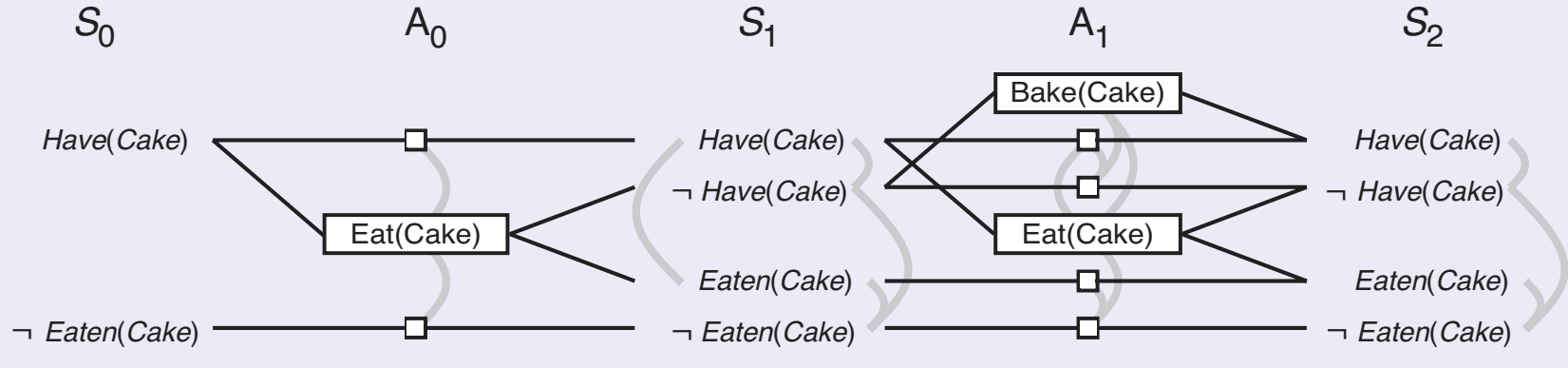
sumu razina *cilja g kao $\sigma_{G(p)}(g) = \sum_{i=1}^{l} l_{G(p)}(g_i)$,*

razinu skupa *cilja g, $s_{G(p)}(g)$ kao indeks prve S razine u kojoj se pojavljuju svi g_i i među njima nema mutex bridova.*

- Možemo konstruirati heuristiku h_x za problem planiranja s ciljem g, gdje je x jedna od funkcija m, σ i s, kao $h_x(p) = x_{G(p)}(g)$

Nedopustivost h_σ

Graf planiranja za "*Eat cake and have cake*"



- Cilj $g = \neg Have(Cake) \wedge Eaten(Cake)$ ima sumu razina 2, ali optimalni plan se sastoji samo od akcije $Eat(Cake)$

Dopustivost h_s

Teorem

h_s je dopustiva heuristika.

Ideja dokaza.

Indukcijom po duljini najkraćeg plana $a_1, \ldots a_n$ od s do g se pokazuje da se na razini S_n nalaze svi literali iz g te da među njima nema *mutex*-a. Za $n = 0$ tvrdnja očito vrijedi.
Pretpostavimo da tvrdnja vrijedi za n.
Neka je $a_1, \ldots, a_n, a_{n+1}$ najkraći plan od s do g. Označimo s $r_a(s)$ stanje koje je rezultat izvršavanja akcije a u stanju s. Znamo da je $a_1, \ldots, a_n$ najkraći plan od s do $g' = r_{a_n}(r_{a_{n-1}}(\ldots r_{a_1}(s)\ldots))$, pa se po pretpostavci indukcije na razini S_n grafa nalaze svi literali koji se nalaze u g' i među njima nema *mutex* bridova. Iz konstrukcije grafa planiranja znamo da se akcija a_{n+1} nalazi u A_n, a tada se (zbog praznih akcija) literali stanja $r_{a_{n+1}}(g') \supseteq g$ nalaze u S_{n+1} i među njima nema mutex bridova. □

Sadržaj

Ideja GRAPHPLAN algoritma

- Cilj nam je iskoristiti graf planiranja za rješavanje problema planiranja, a ne samo kao heuristiku
  - Konstruiramo graf razinu po razinu
  - Kada vidimo da bi na trenutnoj razini moglo biti rješenje (svi potrebni literali su prisutni i među njima nema *mutex* bridova) pokušamo rekonstruirati plan traženjem unatrag od ciljnog do početnog stanja
  - Ako ne možemo rekonstruirati plan, konstruiramo još jednu razinu grafa i pokušamo ponovno

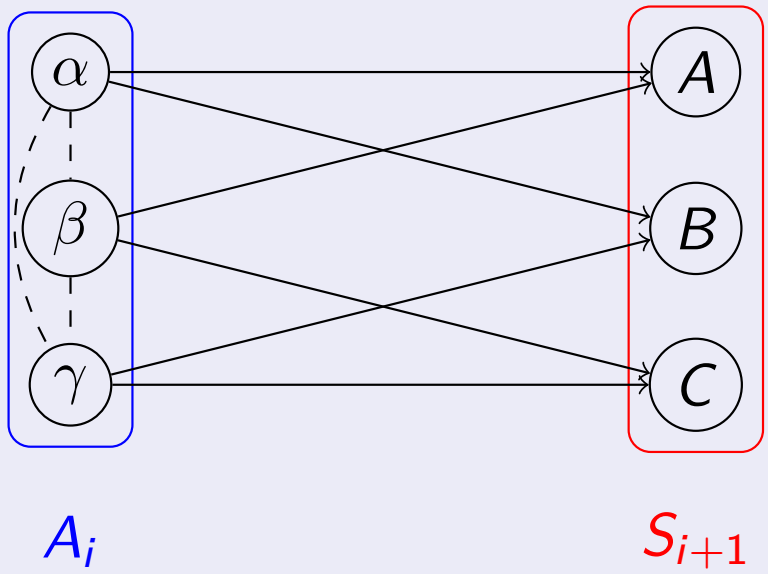
Rekonstrukcija plana

- Kako rekonstruirati plan?
  - Na posljednjoj A razini izaberemo skup akcija kojima se postiže ciljno stanje i koje nisu međusobno isključujuće (takvih skupova može biti više)
  - Ako takvog skupa akcija nema, zaključujemo da nije moguće postići ciljno stanje (barem za potrebe algoritma)
  - Ako smo pronašli takav skup, kao novo ciljno stanje gledamo skup svih preduvjeta akcija u odabranom skupu na prijašnjoj S razini
  - Iz novog u polazno stanje možemo doći tako da iz novog stanja primijenimo izabrane akcije u proizvoljnom redoslijedu
    - To možemo jer među akcijama nema *mutexa*, pa *mutex* pravila osiguravaju da akcije izvršene u bilo kojem redoslijedu imaju isti rezultat
    - Cijena prijelaza je broj nepraznih akcija u skupu
  - Postupak ponavljamo dok ne dođemo do razine S_0 (tj. početnog stanja)

Postoji li uvijek put do prethodne razine

- Ako u nekoj S razini između dva literala ne postoji *mutex*, iz definicije *mutex* bridova znamo da postoje nekontradiktorne akcije koje ostvaruju ta dva literala i između njihovih preduvjeta nema *mutex*-a
- Znači li to da uvijek postoji put od stanja na razini S do početnog stanja?
  - Ne, problem je kada trebamo ispuniti više od 2 literala

Stanje koje ne ispunjava niti jedan skup akcija



Rekonstrukcija plana (2)

- Ako iz nekog vrha (skupa literala na nekoj razini) nema puta do S_0, tog puta neće biti niti kada dodamo još jednu razinu, pa taj vrh dodajemo u skup *neostvarivih* ciljeva i ako opet u nekom trenutku dođemo do njega, odmah stajemo
- Broj različitih podskupova akcija kojima se postiže traženo stanje može biti velik, pa nam treba neka heuristika koja će nam pomoći u odabiru vrhova koje ćemo prve proširiti:
  - prvo odaberi literal s najvišom cijenom razine
  - prvo odaberi akciju koja postiže taj literal, a njezini preduvjeti imaju najmanju sumu razina

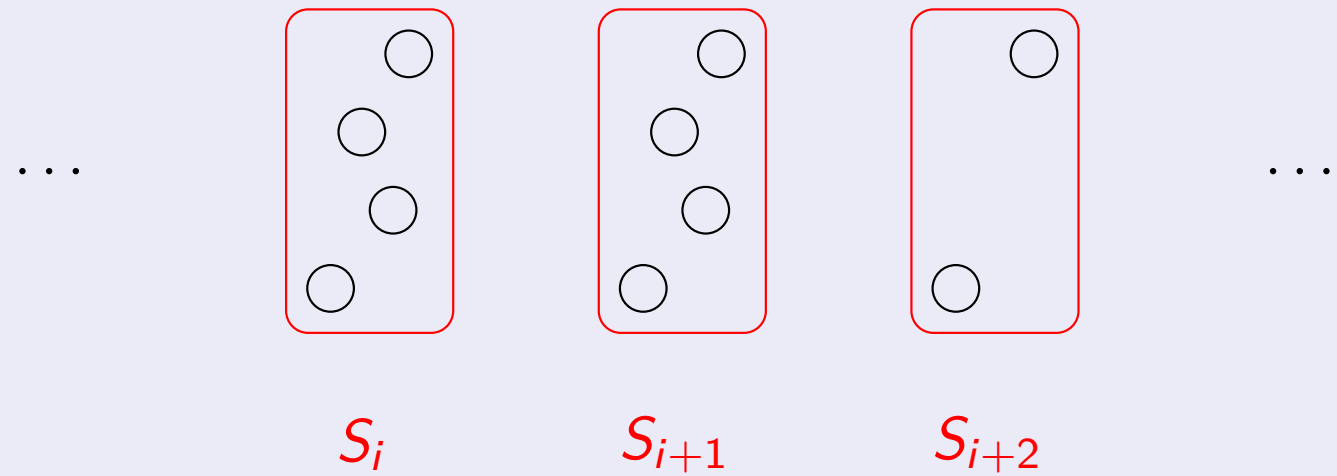
Kada možemo stati s generiranjem novih razina grafa?

- Nije dovoljno samo da su dvije S razine jednake
  - Ima problema u kojima se može brzo postići svaki par ciljeva, ali treba puno vremena za postići sve ciljeve
  - U takvom problemu će se dvije S razine brzo ponoviti, ali još neće biti moguće pronaći rješenje
- Potrebno je gledati i neostvarive ciljeve
  - Možemo stati čim su u nekom trenutku dvije razine jednake i skup neostvarivih ciljeva na njima jednak

Zašto je uvjet zaustavljanja dovoljan?

- Pretpostavimo da smo već postigli prvi uvjet, tj. da su počevši od neke S razine sve slijedeće S razine jednake

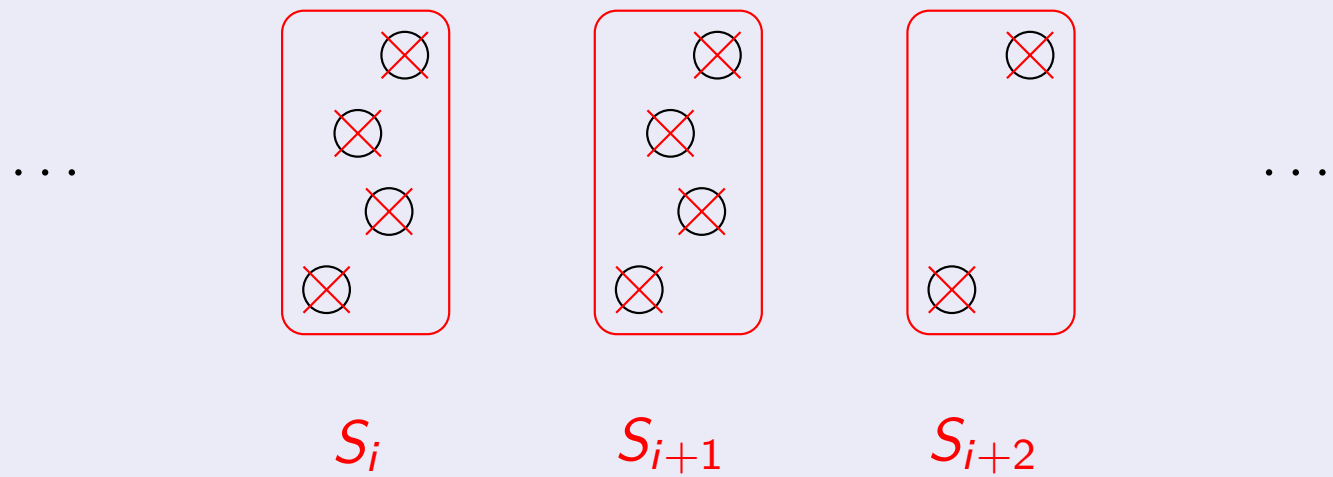
Neispunjivi vrhovi po razinama



Zašto je uvjet zaustavljanja dovoljan?

- Neka je postignut drugi uvjet na razinama S_i i S_{i+1}

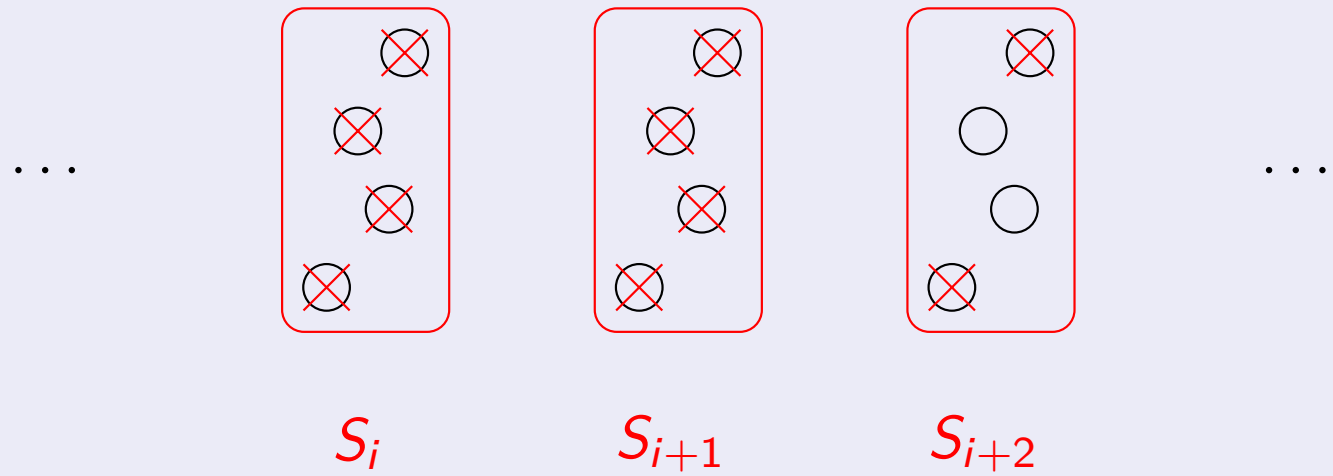
Neispunjivi vrhovi po razinama



Zašto je uvjet zaustavljanja dovoljan?

- U slijedećoj iteraciji algoritma će skup svih posječenih vrhova na razini S_{i+2} biti jednak skupu svih posjćenih vrhova na razini S_{i+1} u prethodnoj iteraciji

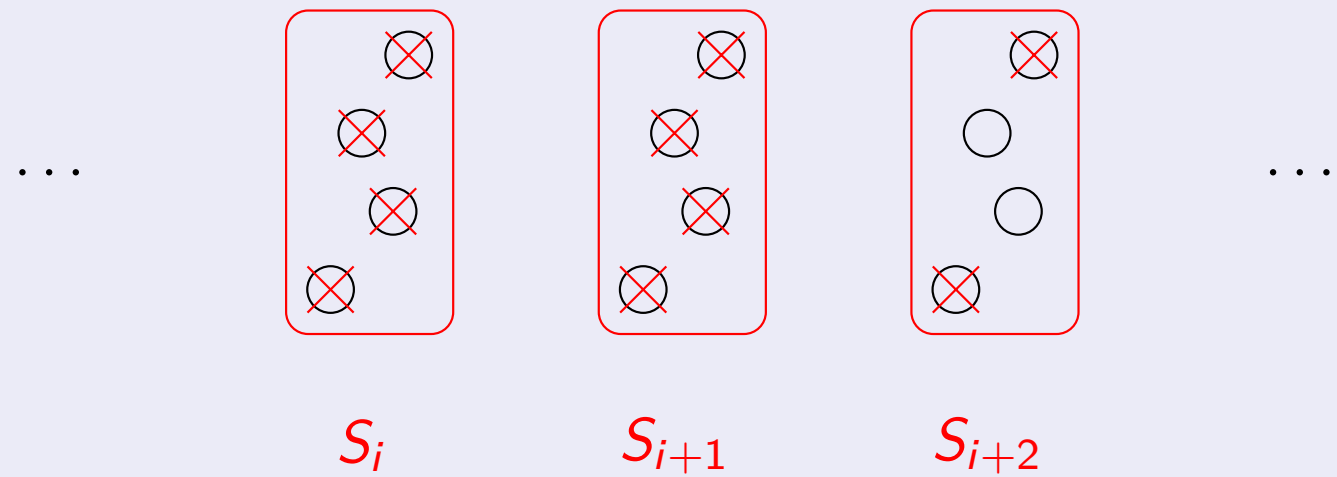
Neispunjivi vrhovi po razinama



Zašto je uvjet zaustavljanja dovoljan?

- Svaki novoposjećeni vrh na razini S_{i+2} je neostvarivi vrh na razini S_{i+1}

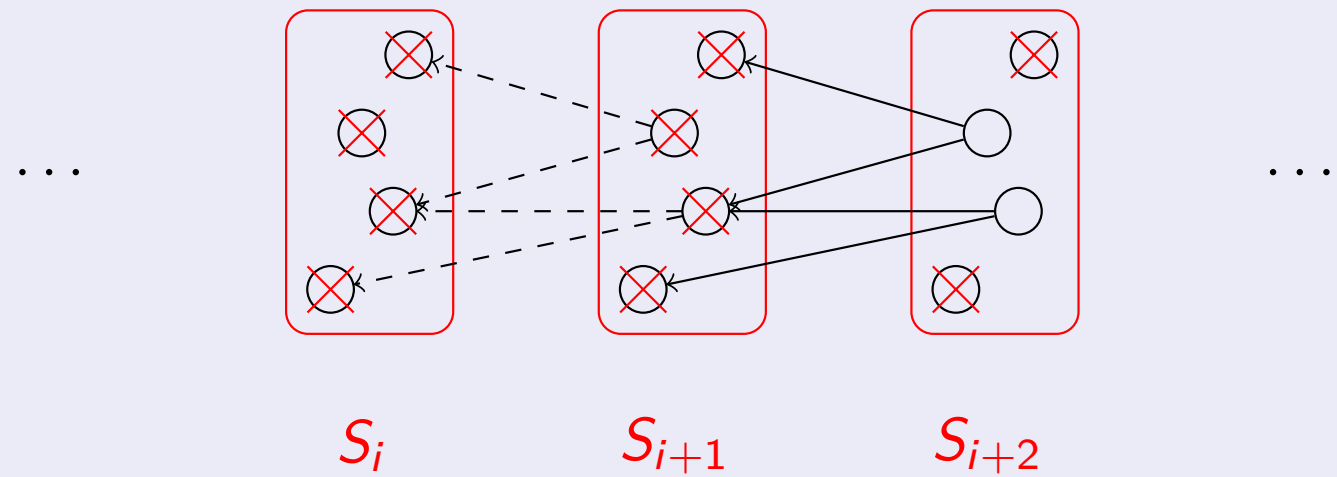
Neispunjivi vrhovi po razinama



Zašto je uvjet zaustavljanja dovoljan?

- Od svakog će se novoposjećenog vrha na razini S_{i+2} algoritam granati na iste vrhove na razini S_{i+1} na koje se granao s tog vrha na razini S_{i+1}

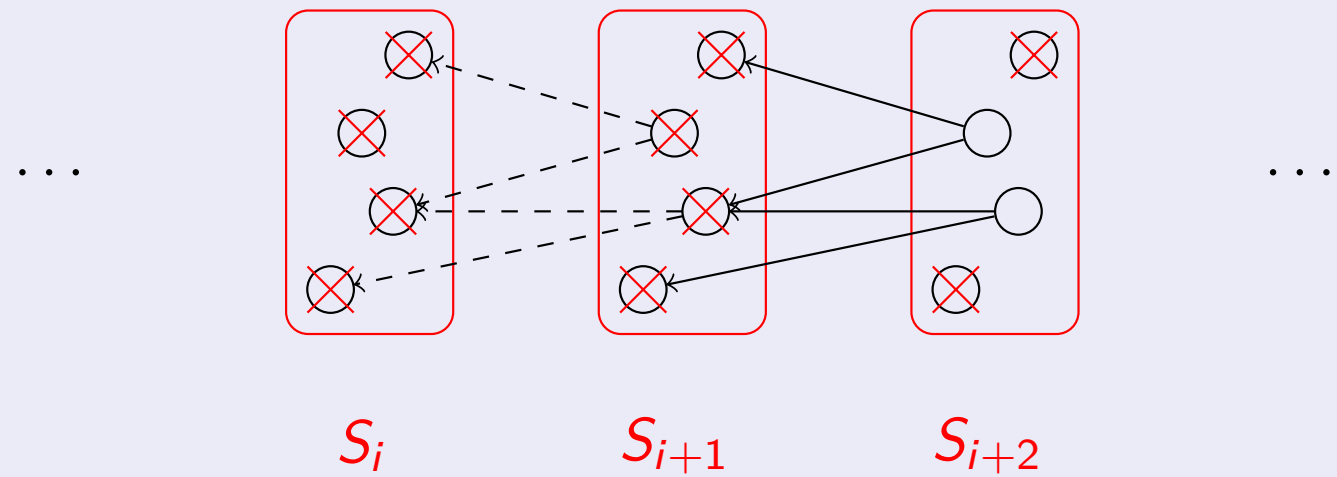
Neispunjivi vrhovi po razinama



Zašto je uvjet zaustavljanja dovoljan?

- Svaki od tih vrhova je neostvariv na razini S_i jer bi inače imali ostvariv vrh na razini S_{i+1}

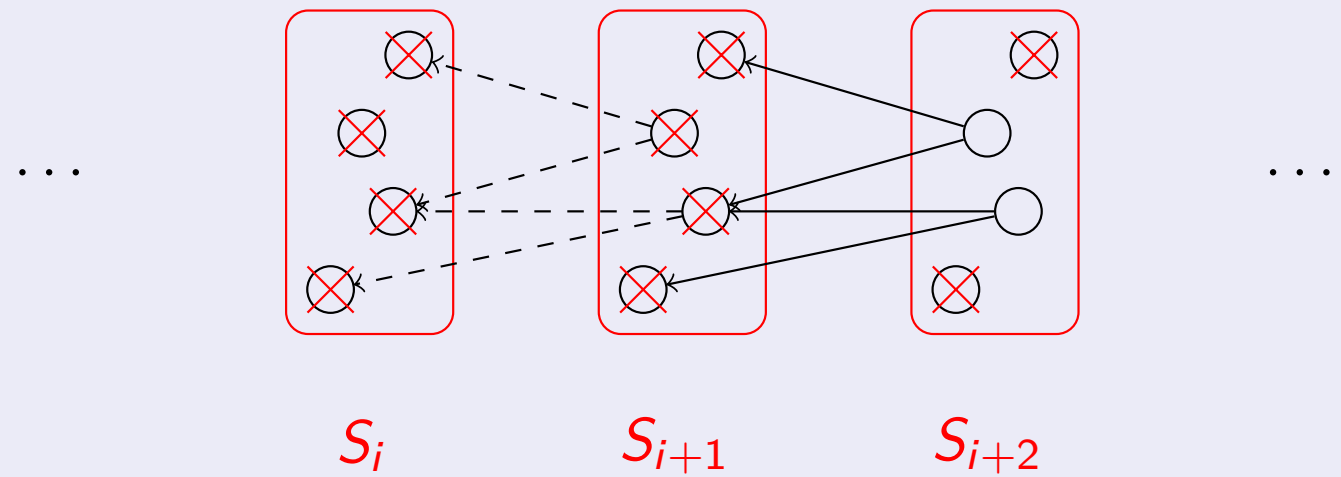
Neispunjivi vrhovi po razinama



Zašto je uvjet zaustavljanja dovoljan?

- Jer je skup svih neostvarivih vrhova na razinama S_i i S_{i+1} jednak, svi ti vrhovi su neostvarivi i na razini S_{i+1}

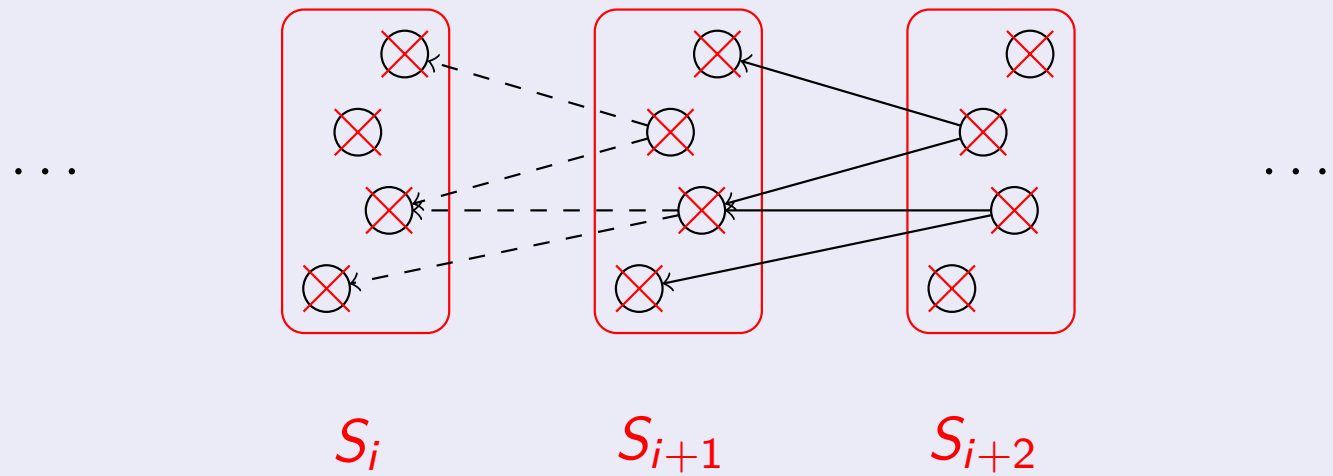
Neispunjivi vrhovi po razinama



Zašto je uvjet zaustavljanja dovoljan?

- Slijedi da je i novoposjećeni vrh neostvariv

Neispunjivi vrhovi po razinama



GRAPHPLAN algoritam

Algoritam

funkcija GRAPHPLAN(p)
    $G \leftarrow$ INICIJALNIGRAF(p)
    $ciljevi \leftarrow p.ciljevi$
    $nstv \leftarrow \emptyset$
    za $t \leftarrow 0 \ldots \infty$ **čini**
        ako *ciljevi* nisu *mutex* u $S_t \in G$ **tada**
            $rj \leftarrow$ NADJIRJESENJE($G, ciljevi, t, nstv$)
            ako *rj* postoji **tada**
                vrati *rj*
        ako su dvije razine G i *nstv* jednake **tada**
            vrati neuspjeh
        $G \leftarrow$ PROSIRIGRAF(G, p)

GRAPHPLAN uvijek završava

Teorem

Algoritam GRAPHPLAN *uvijek završava, tj. uvijek postoje dvije jednake razine grafa i skupa neostvarivih ciljeva.*

Ideja dokaza.

1. Broj literala monotono raste po S razinama.
   - Ako se literal pojavio u nekoj razini, zbog praznih operacija će se pojaviti i u sljedećoj.
2. Broj *mutex*-a među literalima monotono pada.
   - Ako dva literala nisu *mutex* u jednoj razini, zbog praznih operacija neće biti *mutex* ni u slijedećoj
3. Broj neostvarivih vrhova monotono pada.
   - Ako je neki skup ciljeva neostvariv u nekoj S razini, tada je neostvariv i u svim prethodnim razinama. Kada bi bio ostvariv u nekoj od prethodnih razina, onda bi zbog praznih operacija bio ostvariv i u svakoj slijedećoj razini.

Literatura


S. Russell, P. Norvig
Artificial Intelligence: A Modern Approach (3rd edition)
Prentice Hall, 2009.