Hibridni algoritem d22 z uporabo požrešne metode in vračanjem kot rešitev logističnega problema skupnega potovanja

David Slatinek, Marcel Iskrač, Mateja Žvegler

Predstavitev problema

- Za delo v skladišču smo kupili nove drone. Zamislili smo si naslednji test letenja: dva drona bomo postavili nekam v skladišče in jima določili končne koordinate. Nato bomo opazovali njun let do teh koordinat. Program mora izpisati pot dronov, preprečiti moramo trk (drona nikoli ne smeta biti na istih koordinatah).
- Zaporedni poziciji drona se lahko razlikujeta v največ eni koordinati in to za največ 1 (dron se torej lahko premakne za največ 1 v smeri x, y ali z; diagonalni premiki niso mogoči). Dron lahko vedno stoji na mestu.
- Drona se ob istem času nikoli ne smeta nahajati na isti poziciji.

Prenos problema v realni svet

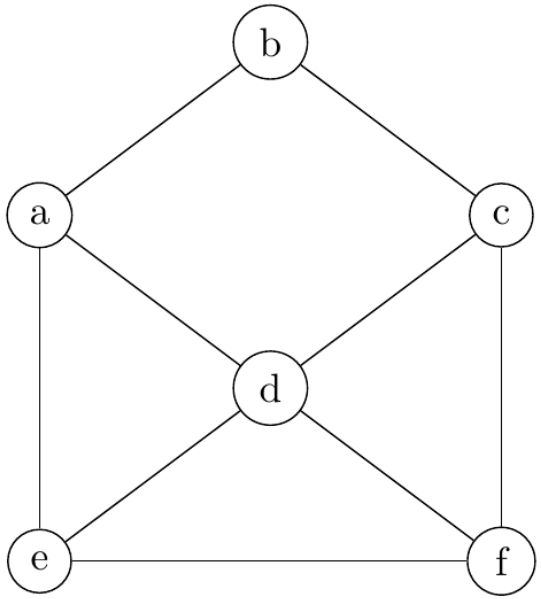
- Logistični problem
  - Amazon – avtomatizirani sistem za shranjevanje in iskanje.
- Iskanje poti – Google Maps.
- Vojska
  - Letala.

Matematični opis

- Teorija grafov
  - V – končna neprazna množica.
  - E – družina dvoelementnih podmnožic množice V.
  - G=(V,E) – graf na množici vozlišč V in z množico povezav E.
  - $uv \in E(G)$ => u in v sta sosedni vozlišči.
  - {u, v} ≠ {v, u} => usmerjen graf.
  - Povezave vsebujejo uteži => utežen graf.

Matematični opis

- Teorija grafov

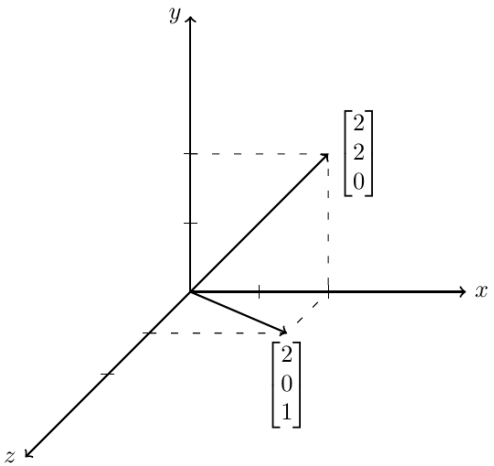


Slika 1: Neutežen, neusmerjen graf G.

Matematični opis

- Vektor
  - Urejen par točk v prostoru.
  - Velikost, smer.
- Tridimenzionalni pravokotni kartezični sistem
  - Os x, y, z.

Slika 2: Vektor $\overrightarrow{AB}$.

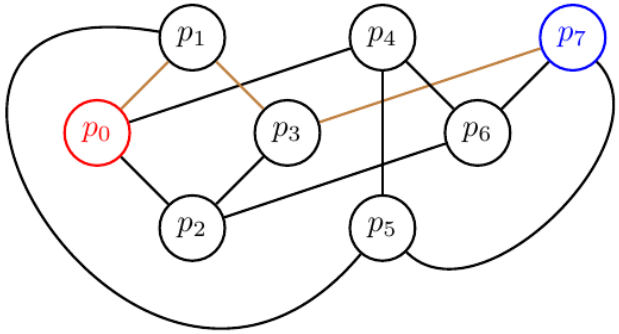


Slika 3: 3d prostor z dvema točkama: $A(2,2,0)$, $B(2,0,1)$ in njuna pripadajoča krajevna vektorja.

$$d(A,B) = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2} = \|\vec{r_b}-\vec{r_a}\|$$

Uporaba teorije grafov in vektorjev

- Položaj drona v 3d prostoru
  - Vozlišča.
- Povezave
  - Vektor.



Slika 4: Graf s pričetkom v $p_0(0,0,0)$ in koncem v $p_7(1,1,1)$. Z rjavo barvo je označena ena od rešitev.

Rešitev z grobim pristopom

- Generiramo vsa vozlišča.
- Generiramo vse povezave.
- Izvedemo grobi pristop.

Algorithm 1 Generiraj vrednosti n-te koordinate v $3d$ prostoru med podanima vrednostma.

```
1:  function GenerateCoordinates(start, end)
2:      A ← ∅
3:      while start ≠ end do
4:          ADD(A, start)                    ▷ Dodamo start v strukturo A
5:          if start < end then
6:              start ← start + 1
7:          else
8:              start ← start − 1
9:          end if
10:     end while
11:     ADD(A, start)                        ▷ Dodamo še zadnjo koordinato
12:     return A
13: end function
```

Rešitev z grobim pristopom

Algorithm 2 Generiraj vozlišča.

```
1:  function GenerateVertices(X, Y, Z)
2:      A ← ∅                                          ▷ Shranjevanje vozlišč
3:      for x ← 0 to len(X) do                         ▷ Za vsako x vrednost
4:          for y ← 0 to len(Y) do                     ▷ Za vsako y vrednost
5:              for z ← 0 to len(Z) do                 ▷ Za vsako z vrednost
6:                  A ← A ∪ CreateVertex(X[x], Y[y], Z[z])   ▷ Ustvarimo vozlišče
7:              end for
8:          end for
9:      end for
10:     return A
11: end function
```

Algorithm 3 Generiraj vse povezave med vozlišči.

```
1: function GenerateEdges(A)                           ▷ A vsebuje vsa vozlišča
2:     for i ← 0 to len(A) do
3:         for j ← i to len(A) do
4:             CreateEdge(A[i], A[j])                  ▷ Ustvarimo povezavo med vozliščema
5:         end for
6:     end for
7: end function
```

Rešitev z grobim pristopom

- Jemljemo vozlišča.
- Če je vozlišče veljavno in ne pride do trka, sprejmemo vozlišča.
- V nasprotnem primeru izvedemo mirovanje ali se vrnemo en korak nazaj.

Algoritem d22

- Dinamično ustvarjaj vozlišča.
- Dinamično ustvarjaj povezave.
- Izvedi algoritem d22.

Algorithm 5 Generiraj vrednosti vozlišča po potrebi.

```
1: function GetValue(start, end)
2:     value ← start
3:     if start < end then
4:         start ← start + 1
5:     else
6:         start ← start − 1
7:     end if
8:     yield value
9: end function
```

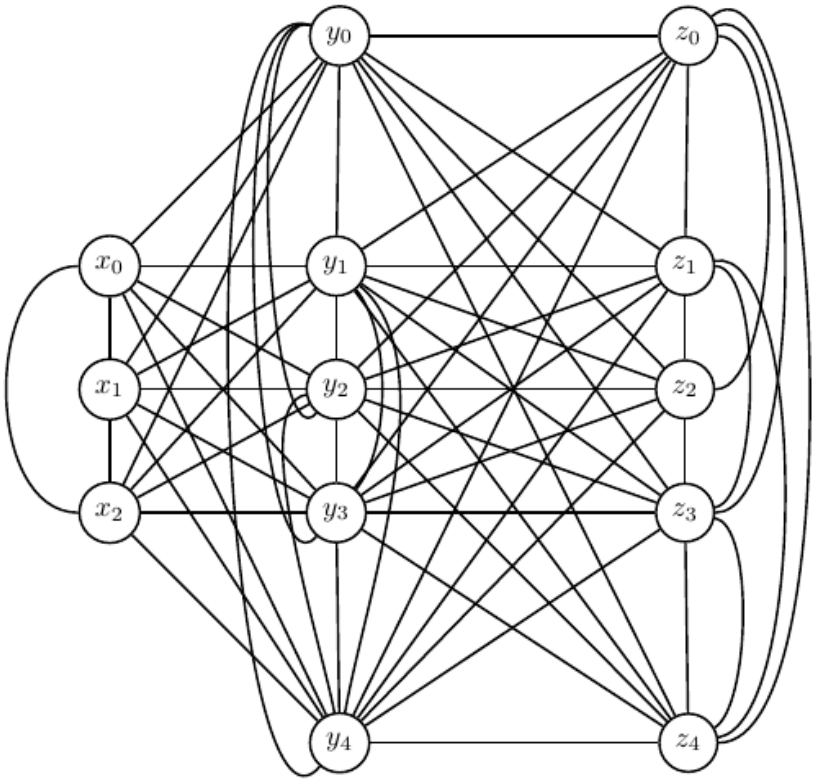
Algorithm 6 Generiraj vozlišča po potrebi.

```
1: function GetVertex(start, end)
2:     for x in GetValue(start.x, end.x) do                ▷ Glej 5
3:         for y in GetValue(start.y, end.y) do
4:             for z in GetValue(start.z, end.z) do
5:                 yield CreateVertex(x, y, z)             ▷ Ustvarimo vozlišče
6:             end for
7:         end for
8:     end for
9: end function
```

Težji problem

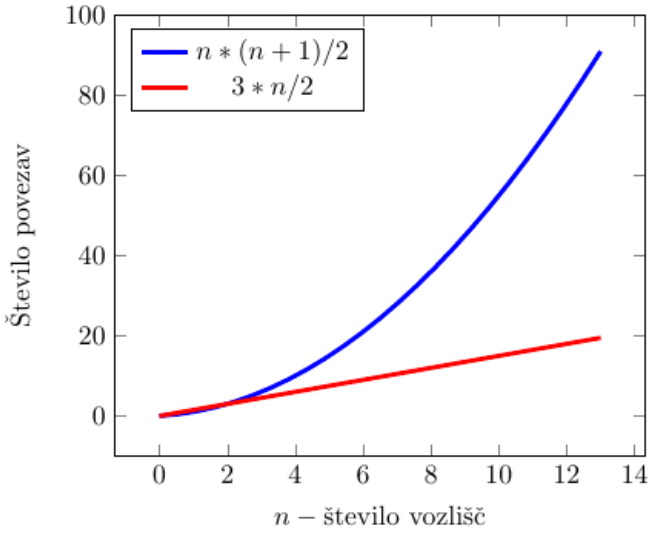
- Generiramo vse povezave => $n * (n+1)/2$ povezav
  - Časovna zahtevnost $O(n^2)$, polni graf
- Dinamično ustvarjanje povezav => $3n/2$
  - Časovna zahtevnost $O(n)$

Težji problem



Slika 5: Polni graf, pri katerem je začetna točka v *(0,0,0)*, končna točka pa v *(2,4,4)*.

Težji problem



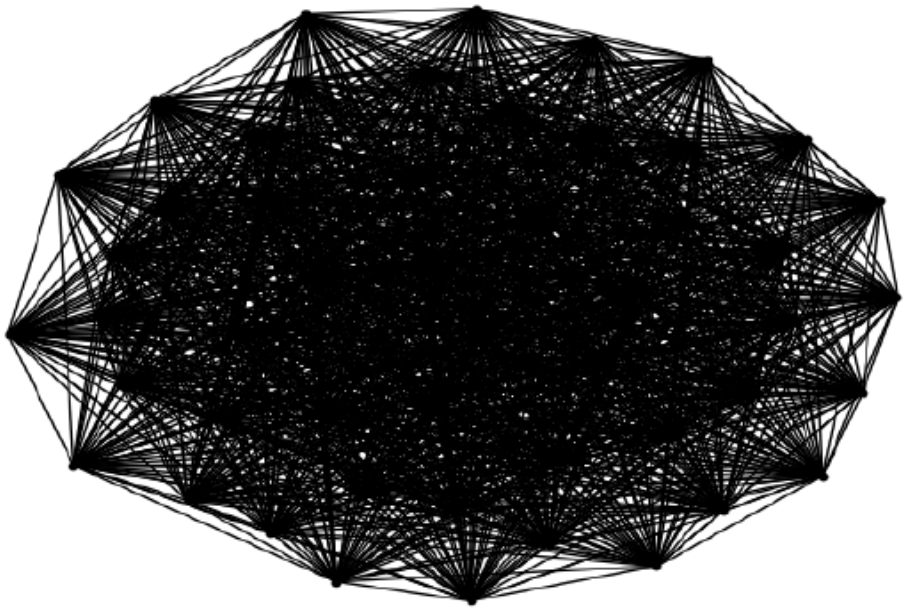
Slika 6: Razlika v številu povezav pri polnem grafu (modra krivulja) v primerjavi z upoštevanjem omejitve problema (rdeča krivulja).

Težji problem

$$A = \begin{bmatrix} 15 & 10 & 20 \\ -20 & 50 & 50 \end{bmatrix} \quad B = \begin{bmatrix} -5 & -3 & -2 \\ 10 & -20 & 20 \end{bmatrix}$$

- Število vozlišč:
  - Dron A: 45.756
  - Dron B: 6.624
- Število povezav:
  - Dron A: 1 miljarda
  - Dron B: 22 miljonov
- Potrebovali bomo 62 oziroma 32 vozlišč.

Težji problem



Slika 7: Polni graf z petdesetimi vozlišči.

Težji problem - d22

- Število vozlišč:
  - Dron A: 108
  - Dron B: 57
- Število povezav:
  - Dron A: 162
  - Dron B: 85

Predstavitev testov

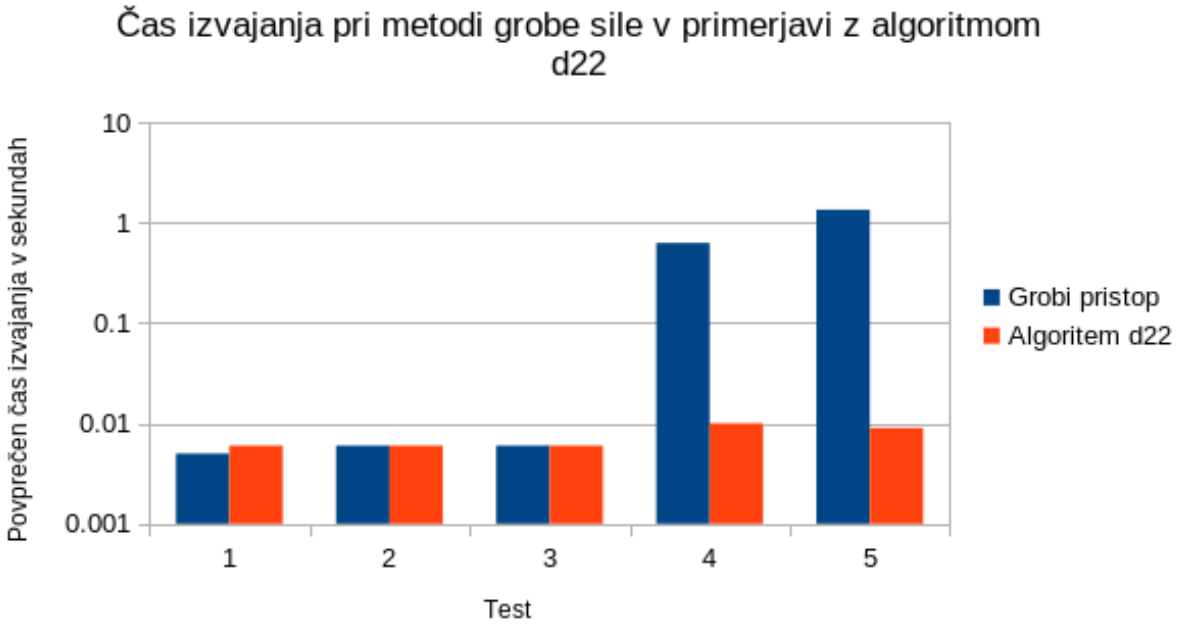
	Metoda grobe sile	Algoritem d22
Povprečje	1.345	0.009
Standardni odklon	0.019	0.004

Tabela 9: Čas izvajanja pri testu 4.5.

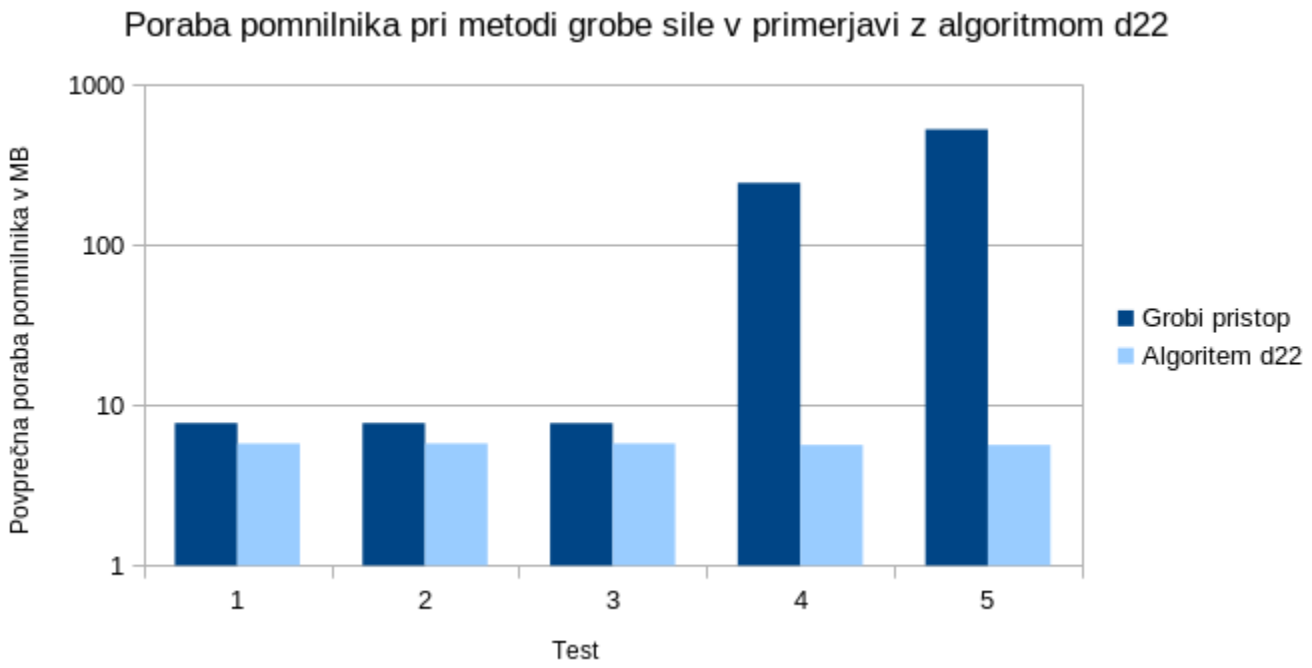
	Metoda grobe sile	Algoritem d22
Povprečje	522.729	5.637
Standardni odklon	1.324	0

Tabela 10: Poraba pomnilnika pri testu 4.5.

Interpretacija



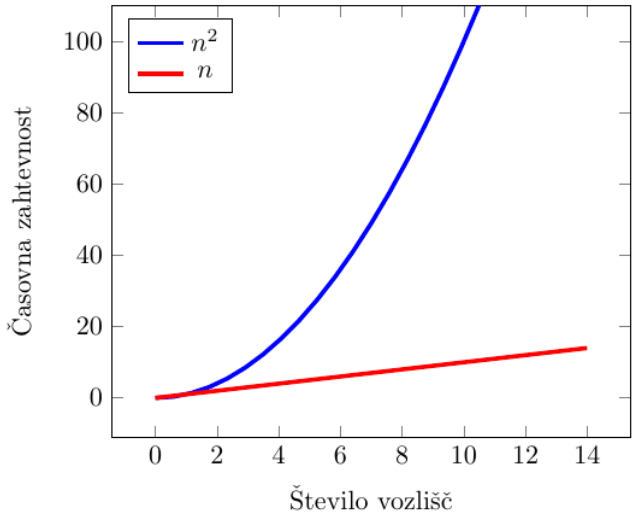
Interpretacija



Povzetek

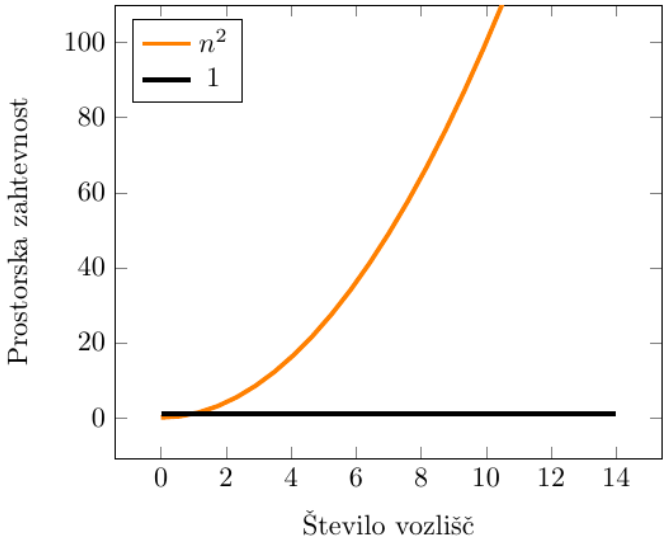
- Časovna zahtevnost:
  - Groba sila: $O(n^2)$
  - d22: $O(n)$
- Prostorska zahtevnost:
  - Groba sila: $S(n^2)$
  - d22: $S(1)$

Povzetek



Slika 10: Razlika v časovni zahtevnosti pri metodi grobega pristopa (modra krivulja) v primerjavi z algoritmom d22 (rdeča krivulja).

Povzetek



Slika 11: Razlika v prostorski zahtevnosti pri metodi grobega pristopa (oranžna krivulja) v primerjavi z algoritmom d22 (črna krivulja).