Seminarska naloga

POVEZANE POTI V QCA STRUKTURAH

FRI UNI, 4. letnik
Optične in nanotehnologije

Matic Tovšak
Gašper Žejn
Tine Kavčič
Jernej Gorički

Datum: 13.1.2009

1.) Uvod

Pri 2. seminarski nalogi pri predmetu Optične in nanotehnologije smo se ukvarjali s povezanimi potmi v QCA strukturah. Cilj je bila uporaba genetskih algoritmov za realiziranje čimbolj optimalnih povezav med točkami v strukturi. Seminarska naloga zavzema realizacijo naslednjih petih točk:

1.) Povezana pot od vhoda do izhoda
2.) Najkrajša povezana pot od vhoda do izhoda (število celic)
3.) Najkrajša povezana pot (vogalna konfiguracija)
4.) Povezana pot od n vhodov do enega izhoda
5.) Povezana pot od n vhodov do m izhodov

Uporabljen je bil naslednji model QCA struktur:

0	0	0	0	0
0	0	0	0	0
1	1	1	0	0
0	0	1	1	1
0	0	0	0	0

Polje velikosti 5×5, kjer imamo poljubno pozicijo vhodov in izhodov(določimo naključno), pri čemer so na desni izhodi, drugje pa vhodi. Polja zavzamejo vrednost 0 – ni povezave
1 – je povezava.

2.) Genetski algoritmi

Ključna stvar pri realizaciji so bili t.i. genetski algoritmi. Na osnovi genetskih algoritmov preferiramo poti brez lukenj (povezane poti) ter krajše poti.

Veliko pomoč pri realizaciji nam je nudila C++ knjižnica objektov genetskih algoritmov, imenovana GAlib (http://lancet.mit.edu/ga). Ta knjižnica vključuje orodja za uporabo genetskih algoritmov pri optimizaciji v C++ programih.

Ko rešujemo problem z genetskimi algoritmi moramo upoštevati 3 točke:

1.) Definirati predstavitev (instance genoma)
2.) Definirati genetske operatorje
3.) Definirati cenilko – funkcijo za željen rezultat (ugotovi, kako dobra je rešitev)

Pri uporabi knjižnice delamo namreč z dvema razredoma: to sta genom in genetski algoritem. Vsaka instanca genoma predstavlja eno rešitev našega problema. Genetski algoritem je objekt, ki definira, kako mora potekati evolucija. Genetski algoritem uporablja cenilko (definirano z naše strani), da ugotovi, koliko je posamezen genom primeren za preživetje. Uporablja genomske operatorje (vgrajene v genom) ter strategijo izbira/zamenjava, z namenom generiranja novih posameznikov.

Izdelava cenilke je odvisna od nas. Ko imamo enkrat instance genoma, operatorje ter nek željen rezultat, lahko uporabimo katerikoli genetski algoritem, da poiščemo boljše rešitve našega problema. V našem primeru je kot cenilka uporabljen modificiran Dijkstrov algoritem.

Objekt genetski algoritem ugotovi, kateri posamezniki lahko preživijo, kateri se lahko razmnožujejo ter kateri morajo poginiti. Prav tako beleži statistiko in se odloča, kako dolgo se lahko evolucija nadaljuje. Kdaj se naj algoritem ustavi, moramo določiti mi sami.

Potek algoritma:

1.) Inicializacija populacije

2.) Izbira posameznikov za razmnoževanje

3.) Križanje posameznikov z namenom ustvarjanja potomcev

4.) Mutacija potomcev

5.) Vstavitev novega v populacijo

6.) Ali se mora algoritem ustaviti (glede na izbran kriterij? – če ne, se vrnemo nazaj k izbiri posameznikov za razmnoževanje, sicer se algoritem konča.

2.1) Uporabljene strukture

V naši kodi smo uporabili genetski algoritem **GASimpleGA**, t.i "Simple genetic algorithm" oziroma preprosti genetski algoritem. Pri kreaciji tega algoritma moramo specificirati bodisi posameznika, bodisi populacijo posameznikov. Nov genetski algoritem bo kloniral našega specificiranega posameznika oziroma posameznike in naredil svojo populacijo. Ta algoritem bo v vsaki generaciji kreiral popolnoma novo populacijo posameznikov in sicer tako, da bo izbral posameznike prejšnje populacije, jih med seboj "razmnožil" ter ustvaril nov podmladek za novo generacijo. Najboljši posameznik iz določene generacije bo prenesen v naslednjo generacijo. Ta proces se nadaljuje, dokler niso izpolnjeni pogoji za ustavitev.

Ko uporabljamo genetske algoritme pri reševanju optimizacijskih problemov, moramo znati predstaviti eno rešitev našega problema v eni podatkovni strukturi. Genetski algoritem bo kreiral populacijo rešitev, temelječih na vzorčni podatkovni strukturi, ki jo določimo. Genetski algoritem nato izvaja operacije na populaciji, da dobi najboljšo rešitev. V GAlib knjižnici se vzorčni podatkovni strukturi reče GAGenome (kromosom). Knjižnica vključuje štiri tipe genomov: GAListGenome, GATreeGenome, GAArrayGenome ter GABinaryStringGenome. Ti razredi so izpeljani iz osnovnega razreda GAGenome ter iz razreda določene podatkovne strukture.
V naši kodi smo uporabili **GA2DbinaryStringGenome** in **GA3DbinaryStringGenome,** ki sta izpeljana iz razredov GABinaryString ter GAGenome. Dejansko gre za matrike, ki jih sestavljata števili 1 in 0. Širina in višina matrike sta lahko fiksni ali pa spremenljivi. Naša uporabljena velikost je 5×5. Geni v teh genomih so biti.

Implementacija cenilke s pomočjo Dijkstrovega algoritma se izkaže za uspešno, vendar je precej časovno zahtevna $O(n^2)$, kjer je n število celic. Implementacija prvih treh tipov povezav je razmeroma preprosta, pri zadnjih dveh točkah pa se pojavijo težave zaradi odvisnosti od velikega števila parametrov.

3.1) Najkrajša povezana pot od vhoda do izhoda (število celic)

Tabele prikazujejo izhod našega programa pri večkrat zagnani konfiguraciji
vhod = (0,1), izhod = (4,3)

0	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	1	1	1	1
0	0	0	0	0

0	0	0	0	0
1	1	0	0	0
0	0	1	1	0
0	0	0	0	1
0	0	0	0	0

0	0	0	0	0
1	0	0	0	0
0	1	1	1	0
0	0	0	0	1
0	0	0	0	0

0	0	0	0	0
1	0	0	0	0
0	1	1	0	0
0	0	0	1	1
0	0	0	0	0

3.2) Najkrajša povezana pot (vogalna konfiguracija – pri spremembi snovi je potreben tudi vogalni element)

Tabele prikazujejo izhod našega programa pri večkrat zagnani konfiguraciji
vhod = (0,1), izhod = (4,3)

0	0	0	0	0
1	1	1	1	0
0	0	0	1	0
0	0	0	1	1
0	0	0	0	0

1	1	1	1	1
1	0	0	0	1
0	0	0	0	1
0	0	0	0	1
0	0	0	0	0

0	0	0	0	0
1	1	1	1	0
0	0	0	1	0
0	0	0	1	1
0	0	0	0	0

0	0	0	0	0
1	1	0	0	0
0	1	0	0	0
0	1	1	1	1
0	0	0	0	0

3.2) Najkrajša povezana pot od n vhodov do enega izhoda

Tabele prikazujejo izhod našega programa pri večkrat zagnani konfiguraciji
vhod = (0,0) (0,4) izhod = (4,2)

1	0	0	0	0
1	1	0	0	0
0	1	1	1	1
0	1	0	0	0
1	0	0	0	0

1	0	0	0	0
1	1	1	0	0
0	0	1	1	1
0	1	1	0	0
1	0	0	0	0

1	0	0	0	0
1	1	1	1	0
0	0	0	1	0
0	1	1	1	1
1	0	0	0	0

1	0	0	0	0
1	0	0	0	0
1	1	1	1	1
0	1	0	0	0
1	0	0	0	0

3.3) Najkrajša povezana pot od n vhodov do m izhodov

Tabele prikazujejo izhod našega programa pri večkrat zagnani konfiguraciji
Linija a = vhod (2,0),(3,0) izhod (4,0)
Linija b = vhod(0,1), (0,4) izhod (4,3)

0	0	a	a	a
b	0	0	0	0
b	0	0	0	0
b	b	b	b	b
b	0	0	0	0

0	0	a	a	a
b	0	0	0	0
b	b	0	0	0
0	b	b	b	b
b	0	0	0	0

0	0	a	a	a
b	0	0	0	0
b	b	b	0	0
0	0	b	b	b
b	b	0	0	0

0	0	a	a	a
b	b	0	0	0
0	b	0	0	0
0	b	b	b	b
b	0	0	0	0

4.) Zaključek

Iz dobljenih rezultatov je razvidno, da je cenilka razmeroma uspešna. Implementacija z Dijkstro se izkaže za zelo uspešno pri prvih treh tipih povezav, pri povezavi n vhodov z enim izhodom se pojavijo problemi zaradi večjega števila parametrov, ki določajo nagrado in kazen glede na celico, posebej pa se to pokaže pri povzavi n vhodov na m izhodov. Za parametre, ki smo jih nastavili, nimamo teoretičnega dokaza, ampak so bili nastavljeni s poskušanjem.

4.1) Viri

http://lancet.mit.edu/ga/
http://lancet.mit.edu/ga/dist/galibdoc.pdf