TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
ARVUTITEADUSE INSTITUUT
TARKVARASÜSTEEMIDE ÕPPETOOL
INFORMAATIKA ERIALA

Dan Bogdanov

Kuupide kaheksandpuul põhinevad ruumi mudelid ja nende kasutamine optimaalse tee otsingul

Bakalaureusetöö (10 AP)

Juhendaja: dots. A. Isotamm

Autor: …… "….." mai 2005
Juhendaja: ……. "….." mai 2005
Õppetooli juhataja: …… "….." …….. 2005

TARTU 2005

Sisukord

Sissejuhatus

Optimaalse teekonna otsingu ülesande püstitus sõltub läbitavast ruumist ja selles liikuvast agendist. Võimalik on kasutada erinevaid lahendusmeetodeid, mis arvestavad ruumi ja agendi eripäradega ning hoiavad selle informatsiooni kasutamisel kokku nii töötlemisaega kui ressursse.

Teekonnaotsingu alamülesanneteks jaotamise teel on võimalik lahendusmeetodite osi üldistada ning taaskasutada. Käesolev bakalaureusetöö jätkab semestritöös [1] alustatud uurimissuunda, mille eesmärgiks on üldistatud teeotsinguülesande püstitamine ja geomeetriliste objektide puudel põhinevate lahendusmeetodite kirjeldamine.

Töös [1] jaotatakse teeotsinguülesande lahendamise neljaks põhiliseks sammuks: ruumi mudeli loomine lähteandmete analüüsi teel, otsingugraafi tippude määramine ruumi mudeli põhjal, otsingugraafi koostamine (servade lisamine) ning graafiotsing. Esitatud on kolmnurkade kahendpuul ja kuupide neljandpuul põhinevad meetodid optimaalse tee leidmiseks maastikul, mida vaadeldakse deformeeritud tasandina.

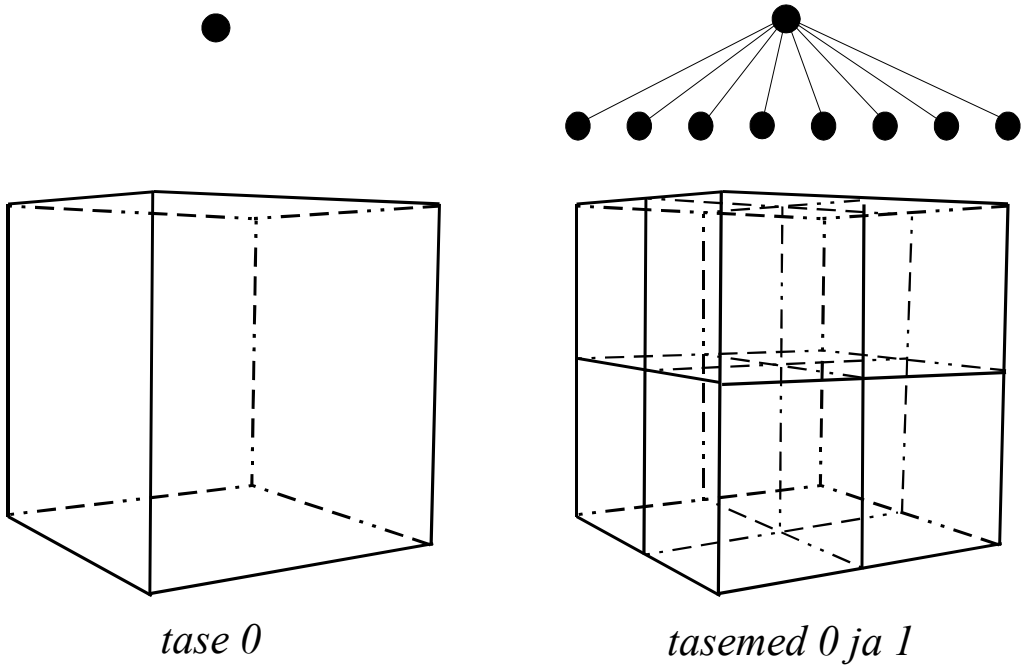
Käesolevas töös esitatakse meetodid otsinguruumi laiendamiseks kolmemõõtmelisse ruumi. Esimeses peatükis defineeritakse lähteandmete vorming ning esitatakse kuupide kaheksandpuul põhinev analüüsi meetod ruumi mudeli loomiseks. Teine peatükk kirjeldab kahte erinevat võimalust otsingugraafi koostamiseks ning loetleb omadusi, mille abil erinevaid graafi koostamise meetodeid võrrelda.

Kolmandas peatükis defineeritakse läbitavus-, kaalu- ja pöörangufunktsiooni abil mobiilse agendi profiil. Need kolm funktsiooni kirjeldavad agendi võimet ruumi läbida ning määravad selle optimaalsuse. Esitatakse reeglid otsingugraafi lihtsustamiseks konkreetse agendi profiili põhjal.

1. Kuupide kaheksandpuu kasutamine ruumi mudeli modelleerimisel

1.1 Kuupide kaheksandpuu andmestruktuur

Kuupide kaheksandpuu (ing k *octree* või *oct-tree*) on puu, mille tippudeks on kuubid. Igal tipul on kaheksa alamtippu, mis tekivad kuubi jaotamisel kaheksaks võrdse suurusega kuubiks. Tipu sügavuseks nimetame tema kaugust juurtipust. Tipud sügavusega *l* moodustavad kaheksandpuu *l* taseme. Joonisel 1.1 on näidatud kaheksandpuu kaks esimest taset.



Joonis 1.1: Täieliku kuupide kaheksandpuu kaks esimest taset

Kaheksandpuid kasutatakse muuhulgas punktidevahelise nähtavuse kontrollimiseks ruumis [2] ja kahemõõtmeliste kujutiste põhjal ruumiliste mudelite loomisel [3]. Need ja ka teised tööd ([4] ja [5]) tunnustavad kaheksandpuudel põhineva geomeetrilise ruumi esitamise meetodi autoritena Jackinsit ja Tanimotot [6].

Täielikuks kuupide kaheksandpuu tipuks nimetame sellist tippu, millel on täpselt kaheksa alamtippu. Mittetäielikuks kuupide kaheksandpuu tipuks nimetame sellist tippu, millel on vähem kui kaheksa alamtippu. Kuupide kaheksandpuu on täielik, kui

selle juurtipp ja kõik mitterippuvad alamtipud on täielikud.

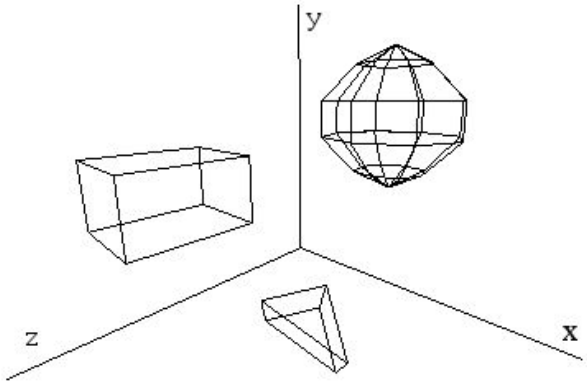
Kaks kuupide kaheksandpuu tippu on külgnevad, kui ükski neile vastavatest kuupidest ei ole teise sees ning ühe kuubi ühel tahul on ühisosa teise kuubi ühe tahuga. Külgnevad tipud ei pea olema samal tasemel. On ilmne, et kahel sama taseme külgneval tipul on külgnev tahk kogu ulatuses ühine.

Kaks kuupide kaheksandpuu tippu on tipnevad, kui ükski nendele vastavatest kuupidest ei ole teise sees ning üks kuubi serv puutub kokku teise kuubi servaga (nende servadel on ühine punkt).

Puus võib leiduda tippe, millel puuduvad nii külgnevad kui tipnevad tipud. Sellel on kaks võimalikku põhjust – kas puu ei ole vastava külgneva või tipneva tipuni välja harunenud või jääks see vaadeldava kuupide kaheksandpuu piiridest üldse välja.

1.2 Objektide ruumi kirjeldus

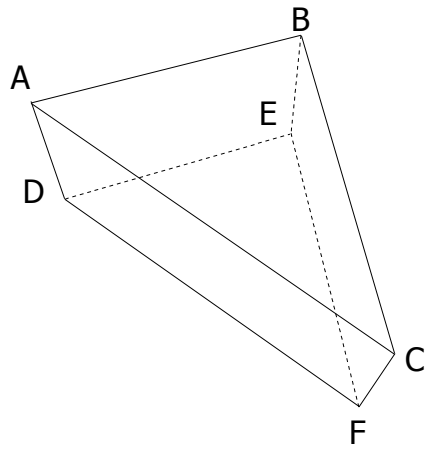
Käesolevas töös kasutame järgnevat kolmemõõtmelise ruumi lihtsustatud mudelit. Vaatleme kolmemõõtmelist ruumi ristkoordinaatsüsteemiga $\{O;\vec{x};\vec{y};\vec{z}\}$, kus $O(0, 0, 0)$. Objekt selles ruumis on kumer suletud geomeetriline keha, mis koosneb kumeratest hulknurkadest. Keerulisemad struktuurid moodustatakse objektide kõrvuti asetamise teel. Joonisel 1.2 on näited defineeritud objektidest.



Joonis 1.2: Kolm näidisobjekti

Hulknurgad esitatakse nende tippude koordinaatide jadana, kusjuures tipud järjestatakse keha väljaspoolt vaadates päripäeva liikudes. Hulknurga, mille tipud on A, C, F ja D, esitus on kujul:

$$P_{ACFD} = <(A_x, A_y, A_z), (C_x, C_y, C_z), (F_x, F_y, F_z), (D_x, D_y, D_z)>.$$



Joonis 1.3: Näidisobjekt tähistustega

Kehad esitatakse hulknurkade kirjelduste hulgana. Järgneb joonisel 1.3 kujutatud keha esitus hulknurkade hulgana.

$$M_{ABCDEF} = \{<(A_x, A_y, A_z), (B_x, B_y, B_z), (C_x, C_y, C_z)>,$$

$$<(A_x, A_y, A_z), (C_x, C_y, C_z), (F_x, F_y, F_z), (D_x, D_y, D_z)>,$$

$$<(A_x, A_y, A_z), (D_x, D_y, D_z), (E_x, E_y, E_z), (B_x, B_y, B_z)>,$$

$$<(B_x, B_y, B_z), (E_x, E_y, E_z), (F_x, F_y, F_z), (C_x, C_y, C_z)>,$$

$$<(E_x, E_y, E_z), (D_x, D_y, D_z), (F_x, F_y, F_z)>\}.$$

Objektide ruumi W kolmemõõtmelises ruumis defineerime sinna kuuluvate kehade hulgana.

$$W = \{ M \mid M \text{ on objekt ruumis } W \}.$$

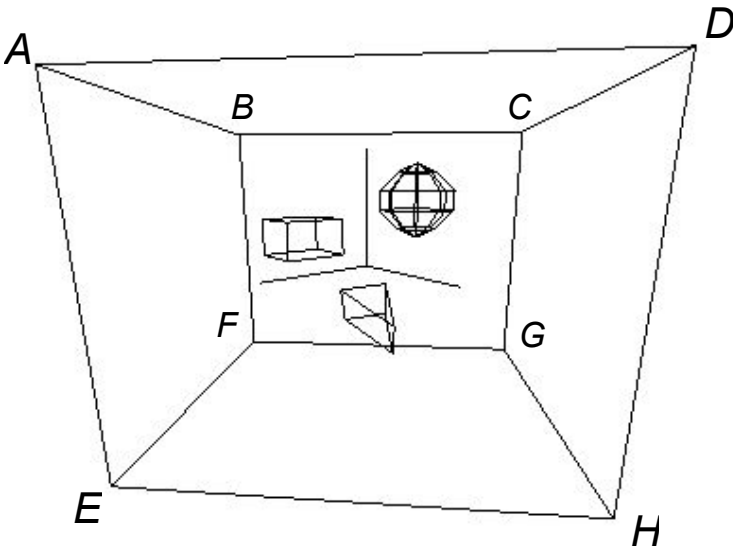
1.3 Kuupide kaheksandpuu paigutus objektide ruumis

Olgu antud objektide ruum W ja kuupide kaheksandpuu G. Kaheksandpuu G paigutus ruumis esitatakse 8 punkti koordinaatidega, mis määravad G juurtipuks oleva kuubi asukoha kolmemõõtmelises ruumis. Joonis 1.4 kujutab joonisel 1.2 toodud näidisobjektide ruumi W_1, millesse on paigutatud kõiki objekte haarav kuupide kaheksandpuu G_1 juur.

Toodud kuupide kaheksandpuu G_1 paigutust objektide ruumis W_1 tähistame

$$G_1(W_1) = <(A_x, A_y, A_z), (B_x, B_y, B_z), (C_x, C_y, C_z), (D_x, D_y, D_z),$$

$$(E_x, E_y, E_z), (F_x, F_y, F_z), (G_x, G_y, G_z), (H_x, H_y, H_z)>.$$



Joonis 1.4: Kaheksandpuu G_1 juur objektide ruumis W_1

Puu paigutusest sõltub kuubi tippude täidetus. Kuupide kaheksandpuu tipp on täidetud, kui tipule vastav kuubi ja ruumi objektide ühisosa on võrdne kuubiga. Kuupide kaheksandpuu tipp on tühi, kui sellele vastaval kuubil ei ole ühisosa ühegi ruumi objektiga. Tipp on osaliselt täidetud, kui see ei ole täidetud ega tühi.

1.4 Kuupide kaheksandpuul põhinevad ruumi mudelid

Sõltuvalt lahendatavast ülesandest võib meid huvitada kas tühja või täidetud ruumi mudel. Objektide ruumi paigutatud kuupide kaheksandpuu on tühja ruumi mudel, kui kõik selle lehed on tühjad. Samasugune kaheksandpuu on täidetud ruumi mudel, kui kõik selle lehed on täidetud.

Ühte tüüpi (tühja / täidetud ruumi) kuupide kaheksandpuu mudel on optimaalne tasemeni *l,* kui kõik tipud on ülimalt *l* taseme tipud ning suvalise kuni *l* taseme tipu lisamisel kaotab mudel oma tüübi (ei ole enam tühi või täidetud). Tühi või täidetud kuupide kaheksandpuu mudel on mitteoptimaalne tasemeni *l,* kui puule on võimalik lisada selline kuni *l* taseme tipp, millele vastav kuup on vastavalt tühi või täidetud. Selline lisamine võib, kuid ei tarvitse, muuta mudeli optimaalseks.

Täidetud optimaalse mudeli ja tühja optimaalse mudeli vahel on seosed, mille abil on ühest puust võimalik tuletada teine. Kehtivad järgmised reeglid:

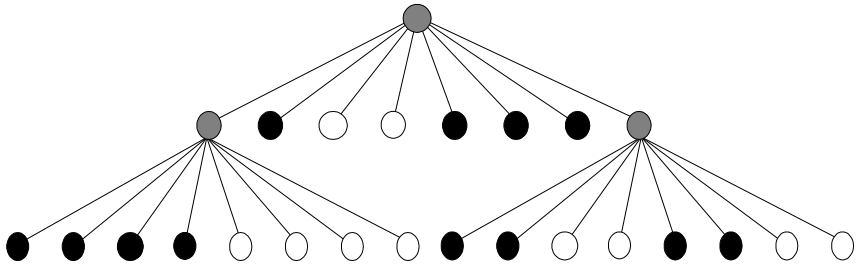
1. Kui tipp on osaliselt täidetud, kuulub ta mõlemasse mudelisse.

2. Kui täidetud ruumi mudeli leht on täidetud, siis sellele vastava tühja ruumi mudeli

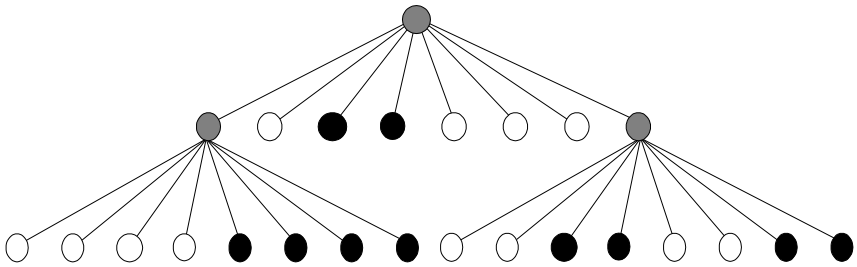
leht on tühi.

3. Kui tühja ruumi mudeli leht on täidetud, siis sellele vastava täidetud ruumi mudeli leht on tühi.

Näidis vastavuses olevatest täidetud ja tühja ruumi mudelitest on joonisel 1.5.



täidetud ruumi mudel



tühja ruumi mudel

Joonis 1.5: Täidetud ja tühja ruumi mudelid

Reaalsetes rakendustes võib vaja minna mõlemat mudelit korraga. Sellisel juhul on otstarbekas kasutada ühte ainsat puustruktuuri, mille tipud on märgendatud vastavalt tühjaks, täidetuks või osaliselt täidetuks.

Kuupide kaheksandpuu mudel on ühtlane, kui iga puu lehttipu kõik eksisteerivad külgnevad tipud on kas samal tasemel või ühe võrra erineval tasemel. Puu on ebaühtlane kui leiduvad kaks külgnevat lehttippu, mille tase erineb kahe või rohkema võrra.

1.5 Kuupide kaheksandpuu mudeli koostamine objektide ruumi kirjelduse põhjal

Teeotsinguülesande lahendamisel on optimaalsel ruumi mudelil oluline roll. Vaatleme lihtsat algoritmi optimaalse kuupide kaheksandpuu mudeli loomisel, kui aluseks on eelpoolkirjeldatud objektide ruum.

Algoritm põhineb ruumi jaotamisel kuupide kaheksandpuu osadeks „jaga ja valitse“ meetodil. Kaheksandpuid üldiselt kasutatakse ka teistes „jaga ja valitse“ tüüpi

algoritmides andmestruktuuride optimeerimiseks. Vastavaid lahendusi kirjeldab [4]. Kuupide kaheksandpuu võimaldab esitada ruumis ühtlaselt täidetud / täitmata piirkondi väiksema mälukuluga. Rohkem detailsust nõudvaid piirkondi saab esitada suurema hulga tippudega.

Mudeli keerukus ja sellest lähtuvalt ka mäluvajadus ning töötlemisaeg sõltuvad tasemest – mida suurem on loodava optimaalse puu tase, seda keerukam on loodav mudel. Soovitav tase on algoritmi oluline parameeter. Veel sõltub algoritmi töö sellest, kas loodav mudel peab olema ühtlane või mitte.

Ühtlase mudeli loomine nõuab rohkem mälu ja aega. Tippude omavahelisele suhtele esitatav piirang tekitab mudelisse täiendavat struktuuri, mida saab ära kasutada näiteks otsingugraafi loomisel. Vastavat meetodit kirjeldatakse teises peatükis.

1.6 Algoritm tipu täidetuse kontrollimiseks

1.6.1 Ülesande kirjeldus

Optimaalse mudeli loomise algoritmi töös on tihti vaja kontrollida, kas mõni tipp on täidetud või tühi. Selleks tuleb vastavalt eelpool antud definitsioonile leida tipule vastava kuubi ja ruumi objektide ühisosa. Selle ülesande lahendamine taandub risttahuka (kuupide kahendpuu tipp) ja kumera hulktahuka (objekt ruumis) lõikumise kontrollimisele. Siinkirjeldatud algoritmi on esitanud Greene [7].

1.6.2 Algoritmi kirjeldus

Algoritm koosneb kahest osast - eeltööst ja lõikumise kontrollist. Eeltöö tulemustena leitud tasandi võrrandid ja siluettservade sirge võrrandid on võimalik algoritmi töö kiirendamiseks salvestada ning taaskasutada. Algoritm on kiire ja efektiivne tänu asjaolule, et selle põhisammudeks on tasandi ja sirge võrrandite väärtustamine ja väärtuste võrdlemine. Võrdluste arv sõltub otseselt etteantud kumera hulknurga P tahkude arvust ja P projektsiooni servade arvust.

Algoritm 1.1: Risttahuka ja kumera hulktahuka lõikumise kontroll

Sisend:

1. Risttahukas R.
2. Kumer hulktahukas P.

Väljund:

1. Tõene, kui *P* ja *R* lõikuvad, vastasel juhul väär.

Algoritmi sammud:

1. Leiame iga *P* tahu jaoks selle tasandi võrrandi.
2. Leiame iga *P* tahu jaoks risttahuka *R* lähima ja kaugeima tipu tasandist (vastavalt *n*-tipp ja *p*-tipp).
3. Vaatleme *P* kolme ristprojektsiooni teljetasanditele. Need projektsioonid on hulknurksed. Hulktahuka *P* serva, mis moodustab projektsiooni serva, nimetame siluettservaks. Leiame iga siluettserva sirge võrrandi.
4. Vaatleme risttahuka *R* projektsiooni samadele teljetasanditele. Need projektsioonid on kuubid. Iga punktis 3 leitud sirge võrrandi jaoks leiame *R* projektsiooni lähima ja kaugeima tipu (vastavalt *n*- ja *p*-tipu).
5. Leiame hulktahukat *P* piirava risttahuka, mille tahud on vastavalt paralleelsed kolme teljetasandiga.
6. Kui *R* ei lõiku *P*-d piirava risttahukaga, siis *P* ja *R* ei lõiku. Lõpetame töö.
7. Vaatleme *P* tahkudele vastavaid tasandeid. Kui *R* asub mõnest tasandist täielikult ühel pool, siis *P* ja *R* ei lõiku. Lõpetame töö.
8. Vaatleme *P* projektsioonide servi. Kui mõnes kolmest projektsioonist asub *R* projektsioon *P* projektsiooni servast täielikult ühel pool, siis *P* ja *R* ei lõiku. Lõpetame töö.
9. Mittelõikumise variandid on ammendatud, millest tulenevalt *P* ja *R* lõikuvad.

1.7 Algoritm kaheksandpuus külgnevate tippude leidmiseks

1.7.1 Ülesande kirjeldus

Ruumi mudeli koostamise algoritmi teine alamülesanne on kaheksandpuu tipuga külgnevate tippude leidmine. Seda operatsiooni kasutatakse ka edaspidi otsingugraafi koostamisel. Ülesandel on mitmeid triviaalseid lahendusi. Näiteks võib kahe tipu külgnevust kontrollida ühise ülemtipu leidmise ning sellele järgneva puuotsingu abil. On selge, et selline lahendus on aeganõudev ja ei sobi seega mahukate ülesannete täitmiseks.

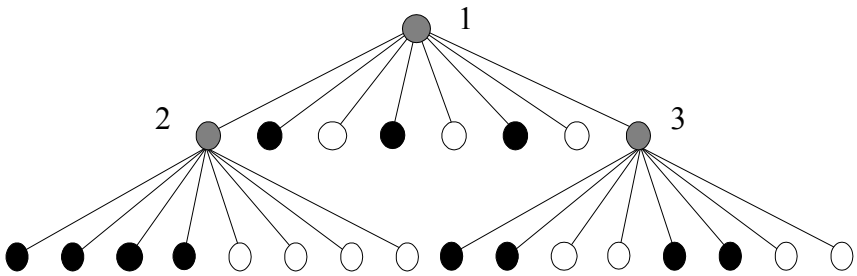
Meetodit külgnevate tippude kiireks leidmiseks kuupide kaheksandpuus kirjeldab [8]. Töö kiirendamiseks esitatakse kaheksandpuu maatrikskujul ning kasutatakse puu lehtede adresseerimist.

1.7.2 Kaheksandpuu maatriksesitus

Kaheksandpuud esitav maatriks V koosneb ridadest $F_i, i \in \mathbb{N}$ mis vastavad puu osaliselt täidetud tippudele. Igal real on kaheksa veergu $S_{i1}...S_{i8}$, mis tähistavad vastava tipu alamtippe. S_{ij} väärtus sõltub sellest, mis tüüpi alamtipuga on tegemist. Võimalusi on kolm:

1. Kui alamtipp on osaliselt täidetud, siis $S_{ij} = k, k \in \mathbb{N}$, kus k on alamtipule vastava rea number maatriksis.
2. Kui alamtipp on tühi, siis $S_{ij} = 0$.
3. Kui alamtipp on täidetud, siis $S_{ij} = -w, w \in \mathbb{N}$.

Tabelis 1.1 on joonisel 1.6 kujutatud kaheksandpuu esitus maatrikskujul. Kasutame w väärtusena naturaalarvu 1.



Joonis 1.6: Kaheksandpuu

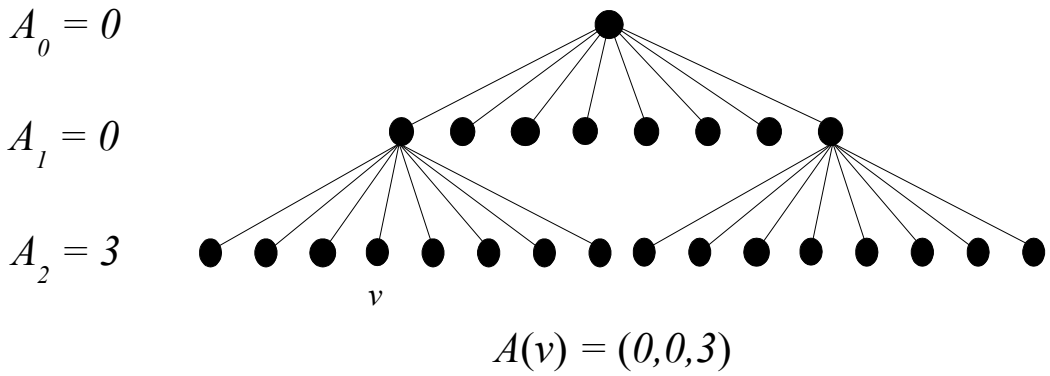
	0	***1***	***2***	***3***	***4***	***5***	***6***	***7***
1	2	-1	0	-1	0	-1	0	3
2	-1	-1	-1	-1	0	0	0	0
3	-1	-1	0	0	-1	-1	0	0

Tabel 1.1: Kaheksandpuu O maatriksesitus

1.7.3 Kaheksandpuu tippude adresseerimine

Kasutame järgnevat kaheksandpuu lehtede adresseerimisviisi. Tipu v aadressiks on jada $A(v) = (A_0, A_1,...,A_{N-1})$, kus $A_i \in \{0...7\}$ näitab, milline i. taseme tipu alamtipp sisaldab

tippu v. Kui $A_i = 0$, siis valiti esimene alamtipp, kui $A_i = 1$, siis teine jne. Näide aadressist on joonisel 2.7.



Joonis 1: Kaheksandpuu lehtede adresseerimine

Käesolevas ülesandes kasutatakse aadress esitust maatriksina, kus A_i väärtused on kahendkujul. Joonisel 2.7 kujutatud näite aadress oleks kujul

$$A(v)=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 1 & 1\end{pmatrix}.$$

1.7.4 Algoritmi kirjeldus

Kirjeldatava meetodi rakendamine käesolevas ülesandes eeldab, et kaheksandpuu esitused graafina ja maatriksina on igal hetkel identsed.

Algoritm 1.2: Külgneva tipu leidmine kaheksandpuu maatriksesituse abil

Sisend:

1. Kaheksandpuu O esitus nii graafi kui maatriksina.
2. Puu O tipp u.

Väljund:

1. Tipule u puus O külgnevate tippude hulk U.

Algoritmi sammud:

1. Leitakse u aadress $A(u)$ kahendkujul maatriksina.
2. Kasutades kirjeldatud reeglite hulka ([7]) leitakse võimalike u külgnevate tippude aadresside hulk $N=\{A(v_i) \mid v_i \in V(O)\}$.
3. Aadressi järgi leiame puu maatrikesitusest tipud ning väljastame hulga U.

1.8 Algoritm optimaalse kuupide kaheksandpuu mudeli koostamiseks

1.8.1 Ülesande kirjeldus

Olgu antud objektide ruum *W*. Algoritmi ülesandeks on selle põhjal koostada ruumi mudel *D*. Vastavalt vajadusele koostab algoritm täidetud või tühja ruumi ühtlase või ebaühtlase mudeli. Lisaks võib määrata mudeli maksimaalse sügavuse.

1.8.2 Algoritmi kirjeldus

Antud objektide ruumi analüüsiks paigutame sellesse kuupide kaheksandpuu *G*. Analüüsipuu *G* paigutus määrab, millist ruumi osa modelleeritakse.

Ruumi mudeli koostamise alamülesandeks on loodava mudeli osa ühtlustamine. Vastav protseduur kasutab põhialgoritmi andmeid ning jaotab ühtluse tingimust rikkuvad tipud alamtippudeks.

Algoritm 1.3: Ruumi mudeli alampuu ühtlustamine.

Sisend:

1. Analüüsigraaf *G*.
2. Vaatlemata tippude nimekiri *O*.
3. Vaadeldav tipp *c*.

Väljund:

Puudub.

Algoritmi sammud:

1 Vaatleme tipu *c* kõiki külgnevaid tippe puus *G*. Iga sellise tipu *c′* jaoks leiame tema taseme.

1.1 Kui *c* tase on *c′* tasemest väiksem rohkem kui ühe võrra, jaotame tipu *c* kaheksaks võrdseks alamtipuks $c_1 ... c_8$. Lisame need tipud nimekirja *O* lõppu.

1.2 Kui *c* tase on *c′* tasemest suurem rohkem kui ühe võrra, jaotame tipu *c′* kaheksaks võrdseks alamtipuks $c'_1 ... c'_8$. Lisame need tipud nimekirja *O* lõppu.

Algoritm 1.4: Kuupide kaheksandpuu mudeli koostamine etteantud objektide ruumi põhjal.

Sisend:

1. Objektide ruum *W*.
2. Objektide ruumi *W* paigutatud kuupide kaheksandpuu *G*, kuhu töö alguses kuulub vaid üks tipp *r*.
3. Loodava optimaalse mudeli tase *l*.
4. Tõeväärtus *u*, mille tõese väärtuse korral koostatakse ühtlane mudel ning väära väärtuse korral ebaühtlane mudel.
5. Tõeväärtus *v*, mille tõese väärtuse korral koostatakse tühja ruumi mudel ning väära väärtuse korral täidetud ruumi mudel.

Väljund:

1. Soovitud parameetritega ruumi mudel *D*.

Algoritmi tööandmed:

1. Vaatlemata tippude nimekiri *O,* kuhu algoritmi töö alguses kuulub *G* juurtipp *r*.
2. Vaadeldud tippude nimekiri *P*, mis on töö alguses tühi.
3. Koostatava mudeli graaf *D*, mis on töö alguses tühi.
4. Vaadeldav tipp *c*.

Algoritmi põhitsükkel:

1 Võtame nimekirja *O* esimese tipu vaadeldavaks tipuks *c*.

2 Kontrollime eelpoolkirjeldatud meetodi abil, kas *c* on täidetud objektide ruumis *W*.

3 Kui *c* on osaliselt täidetud:

3.1 Lisame *c* graafile *D*. Kui *c* on *G* juurtipp, siis lisame ta graafi *D* juurtipuna. Vastasel juhul leiame *c* ülemtipu graafis *G* (tähistame seda *d*) ning lisame *c* graafile *D* tipu *d* alamtipuna. Graafis *D* leidub tipp *d*, sest see on lisatud algoritmi senise töö käigus.

3.2 Kui *c* tase on väiksem kui *l*, siis jaotame *c* kaheksaks võrdseks alamtipuks $c_1 ... c_8$. Lisame need tipud nimekirja *O* lõppu.

4 Kui c on tühi:

4.1 Kui *v* on tõene:

- Lisame *c* graafile *D*. Kui *c* on *G* juurtipp, siis lisame ta graafi *D* juurtipuna. Vastasel juhul leiame *c* ülemtipu graafis *G* (tähistame seda *d*) ning lisame *c* graafile *D* tipu *d* alamtipuna. Graafis *D* leidub tipp *d*, sest see on lisatud algoritmi senise töö käigus.

- Kui *u* on tõene, siis *Ühtlusta* (*G, O, c*)

5 Kui *c* on täidetud:

5.1 Kui *v* on väär:

- Lisame *c* graafile *D*. Kui *c* on *G* juurtipp, siis lisame ta graafi *D* juurtipuna. Vastasel juhul leiame *c* ülemtipu graafis *G* (tähistame seda *d*) ning lisame *c* graafile *D* tipu *d* alamtipuna. Graafis *D* leidub tipp *d*, sest see on lisatud algoritmi senise töö käigus.

- Kui *u* on tõene, siis *Ühtlusta* (*G, O, c*)

6 Eemaldame tipu *c* nimekirjast *O* ning paigutame selle nimekirja *P*.

7 Kui nimekiri *O* on tühi, siis lõpetame algoritmi töö.

8 Jätkame algoritmi tööd esimeselt sammult.

2. Otsingugraafi koostamine kuupide kaheksandpuu mudelil

2.1 Sissejuhatus

Otsinguülesande järgmiseks oluliseks etapiks pärast ruumi mudeli koostamist on otsingugraafi loomine. Graafi koostamisel rakendatavad meetodid sõltuvad otseselt kasutatud mudelist. Samas võib ühe mudeli põhjal koostada mitmeid erinevaid graafe. Saadud graafid võivad erineda tippude arvu ja tiheduse ning servade arvu ja kaalu järgi.

Lisaks loendatavatele ja mõõdetavatele väärtustele on otsingugraafil ka muid parameetreid, mis kirjeldavad selle sobivust reaalse maailma tingimustega ning otsingutulemuste rakendatavust reaalses maailmas. Erinevad otsingugraafi koostamise süsteemid võivad anda erinevate otsingukeskkondade ja mobiilsete agentide puhul väga erinevaid tulemusi.

Käesolevas töös kirjeldatakse kahte graafi koostamise meetodit ning uuritakse mõlema tugevaid ja nõrku külgi. Mõlemad meetodid tuginevad kuupide kaheksandpuu andmestruktuuril põhinevale ruumi mudelile. Selgitatakse graafi tippude paigutamist ja servade valikut. Eraldi vaadeldakse serva kaalu arvutamise võimalusi erinevate mobiilsete agentide jaoks.

2.2 Otsingugraafi koostamine ühtlase mudeli meetodil

2.2.1 Nõudmised mudelile

Kirjeldatav meetod vajab tööks eelmises peatükis kirjeldatud ühtlast tühja ruumi mudelit. Tühja ruumi mudel on graafi koostamiseks sobivam, sest selle mudeli piires paigutatud graafi tipud asetsevad ka reaalses maailmas tühjas (läbitavas) ruumis. Ühtlase ruumi mudel tagab meile kindla seose külgnevate kaheksandpuu tippude vahel (nende tippude tase ei erine rohkem kui ühe võrra).

2.2.2 Graafi tippude määramine

Kuupide kaheksandpuul põhinev tühja ruumi mudel võimaldab meil valida ruumis teatud punktid graafi tippudeks. Sellel valikul saame lähtuda kaheksandpuu lehtedest –

erineva suurusega kuupidest, mis katavad tühja ruumi osi. Sellele teadmisele tuginedes võime kuubi sisse graafi tipu paigutamisel olla kindlad, et see asub otsitavas ruumis.

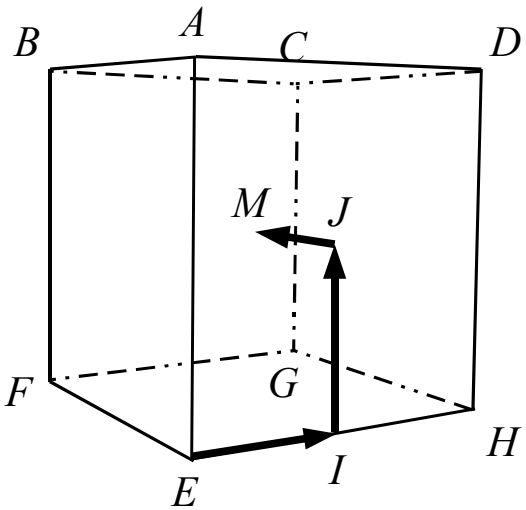
Graafi tipu või tippude asukoha valikuks on mitmeid võimalusi. Neid kaaludes tuleb arvestada järgmiseid tegureid:

- tekkivate tippude arv,
- tekkivate tippude tihedus,
- tipu koordinaatide arvutamise keerukus.

Kui ühe kuubi sisse panna mitu tippu, võib graafi tippude koguarv kasvada väga suureks. Selline graaf nõuab palju ressursse (mälu) ning tingib tõenäoliselt ka suure servade hulga. Suure tippude hulgaga võib kaasneda ka liigne tihedus (kuigi viimane võib esineda ka väiksema tippude arvu juures). Sõltuvalt tippude valikul kasutatud reeglitest võivad graafi tipud sattuda nii lähtestikku, et mõistlikum oleks kahe tipu asemel kasutada ühte.

Tihedus ei tohi olla ka liiga väike, sest sellisel juhul kannatab graafi põhjal leitud teede kvaliteet ning optimaalsus. Olulised on ka arvutusmeetodid, mille abil leitakse graafi tipu koordinaadid ruumis. Koordinaate on vaja graafi servade lisamisel ning kaalu arvutamisel. Arvutusmahukate valemite puhul pikeneb otsingugraafi koostamise aeg.

Kirjeldatav meetod ei tekita liigselt palju tippe, ei paiguta tippe põhjendamatult tihedalt ning on lisaks veel kerge arvutada. Aluseks võetakse ühtlane tühja ruumi mudel ning vaadeldakse kõiki selle mudeli lehti. Graafi tipp paigutatakse kõikide mudeli lehttippude (kuupide) geomeetrilistesse keskpunktidesse (joonis 2.1).



$$|\vec{EI}| = \frac{1}{2} \cdot |\vec{EH}|$$

$$|\vec{IJ}| = \frac{1}{2} \cdot |\vec{HD}|$$

$$|\vec{JM}| = \frac{1}{2} \cdot |\vec{DC}|$$

Joonis 2.1: Kuubi keskpunkt

Kuubi keskpunkti *M* koordinaadid arvutatakse kuubi tippude koordinaatide järgi:

$$M = (\frac{x_1 + x_2}{2}; \frac{y_1 + y_2}{2}; \frac{z_1 + z_2}{2}),$$
$$kus\,(x_1; y_1; z_1)\; ja\,(x_2; y_2; z_2)\; on\; kuubi\; diagonaali\; otstipud.$$

Valem 2.1: Kuubi keskpunkti arvutamise valem

Saadud graafi tippude arv sõltub kaheksandpuu lehttippude arvust, tihedus kaheksandpuu sügavusest. Mida rohkem on puul tasemeid, seda rohkem on väikseid (kuubi mõttes) tippe, mis asetsevad tihedalt kõrvuti.

2.2.3 Graafi servade lisamine

Kui graafi tipud on valitud, peame need omavahel ühendama. Iga loodud ühendus peab tähistama teekonda, mida liikuv agent suudab reaalses maailmas läbida. Selle tingimuse täitmiseks on mitmeid võimalusi. Kõige triviaalsem on iga serva loomisel kontrollida, kas kahe graafi tipu vahel on võimalik liikuda. Otstarbekam on siiski tugineda teadaolevatele seostele. Paneme tähele, et koostatav graaf on suunatud, sest mingi ruumi läbitavus eri suundades võib olla erineva kuluga (või isegi võimatu).

Kasutades ühtlase mudeli eeldust, teame, et nende lehttippude poolt kaetav ruum on reaalses maailmas läbitav. Samuti teame, et igal mudeli lehttipuga külgnevad tipud on sama või ühe võrra erineva tasemega.

Algoritm 2.1: Otsingugraafi koostamine ühtlase tühja ruumi mudeli põhjal.

Sisend:

1. Ühtlane täidetud ruumi mudel *G*.

Väljund:

1. Otsingugraaf *S*.

Algoritmi sammud:

1. Leiame antud ruumi *G* mudelit sügavuti läbides kõik lehttipud *v*.

1 Leiame tipu *v* taseme *l*.

2 Vaatleme tipuga *v* külgnevaid tippe kõigis kuues suunas (iga kuubi tahu kohta on tipul üks suund). Suundi tähistame numbritega 1...6. Tähistame vaadeldava suuna *d*.

Leiame külgneva tipu taseme k.

3 Kui $l = k$ (u tase on võrdne v tasemega) või $l = k + 1$ (u tase on väiksem kui v tase):

3.1 Leiame mudeli tipuga v külgneva mudeli tipu u suunas d.

3.2 Lisame otsingugraafile S mudeli tipu v keskpunkti v' ja tipu u keskpunkti u'.

3.3 Lisame otsingugraafile S servad tipust v' tippu u' ja vastupidi.

4 Kui $l = k + 1$ (u tase on suurem kui v tase):

4.1 Leiame mudeli tipuga v suunas d külgnevad neli mudeli tippu u_i, $i = 1...4$.

4.2 Iga mudeli tipu u_i ($i = 1...4$) jaoks:

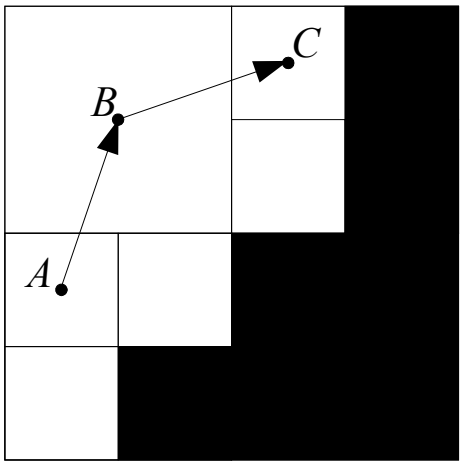
- Lisame otsingugraafile S mudeli tipu v keskpunkti v'.
- Lisame otsingugraafile S mudeli tippude u_i keskpunktid u_i', $i = 1...4$.
- Iga u_i', $i = 1...4$ jaoks lisame otsingugraafile S serva tipust v' tippu u_i'.

Serva lisamisel tuleb määrata ka serva kaal, mis võib kajastada näiteks läbitava tee pikkust, läbimisaja, läbimiseks vajaliku energia või mingi muu ressursi kulu. Kasutatav funktsioon sõltub liikuvast agendist. Võimalik on lisada piiranguid lähtuvalt agendi profiilist. Kui mingi serva kaal on liiga suur, või ei ole teelõik antud agendile läbitav, võib serva jätta graafi lisamata.

2.2.4 Meetodi analüüs

Kirjeldatud meetod sobib väga erinevate lähteruumide analüüsiks. Ühtlasel mudelil põhinevat analüüsi on võimalik kasutada nii väliskeskkonna (mäed, orud, metsad, linnad) kui ka sisekeskkonna (ruumid, tunnelid, labürindid) kaardistamiseks. Mudeli koostamine on arvutuslikult lihtne ning ei nõua keerukaid lisastruktuure.

Kuupide kaheksandpuu mudel on ressursside kasutuselt ratsionaalsem ja annab detailsema mudeli kui kogu ruumi jaotus võrdseteks ühtlasteks alamosadeks. Üheks ühtlase mudeli probleemiks on väiksem paindlikkus suurtes tühjades ruumides (väliskeskkonnas). Suur tühi ruum esitatakse otsingugraafil väheste tippudega ning seda läbivad teekonnad võivad sisaldada ebaotstarbekaid murdjooni. Näide sellisest probleemist (kahemõõtmelisel projektsioonil) on joonisel 2.2



Joonis 2.2: Ühtlase mudeli probleem

Joonisel on toodud lihtne ruumi mudeli projektsioon, kus valged ruudud tähistavad tühja ruumi ja mustad ruudud täidetud ruumi. Tegemist on lihtsustatud näitega olukorrast, kus lühimaks teeks osutub murdjoon, samas kui on ilmne, et punktide *A* ja *C* vahele võiks tõmmata sirge. Probleemi lahendamiseks on mitmeid meetodeid, millest mõned laiendavad ja täiendavad mudelit ja ruumi analüüsi, teised otsingumeetodeid.

2.3 Otsingugraafi koostamine punktidevahelise nähtavuse meetodil

2.3.1 Nõudmised mudelile

Kirjeldatud meetod on ette nähtud teede leidmiseks sisekeskkonnas (hooned, tunnelid, labürindid). Nii tippude paigutamisel kui servade loomisel rõhutakse asjaolule, et lühim tee kahe punkti vahel on sirgjoon. Ruumi analüüsil kasutatakse kahe punkti vahelist nähtavust, et veenduda, kas tippude vaheline tee on läbitav.

Meetodi kasutamiseks on vaja tühja ruumi mudelit. Täiendavad nõuded mudelile puuduvad. Loomulikult on nii selle kui ka kõigi teiste meetodite puhul võimalik mudeli koostamist vajalikus suunas „juhtida“. Nõuda saaks näiteks minimaalset puu sügavust kogu puu ulatuses või ainult tühja ning täidetud ruumi piiril asuvate tippude jaoks.

2.3.2 Graafi tippude määramine

Graafi tipud paigutatakse tühja ruumi äärtesse „seinade“ lähedale. Põhjenduseks on järgmised kaks tähelepanekut:

1. Kui me näeme lähtepunktist sihtpunkti ning teel ei ole takistusi, võime minna otse.
2. Kui sihtpunkti ei ole näha või teel on takistus, siis liigume võimalikult otsemat teed pidi takistuseni, möödume sellest ja jätkame sihtpunkti (või järgmise takistuse) suunas.

Sellest lähtuvalt seamegi otsingugraafi tipud nendesse tühja ruumi mudeli tippudesse, millega vähemalt ühes suunas ei külgne teine sama mudeli tipp.

Algoritm 2.2: Otsingugraafi tippude määramine punktidevahelise nähtavuse meetodi puhul.

Sisend:

1. Tühja ruumi mudel *G*.

Väljund:

1. Ainult tippe sisaldav otsingugraaf *S*.

Algoritmi sammud:

1 Leiame antud ruumi mudelit *G* sügavuti läbides kõik lehttipud *v*.

2 Iga tipu *v* jaoks:

2.1 Iga suuna $d = 1...6$ kohta leiame, kas vastavas suunas külgnev tipp v' kuulub mudelisse või mitte. Kui v' ei kuulu samasse mudelisse, siis lisame tipu *v* keskpunkti otsingugraafi *S*.

Tipu keskpunkti leidmiseks kasutame sama arvutuskäiku kui ühtlase mudeli meetodi puhul. Saadud graafis on tippe vähem või sama palju kui mudelis lehttippe. Tippude tihedus on suhteliselt suur, kuid nad kuhjuvad ruumis kindlastesse piirkondadesse („seinte“ äärde). Meetod on ka arvutuslikult lihtne, ning vajab lisaks kuubi keskpunkti leidmisele veel ainult kuupide kaheksandpuus külgnevate tippude leidmise operatsiooni.

2.3.3 Graafi servade lisamine

Graafi servade loomisel kasutame ühte üldist reeglit. Iga tipuga ühendame kõik teised tipud, mis on sellest tipust nähtavad. Nähtavuse defineerime järgnevalt. Olgu antud tipud *A* ja *B* ning objektide ruum *W*. Tipp *B* on tipust *A* nähtav, kui sirge *AB* ei lõiku ühegi ruumi *W* objektiga.

Graafi servade lisamisel on vaadeldava objektide ruumi objektideks täidetud ruumi mudeli lehttippudes asuvad kuubid. Järgnevalt kirjeldatav algoritm nähtavuse kontrolliks on optimeeritud töötama täidetud ruumi mudelil.

Algoritmi alamülesandeks on sirge ja kuubi lõikumise kontroll. Sobivat lahendust sellele ülesandele kirjeldab Woo [9]. Alamülesande parameetriteks on kuubi ja sirge

koordinaadid ning tagastatavaks väärtuseks tõeväärtus, mille tõese väärtuse korral joon lõikub kuubiga.

Algoritm 2.3: Otsingugraafi servade lisamine punktidevahelise nähtavuse meetodil.

Sisend:

1. Täidetud ruumi mudel *F*.
2. Otsingugraaf *S*, mis sisaldab algoritmis 2.2 määratud tippe.

Väljund:

1. Otsingugraaf *S*, millele on lisatud ka servad.

Algoritmi sammud:

1 Vaatleme kõiki graafi *S* tippe *a*.

2 Iga sellise tipu *a* puhul vaatleme kõiki teisi graafi *S* tippe *b*.

3 Iga sellise paari (*a*, *b*) jaoks leiame nende asukohad ruumis *A* ja *B*.

3.1 Leiame sirge võrrandi sirgele *t*, mis läbib tippe *A* ja *B*.

3.2 Leiame tasandi võrrandi tasandile n_1, mis on risti joonega *t* ning läbib tippu *A*.

3.3 Leiame tasandi võrrandi tasandile n_2, mis on risti joonega *t* ning läbib tippu *B*.

3.4 Leiame ruumi mudelit *F* sügavuti läbides kõik lehttipud *v*.

3.5 Kontrollime, kas tipule *v* vastav kuup asub täielikult või osaliselt tasandite n_1 ja n_2 vahel. Selleks kontrollime, kas mõni selle kuubi tipp asub nende tasandite vahel.

3.6 Kui vastav kuup või selle osa asub tasandite n_1 ja n_2 vahel, kontrollime, kas lõik *t* lõikub selle kuubiga.

3.6.1 Kui ei lõiku, siis lisame otsingugraafi serva tippude *a* ja *b* vahele.

Lahenduse praktilisust piirab algoritmi keerukus. Põhiline töötlemisaeg kulub punktidevahelise nähtavuse arvutamisele. Seega saavutaksime suurima võidu graafi tippude arvu vähendamisega. Otstarbekas on kasutada mobiilsest agendist lähtuvaid optimisatsioone. Selliseid meetodeid kirjeldatakse peatükis 3.

Teine võimalus tippude arvu vähendamiseks on nende määramise algoritmi 2.2 asendamine sellisega, mis teeb analüüsi teel täpsemat eristust oluliste ja vähem oluliste punktide vahel ruumis.

2.3.4 Meetodi analüüs

Meetod erineb oluliselt ühtlasel mudelil põhinevast. Ühtlase tippude paigutuse asemel eristatakse olulised piirkonnad heuristilise algoritmiga. Servade lisamisel ei kasutata kaheksandpuu külgnevate tippude suhteid, vaid kontrollitakse tippudevahelist nähtavust. Servade arvu piiramiseks kontrollitakse tingimusi nagu minimaalne lubatud distants ühendatavate punktide vahel ja sobivus mobiilse agendi profiiliga.

Punktidevahelise nähtavuse meetodil on mitmeid eeliseid ühtlase mudeli kasutamise ees. Tippude arv on väiksem (seevastu on potentsiaalselt suurem servade arv). Otsingu tulemusena tekib vähem ebaotstarbekaid murdjooni.

Kaheksandpuu külgnevate tippude puhul võisime eeldada, et ühest tipust teise viiv serv asub teatud ulatuvusega tühjas ruumis (kõrvuti asuvad tipud/kuubid garanteerisid selle). Nähtavus sellisel kujul, nagu ta servade loomise juures defineeritud on, seda garanteerida ei saa. Seega tuleb põhjalikumalt kontrollida, kas mobiilne agent on üldse võimeline otsingugraafil liikuma (kas ta mahub igalt poolt läbi).

Kokkuvõttes on tegemist spetsiifiliseks eesmärgiks optimeeritud lahendusega. Väliskeskkonnas ei ole nähtavusel põhinev meetod otstarbekas suure servade arvu tõttu. Kui servade arvu on võimalik (näites agendist lähtuvalt) piirata, siis võib antud meetod sobida ka praktilises töös kasutamiseks.

3. Meetodeid otsingugraafi kohandamiseks ja optimeerimiseks

3.1 Mobiilse agendi profiil

Otsingugraafi koostades on võimalik seda kohandada konkreetsele kasutajale. Kasutajana vaatleme autonoomset mobiilset agenti, kes on võimeline modelleeritavas maailmas liikuma. Erinevatel agentidel on erinevad võimed ning nende jaoks on läbitavatel teekondadel erinevad optimaalsuse kriteeriumid.

Mobiilse agendi profiil koondab olulisemad agendi liikumist mõjutavad andmed. Järgnev nimekiri esitab agenti modellerivatest funktsioonidest koosneva profiili struktuuri. Agendi profiili kuuluvad:

1. Läbitavusfunktsioon *p*(*N*), mis seab otsingugraafi tipule *N* vastavusse tõeväärtuse $b \in \{t, v\}$, mille tõese väärtuse korral on tipp agendi jaoks läbitav. Täpsemalt, agent on võimeline antud tipus peatuma ja suunda muutma.

2. Kaalufunktsioon *c*(*E*), mis seab otsingugraafi servale *E* vastavusse reaalarvu $d \in \mathbb{R} \cup \{\infty\}$, mis on serva *E* kaal antud agendi jaoks. Kaalufunktsiooni väärtus võib mõõta teepikkust, kuluvat energiat, aega või näiteks kütusekulu.

3. Pööragufunktsioon *r*(*E, F*) on defineeritud juhul kui *E* ja *F* on kaks otsingugraafi serva ning serva *E* lõpptipp *N* on serva *F* algtipuks. Pööragufunktsioon seab kahele otsingugraafi servale vastavusse väärtuse $e \in \mathbb{R} \cup \{\infty\}$, mis on servas *E* mööda tulles ning tipus *N* serva *F* suunas pööramise kaal.

Need kolm funktsiooni lubavad kirjeldada küllalt erinevaid agente. Läbitavusfunktsiooni abil saame eristada lendavad agendid maa peal liikuvatest ning funktsioonile *p* vastava ruumi kirjeldava lisainformatsiooni andmise teel keelata mõningate piirkondade läbimist. Lendamise keelamiseks märgime mitteläbitavateks tipud, mis asuvad lähimast horisontaalpinnast kõrgemal kui mingi konstant $h \in \mathbb{R}, h > 0$. Erijuhuna võime siiski lubada „õhus“ olemist, kui piisavalt lähedaval on ronimiseks sobiva kaldega vertikaalpind ning agendiks on näiteks varustusega alpinist.

Kaalu ja pöörangufunktsioon määravad otsingugraafi osade läbitavuse. Kaalufunktsiooni abil väärtustatakse servad, mida mööda agent liigub. Pöörangufunktsiooni abil leitakse servadevahelise ülemineku kaal võrreldes muu liikumisega. Mõlema funktsiooni väärtusena võib defineerida lõpmatuse. Sellisel juhul on serva läbimine või pöörde tegemine lõpmatult kulukas ehk võimatu. Funktsioonide negatiivsed väärtused on lubatud juhtudeks, kus mõõdetav väärtus on näiteks energia. Kaldpinda mööda alla liikudes võib agendi energiahulk suureneda. Sellisel juhul on tegemist negatiivse kuluga, mis tuleb vastavalt tähistada.

Kaalufunktsiooni ja pöörangufunktsiooni defineerimisele tuleb panna rõhku, sest need peavad tagama otsingugraafi põhjal leitavate teede optimaalsuse valitud agendi jaoks. Ka neid funktsioone saab täiendada ruumi kirjeldava lisainfoga, mis võimaldab erinevatele ruumi osade läbimist muuta lihtsamaks või raskemaks. Näitena võib siin vaadelda inimest, kes liigub siledal teel ja soisel pinnal. Esimese liikudes võib graafi serva kaalu korrutada läbi ühest väiksema koefitsiendiga, mis vähendab serva kaalu. Teisel juhul tuleks kasutada ühest suuremat koefitsienti, mis kajastaks pinna raskemat läbitavust.

Erinevate agentide puhul on kaalu- ja pöörangufunktsiooni osatähtsused erinevad. Inimese puhul on olulisem kaalufunktsioon, sest pöörang on sõltuvalt kiirusest vähe energiat kulutav tegevus. Sõidukite ja lennukitega on olukord peaaegu vastupidine. Nende jaoks on kiiruse säilitamise seisukohalt väga oluline järskude pöörete vältimine.

3.2 Otsingugraafi kärpimine agendi profiilist lähtuvalt

Kui graafi koostamisel on võimalik kasutada agendi profiili, saame seda kasutada otsingugraafist ülearuse informatsiooni eemaldamiseks. Otsingugraafi kärpides hoiame kokku salvestusmahtu ja töötlemisaega. Graafi koostamise erinevatel etappidel kasutame erinevaid agendi profiili funktsioone. Tippude lisamise ajal saame kasutada funktsiooni $p(N)$ ning servade loomisel funktsiooni $c(E)$. Vastavad reeglid on järgnevad.

1. Iga graafi lisatava kandidaattipu V jaoks leiame $b = p(V)$. Kui b on väär, siis jätame tipu graafi lisamata.

2. Iga graafi lisatava kandidaatserva E jaoks leiame $d = c(E)$. Kui $d = \infty$, siis jätame serva lisamata.

Pöörangufunktsiooni kasutamine on keerulisem, sest kui üks pööranguvõimalus tipus on välistatud, siis ei tähenda see, et ka teised on välistatud.

Suur võit saavutatakse otsingugraafi kärpimisel kõigi maa peal liikuvate agentide puhul. Välistada saab kõik õhus asuvad tipud.

Juhul kui agendi profiili laiendada tema mõõtmetega, võib otsingugraafi kärpimiseks kasutada veel ühte täiendavat võimalust. Iga kandidaatserva lisamisel tuleb eelnevalt kontrollida, kas agent suudab seda serva mööda liikuda nii, et see mahub igalt pool läbi.

3.3 Märkuseid otsingu läbiviimise kohta

Agent võib otsingu alguses asuda sellises ruumipunktis, mille jaoks graafil ei ole vastavat tippu. Sellisel juhul tuleks graafi dünaamiliselt täiendada, lisades algustipu ning sidudes selle lähimate graafi tippudega (või kõigi nähtavate tippudega). Samasugune vajadus võib tekkida lõpptipu määramisel. Arvesse tuleb siiski võtta agendi profiili iseärasusi. Kui agent ei lenda, siis ei ole võimalik leida teed tippu, mis asub õhus.

Otsingutulemuse kvaliteeti triviaalsete ülesannete puhul on võimalik parandada järgneva lihtsa meetodiga. Enne otsingut kontrollitakse, kas lõpptipp on algustipust nähtav. Kui on, siis sellisel juhul on optimaalseks teekonnaks sirgjoon alg- ja lõpptipu vahel.

Täiendavate meetoditena võib kaaluda servadele minimaalse pikkuse seadmist (just pikkuse, mitte kaalu). Näiteks eelpoolkirjeldatud nähtavusel põhineva otsingugraafi puhul võib liiga lühikeste servade ärajätmine anda olulist kokkuhoidu.

Kokkuvõte

Käesolevas töös esitatakse meetod optimaalse tee otsinguks kolmemõõtmelises objektide ruumis. Kirjeldatakse kuupide kaheksandpuu mudeli koostamist etteantud objektide ruumi analüüsi teel. Võimalik on koostada tühja või täidetud ruumi mudeleid, mis võivad olla ühtlased või mitteühtlased. Mudeli keerukust ja suurust on võimalik kontrollida kaheksandpuu sügavusele piirangute seadmisega.

Kirjeldatud on kahte erinevat meetodit ruumi mudeli põhjal otsingugraafi koostamiseks. Esimene kasutab ühtlast tühja ruumi mudelit ning paigutab tippe vastavalt mudeli tihedusele kogu läbitava ruumi ulatuses. Tippude servadega ühendamisel kasutatakse ühtlase mudeli puhul kehtivaid seoseid kõrvuti asuvate kaheksandpuu kuupide vahel.

Teine meetod koostab otsingugraafi, mille tipud on paigutatud tühja ruumi ja täidetud ruumi piirile. Kahe tipu vahele lisatakse serv siis, kui nende vahel on nähtavus (sirgjoon kahe tipu vahel läbib ainult tühja ruumi).

Esimene kirjeldatud meetoditest on rakendatav kõigis otsinguruumides, kuid ei arvesta asjaolu, et kui ruum kahe punkti vahel on läbitav, siis neid ühendav lühim tee on sirglõik. Selle probleemi lahendamiseks välja töötatud nähtavusel põhineva meetodi puuduseks on otsingugraafi suur servade hulk, mis muudab selle praktiliseks vaid piiratud keskkonnas nagu hooned ja tunnelid.

Lõpuks kirjeldatakse agendi profiili, mis läbitavus-, kaalu- ja pöörangufunktsiooni abil väärtustab graafi nii, et kõik servad on agendi jaoks läbitavad. Antakse reeglid, mida rakendades saab agendi profiili põhjal otsingugraafi lihtsustada.

Saadud otsingugraafil optimaalse tee leidmiseks saab kasutada graafiotsingu algoritme. Kui agendi profiil sisaldab ainult läbitavus- ja kaalufunktsiooni, sobivad kasutamiseks kõik suunatud graafil kasutatavad algoritmid. Kui lisandub pöörangufunktsioon, tuleb kasutatavat algoritmi täiendada nii, et tippudes suunda muutes võetakse arvesse tekkivat energiakulu.

Octree-based space models and their use in solving path finding problems

Bachelor thesis

Dan Bogdanov

Abstract

This work explores additional methods for solving the unified path finding problem proposed in [1]. The former paper concentrated on finding the optimal path on a terrain presented as a heightmap. We extend the searchable area to three-dimensional space populated by convex geometrical objects.

We define octree-based models to represent both filled and void space and introduce a special homogeneous type of these models which enforces relations between adjacent nodes in the model octree. Algorithms for constructing these models from the object space are given.

We present two different methods for constructing the search graph from the created model. The first method distributes graph nodes depending on the density of the model and makes use of the homogeneous property of the model when interconnecting the nodes. The second method requires less graph nodes but connects every pair of nodes when one of them is directly visible from the other.

We reach the conclusion that although the first method can be used in any environment, it can provide inaccurate results in trivial situations, suggesting a broken line as the optimal path when a straight line is the obvious choice. The second method solves this problem, but requires a large amount of graph edges in the process.

The agent profile is introduced as means of optimizing and adjusting the search graph. This profile, consisting of three functions, describes the agent's capability of moving through the search space. A set of rules is listed for pruning the search graph by using the functions in the profile.

Kasutatud kirjandus

[1] Bogdanov, D. Optimaalse teekonna leidmise ülesande lahendamine geomeetriliste objektide puude abil, Tartu Ülikool, 2004, (käsikiri).

[2] Szeliski, R. Real-Time Octree Generation from Rotating Objects, Digital Equipment Corporation Cambridge Research Lab Technical Report Series 90/12, 1990.

[3] Saona-Vázquez, C., Navazo, I., Brunet, P. The visibility octree: a data structure for 3D navigation, Computers & Graphics, vol. 23, n. 5, pp 635-643, 1999.

[4] Globus, A. Octree Optimization, Computer Sciences Corporation, NASA Ames Research Center, 1991.

[5] Ashton, T., Cantarella, J. A Fast Octree-Based Algorithm for Computing Ropelength, Department of Mathematics, University of Georgia, 2003 (täiendatud versioon 2005).

[6] Jackins, C.L., Tanimoto, S.L. Oct-trees and their use in representing three-dimensional objects, Computer Graphics and Image Processing, vol 14, pp 249–270, 1980.

[7] Greene, N., Detecting intersection of a rectangular solid and a convex polyhedron, Graphics Gems IV, Academic Press, pp 74-82, 1994.

[8] Vörös, J. A strategy for repetitive neighbor finding in octree representations, Image and Vision Computing, vol. 18 , pp 1085–1091, 2000.

[9] Woo, A. Fast Ray-Box Intersection, Graphics Gems, Academic Press, pp 395-396, 1990.