



Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

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

David Slatinek

Marcel Iskrač

Mateja Žvegler

Maribor, januar 2022

Povzetek

Ta dokument predstavi hibridni algoritem **d22** z uporabo požrešne metode in vračanjem kot rešitev logističnega problema skupnega potovanja. Najprej predstavimo problem, opišemo njegovo predstavitev v realnem svetu in podamo njegov matematični opis. Nato podamo rešitev z metodo grobe sile in z algoritmom d22. Na koncu opišemo različne testne primere in njihove rezultate, ki jih tudi interpretiramo.

1 Predstavitev problema

Opis scenarija:

Za delo v skladišču smo kupili nove drone. Želimo jih preizkusiti. 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. Ker so droni dragi, se želimo za vsako ceno izogniti trčenju. Doletela vas je odgovorna naloga in sicer napisati morate programa za oba drona. Podane so začetne in končne koordinate obeh dronov, vaš program pa naj izpiše pot obeh dronov, tako da se drona nikoli ne znajdeta na istih koordinatah ali izmenjata svojih lokacij (tj. trčita).

Omejitve:

Program naj prebere začetne koordinate drona 1 in drona 2 ter izpiše zaporedje pozicij obeh dronov, tako da veljajo naslednje lastnosti:

- Prvi poziciji dronov sta enaki njunim začetnim koordinatam.
- Zadnji poziciji dronov sta enaki njunim končnim koordinatam.
- 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.

- Drona nikoli ne smeta izmenjati lokacij (da bi se prvi dron ob času t nahajal na lokaciji A in drugi na lokaciji B , ob času $t + 1$ pa prvi na lokaciji B in drugi na lokaciji A).
- Seznam obiskanih koordinat ni daljši od 7000 vrstic.
- Absolutna vrednost koordinat dronov na nobeni točki ne preseže 10^6 .

Vhodni podatki:

V prvi vrstici se nahaja 6 celih števil. To so X_1, Y_1 in Z_1 , tj. koordinate začetne pozicije prvega drona, in X_2, Y_2 in Z_2 , tj. koordinate končne pozicije prvega drona.

V drugi vrstici sledi enak opis začetne in končne pozicije drugega drona. Zagotovljeno je, da bodo začetne koordinate prvega drona različne od začetnih koordinat drugega drona. Prav tako je zagotovljeno, da bodo končne koordinate prvega drona različne od končnih koordinat drugega drona.

Absolutna vrednost nobene koordinate ne bo presegla 1000.

Izhodni podatki:¹

Izpišite seznam obiskanih koordinat (glejte primere za obliko izpisa), tako da ustrezajo zahtevam naloge. V i -ti vrstici naj se nahajajo koordinate prvega in drugega drona ob času i .

Vhod	Izhod
0 0 0 2 2 2	(0 0 0) (1 1 2)
1 1 2 0 0 0	(1 0 0) (1 1 1)
	(1 1 0) (0 1 1)
	(1 1 1) (0 1 0)
	(1 1 2) (0 0 0)
	(1 2 2) (0 0 0)
	(2 2 2) (0 0 0)

Vhod	Izhod
-2 0 0 1 0 0	(-2 0 0) (3 0 0)
3 0 0 -1 0 0	(-1 0 0) (2 0 0)
	(0 0 0) (1 0 0)
	(1 0 0) (1 0 -1)
	(1 0 0) (0 0 -1)
	(1 0 0) (-1 0 -1)
	(1 0 0) (-1 0 0)

Vhod	Izhod
0 0 0 1 0 0	(0 0 0) (1 0 0)
1 0 0 0 0 0	(0 1 0) (0 0 0)
	(1 1 0) (0 0 0)
	(1 0 0) (0 0 0)

1.1 Prenos problema v realni svet

Obravnavani problem je eden večjih problemov logistike, na primer v avtomatiziranih sistemih za shranjevanje in iskanje, kot na primer skladišče trgovca Amazon². Avtomatizirani sistem za shranjevanje in iskanje je vrsta ali zvrst tehnologije za avtomatizacijo skladišča, ki je posebej zasnovana za shranjevanje, hranjenje in pridobivanje izdelkov in zalog na zahtevo [1] z uporabo robotov ali dronov. Avtomatizirani sistemi prinesejo veliko prednosti, med drugim [2, 3]:

- Nižji stroški.

¹Rešitev je več, podane so samo določene.

²Več o tem: <https://www.youtube.com/watch?v=IMPbKVb8y8s> (dostop: 29. 12. 2021)

- Lažje vzdrževanje.
- Večja kakovost storitev.
- Večja produktivnost in varnost.

Nalogo prav tako lahko prevedemo na problem iskanja poti, na primer platforma Google Maps ali za namen vojske. Google Maps je navigacijska storitev, ki omogoča prikaz poti med točko A in točko B . Pri tem seveda omogoča določanje parametrov, na primer tip potovanja, izogibanje avtocestam, določanje vmesnih točk ...

Vojska je naslednji uporabnik rešitve tega problema. Tudi njih zanima najhitrejša pot med točkama A in B , pri čemer je ta naloga veliko bolj podobna obravnavani tematiki - tudi tam se, na primer, v primeru obravnave leta letalov, želimo izogniti trčenju, pri čemer morata biti letali čim bližje zaradi varnosti in podpore v primeru težav.

1.2 Matematični opis problema

1.2.1 Teorija grafov

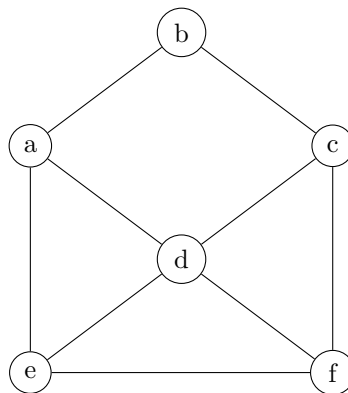
Definirajmo V kot končno neprazno množico in E kot poljubno družino³ dvoelementnih podmnožic množice V . Urejenemu paru $G=(V,E)$ pravimo graf na množici vozlišč (točk) V in z množico povezav E . [4]

Graf $G = (V(G), E(G))$ sestavljata neprazna množica vozlišča $V(G)$ in množica povezav $E(G)$. V množici vozlišč $V(G)$ lahko imamo poljubne elemente, medtem ko so elementi množice povezav delno določeni z množico vozlišč $V(G)$. Tako so v $E(G)$ lahko le neurejene podmnožice množice vozlišč $V(G)$ z dvema elementoma. Elementom množice $V(G)$ pravimo vozlišča grafa G , elementom $E(G)$ pa imenujemo povezave grafa G . [5]

Naj bo G graf. Če vozlišči $u, v \in V(G)$ tvorita povezavo grafa G , torej če je $uv \in E(G)$, potem rečemo, da sta u in v sosednji vozlišči oziroma sosed. [5]

Graf je usmerjen, če povezava $\{u, v\}$ določa smer, torej je $\{u, v\} \neq \{v, u\}$, v nasprotnem primeru je graf neusmerjen - v tem primeru je $\{u, v\} = \{v, u\}$. Graf je utežen, če povezave vsebujejo uteži, ki predstavljajo razdaljo med vozliščema. V nasprotnem primeru je graf neutežen.

Primer grafa je viden na sliki 1. Graf G je podan z množico vozlišč $V(G) = \{a, b, c, d, e, f\}$ in množico povezav $E(G) = \{\{a, b\}, \{a, d\}, \{a, e\}, \{b, c\}, \{c, d\}, \{c, f\}, \{d, e\}, \{d, f\}, \{e, f\}\}$. Vozlišči a in b sta sosednji, kar zapišemo kot $a \sim b$, medtem ko vozlišči a in f nista sosedni, kar zapišemo kot $a \not\sim f$.



Slika 1: Neutežen, neusmerjen graf G .

³Dvoelementne podmnožice množice V se lahko ponovijo.

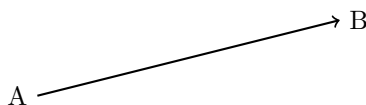
1.2.2 Vektorji

V znanosti in tehniki se srečujemo s številnimi količinami, za katere sta potrebna velikost in smer, npr. hitrost, pospešek ... Za pravilno nadaljnjo obravnavo potrebujemo zakone za njihovo predstavitev, združevanje in upravljanje. Namesto da bi ustvarili formule za vsako količino posebej, je bolj smiselno ustvariti matematični model, ki določa skupne zakone za vse veličine, ki zahtevajo tako velikost in smer. [6]

Vektor lahko definiramo na dva načina: 1.1 in 1.2, primer vektorja v 2d prostoru je viden na sliki 2.

Definicija 1.1. Vektor je fizikalna oziroma matematična količina, ki je določen oziroma z dvema neodvisnima komponentama: velikostjo in smerjo. [6]

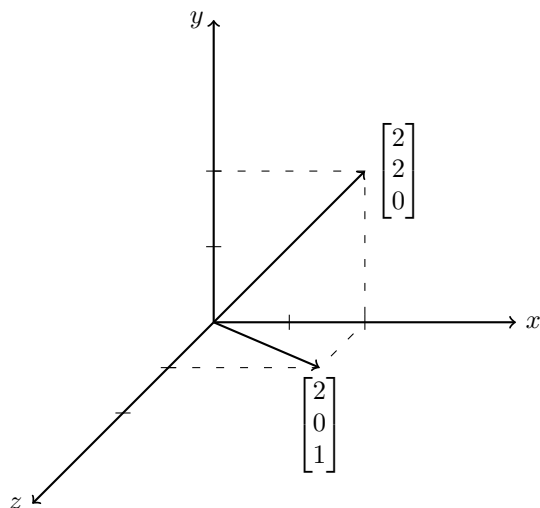
Definicija 1.2. Vektor je urejen par točk v prostoru. Vektor, ki ga določa urejen par (A, B) označimo z \overrightarrow{AB} . [7]



Slika 2: Vektor \overrightarrow{AB} .

Krajevni vektor točke T je usmerjena daljica od koordinatnega izhodišča (oznaka O) do točke T . Označimo ga z $\vec{r}_T = \overrightarrow{OT}$ [7]. Vektorji lahko nastopajo v poljubnem N razsežnostnem prostoru, ogledali si bomo tridimenzionalni prostor.

Tridimenzionalni pravokotni kartezični sistem (oznaka \mathbb{R}^3) sestavljajo tri pravokotne premice - x, y, z . Osi imenujemo abscisna os (os x), ordinatna os (os y) in aplikatna os (os z). Primer 3d prostora prikazuje slika 3.



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

Če definiramo točko $A(x_1, y_1, z_1)$ in točko $B(x_2, y_2, z_2)$, lahko razdaljo med njima, to je dolžina vektorja, izračunamo po enačbi 1,

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} = \|\vec{r}_b - \vec{r}_a\| \quad (1)$$

kjer \vec{r}_a in \vec{r}_b predstavljata krajevni vektor točke A oziroma točke B .

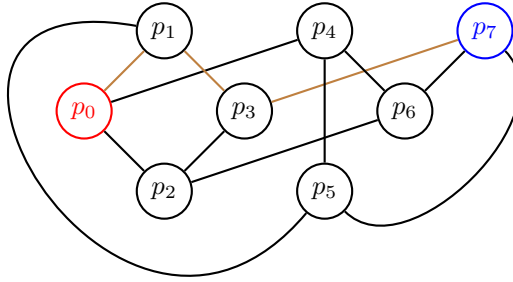
1.2.3 Uporaba teorije grafov in vektorjev

Teorijo grafov in vektorje lahko uporabimo za sestavo rešitve podanega problema. Položaj drona v 3d prostoru lahko predstavimo kot vozlišča, povezave med vozlišči pa kot vektorje.

Predpostavimo, da imamo vozlišče $p_A(0,0,0)$, ki predstavlja začetno lokacijo drona in vozlišče $p_B(1,1,1)$, ki predstavlja cilj oziroma končno lokacijo drona. Prvi korak je generiranje vseh vozlišč - postopek je predstavljen v 1. Nato je treba med vozlišči ustvariti ustrezne povezave. Glede na omejitve problema⁴, lahko uporabimo formulo 1, ki jo uporabimo za vse kombinacije vozlišč. Formulo 2 uporabimo za odločitev, ali bomo povezavo sprejeli.

Primer takšnega grafa je viden na sliki 4.

$$d(A, B) = \begin{cases} \text{sprejmi povezavo} & d(A, B) = 1 \\ \text{zavrni povezavo} & d(A, B) \neq 1 \end{cases} \quad (2)$$



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.

2 Rešitev z grobim pristopom

Prvi korak rešitve problema z metodo grobe sile oziroma z grobim pristopom je generiranje vseh vozlišč od začetne do končne točke. Ta števila najenostavneje generiramo za vsako razsežnost N - algoritem 1 -, tj. za X , Y , Z , ki jih nato združimo - algoritem 2. Število generiranih števil določimo po enačbi 3, število vseh vozlišč pa po enačbi 4. Postopek za generiranje koordinat ponovimo trikrat, celoten postopek pa dvakrat⁵.

$$\text{len}(A) = |\text{start} - \text{end}| + 1 \quad (3)$$

kjer:

A = struktura za shranjevanje števil,

start = začetna koordinata,

end = končna koordinata.

$$\text{len}(V) = \text{len}(X) * \text{len}(Y) * \text{len}(Z) \quad (4)$$

kjer:

V = struktura za shranjevanje vozlišč,

X = število koordinat X parametra,

Y = število koordinat Y parametra,

Z = število koordinat Z parametra.

⁴Zaporedni poziciji drona se lahko razlikujeta v največ eni koordinati in to za največ 1, glej Predstavitev problema.

⁵Imamo dva drona v 3d prostoru.

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

```
1: function GENERATECOORDINATES( $start, end$ )
2:    $A \leftarrow \emptyset$ 
3:   while  $start \neq end$  do
4:     ADD( $A, start$ ) ▷ Dodamo  $start$  v strukturo  $A$ 
5:     if  $start < end$  then
6:        $start \leftarrow start + 1$ 
7:     else
8:        $start \leftarrow start - 1$ 
9:     end if
10:  end while
11:  ADD( $A, start$ ) ▷ Dodamo še zadnjo koordinato
12:  return  $A$ 
13: end function
```

Algorithm 2 Generiraj vozlišča.

```
1: function GENERATEVERTICES( $X, Y, Z$ )
2:    $A \leftarrow \emptyset$  ▷ Shranjevanje vozlišč
3:   for  $x \leftarrow 0$  to  $len(X)$  do ▷ Za vsako  $x$  vrednost
4:     for  $y \leftarrow 0$  to  $len(Y)$  do ▷ Za vsako  $y$  vrednost
5:       for  $z \leftarrow 0$  to  $len(Z)$  do ▷ Za vsako  $z$  vrednost
6:          $A \leftarrow A \cup \text{CREATEVERTEX}(X[x], Y[y], Z[z])$  ▷ Ustvarimo vozlišče
7:       end for
8:     end for
9:   end for
10:  return  $A$ 
11: end function
```

Naslednji korak je generiranje povezav, postopek je predstavljen v 3. Na koncu izvedemo algoritem grobe sile, ki je prikazan v 4.

Algorithm 3 Generiraj vse povezave med vozlišči.

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

Algorithm 4 Algoritem z grobim pristopom.

```
1: function COLLISION( $a, b$ )
2:   return  $a.x == b.x$  &  $a.y == b.y$  &  $a.z == b.z$ 
3: end function
4: function VALID( $a, b$ )
5:   return  $edge(a, b) == 1$ 
6: end function
7: function BRUTEFORCE( $A, B$ )
8:    $M \leftarrow A[0], N \leftarrow B[0]$  ▷ Sprejeta vozlišča drona  $A, B$ , sprejmemo začetno vozlišče
9:   while  $last(M) \neq last(A)$  &  $last(N) \neq last(B)$  do ▷ Začetni argumenti, zmanjšamo gnezdenje
10:     $a \leftarrow MIROVANJE$  ▷ Začetni argumenti, zmanjšamo gnezdenje
11:     $b \leftarrow MIROVANJE$ 
12:    if  $isNext(A)$  then
13:       $a \leftarrow next(A)$  ▷ Dron še ni na cilju
14:    end if
15:    if  $isNext(B)$  then
16:       $b \leftarrow next(B)$ 
17:    end if
18:    if  $a == MIROVANJE$  &  $b \neq MIROVANJE$  then
19:      if  $VALID(b, last(N))$  & not  $COLLISION(a, b)$  then ▷ Preverimo, če je povezava ustrezna
20:         $N \leftarrow N \cup b$  ▷ Sprejmemo vozlišče  $b$ 
21:      else
22:         $BACKTRACK(N)$  ▷ Vrnemo se na prejšnje vozlišče
23:      end if
24:    else if  $a \neq MIROVANJE$  &  $b == MIROVANJE$  then
25:      if  $VALID(a, last(M))$  & not  $COLLISION(a, b)$  then
26:         $M \leftarrow M \cup a$ 
27:      else
28:         $BACKTRACK(M)$ 
29:      end if
30:    else
31:       $a \leftarrow next(A), b \leftarrow next(B)$ 
32:      if  $COLLISION(a, b)$  then ▷ Preverimo za trk dronov
33:         $a \leftarrow MIROVANJE$  ▷ Dron se to iteracijo ne bo premikal
34:        if  $VALID(b, last(N))$  then
35:           $N \leftarrow N \cup b$ 
36:        else
37:           $BACKTRACK(N)$ 
38:        end if
39:      else
40:        if  $VALID(a, last(M))$  then
41:           $M \leftarrow M \cup a$ 
42:        else
43:           $BACKTRACK(M)$ 
44:        end if
45:        if  $VALID(b, last(N))$  then
46:           $N \leftarrow N \cup b$ 
47:        else
48:           $BACKTRACK(N)$ 
49:        end if
50:      end if
51:    end if
52:  end while
53:  return  $M, N$ 
54: end function
```

3 Rešitev z algoritmom d22

Delovanje algoritma d22 lahko predstavimo v dveh korakih:

1. Dinamično ustvarjaj vozlišča.
2. Dinamično ustvarjaj povezave.
3. Izvedi algoritem d22.

Algoritem d22 deluje na osnovi požrešne metode [8] in na podlagi vračanja [9]. Namesto da ustvarimo vsa vozlišča, jih sedaj ustvarjamo po potrebi - algoritem 5 in 6.

Drugi korak je prav tako podoben algoritmu grobe sile, vendar pri algoritmu d22 upoštevamo omejitve, da se zaporedna položaja drona lahko razlikujeta v največ eni koordinati in to za največ 1 - glej 1.2.3. Na podlagi tega zelo zmanjšamo velikost problema, saj moramo ustvariti manj povezav. V primeru generiranja vseh povezav pridobimo $\frac{n(n+1)}{2}$ povezav (časovna zahtevnost $O(n^2)$) oziroma polni graf - primer takšnega grafa prikazuje slika 5, v primeru upoštevanja omejitve pa $\lfloor \frac{3n}{2} \rfloor$ povezav (časovna zahtevnost $O(n)$), kjer n predstavlja število vozlišč. Razliko med številom povezav obeh algoritmov prikazuje slika 6.

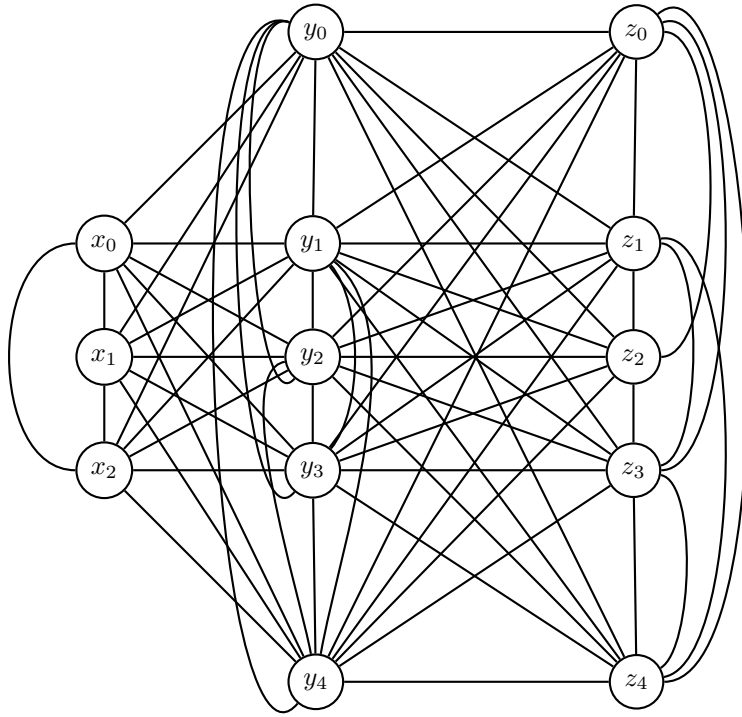
Na koncu izvedemo algoritem d22, ki je predstavljen v 7.

Algorithm 5 Generiraj vrednosti vozlišča po potrebi.

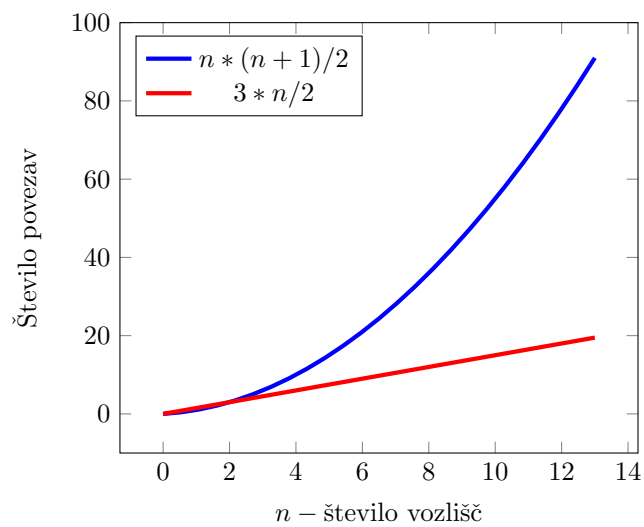
```
1: function GETVALUE(start, end)
2:   value  $\leftarrow$  start
3:   if start < end then
4:     start  $\leftarrow$  start + 1
5:   else
6:     start  $\leftarrow$  start - 1
7:   end if
8:   yield value
9: end function
```

Algorithm 6 Generiraj vozlišča po potrebi.

```
1: function GETVERTEX(start, end)
2:   for x in GETVALUE(start.x, end.x) do                                 $\triangleright$  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)                                 $\triangleright$  Ustvarimo vozlišče
6:       end for
7:     end for
8:   end for
9: end function
```



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



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

Algorithm 7 Algoritem d22.

```
1: function FEASIBLE( $a, b$ )
2:   return  $a.x \neq b.x \ \& \ a.y \neq b.y \ \& \ a.z \neq b.z$ 
3: end function
4: function VALID( $a, b$ )
5:   return  $\text{sqrt}((b.x - a.x)^2 + (b.y - a.y)^2 + (b.z - a.z)^2) == 1$ 
6: end function
7: function D22( $A1, A2, B1, B2$ )  $\triangleright A1 =$  začetne koordinate drona A,  $A2 =$  končne koordinate drona A
8:    $M \leftarrow [(\text{GETVERTEX}(A1, A2))], N \leftarrow [(\text{GETVERTEX}(B1, B2))]$   $\triangleright$  Premiki drona A, B
9:   while  $\text{last}(M) \neq A2 \ \& \ \text{last}(N) \neq B2$  do
10:     $a \leftarrow \text{MIROVANJE}$   $\triangleright$  Koordinate drona A, dron se to iteracijo ne bo premikal
11:     $b \leftarrow \text{MIROVANJE}$   $\triangleright$  Koordinate drona B
12:    if  $\text{last}(M) \neq A2$  then  $\triangleright$  Pogledamo, če obstaja naslednja koordinata
13:       $a \leftarrow \text{GETVERTEX}(A1, A2)$ 
14:    end if
15:    if  $\text{last}(N) \neq B2$  then
16:       $b \leftarrow \text{GETVERTEX}(B1, B2)$ 
17:    end if
18:    if  $a == \text{MIROVANJE} \ \& \ b \neq \text{MIROVANJE}$  then
19:      if  $\text{VALID}(b, \text{last}(N)) \ \& \ \text{FEASIBLE}(a, b)$  then  $\triangleright$  Preverimo, če je povezava ustrezna
20:         $N \leftarrow N \cup b$   $\triangleright$  Sprejmemo vozlišče  $b$ 
21:      else
22:         $\text{BACKTRACK}(N)$   $\triangleright$  Vrnemo se na prejšnje vozlišče
23:      end if
24:    else if  $a \neq \text{MIROVANJE} \ \& \ b == \text{MIROVANJE}$  then
25:      if  $\text{VALID}(a, \text{last}(M)) \ \& \ \text{FEASIBLE}(a, b)$  then
26:         $M \leftarrow M \cup a$ 
27:      else
28:         $\text{BACKTRACK}(M)$ 
29:      end if
30:    else
31:       $a \leftarrow \text{next}(A), b \leftarrow \text{next}(B)$ 
32:      if  $\text{FEASIBLE}(a, b)$  then  $\triangleright$  Preverimo za trk dronov
33:         $a \leftarrow \text{MIROVANJE}$ 
34:        if  $\text{VALID}(b, \text{last}(N))$  then
35:           $N \leftarrow N \cup b$ 
36:        else
37:           $\text{BACKTRACK}(N)$ 
38:        end if
39:      else
40:        if  $\text{VALID}(a, \text{last}(M))$  then
41:           $M \leftarrow M \cup a$ 
42:        else
43:           $\text{BACKTRACK}(M)$ 
44:        end if
45:        if  $\text{VALID}(b, \text{last}(N))$  then
46:           $N \leftarrow N \cup b$ 
47:        else
48:           $\text{BACKTRACK}(N)$ 
49:        end if
50:      end if
51:    end if
52:  end while
53:  return  $M, N$ 
54: end function
```

3.1 Primer težjega problema in njegova rešitev

3.1.1 Metoda grobega pristopa

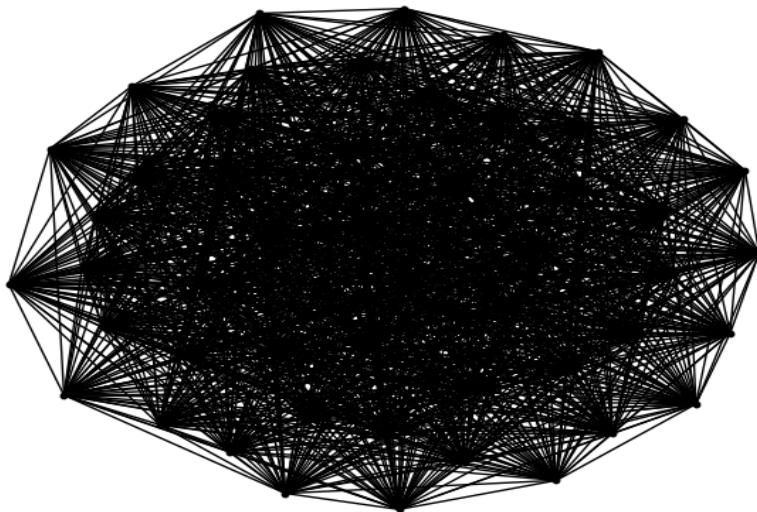
Definirajmo začetne koordinate drona A kot prvo vrstico matrike A , njegove končne koordinate pa kot drugo vrstico matrike v 5. Enako storimo za dron B.

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

Z uporabo formule 3 pridobimo 36 x koordinat, 41 y koordinat in 31 z koordinat za dron A. Za dron B pridobimo 16 x koordinat, 18 y koordinat in 23 z koordinat. Z uporabo formule 4 pridobimo 45.756 vozlišč za dron A in 6.624 vozlišč za dron B, skupaj pa 52.380 vozlišč.

Če uporabimo algoritem 3 in upoštevamo 3, pridobimo $1.046.828.646 \approx 1$ milijardo povezav za dron A in $21.942.000 \approx 22$ milijonov povezav za dron B.

Na podlagi formule 1 izračunamo, da bomo za dron A opravili 62 premikov, za dron B pa 32 premikov. Metoda grobega pristopa tako izračuna občutno preveč vozlišč in povezav. Primer polnega grafa z petdesetimi vozlišči je viden na sliki 7.



Slika 7: Polni graf s petdesetimi vozlišči.

3.1.2 Algoritem d22

Koordinate dronov ostanejo enake kot v 3.1.1. Z dinamičnim ustvarjanjem vozlišč sedaj pridobimo 108 vozlišč ($36 + 41 + 31$) za dron A in 57 vozlišč ($16 + 18 + 23$) za dron B. Z dinamičnim ustvarjanjem povezav - algoritem 6 in upoštevanjem 1.2.3 in 3, sedaj pridobimo 162 povezav za dron A in 85 povezav za dron B.

Z algoritmom d22 tako ustvarimo 0.24% vozlišč in $1.54E^{-5}\%$ povezav za dron A v primerjavo z grobim pristopom. Za dron B ustvarimo 0.86% vozlišč in 0.004% povezav v primerjavi z grobim pristopom.

4 Predstavitev testov, njihovih rezultatov in njihova interpretacija

Testi so bili izvedeni na operacijskem sistemu Manjaro različica 21.2.0, g++ različica je bila 11.1.0. Zanimala nas je časovna razlika med algoritmoma in razlika v porabi pomnilnika. Vsak test je bil izveden desetkrat. Pri času izvajanja so rezultati navedeni v sekundah, pri porabi pomnilnika so rezultati navedeni v megabajtih.

Računalniška strojna oprema:

CPU: Intel Core i7-9750H, 2.60GHz \times 6

RAM: 16GB

4.1 Test št. 1

Testni podatki:

0 0 0 0 0 0

0 1 0 0 -1 0

	Metoda grobe sile	Algoritem d22
Povprečje	0.005	0.006
Standardni odklon	0.001	0

Tabela 1: Čas izvajanja pri testu 4.1.

	Metoda grobe sile	Algoritem d22
Povprečje	7.711	5.754
Standardni odklon	0.023	0

Tabela 2: Poraba pomnilnika pri testu 4.1.

4.2 Test št. 2

Testni podatki:

1 0 0 0 0 50

0 0 0 1 0 50

	Metoda grobe sile	Algoritem d22
Povprečje	0.006	0.006
Standardni odklon	0.002	0.002

Tabela 3: Čas izvajanja pri testu 4.2.

	Metoda grobe sile	Algoritem d22
Povprečje	7.711	5.758
Standardni odklon	0.023	0

Tabela 4: Poraba pomnilnika pri testu 4.2.

4.3 Test št. 3

Testni podatki:

952 0 0 952 0 0

953 0 0 951 0 0

	Metoda grobe sile	Algoritem d22
Povprečje	0.006	0.006
Standardni odklon	0.001	0.001

Tabela 5: Čas izvajanja pri testu 4.3.

	Metoda grobe sile	Algoritem d22
Povprečje	7.703	5.758
Standardni odklon	0	0

Tabela 6: Poraba pomnilnika pri testu 4.3.

4.4 Test št. 4

Testni podatki:
-10 -10 -10 20 20 20
20 20 20 40 40 40

	Metoda grobe sile	Algoritem d22
Povprečje	0.627	0.01
Standardni odklon	0.019	0.002

Tabela 7: Čas izvajanja pri testu 4.4.

	Metoda grobe sile	Algoritem d22
Povprečje	242.544	5.637
Standardni odklon	0.5	0

Tabela 8: Poraba pomnilnika pri testu 4.4.

4.5 Test št. 5

Testni podatki:
15 10 20 -20 50 50
-5 -3 -2 10 -20 20

	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.

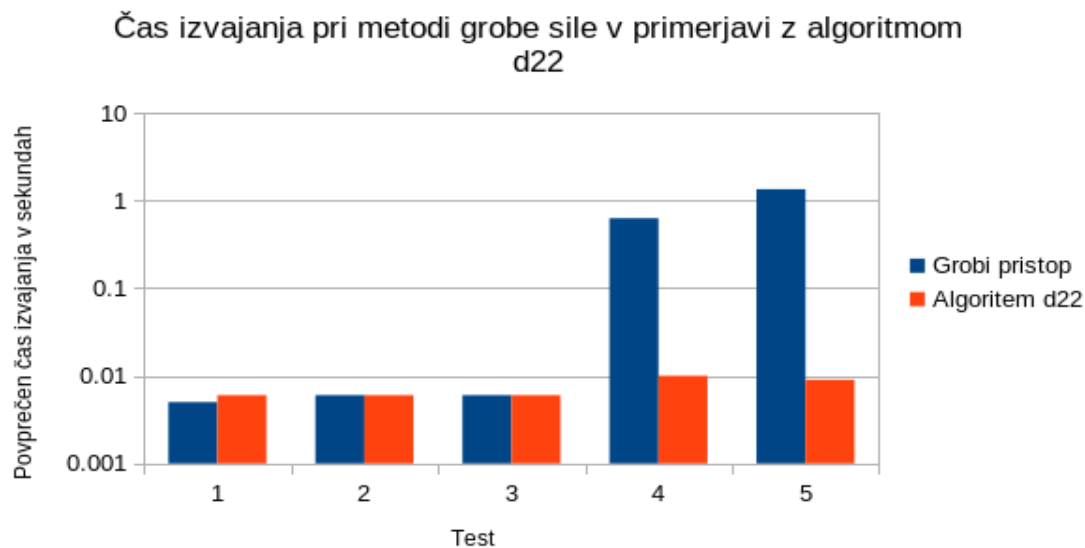
4.6 Interpretacija in vizualizacija rezultatov testov

Pri vseh testih so rezultati bili zelo podobni, na podlagi tega je standardni odklon v vseh testih blizu nič. Pri prvih treh testih sta si algoritma zelo podobna - čas, da se algoritem izvede, v povprečju traja 0.006 sekund, povprečna poraba pomnilnika je okoli 6 MB.

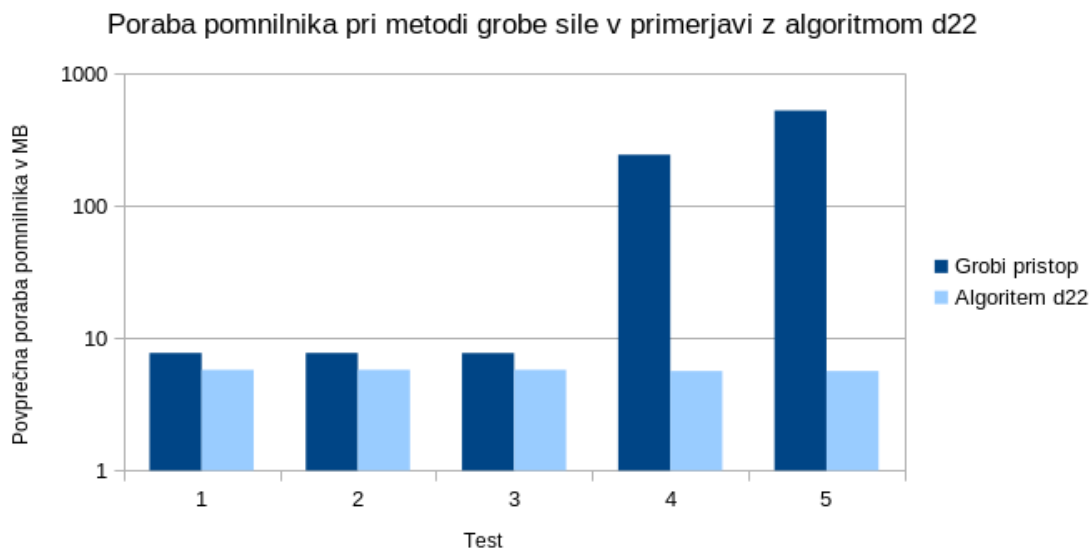
Razlika postane opaznejša pri zadnjih dveh testih. Algoritem grobe sile se je izvedel v 0.0627 sekunde, algoritem d22 pa v 0.01, kar je 627 % hitreje. Algoritem grobe sile je porabil 242 MB pomnilnika, medtem ko je algoritem d22 porabil 6 MB pomnilnika, kar je za 4.033 % manj.

Razliko med algoritmoma pa najbolje prikaže test 5. Algoritem grobe sile se je izvedel v 1.345 sekundah, algoritem d22 pa v 0.009, kar je 14.944 % hitreje. Algoritem grobe sile je porabil 522 MB pomnilnika, medtem ko je algoritem d22 porabil 6 MB pomnilnika, kar je za 8.700 % manj.

Prikaz rezultatov vseh testov prikazujeta sliki 8 in 9 - logaritemska skala za os y .



Slika 8: Čas izvajanja pri metodi grobe sile v primerjavi z algoritmom d22.



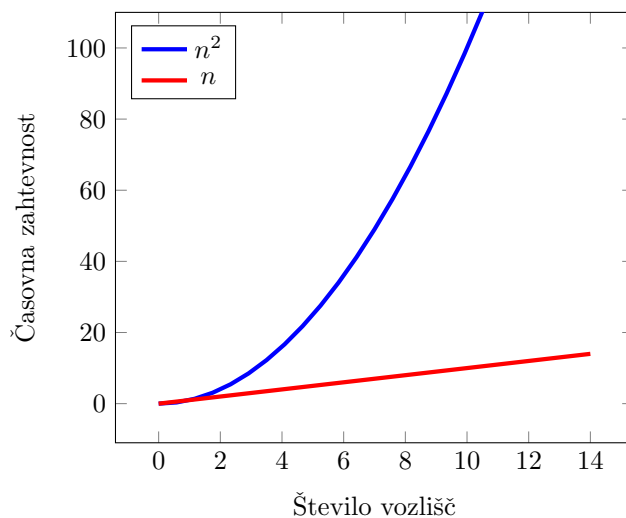
Slika 9: Poraba pomnilnika pri metodi grobe sile v primerjavi z algoritmom d22.

5 Zaključek

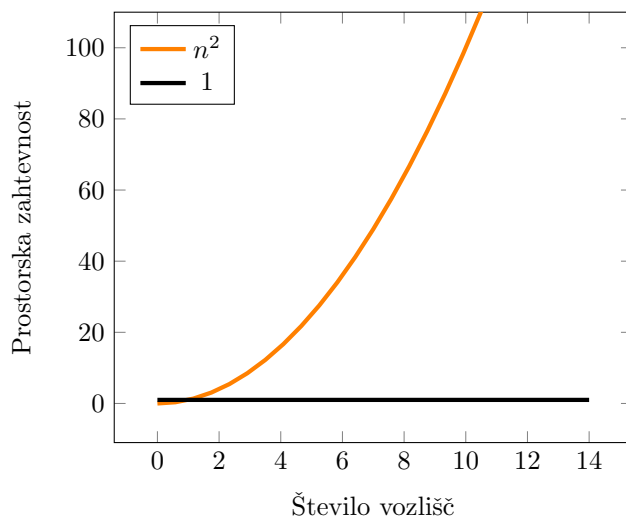
Podana problematika je zelo aktualna in njena rešitev bo vedno pomembnejša zaradi želje trgovcev po zmanjševanju stroškov. Prav tako te rešitve uporabljajo velika podjetja na področju logistike in državne ustanove, v prvi vrsti je to vojaška industrija.

Z algoritmom d22 podamo rešitev tega problema, pri čemer temeljito izboljšamo metodo grobega pristopa glede uporabe pomnilnika in časa izvajanja.

Na podlagi analize lahko zaključimo, da ima algoritem grobega pristopa časovno zahtevnost $O(n^2)$ in prostorsko zahtevnost $S(n^2)$ (kvadratna zahtevnost), algoritem d22 pa $O(n)$ in $S(1)$ (linearna in konstantna zahtevnost) - sliki 10 in 11.



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



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

Literatura

- [1] Ed Romaine. *Automated Storage and Retrieval System (AS/RS) Types and Uses*. 18. avg. 2020. URL: <https://www.conveyco.com/automated-storage-and-retrieval-types/> (pridobljeno 20. 11. 2021).
- [2] Ales Vysocky in Petr Novak. "Human - Robot collaboration in industry". V: *MM Science Journal* 2016 (jun. 2016), str. 903–906. DOI: 10.17973/MMSJ.2016_06_201611.
- [3] Mark Stevens. *Pros and cons of using industrial robots in your manufacturing operation*. 21. sep. 2021. URL: <https://www.wipfli.com/insights/articles/mad-pros-and-cons-of-using-industrial-robots-in-manufacturing> (pridobljeno 20. 11. 2021).
- [4] izr. prof. dr. Aleksandra Tepeh. *Teorija grafov*. URL: <https://studij.um.si/course/view.php?id=11377> (pridobljeno 28. 11. 2021).
- [5] Iztok Peterin. "Diskretne strukture". V: (2020). DOI: doi.org/10.18690/978-961-286-400-2. URL: <https://dk.um.si/IzpisGradiva.php?lang=slv&id=78112>.
- [6] Pramod S. Joag. *An Introduction to Vectors, Vector Operators and Vector Analysis*. Cambridge University Press, 2016. DOI: 10.1017/9781316650578.
- [7] Aleksandra Tepeh in Riste Škrekovski. *Diskretna matematika*. 2018. ISBN: 978-961-286-152-0. DOI: 10.18690/978-961-286-152-0. URL: <https://dk.um.si/IzpisGradiva.php?lang=slv&id=70165>.
- [8] Tim Roughgarden. *Algorithms Illuminated (Part 3): Greedy Algorithms and Dynamic Programming*. 1. izd. Soundlikeyourself Publishing, LLC, 2019. ISBN: 0999282948, 978-0999282946.
- [9] Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford Stein. *Introduction to Algorithms 3rd Edition*. 3. izd. 2009. ISBN: 0262033844, 9780262033848.