nedeljainformatike v6.0

Planiranje i navigacija autonomnih robota

Vladimir Milenković

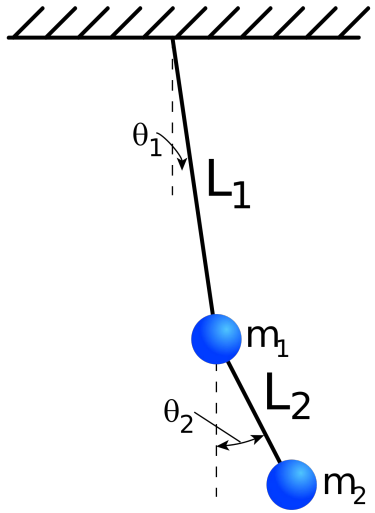
Matematička gimnazija

28. 04. 2021.

Sadržaj

Kako možemo matematički opisati robota?

- Kako uopšte da razmišljamo o robotima? Kako da ih predstavimo?
- Želimo da matematički, formalno, definišemo prostor u kome naš robot obitava -- i to tako da ga opišemo u što je manje moguće dimenzija radi lakšeg izračunavanja
- Pitanje: u koliko najmanje dimenzija možemo opisati kretanje objekta na slici?



Holonomičnost robota

- Za nekog robota kažemo da je **holonomičan**, ukoliko ima isti broj stepena slobode kao i broj stepena slobode koje možemo da kontrolišemo.
- Na primer, automobil ima tri stepena slobode: pozicija u ravni (2) i orijentacija, a samo dva stepena slobode koje možemo da kontrolišemo (ubrzanje -- gas), i ugao za koji je zarotiran volan. Dakle, automobil nije holonomičan.
- Voz ima jedan stepen slobode -- poziciju na šinama (pretpostavke), i jedan stepen slobode koji možemo da menjamo -- brzinu (ili ubrzanje), tako da je voz holonomičan.
- Pitanje: šta od navedenih “robota” je holonomično? (klatno, helikopter, avion, ruka)

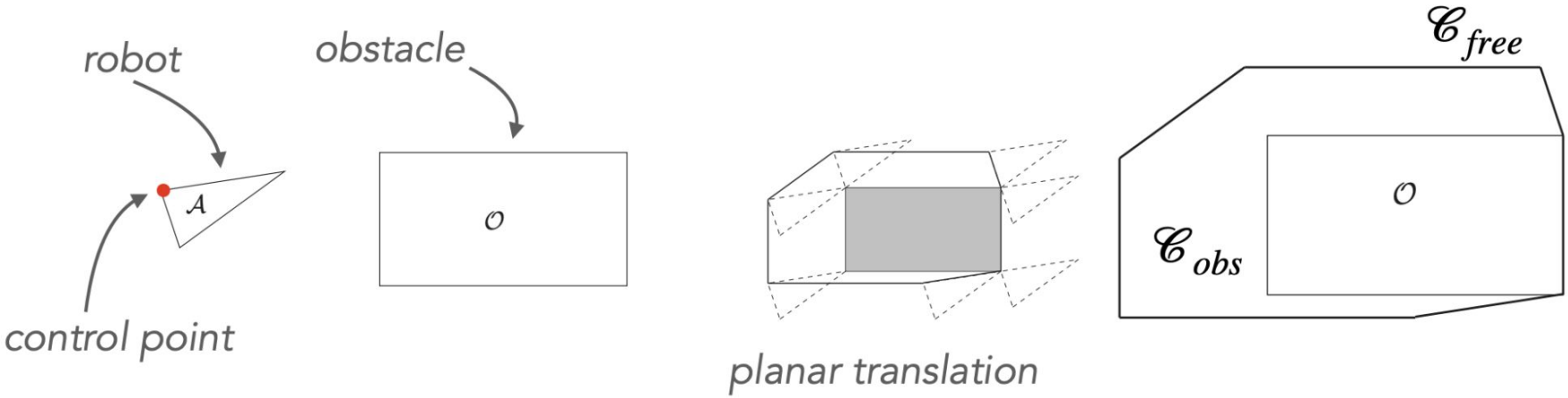
Prostor svih stanja

- Dakle, najrazumniji način kojim možemo opisati naš objekat (ili našeg robota) je vektorom vrednosti, gde svaka vrednost odgovara trenutnom stanju svakog od promenljivih parametara našeg robota
  - Ugaona vrednost za svaki zglob koji može da se rotira
  - Dužina svakog dela koji može da se isteže
  - Pozicija centra robota ukoliko može da se kreće
  - Itd...

- Ukoliko imamo **N** promenljivih parametara, naš robot će zauzimati tačno jednu tačku u prostoru **W** = R^N, a naš prostor svih mogućih stanja **C** će biti podskup istog tog skupa
- Neki deo **W** će nam biti nepristupačan (fizički, ili zbog izgradnje našeg modela), nazovimo taj deo prostora **O**, podskup **W**
- Ukoliko je A(q) skup svih tačaka koje robot zauzima dok je u stanju q, možemo definisati skup svih stanja u koje ne možemo da dođemo kao $\mathbf{C_{obs}}$= {q \in **C** | A(q) \cap O = 0}, a ostatak **C** kao $\mathbf{C_{free}}$= **C** / $\mathbf{C_{obs}}$

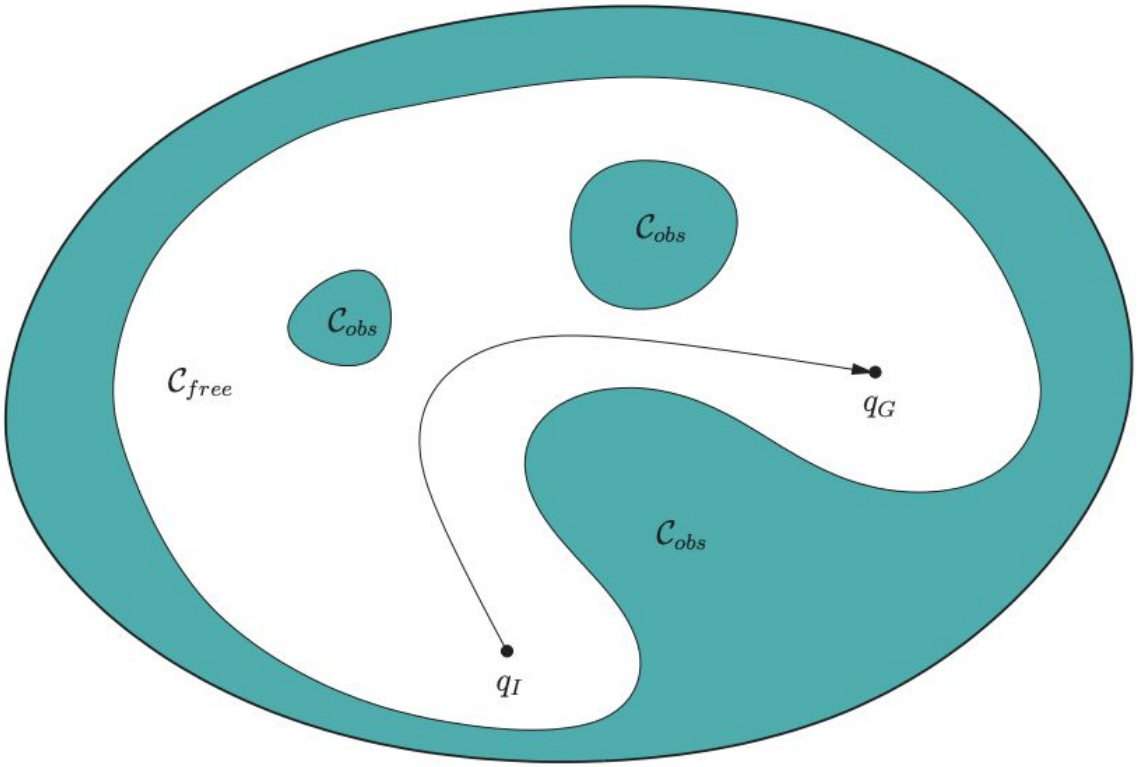
Prostor svih stanja

- Kao što smo rekli, robota možemo predstaviti nekom kontrolnom tačkom i skupom svih tačaka u odnosu na kontrolnu koju on obuhvata, kao na slici ispod
- Takozvanom Minkowski sumom (za skupove vektora A i B, Minkowski suma vektora A i B je skup svih vektora a + b, gde a pripada A i b pripada B), možemo odrediti $\mathbf{C_{obs}}$ ukoliko znamo izgled našeg robota i **O**



Problem koji rešavamo

- Neka nam je dato **C** = $\mathbf{C_{obs}}$ + $\mathbf{C_{free}}$, kao i početna i krajnja tačka $\mathbf{q_{initial}}$, $\mathbf{q_{goal}}$ u $\mathbf{C_{free}}$
- Naš zadatak je da nađemo put *p* : [0, 1] -> $\mathbf{C_{free}}$, tako da je *p* neprekidna funkcija, *p*(0) = $\mathbf{q_{initial}}$, *p*(1) = $\mathbf{q_{goal}}$



Šta očekujemo od našeg rešenja?

- Ovaj zadatak je možda jako dobro definisan, ali savršeno rešenje ne postoji (zašto?)
- Svakako, iako savršeno ne postoji, to ne znači da razna predložena rešenja ne možemo porediti...
- Neka svojstva koje neko rešenje problema može imati ili ne, su:
  - Kompletno: ukoliko neko rešenje (validan put) postoji, nalazi neki put, ukoliko ne, vraća da nema rešenja
  - Polu-kompletno: ukoliko neko rešenje (validan put) postoji, nalazi neki put, ukoliko ne, može raditi zauvek
  - Probabilistički kompletno: ukoliko rešenje postoji, verovatnoća da će biti nađeno teži 1 kako vreme teži beskonačnosti
  - Rezolucijski kompletno: ukoliko rešenje postoji, nalazi ga, ukoliko ne, vraća da rešenje postoji sa datom rezolucijom

Egzaktne metode (tačne)

- Kako možemo rešavati ovaj zadatak? Ideja: napravimo graf od svih (ili bar razumno mnogo) tačaka u kojim ima smisla da se nalazimo
  - Šta ako znamo da su sve prepreke krugovi? Deluje da možemo da se iskobeljamo..
  - Šta ako su sve prepreke pravougaonici? Možda, ali malo teže..
  - Ukoliko uspemo da napravimo graf od onoliko stanja za koje mislimo da nam je dovoljno, možemo pustiti neki od poznatih algoritama za traženje najkraćeg puta
    - Dijkstra? Radiće zauvek
    - A*? Radiće zauvek / 2
    - Varijante A*... nekad dovoljno dobro
- Koliko je zapravo teško uraditi ovo u opštem slučaju? **PSPACE-hard**
  - Jako jako teško... Prisetimo se da je NP podskup od PSPACE-hard
- Šta su problemi: mnogo potrebnih dimenzija da opišemo ne toliko komplikovane robote, neprijatno predstavljanje prepreka u **C**, itd..

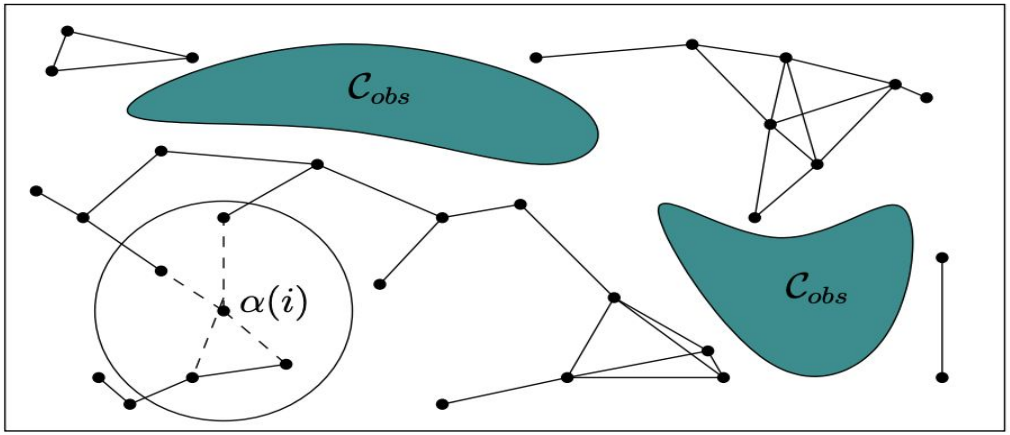
Sadržaj

Motivacija za neegzaktnošću

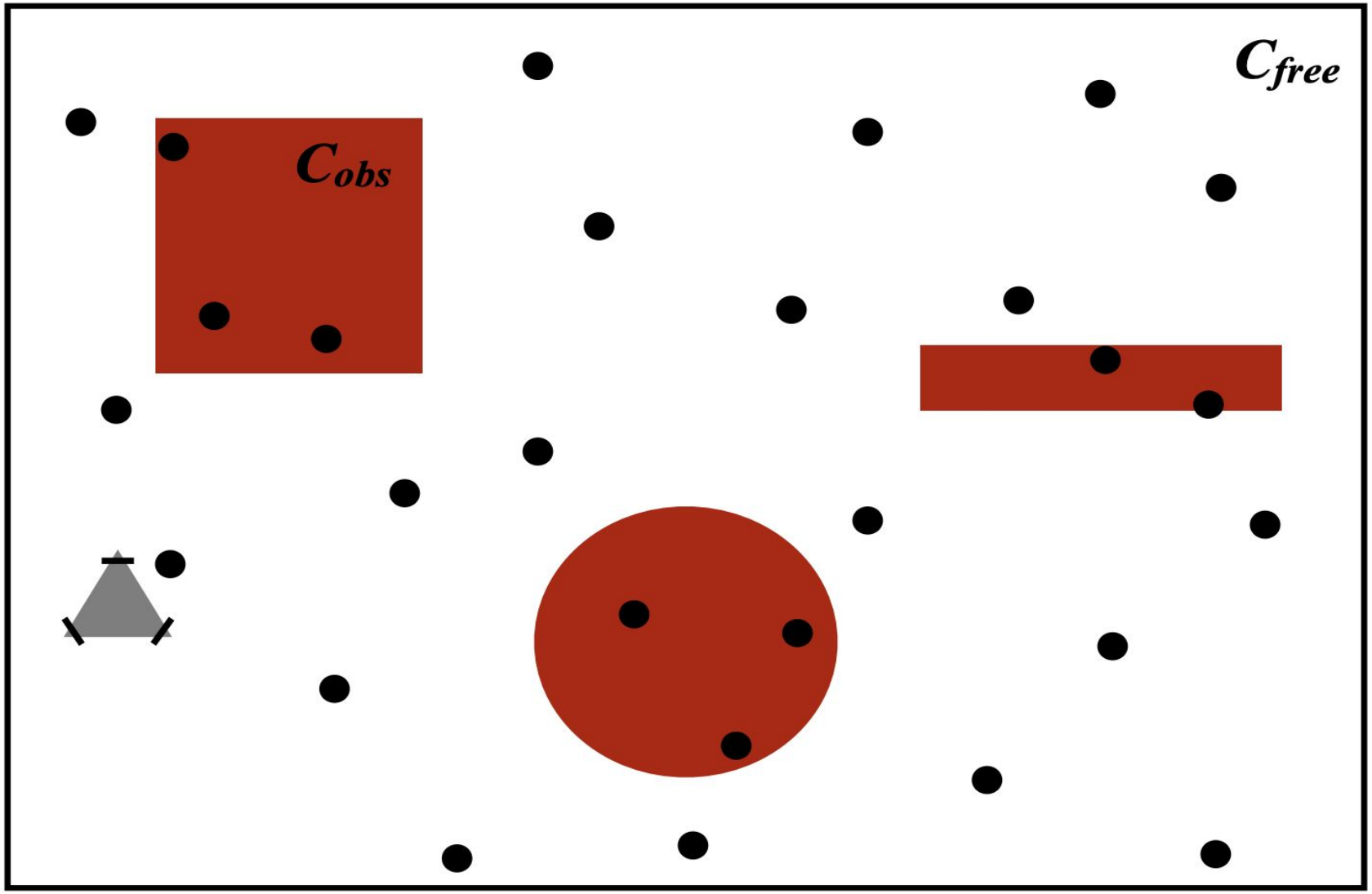
- Kao što smo videli, nema previše vajde od traženja algoritma koji će nam naći **tačno** rešenje u nekoj razumnoj složenosti
- Kao i u mnogim drugim sličnim problemima pribegavamo alternativnim metodama rešavanja -- manje više, olabavljujemo uslov da nam algoritam mora biti potpun i tačan kako bismo, nadamo se, dobili ogromne uštedu na efikasnosti
  - Loš (ili ne toliko) primer: provera da li je broj prost
  - Malo bolji: za date matrice A, B, C, proverimo da li je AB = C
  - I mnogi drugi... mogu navesti još zanimljivih primera nakon predavanja
- Imamo nekoliko različitih ideja koje možemo probati ovde, i na ovom predavanju ćete videti neke od njih

PRM (Probabilistic roadmaps)

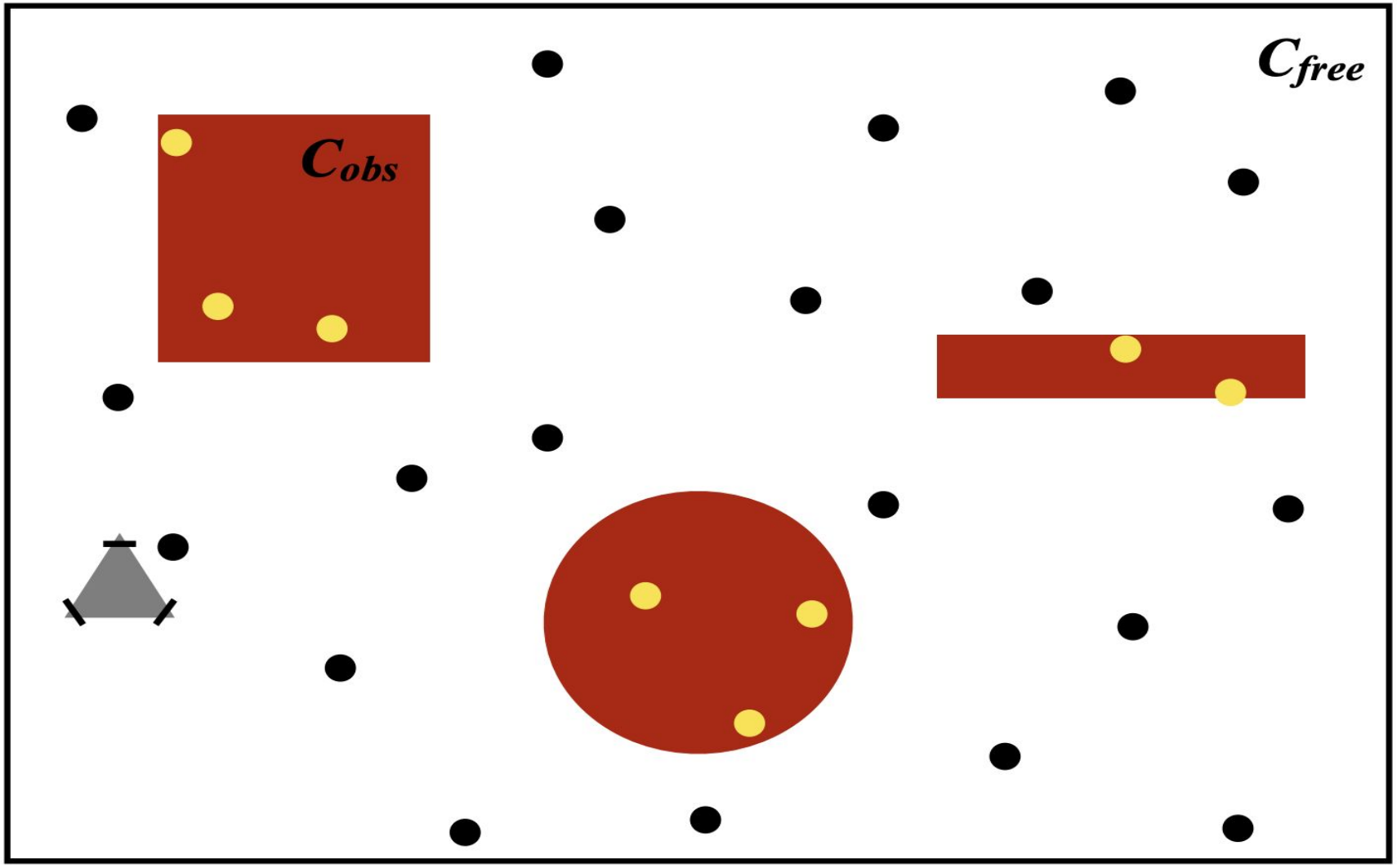
- Ideja: uzmimo neki broj **N** i generišimo **N** tačaka u prostoru nasumično (ne tako lak problem, psst). Te tačke, uključujući i početnu i krajnju, će biti čvorovi našeg grafa koje pravimo
- Za svaku tačku, nađimo njeno blisko okruženje (potrebno: definisati metriku u **C**). Ovaj korak možemo i preskočiti i pretvarati se da je svaka tačka sused svake druge tačke, još jedan “tradeoff” koji imamo
- Za svaki par tačaka u bliskom okruženju, dodati granu u grafu ukoliko je moguće otići iz jedne u drugu bez sudaranja sa nekom preprekom
- Ponavljajmo ovaj proces dok ne odlučimo da prekinemo (što duže to tačnije, ali i duže traje): obično je uslov za prekid neki fiksan broj dodatih grana!



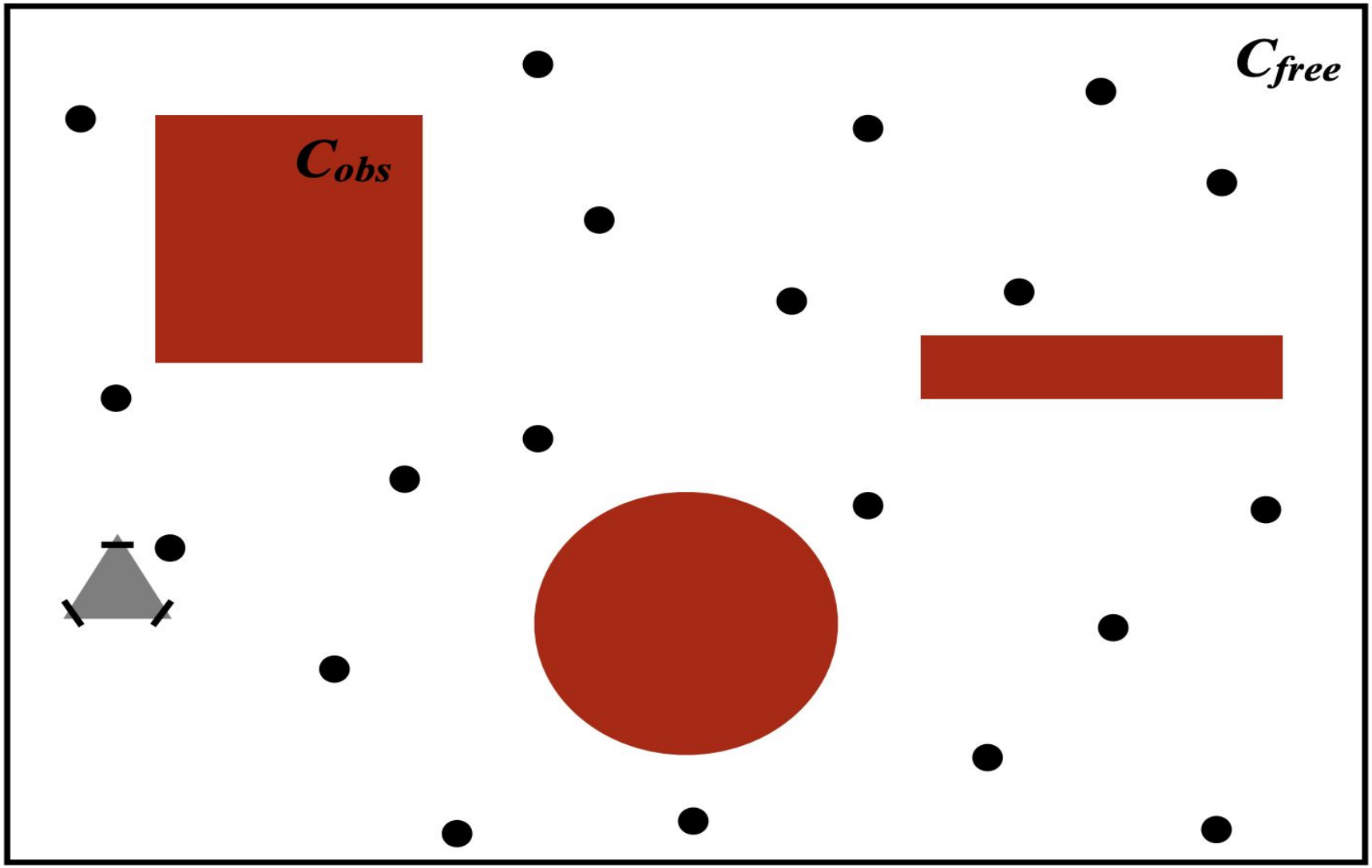
PRM u akciji



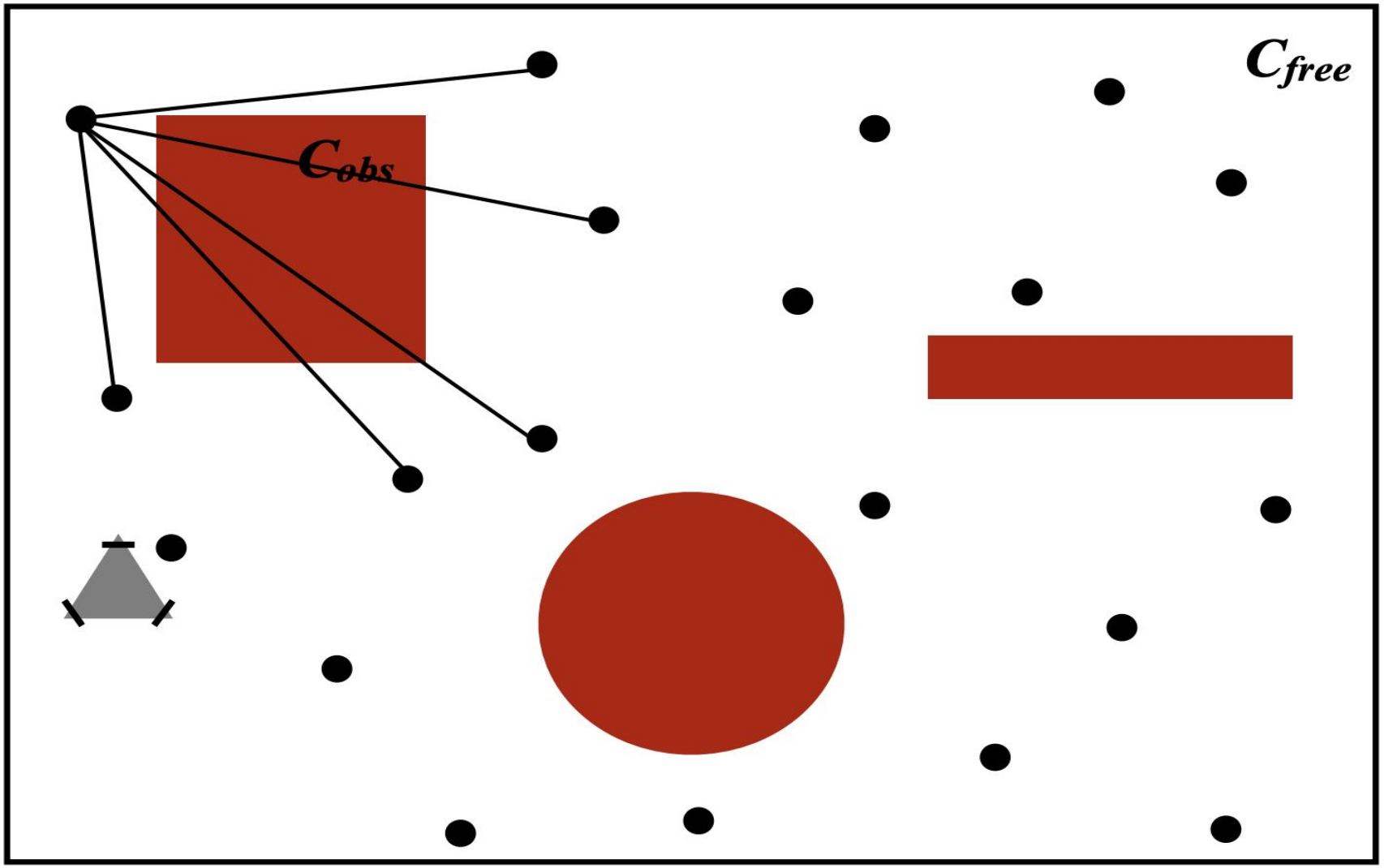
PRM u akciji



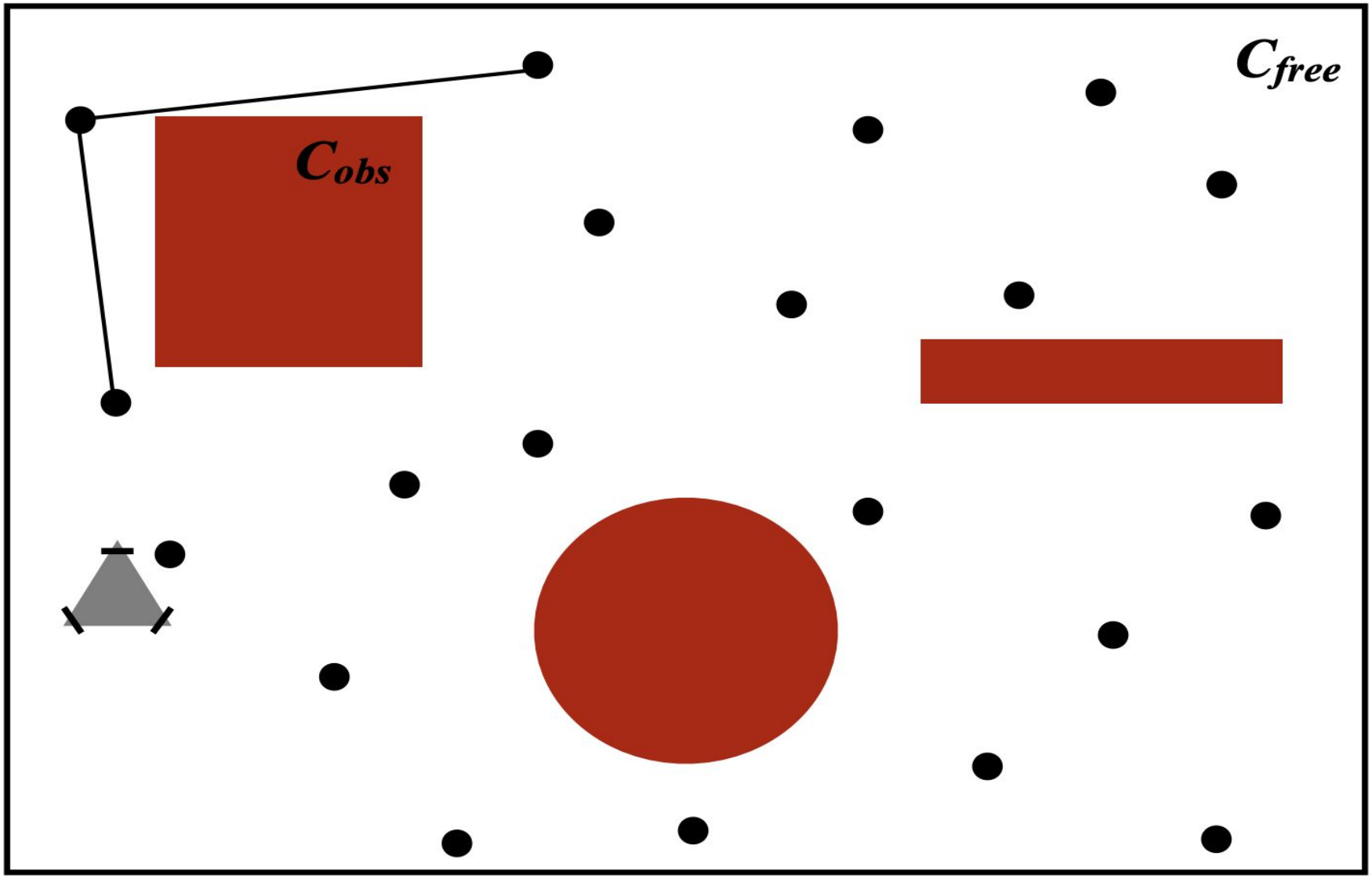
PRM u akciji



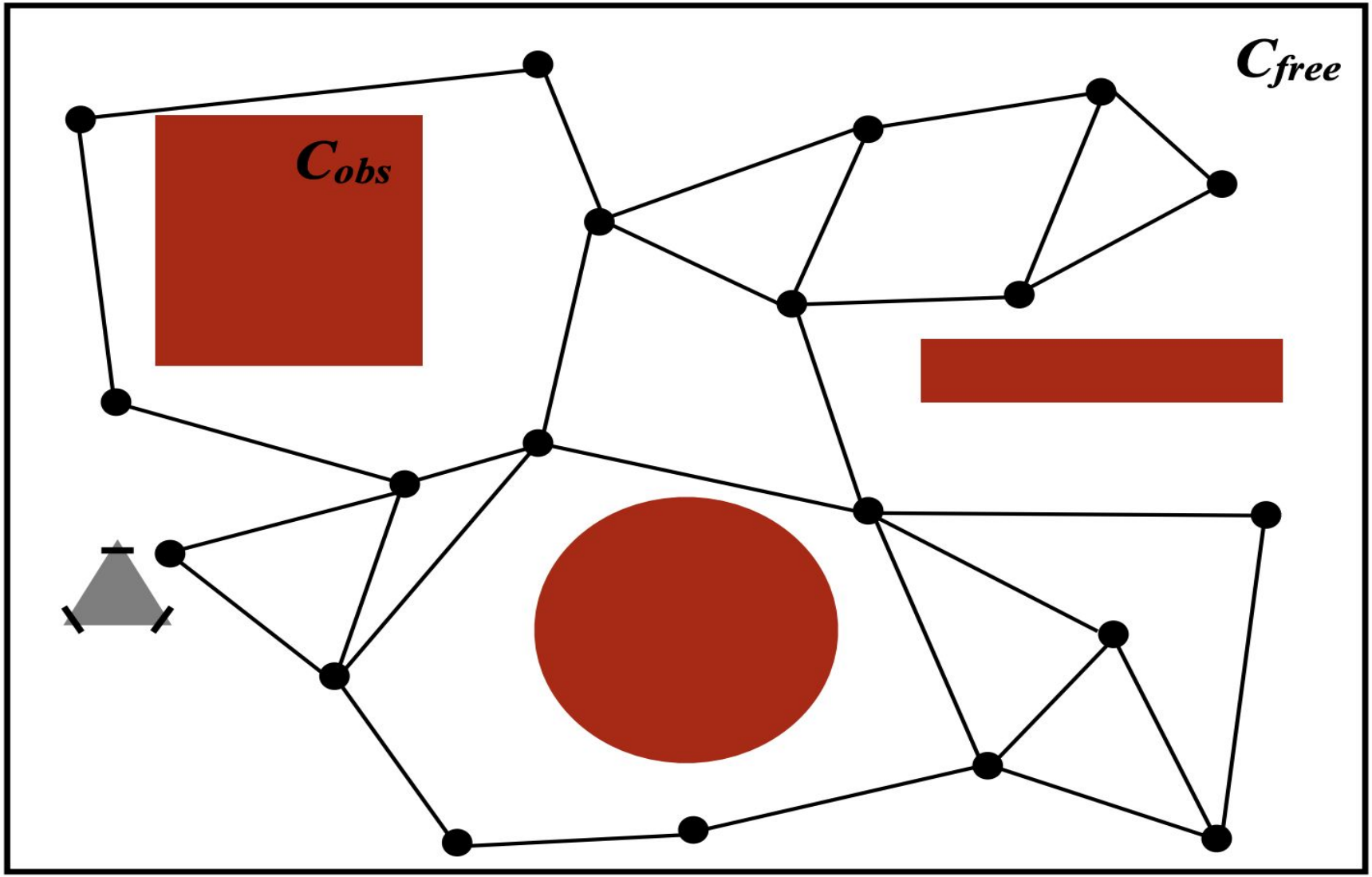
PRM u akciji



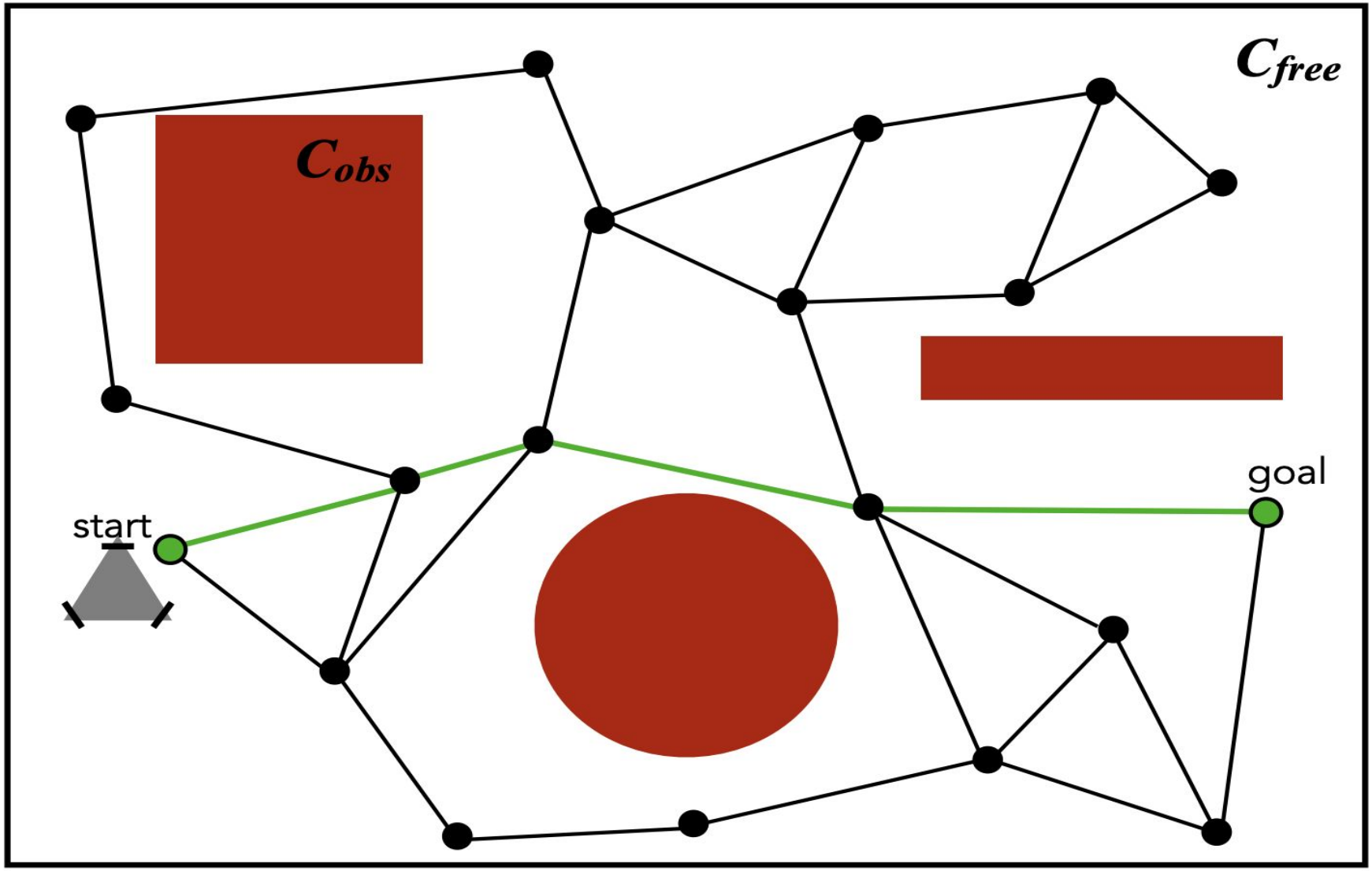
PRM u akciji



PRM u akciji



PRM -- gotovo!!



PRM -- pros and cons

- Potrebne stvari (koje ne moraju da budu lake da se implementiraju):
  - Uzorkovanje nasumične tačke u nekom prostoru (veliki problem?)
  - Provera da li ne postoji kolizija između susednih tačaka (možemo aproksimirati..)
- Pozitivne stvari:
  - Probabilistički kompletan: ukoliko bismo hipotetički pravili ovaj graf beskonačno dugo, našli bismo optimalno rešenje (..svaka tačka prostora bi bila i tačka u grafu)
  - Nema nikakav problem sa brojem dimenzija, brojem prepreka, oblikom, izgledom prepreka (uglavnom)
  - Kada se jednom napravi, možemo efikasno
- Negativne stvari:
  - Radi jako loše ukoliko imamo neke uske prolaze: treba “nabosti” tačku baš u tom prolazu kako bismo prošli
  - Generisane bliske tačke su nezavisne jedna od druge -- verovatnoća da ćemo moći da dođemo od neke do neke njoj bliske je mala. Ideja: generišimo bliske tačne koje su “slične”?

Sadržaj

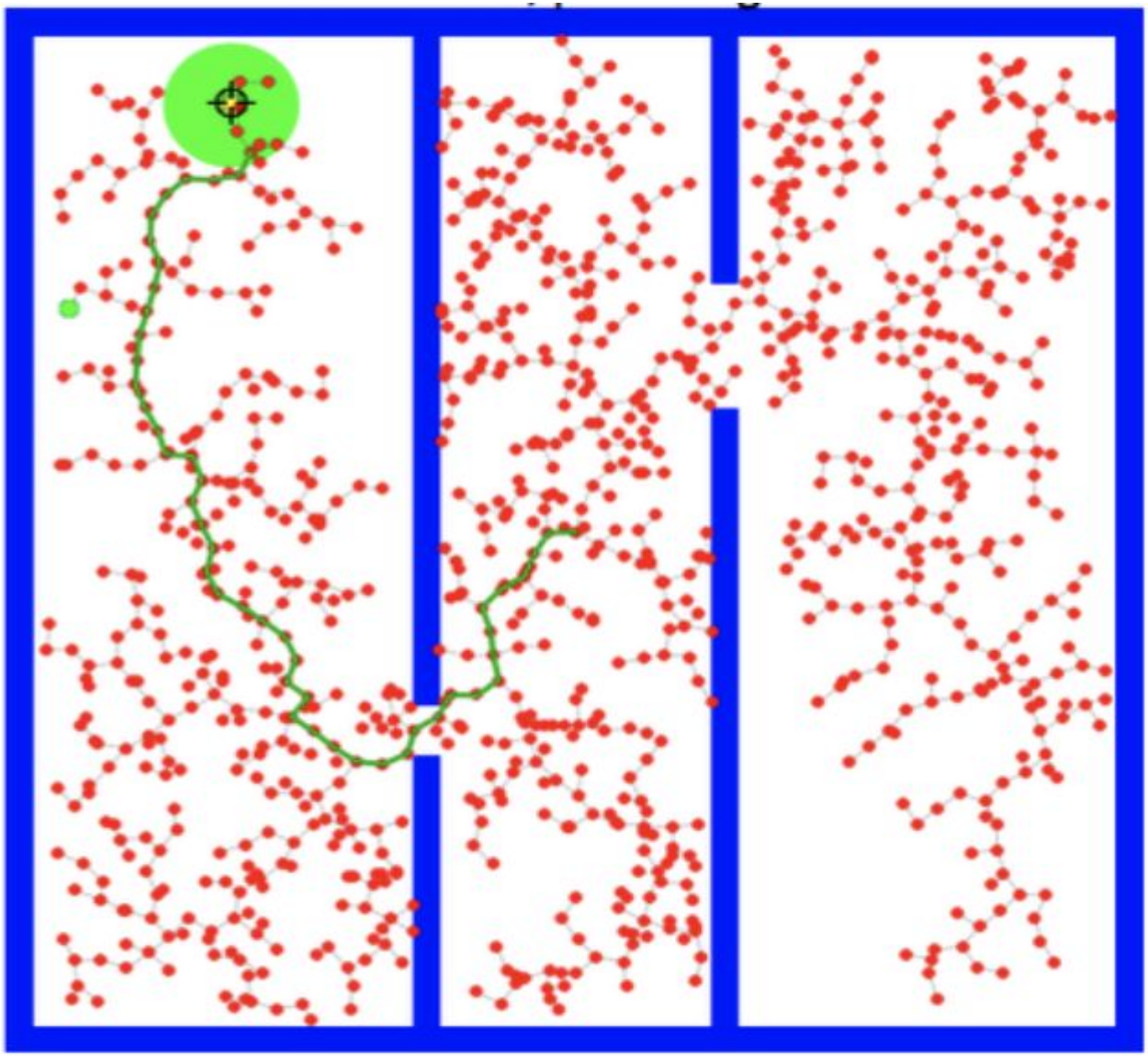
1. Problem koji rešavamo
2. Probabilistic RoadMaps
3. Rapidly-exploring Random Trees

RRT -- ideja

- Generalno, slična ideja kao PRM -- ne želimo da predstavljamo ceo graf koji bi se mogao napraviti u **C**, već samo nasumičan njegov podgraf
- Želimo da optimizujemo naš graf za tačno jedan put (tačno jedan par početne i krajnje tačke): ne treba nam graf nego **stablo**
- Na “pametan” nasumičan način generišemo tačke i produžavamo puteve u stablu, sve dok ne pronađemo put do ciljne tačke
- I na kraju želimo da to izgleda nekako...

RRT -- primer

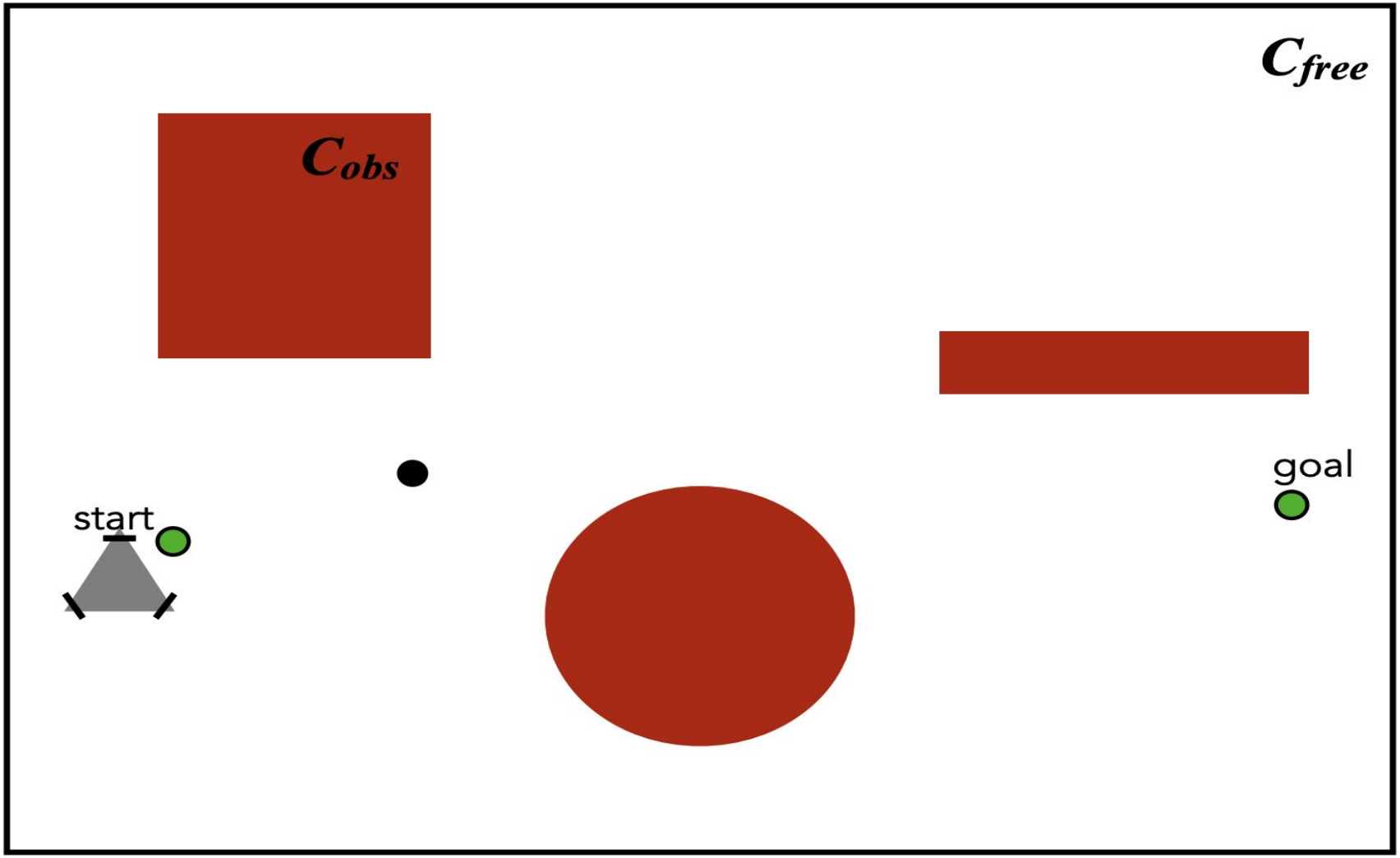
..ovako



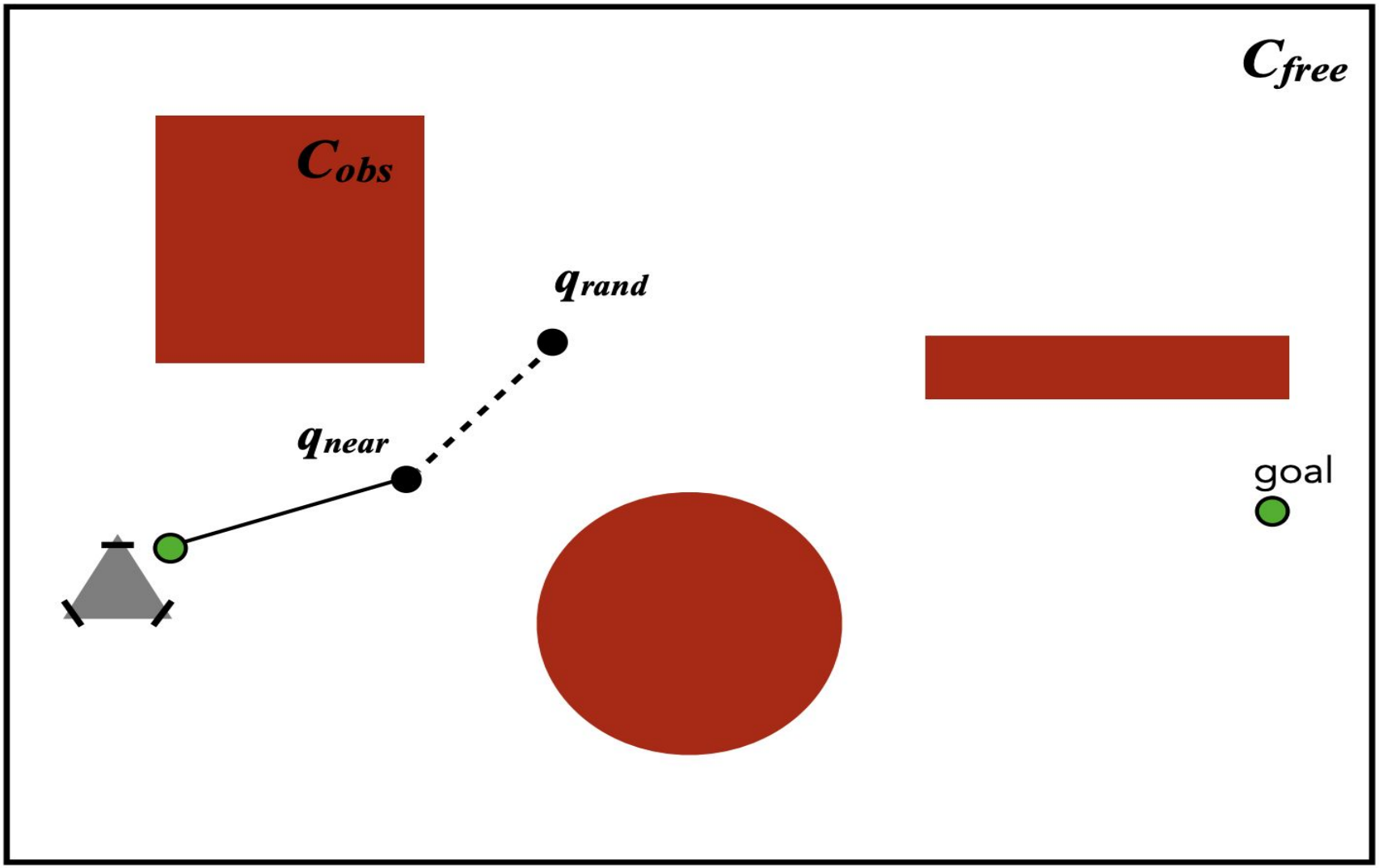
RRT -- algoritam

1. Krenimo sa grafom **G**, koji na početku sadrži samo početnu tačku $\mathbf{q_{initial}}$
2. Izaberimo random tačku (problem generisanja nasumične tačke) $\mathbf{q_{rand}}$ u **C** (sa nekom verovatnoćom, izaberimo $\mathbf{q_{goal}}$ kako bismo navodili pretragu ka cilju)
3. Nađimo najbližu tačku $\mathbf{q_{near}}$ u **G** tački $\mathbf{q_{rand}}$
4. Nađimo neku tačku $\mathbf{q_{new}}$ u okolini $\mathbf{q_{rand}}$ tako da možemo doći iz $\mathbf{q_{near}}$ u $\mathbf{q_{rand}}$ (ukoliko je robot holonomičan, uvek možemo u svaku)
5. Ukoliko postoji put bez kolizije od $\mathbf{q_{near}}$ do $\mathbf{q_{new}}$, dodamo tu granu i nastavimo dalje ukoliko nismo stigli do cilja

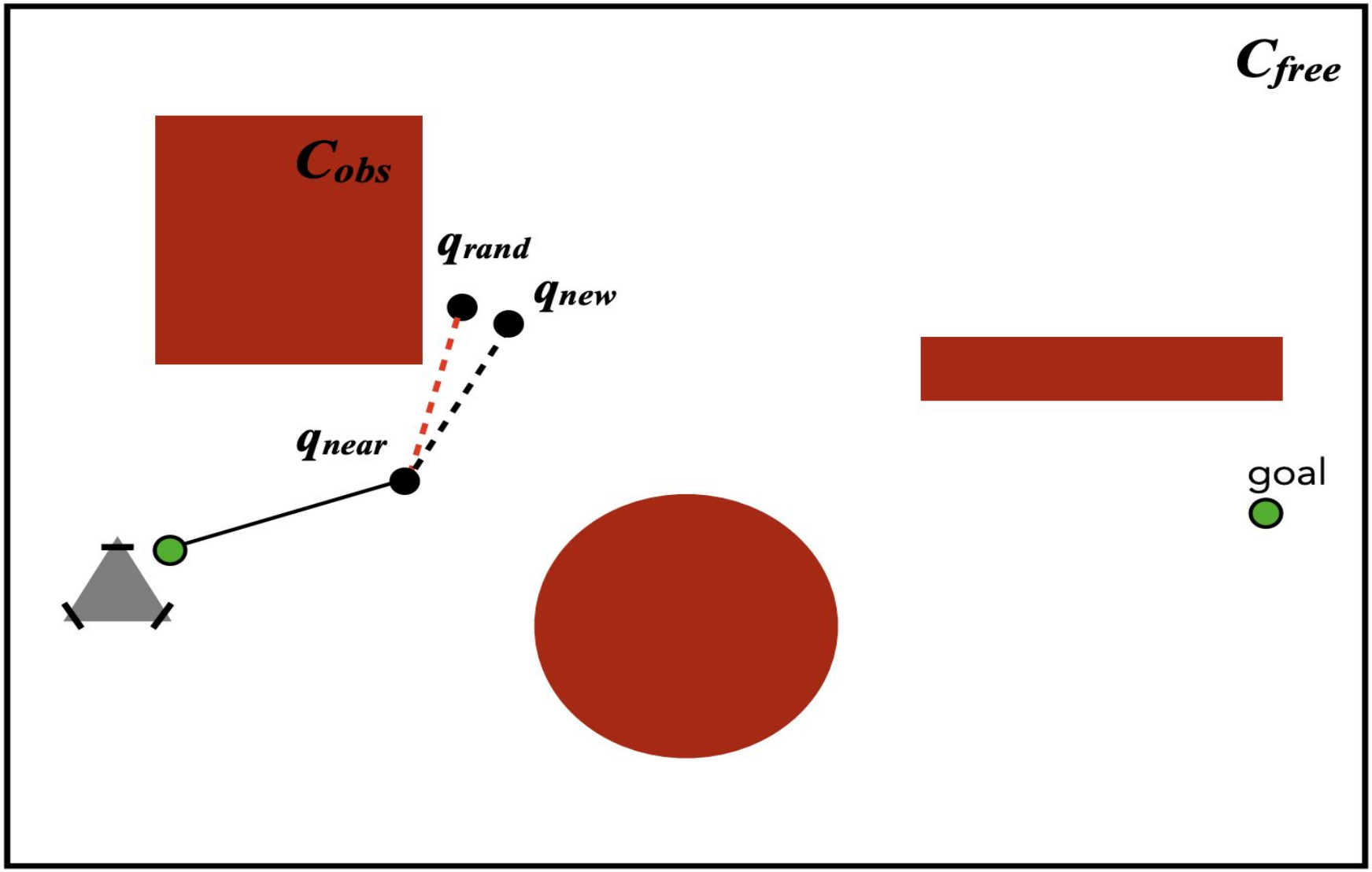
RRT u akciji



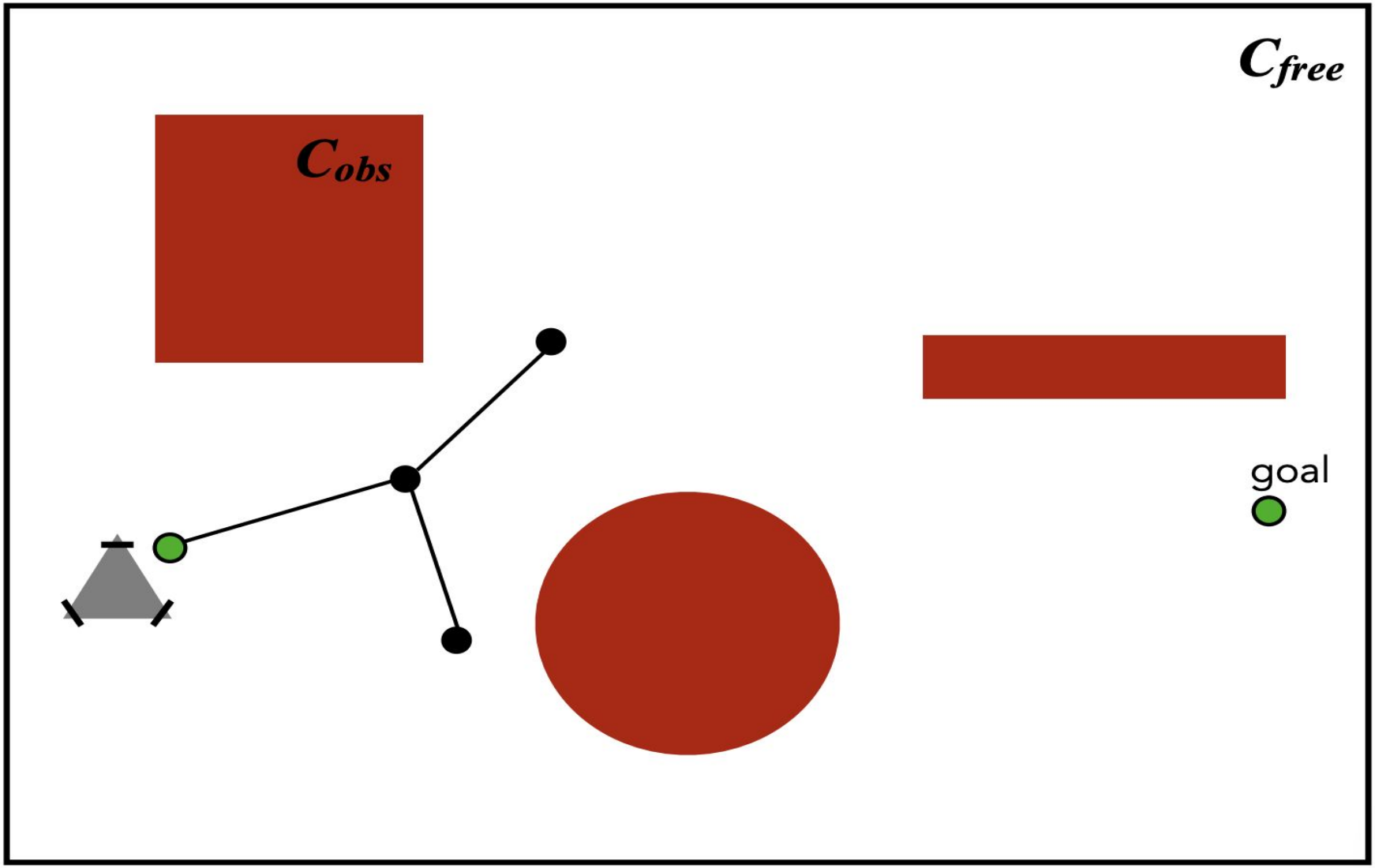
RRT u akciji



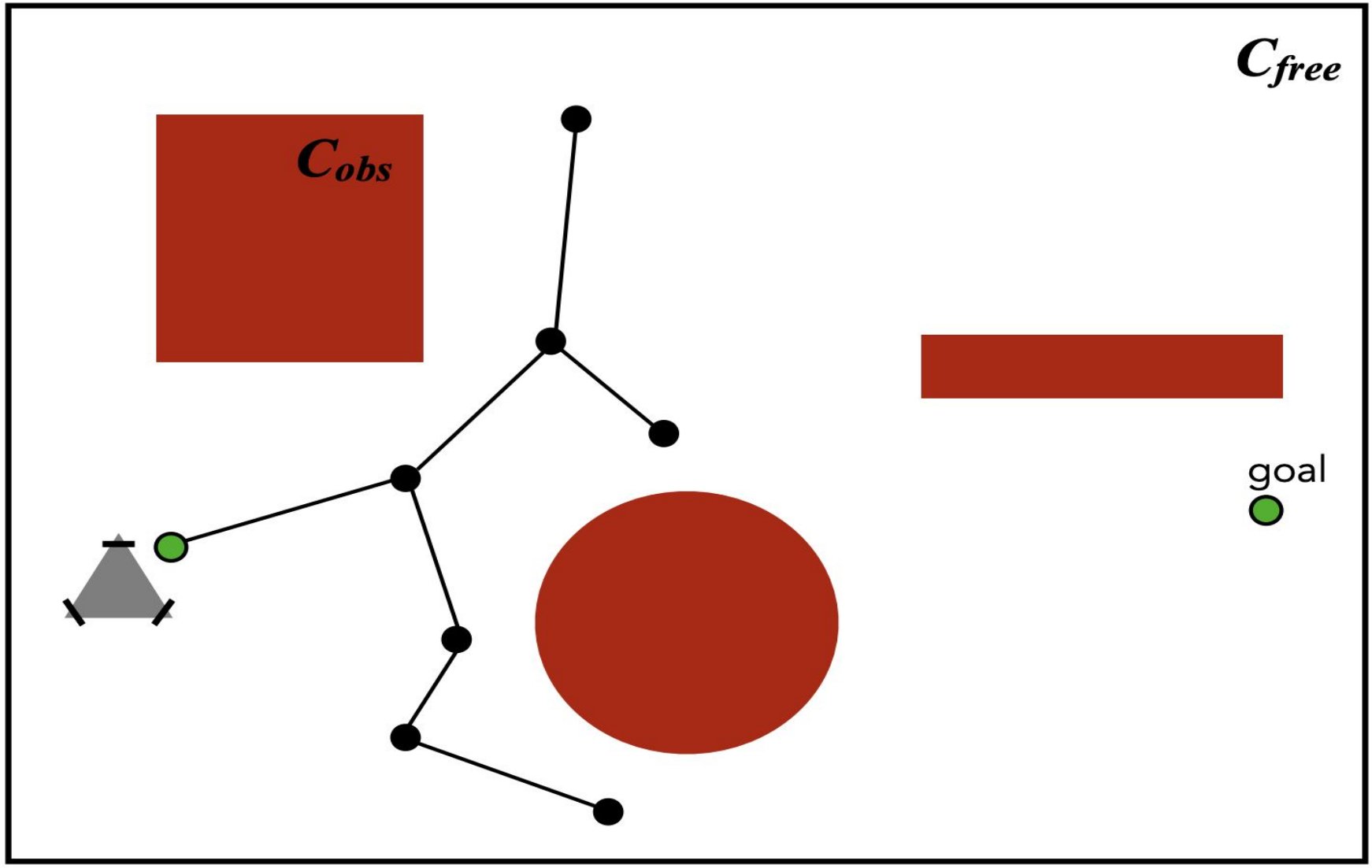
RRT u akciji



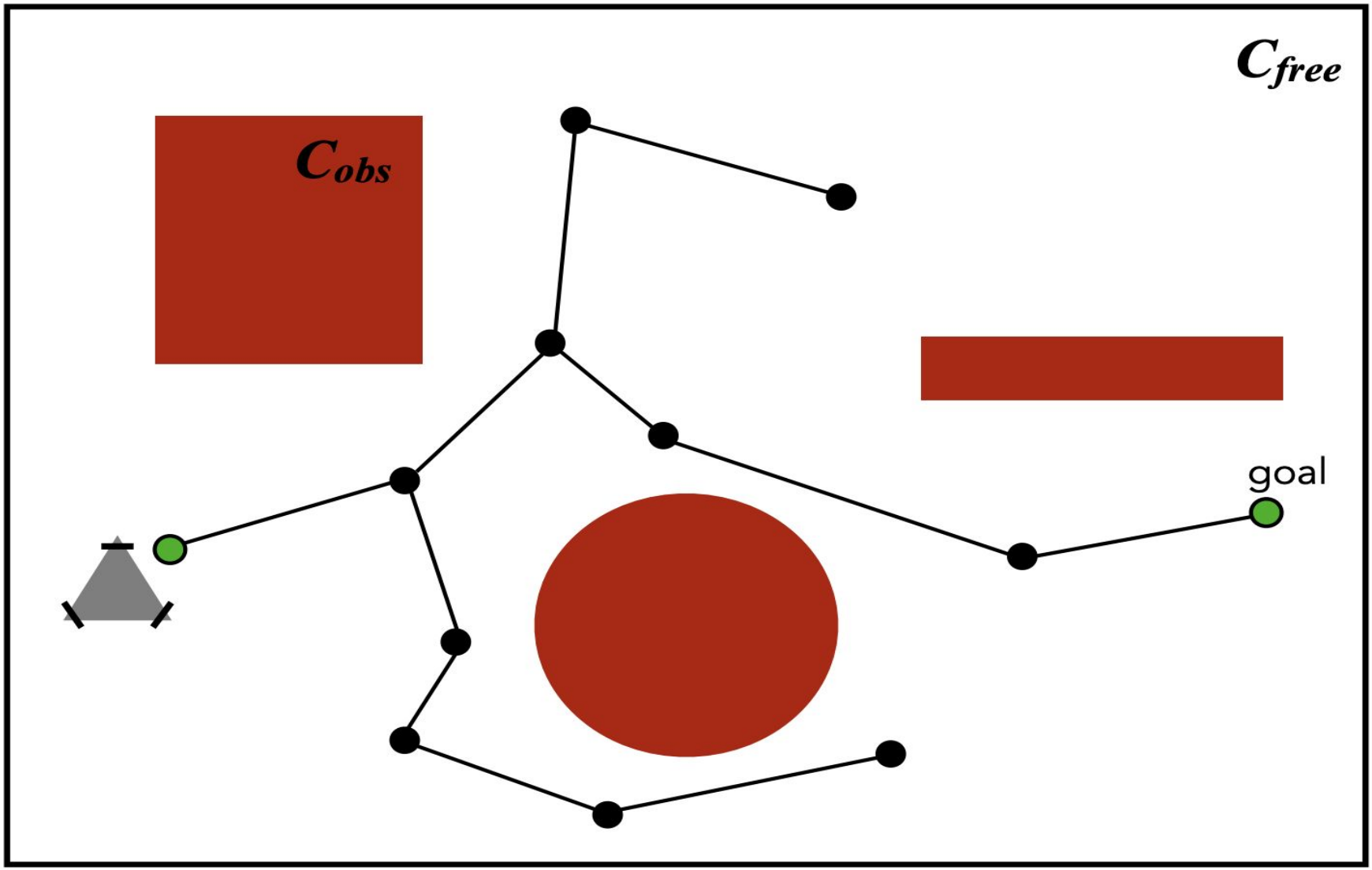
RRT u akciji



RRT -- skoro je kraj



RRT -- gotovo



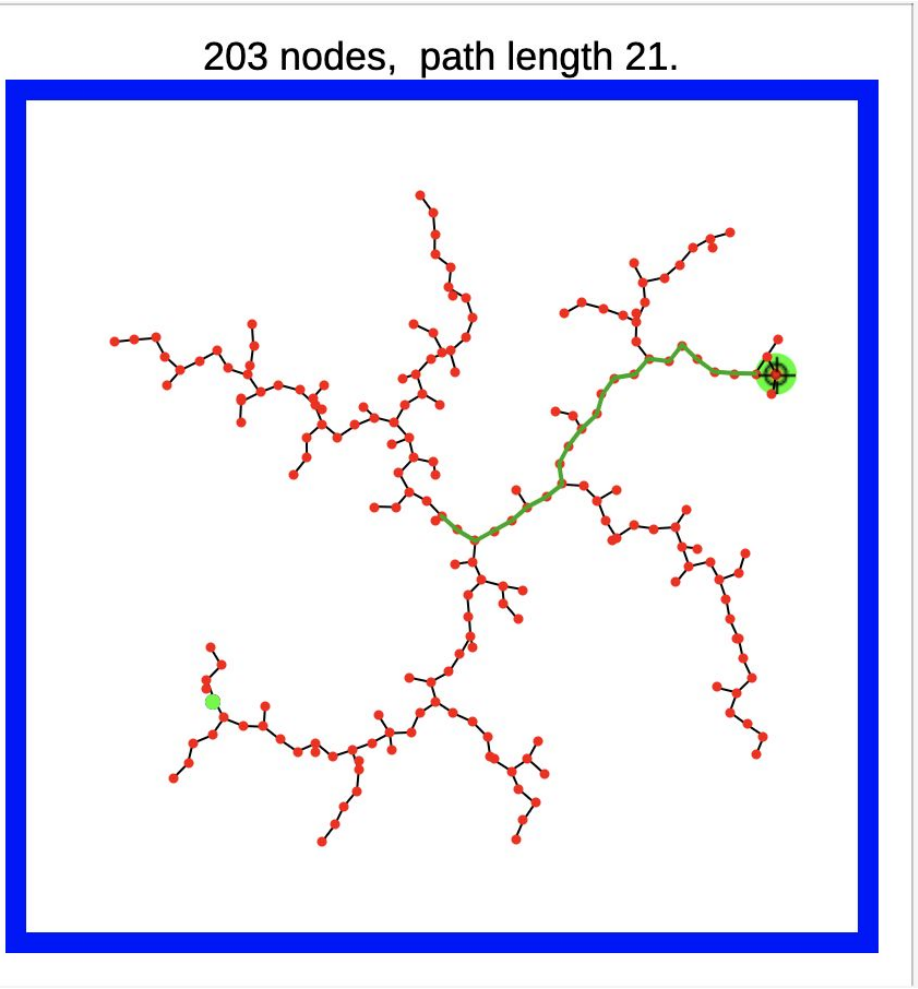
RRT -- pros and cons

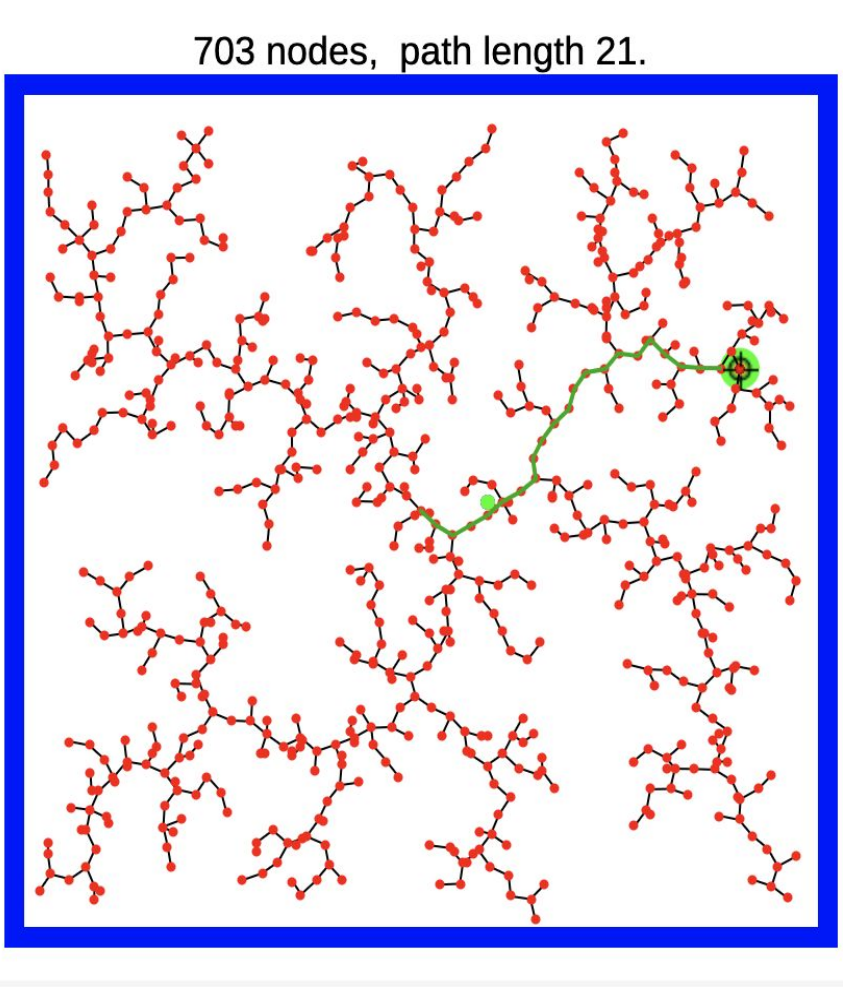
- Potrebne stvari:
  - Generisanje nasumične tačke u prostoru (isto kao PRM, ovo nam ne gine..)
  - Traženje tačke u okolini neke kojoj možemo prići (uglavnom lakši problem nego provera postojanja puta bez kolizije)
- Pozitivne stvari:
  - Probabilistički kompletan (kao i PRM), sa eksponencijalnim padom šanse za grešku
  - U proseku, radi brže nego PRM za traženje puta do na neku aproksimaciju (jako klimava rečenica, znam)
  - Bitno: RRT* je probabilistički optimalan!
- Negativne stvari:
  - Jedan RRT radi samo za jednu putanju -- ako želimo novi put u istom prostoru, moramo praviti celo novo stablo
  - Mogu se dešavati problemi i lošije performanse kada robot nije holonomičan

RRT* -- optimalna verzija RRT-a

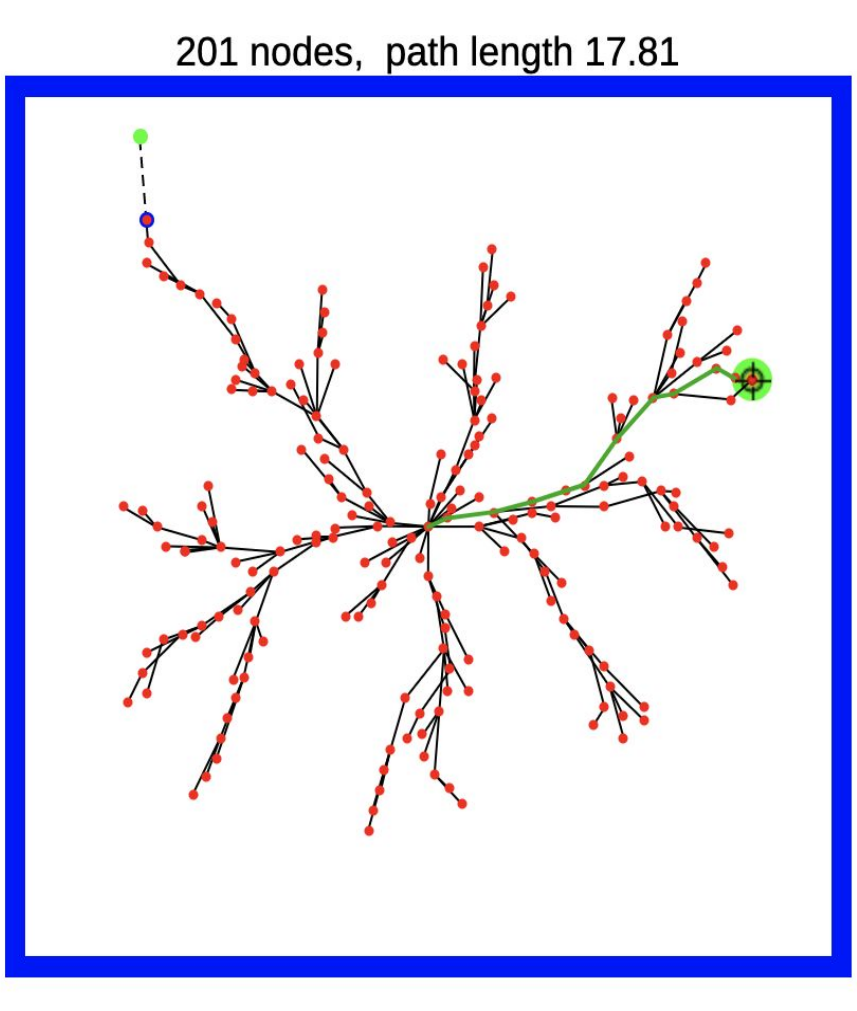
- Pri “običnom” RRT-u, prvi put nađen do nekog čvora ostaje i jedini put zauvek, iako se dodavanjem novih čvorova taj put može skratiti
- Ne želimo ovo, želimo da kako dodajemo čvorove svi putevi se eventualno smanjuju, pa..
- prilikom dodavanja novog čvora, želimo da vidimo da li taj čvor može da skrati neki već postojeći put (detalji kako možemo ovo da uradimo ostavljeni kao vežba, ali nije teško, za svaki novi čvor želimo da pogledamo koje puteve on može da skrati)
- Malo duže traje izvršavanje (zbog dodatnog izračunavanja pri skraćivanju puteva), ali sada imamo garanciju da je probabilistički optimalno! (uz dovoljno vremena, sa verovatnoćom 1 ćemo imati nađen najkraći put)
- Odličan simulator na kome se sve ovo može probati (mnogo lepši UI od mog pokušaja): https://demonstrations.wolfram.com/RapidlyExploringRandomTreeRRTAndRRT/

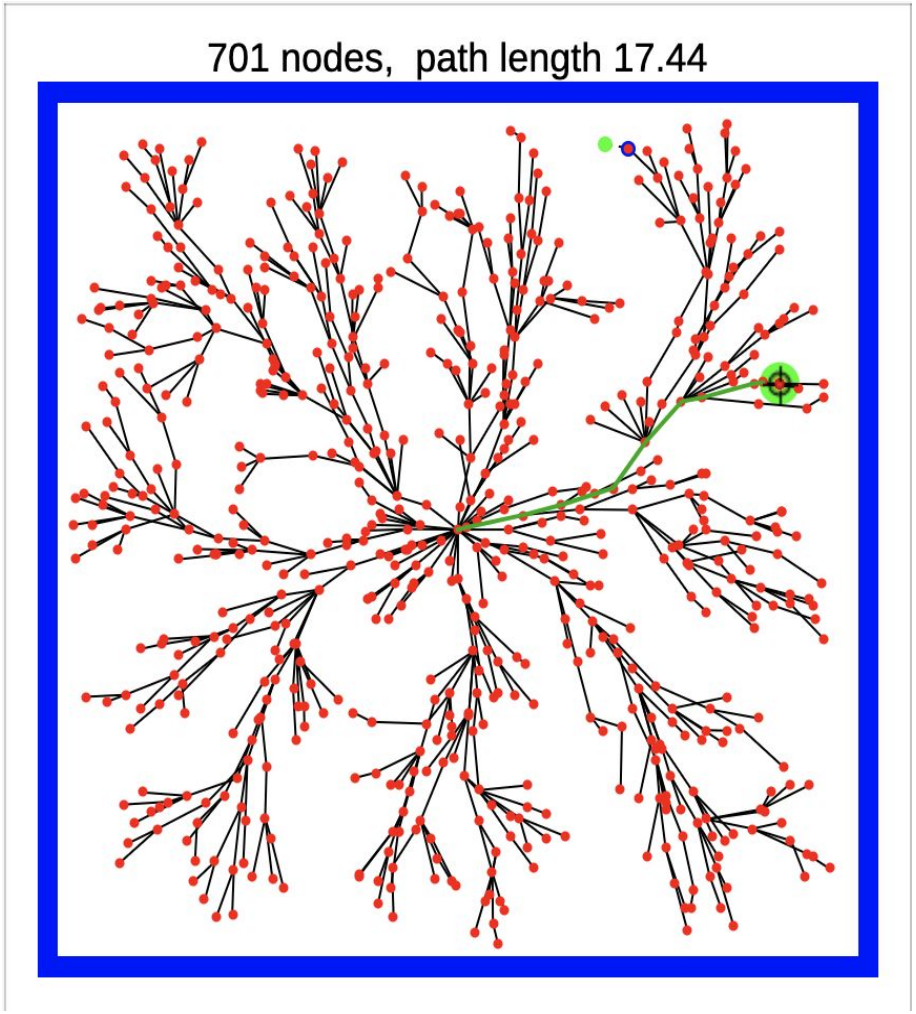
[RRT] vs RRT*





RRT vs [RRT*]





Hvala na pažnji!

Pitanja?