Elektrotehniški vestnik 69(1): 7–12, 2002
Electrotechnical Review, Ljubljana, Slovenija

Optimizacija elektronskih vezij z vzporednim omejenim simpleksnim postopkom

Arpad Bűrmen, Iztok Fajfar, Janez Puhan, Andrej Nussdorfer, Tadej Tuma

Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenia
E-pošta: arpadb@fides.fe.uni-lj.si

Povzetek. Optimizacija vezij je postopek, ki se ga načrtovalci vezij izogibajo predvsem zaradi njegove računske zahtevnosti. Težavi se da izogniti, če uporabimo vzporedne postopke za optimizacijo. Članek obravnava vzporedno enačico omejenega simpleksnega postopka, prirejeno za uporabo na skupinah računalnikov, povezanih v lokalno omrežje. Taka strojna oprema je na voljo v vsakem laboratoriju, ki se ukvarja z računalniškim načrtovanjem (integriranih) vezij, saj lahko uporabimo vse računalnike, ki so opremljeni z vmesnikom za mrežo in operacijskim sistemom, ki podpira protokol TCP/IP (UNIX, Windows). Postopek je preiskušen na naboru testnih funkcij in optimizacijskih problemov s področja načrtovanja analognih vezij. Prikazan je vpliv zakasnitve pri prenosu podatkov in neenakomerne porazdelitve dela med računalniki na faktor pospešitve. Podane so smernice za nadaljnji razvoj postopka.

Ključne besede: omejena optimizacija, elektronska vezja, računalniško podprto načrtovanje, vzporedno računanje

Constrained Optimisation of Electronic Circuits by Means of Parallel Simplex Algorithm

Extended abstract. Probably the largest obstacle to overcome when optimisation is applied to real-world circuits is the long time taken to complete an optimisation run. When a particular optimisation algorithm is considered, there is only one alternative to using a faster computer - to use parallel computing.

One of the algorithms that have given good results in conjunction with the SPICE circuit simulator is the constrained simplex algorithm. The main goal of the research described in this paper was to develop a parallel implementation of the constrained simplex algorithm that can be efficiently run on a group of workstations connected by a local area network (LAN). The biggest advantage of the approach is that such networks are available in every IC design group.

The major drawback of such approach is the relatively long response time of the network when compared to response time of a system bus in multiprocessor computers and other dedicated hardware. This fact requires the amount of network traffic to be kept as low as possible. Another requirement that the parallel algorithm must fulfil is to distribute the load uniformly among workstations so that the time a workstation spends waiting on a new task is minimal.

An algorithm that satisfies all of the mentioned requirements is devised. The performance of the proposed algorithm is evaluated on test functions (Equations (8) and (9), Figs. 1 and 2) and real-world circuit design problems (Tables 1-5). The results are compared to the results of the original algorithm (see 1st column of Tables 2 and 3). The acceleration factor is determined (Table 5). The effect of the overhead caused by the slow response time of the LAN is illustrated by the results obtained for the two analytical test functions (Tables 1-5). The uneven distribution of the computational load causes some computers to wait for an excessive amount of time before they are allowed to continue. The latter results in a lower acceleration factor (Table 6). Further research efforts are to be focused at devising an algorithm that guarantees uniform distribution of the computational load.

Prejet 3. marec, 2001
Odobren 12. november, 2001

Key words: constrained optimisation, electronic circuits, CAD, parallel computing

1 Uvod

Z rastjo računske moči osebnih računalnikov se je simulacija vezij razvila iz pripomočka, ki ga je bilo mogoče uporabljati le z najzmogljivejšimi delovnimi postajami, v splošno uporabljano orodje.

Optimizacija vezij je še korak naprej v tem razvoju. Ker zahteva veliko računanja kriterijske funkcije in posledično tudi veliko simulacij vezja, je računsko zelo zahtevna. Nekakšno optimizacijo pri svojem delu izvaja takorekoč vsak načrtovalec integriranih vezij. Običajno je to spreminjanje parametrov vezja na podlagi načrtovalskih izkušenj, ki jih načrtovalec pridobi pri svojem delu.

Avtomatizirani postopki se redko uporabljajo. Deloma gre vzrok iskati v pomanjkanju učinkovitih orodij za optimizacijo. Z razvojem načrtovalskega orodja SPICE OPUS [1,2,3] smo to oviro vsaj delno odpravili. Drugi razlog pa je seveda sama računska zahtevnost postopka, ki močno omejuje družino problemov, rešljivih v nekem doglednem času z orodjem SPICE OPUS.

Eden izmed načinov, da se slednja ovira odstrani, je izboljšanje postopkov simulacije in optimizacije v smeri večje računske učinkovitosti. V preteklosti se je veliko raziskovalcev ukvarjalo s to temo predvsem na področju analize Monte Carlo (izčrpen pregled je na voljo v [4]) in iskanja optimalnih vrednosti parametrov ob upoštevanju toleranc elementov [5].

Drugi pristop je uporaba vzporednih postopkov. Tako se večji problem rešuje na večih računalnikih hkrati, kar skrajša čas računanja sorazmerno s številom računalnikov.

Ta pristop je bil že večkrat uporabljen pri globalnih optimizacijskih postopkih, kot so genetski algoritmi ali simulirano ohlajanje, ki že po svoji naravi zahtevajo veliko število računanj kriterijske funkcije.

Veliko manj prizadevanj v tej smeri je bilo na področju direktnih optimizacijskih metod [6], ki so zaradi težavnega računanja občutljivosti karakteristik vezja na parametre trenutno edina resna alternativa za splošno optimizacijo vezij. V družino direktnih optimizacijskih metod spadajo tudi simpleksne metode.

Čeprav so bile simpleksne optimizacijske metode [7,8,9] razvite v 60. letih, se še zmeraj uporabljajo za premagovanje optimizacijskih problemov. Razloge za njihovo priljubljenost gre iskati v njihovi enostavnosti in dejstvu, da ne zahtevajno računanja odvodov kriterijske funkcije.

V prispevku se osredotočimo na vzporedno izvedbo omejenega simpleksnega postopka (OSP), prilagojeno za računalnike, povezane v lokalno mrežo (tipično 10/100Mbit Ethernet). Na kratko podamo sorodne raziskave na tem področju. Opišemo omejen vzporeden simpleksni postopek. Predstavimo testne probleme in rezulate, dobljene z uporabo vzporednega postopka. Na koncu podamo ugotovitve, ki sledijo iz rezultatov in smernice za nadaljnje delo.

2 Sorodne raziskave

Omejen simpleksni postopek (OSP) [9] je razširitev simpleksnega postopka, ki sta ga objavila Nelder in Mead (NMSP) [8]. Oba postopka sta namenjena iskanju minimuma nelinearne kriterijske funkcije v N-dimenzionalnem prostoru. NMSP običajno manipulira s simpleksom N+1 ali več točk. Na splošno večje število točk pomeni učinkovitejše preiskovanje prostora in več računanj kriterijske funkcije.

Vzporedna enačica NMSP je bila predsavljena v [10]. Pri tej metodi so bile operacije izvajane vzporedno na več najslabših točkah hkrati. Rezultati so bili različni. Vzporeden postopek je v nekaterih primerih deloval bolje, prikazani pa so bili tudi primeri, kjer se je vzporedni postopek obnašal slabše kot prvotna enačica NMSP.

V [11] je bila uporabljena nekoliko spremenjena enačica NMSP, izvedena na vzporednem računalniku. Ker se je raziskava osredotočala na učinkovitost preiskovanja prostora, podatkov o pospešitvi v članku ni. Pokazano je bilo, da vzporedni postopek v manj računanjih kriterijske funkcije pride do enake vrednosti le-te kot prvotni postopek.

V literaturi nismo zasledili vzporedne enačice OSP. OSP je preprostejši kot NMSP, saj se poslužuje le zrcaljenja in primikanja. Da je vzorčenje prostora, razsekanega z implicitnimi omejitvami, učinkovito, OSP dela z nekoliko večjim simpleksom (običajno 2N točk). Tudi pri OSP število računanj kriterijske fukcije in učinkovitost iskanja raste s številom točk v simpleksu.

Vzporedna enačica simpleksnega postopka je bila preiskušena s programskim paketom SPICE OPUS, ki je na voljo za opreacijske sisteme Windows 95/98/NT, LINUX in SOLARIS. Več informacij je na voljo na strani http://fides.fe.uni-lj.si/spice.

3 Opis postopka

Omejeni simpleksni postopek je namenjen iskanju minimuma nelinearne kriterijske funkcije (KF) N spremenljivk:

$$\mathbf{T} = [x_1, x_2, ..., x_N]$$

$$\mathbf{E}(\mathbf{T}) = \mathbf{E}(x_1, x_2, ..., x_N) \tag{1}$$

Rešitev mora zadoščati M implicitnim mejitvam:

$$\mathbf{f}_i(x_1, x_2, ..., x_N) \geq 0, \quad i = 1, ..., M \tag{2}$$

Postopek se začne z inicializacijo simpleksa:

1. Podana začetna točka ($\mathbf{T}_{zacetna}$) mora zadostiti vsem implicitnim omejitvam.
2. Naključno izberi $k - 1 \geq N$ točk. Če katera od točk ne izpolnjuje vseh implicitnih omejitev, izvajaj primikanje k začetni točki, dokler niso le te izpolnjene:

$$\mathbf{T}^{i+1}_{izbrana} = \frac{1}{2}(\mathbf{T}^{i}_{izbrana} + \mathbf{T}_{zacetna}) \tag{3}$$

Nato postopek ponavlja naslednje korake:

1. Uredi točke po velikosti KF, tako da velja:

$$\mathbf{E}(\mathbf{T}_1) \geq \mathbf{E}(\mathbf{T}_2) \geq ... \geq \mathbf{E}(\mathbf{T}_k)$$

2. Izračunaj centroid ($\mathbf{T}_c$) točk $\mathbf{T}_2, \mathbf{T}_3, ..., \mathbf{T}_k$:

$$\mathbf{T}_c = \frac{1}{k-1}\sum_{i=2}^{k}\mathbf{T}_i \tag{4}$$

3. Zrcali najslabšo točko ($\mathbf{T}_1$) čez centroid s faktorjem $\alpha > 1$:

$$\tilde{\mathbf{T}}^1_1 = (1 - \alpha)\mathbf{T}_1 + \alpha\mathbf{T}_c \tag{5}$$

4. Dokler $\tilde{\mathbf{T}}_1^i$
   - krši (2) ali
   - $\mathbf{E}(\tilde{\mathbf{T}}_1^i) \geq \mathbf{E}(\mathbf{T}_1)$,

   jo primikaj k centroidu:

$$\tilde{\mathbf{T}}_1^{i+1} = \frac{1}{2}(\tilde{\mathbf{T}}_1^i + \mathbf{T}_c) \tag{6}$$

   Primikanje je omejeno bodisi s številom korakov (6) ali pa vstopom točke $\tilde{\mathbf{T}}_1^{i+1}$ v dano okolico $\mathbf{T}_c$. Če se to zgodi, začnemo primikati $\tilde{\mathbf{T}}_1^1$ proti najboljši točki ($\mathbf{T}_k$):

$$\tilde{\mathbf{T}}_1^{i+1} = \frac{1}{2}(\tilde{\mathbf{T}}_1^i + \mathbf{T}_k). \tag{7}$$

   Tudi to primikanje je omejeno na enak način kot primikanje k centroidu. Če tudi s primikanjem (7) ne najdemo točke, ki je boljša od $\mathbf{T}_1$ in izpolnjuje implicitne omejitve (2), postavimo v simpleks namesto točke $\mathbf{T}_1$ najboljšo točko ($\mathbf{T}_k$).

5. Korake 1-4 ponavljamo, dokler ni izpolnjen ustavitveni kriterij. To je lahko bodisi tedaj, ko vrednost KF v $\mathbf{T}_k$ pade pod neko dano vrednost, ali pa ko srednja razdalja točk simpleksa od centroida postane manjša od neke dane razdalje.

Iz samega opisa postopka je razvidno, da je za vsak prehod skozi korake 1-4 potreben vsaj en izračun kriterijske funkcije.

Postopek lahko paraleliziramo tako, da hkrati zrcalimo in primikamo m najslabših točk. Sedaj lahko za vsako od teh m točk korake 2-4 izvaja en procesor neodvisno od drugih. Paraleliziramo lahko tudi inicializacijo postopka, in sicer tako, da hkrati izberemo in ovrednotimo več začetnih točk simpleksa. Postopek zahteva torej m procesorjev (računalnikov). Pri tem računamo centroid iz $k - m$ najboljših točk.

4 Izvedba

Pri izvedbi smo se odločili, da bo en proces (gospodar) razdeljeval delo preostalim (delavcem) in zbiral rezultate. Ker to opravilo samo po sebi računsko ni zahtevno, na istem računalniku poganjamo tudi en delavski proces, tako kot na vseh drugih. V vsaki iteraciji dobi delavec centroid, najboljšo točko in eno od m najslabših točk.

Ko delavec konča niz korakov 1-4, vrne gospodarju kot rezultat novo točko in vrednost KF v tej točki. Gospodar zbere rezultate od vseh delavcev in odloči, ali je postopek treba nadaljevati oziroma ga prekine na podlagi ustavitvenega kriterija v 5. koraku postopka.

Komunikacija med računalniki poteka prek lokalne mreže in protokola TCP/IP. Program SPICE OPUS je bil spremenjen tako, da ponuja dva načina delovanja: gospodar in delavec.

Delavce povežemo z gospodarjem tako, da v oknu gopodarja vnesemo ukaz 'drone'. Gospodar se pri tem poveže z nadzornim programom na ustreznem računalniku in zahteva novega delavca. Nadzorni program teče na vseh računalnikih v mreži in na zahtevo gospodarja zažene delavca, ki nato vzpostavi povezavo z gospodarjem, prek katere bo sprejemal ukaze in vračal rezultate.

V načinu gospodarja deluje program kot običajno, le da vnešene ukaze izvedejo tudi vsi delavci, ki so povezani z gospodarjem. Tako dosežemo pred optimizacijo pri vseh delavcih enako stanje vezja, kot je pri gospodarju. Ko gospodar prejme ukaz 'optimize' in ga razdeli delavcem, ti začnejo sprejemati točke od gospodarja in opravljati operacije zrcaljenja in primikanja z njimi.

Za vsako točko, ki jo gospodar dodeli enemu od delavcev, prejme rezultat. Ko prejme rezultate od vseh delavcev, preveri, ali je izpolnjen ustavitveni kriterij. Če je, potem o tem obvesti delavce in jim pošlje končni rezultat. Tako dosežemo, da so na koncu optimizacijskega teka delavci v istem stanju kot gospodar.

Na zgoraj opisani način dosežemo, da lahko na zelo preprost način zaženemo vzporedno optimizacijo na problemih, ki smo jih prej reševali z navadnim OSP. Vse, kar je potrebno spremeniti v datotekah z vezji in ukazi za simulator, je:

1. v oknu gospodarja izvedemo niz ukazov 'drone', ki zaženejo delavce in jih povežejo z gospodarjem,
2. za optimizacijski postopek pri ukazu 'optimize method...' navedemo vzporedni OSP.

5 Rezultati

Postopek smo preiskusili na dveh analitično podanih funkcijah: kvadratni funkciji in posplošeni Rosenbrockovi funkciji. Opazovali smo najnižjo doseženo vrednost KF po nekem danem številu računanj le-te.

Druga polovica testov je bila izvedena v skupini štirih enakih računalnikov. Merili smo čas optimizacije in število računanj kriterijske funkcije.

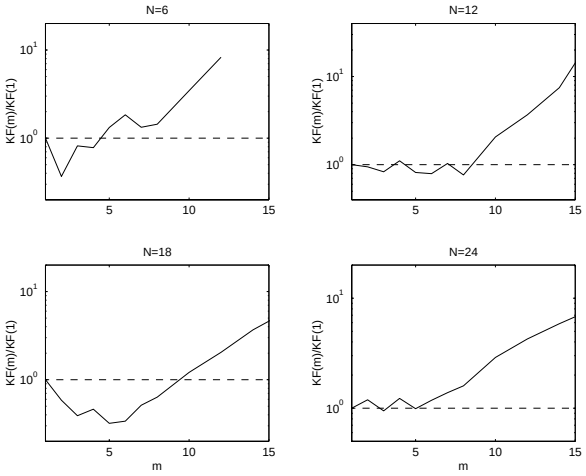
5.1 Konvergenčne lastnosti

Prva testna funkcija, na kateri smo preiskusili postopek, je bila kvadratna funkcija v N-dimenzionalnem prostoru:

$$\mathbf{E}(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} x_i^2 \tag{8}$$

Z njeno pomočjo smo poskušali zajeti konvergenčne lastnosti v okolici minimuma splošne KF. Postopek smo ustavili po 1000 računanjih KF in opazovali najnižjo doseženo vrednost v odvisnosti od števila zrcaljenih točk

(m). Simpleks je pri tem obsegal $k = N + m$ točk. Ker OSP začne z naključno izbranim simpleksom iz množice $[-10, 10]^k$, smo povprečili rezultate 30 tekov. Povprečili smo desetiški logaritem vrednosti kriterijske funkcije, ker se dosežena vrednost lahko giblje v območju več dekad.



Slika 1. Razmerje med doseženo vrednostjo KF za m hkrati zrcaljenih točk in 1 zrcaljeno točko za kvadratno funkcijo (8)
Figure 1. Ratio of the resulting cost functon (CF) for m simultaneously mirrored points and 1 mirrored point for the quadratic function (8).

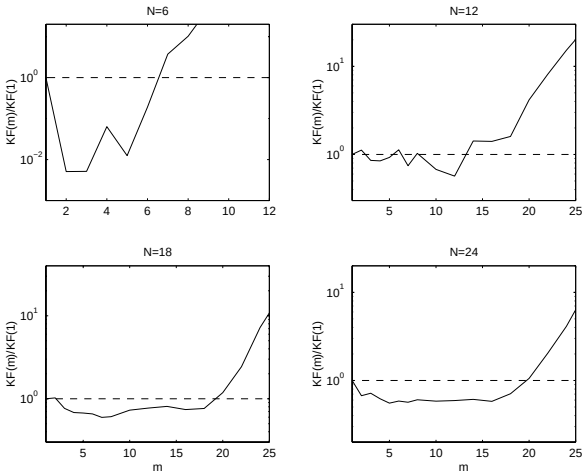
Druga testna funkcija, ki smo jo preiskusili, je posplošena Rosenbrockova funkcija, ki je definirana za $N = 2n$ dimenzionalen prostor:

$$\mathbf{E}(x_1, x_2, ..., x_N) = \sum_{i=1}^{n} (100(x_{2i} - x_{2i-1}^2)^2 + (1 - x_{2i-1}^2) \quad (9)$$

Rosenbrockova funkcija je zaradi dolge ozke ukrivljene doline zelo težavna za vse optimizacijske postopke. Postopek smo ustavili po 3000 računanjih KF, začetni simpleks pa je bil izbran iz množice $[-9, 11]^k$.

Kot je razvidno iz slike 1, postopek za določene vrednosti m doseže enako vrednost kriterijske funkcije pri več hkrati zrcaljenih točkah. Ker lahko m zrcaljenj razdelimo med m računalnikov, lahko pričakujemo pospešitev za tiste vrednosti m, kjer je dosežena vrednost KF manjša ali pa v istem razredu velikosti kot za $m = 1$. To je za vse vrednosti m, pri katerih ležijo krivulje na sliki 1 pod ali v bližini črtkane črte.

Ker je v nekaterih primerih dosežena vrednost kriterijske funkcije celo močno pod doseženo vrednostjo za $m = 1$, lahko pričakujemo v teh primerih pospešitve, ki so večje od števila vzporedno računajočih računalnikov. V nekaterih primerih lahko torej pričakujemo, da bo vzporedni OSP celo učinkovitejši od prvotnega OSP (glede števila računanj kriterijske funkcije med optimizacijskim tekom).



Slika 2. Razmerje med doseženo vrednostjo KF za m hkrati zrcaljenih točk in 1 zrcaljeno točko za posplošeno Rosenbrockovo funkcijo (9)
Figure 2. Ratio of the resulting cost functon (CF) for m simultaneously mirrored points and 1 mirrored point for the generalized Rosenbrock function.

Slika 2, ki prikazuje obnašanje postopka za posplošeno Rosenbrockovo funkcijo (9), potrjuje ugotovitve s slike 1.

5.2 Testni primeri

Postopek smo testirali na dveh testnih funkcijah - Rosenbrockovi in Woodsovi. Poleg testnih funkcij smo opravili preiskuse tudi na štirih elektrotehniških problemih: načrtovanju Schmittovega prožilnika, načrtovanju pretvornika trikot-sinus, optimizaciji CMOS ojačevalnika in robustnem načrtovanju nizkoprepustnega filtra. Vsi elektrotehniški problemi z izjemo zadnjega so opisani v [12].

Zadnji problem je nov in temelji na primeru optimizacije NP filtra petega reda. Tokrat načrtujemo NP filter ob predpostavki, da imajo vse kapacitivnosti 2% tolerance. Cilj je določiti nazivne vrednosti kapacitivnosti tako, da bo valovitost pod 2dB, dušenje v zapornem pasu pa nad 70dB. Razen načrtovalskih zahtev in zelo širokih eksplicitnih omejitev za vrednosti kapacitivnosti optimizacijski postopek ne dobi nobene druge izhodiščne informacije. Kriterijska funkcija se računa na podlagi analize 32 vogalnih točk vezja in nazivne točke. Če katera od karakteristik vezja pade zunaj načrtovalskih zahtev, se k vrednosti kriterijske funkcije prišteje ustrezen kazenski prispevek, ki je sorazmeren prekoračitvi. Cilj optimizacije je najti take nazivne vrednosti parametrov, da je vrednost kriterijske funkcije enaka 0.

Postopek smo ustavili, ko je vrednost kriterijske funkcije (KF) padla pod vrednost, podano v tabeli 1, oziroma ko je relativna velikost simpleksa (SS) padla pod vrednost, podano v tabeli. Rezultati so povprečje 10

tekov, pri zadnjem primeru pa 5 tekov. Število zrcaljenih točk je bilo enako številu računalnikov, ki so računali vzporedno. Vsi računalniki so bili enaki in povezani v omrežje 10Mbit Ethernet.

	N	M	k	Ustavitev
Rosenbrock	6	0	$6+m$	$KF < 10^{-9}$
Woods	4	0	$4+m$	$KF < 10^{-9}$
Schmitt	2	1	$2+m$	$KF < 10^{-4}$
Trikotnik	12	0	16	$SS < 0.005$
CMOS oj.	11	3	15	$SS < 0.1$
NP filter	5	0	10	$KF < 10^{-9}$ ali $SS < 0.01$

Tabela 1. Testni primeri
Table 1. Test cases.

Rezultati so podani v tabelah 2-5. Iz tabele 2 je razvidno, da število računanj kriterijske funkcije nekaj časa ne raste s številom računalnikov (vzporednih zrcaljenj). V tem območju lahko torej pričakujemo pospešitev.

Št. rač.	1	2	3	4
Rosenbrock	3293	2479	2454	6445
Woods	823	815	1010	1278
Schmitt	136	173	300	449
Trikotnik	3097	2804	3072	3325
CMOS oj.	1634	1557	1391	1858
NP filter	245	354	386	227

Tabela 2. Povprečno število računanj kriterijske funkcije
Table 2. Mean number of cost function evaluations.

Št. rač.	1	2	3	4
Rosenbrock	18.58	12.85	9.50	18.49
Woods	4.21	3.71	3.52	4.13
Schmitt	22.67	17.64	23.05	24.51
Trikotnik	46.69	31.89	27.02	24.51
CMOS oj.	290.01	173.25	112.54	120.59
NP filter	713.40	590.67	466.30	209.70

Tabela 3. Povprečen čas optimizacije
Table 3. Mean optimisation time.

Tabela 4 podaja povprečen čas računanja kriterijske funkcije, ki ga dobimo iz podatkov o številu računanj kriterijske funkcije in trajanju optimizacije za en računalnik.

Omeniti velja, da je pri zadnjem primeru za en in dva računalnika program našel rešitev v štiri od petih tekov, za tri in štiri računalnike pa v vseh tekih.

Pospešitev glede na trajanje optimizacije pri uporabi enega samega računalnika je podana v tabeli 5. Iz primerjave tabel 2, 4 in 5 je razvidno, da lahko zadovoljivo pospešitev pričakujemo le za primere, kjer število računanj kriterijske funkcije ne raste bistveno s številom računalnikov, trajanje enega izračuna kriterijske funkcije pa je veliko v primerjavi z zakasnitvijo pri prenosu podatkov po mreži (10ms). Prav tako je razvidno, da od nekega števila računalnikov naprej (odvisno od primera) pospešitev ne raste več oziroma začne padati zaradi naraščanja števila izračunov kriterijske funkcije, kar je bilo ugotovljeno tudi na podlagi grafov na slikah 1 in 2.

	$t_{CF}[\mathrm{ms}]$
Rosenbrock	5.6
Woods	5.1
Schmitt	167
Trikotnik	15
CMOS oj.	177
NP filter	2910

Tabela 4. Povprečno trajanje izračuna kriterijske funkcije
Table 4. Mean cost function evaluation time.

Št. rač.	2	3	4
Rosenbrock	1.45	1.96	1.00
Woods	1.13	1.20	1.02
Schmitt	1.29	0.98	0.92
Trikotnik	1.46	1.73	1.90
CMOS oj.	1.67	2.58	2.40
NP filter	1.21	1.53	3.40

Tabela 5. Povprečen faktor pospešitve
Table 5. Mean acceleration factor.

5.3 Vpliv neenakomerne porazdelitve računanja KF med računalniki

Ker vsi računalniki ne opravijo enakega števila primikanj zrcaljene točke, je tudi število računanj KF različno od računalnika do računalnika. Tisti računalniki, ki končajo prej, morajo na novo točko za zrcaljenje čakati. Zato so faktorji pospešitve manjši, kot bi pričakovali zgolj na podlagi števila računanj KF.

To se lepo vidi, če si ogledamo zadnji primer, kjer je zakasnitev pri prenosu po mreži zanemarljiva v primerjavi s trajanjem enega računanja KF. Če poznamo odvisnost števila računanj KF ($N(m)$) od števila hkrati zrcaljenih točk oziroma računalnikov (m), lahko pričakovano pospešitev izračunamo kot:

$$FP(m) = mN(1)/N(m) \qquad (10)$$

Le-ta je namreč sorazmerna razmerju števila računanj KF pri enem računalniku in m računalnikih ter samemu številu računalnikov. Dosegli bi jo lahko, če bi delo enakomerno porazdelili med računalnike. Primerjavo pričakovane in dobljene pospešitve za zadnji primer ilustrira tabela 6.

Št. rač.	2	3	4
Pričakovano	1.38	1.91	4.33
Dobljeno	1.21	1.53	3.40

Tabela 6. Pričakovan in dobljen faktor pospešitve
Table 6. Expected and obtained acceleration factor.

6 Sklep

Prikazana je bila vzporedna enačica omejenega simpleksnega postopka. Postopek je bil preiskušen na testnih funkcijah in elektrotehniških primerih. Pri uporabi več kot enega računalnika v optimizacijskem postopku je bila dobljena pospešitev, ki je bila sorazmerna številu računalnikov. Kot je razvidno iz rezultatov, poljubna pospešitev ni mogoča. Le-ta je na splošno odvisna od samega problema. Postopek je še zlasti učinkovit pri primerih, kjer je računanje kriterijske funkcije dolgotrajno.

Glavna težava, ki zmanjšuje učinkovitost postopka, je neenakomerna porazdelitev računanja med računalnike. Ta bi še posebej prišla do izraza tedaj, ko bi bili uporabljeni različno zmogljivi računalniki. V tem primeru bi se zgodilo, da bi bili nekateri računalniki zelo slabo izkoriščeni. Ta pomanjkljivost je tudi glavno izhodišče za nadaljnje delo. Končni cilj je razviti optimizacijski postopek, ki bi učinkovito izrabil procesorski čas vseh računalnikov v omrežju in obenem obdržal čim več dobrih lastnosti simpleksnega postopka.

7 Literatura

[1] J. Puhan, T. Tuma, Optimization of analog circuits with SPICE 3f4, *Proceedings of the ECCTD'97*, vol. 1, pp. 177-180, 1997.

[2] J. Puhan, T.Tuma, I. Fajfar, Optimisation Methods in SPICE, a Comparison, *Proceedings of the ECCTD'99*, vol. 1, pp. 1279-1282, 1999.

[3] T. Quarles, A. R. Newton, D. O. Pederson, A. Sangiovanni-Vincentelli, *SPICE3 Version 3f4 User's Manual*, Berkeley, University of California, 1989.

[4] M. Keramat, R. Kielbasa, A Study of Stratified Sampling in Variance Reduction Techniques for Parametric Yield Estimation, *IEEE Transactions on Circuits and Systems II*, vol. 45, no. 5, pp. 575-583, 1998.

[5] *IEE Proceedings G (Electronic Circuits and Systems)*, vol.129, no.4, 1982.

[6] R. M. Lewis, V. Torczon, M. W. Trosset, Direct search methods: then and now, *Journal of Computational and Applied Mathematics*, vol.124, no.1-2, pp.191-207, 2000.

[7] W. Spendley, G. R. Hext, F. R. Himsworth, Sequential Applications of Simplex Designs in Optimisation and Evolutionary Operation, *Technometrics*, vol. 4, no. 4, pp. 441-461, 1962.

[8] J. A. Nelder, R. Mead, A Simplex Method for Function Minimization, *Computer Journal*, vol. 7, no. 4, pp. 308-313, 1965.

[9] M. J. Box, A New Method of Constrained Optimization and a Comparison with Other Methods, *Computer Journal*, vol. 7, pp. 42-52, 1965.

[10] J. E. Dennis, Jr., V. Torczon, Parallel Implementations of the Nelder-Mead Simplex Algorithm for Unconstrained Optimization, *Proceedings of the SPIE*, Vol. 880, pp. 187-191, 1988.

[11] L. Coetzee, E. C. Botha, The Parallel Downhill Simplex Algorithm for Unconstrained Optimisation, *Concurrency: Practice and Experience*, vol. 10, no. 2, pp. 121-137, 1998.

[12] J. Puhan, *Optimizacija vezij v programskem okolju SPICE*, doktorsko delo, Univerza v Ljubljani, Fakulteta za elektrotehniko, 2000.

Arpad Bűrmen je diplomiral leta 1999 na Fakulteti za elektrotehniko Univerze v Ljubljani, kjer je od leta 1999 zaposlen kot mladi raziskovalec. Ukvarja se z analogno in mešano simulacijo, modeliranjem in optimizacijo vezij in sistemov.

Iztok Fajfar je diplomiral leta 1991, magistriral leta 1994 in doktoriral leta 1997 na Fakulteti za elektrotehniko Univerze v Ljubljani. Od leta 1992 je na tej fakulteti tudi zaposlen. Trenutno je izvoljen v naziv docent in v prvem letniku poučuje računalništvo. Raziskovalno se ukvarja z načrtovanjem in optimizacijo elektronskih vezij, sodeluje pa tudi pri razvoju programskega orodja SPICE OPUS v Laboratoriju za elektroniko.

Janez Puhan je diplomiral leta 1993, magistriral leta 1998 in doktoriral leta 2000 na Fakulteti za elektrotehniko Univerze v Ljubljani. Trenutno je izvoljen v naziv asistent in vodi vaje iz več predmetov v prvem letniku in na študijski smeri Elektronika. Raziskovalno se ukvarja z računalniškim načrtovanjem analognih vezij, optimizacijskimi metodami in računalniško analizo vezij.

Andrej Nussdorfer je leta 2000 diplomiral na Fakulteti za elektrotehniko Univerze v Ljubljani. Področje njegovega raziskovalnega dela obsega mikrokrmilniške razvojne sisteme, ukvarja pa se tudi s splošnimi mikrokrmilniškimi aplikacijami. Svoje raziskovalno delo opravlja v Laboratoriju za elektroniko 2 Fakultete za elektrotehnko Univerze v Ljubljani.

Tadej Tuma se je rodil 11.4.1964 v Ljubljani. V šolskem letu 1984/85 se je vpisal na visokošolski študij Fakultete za elektrotehniko v Ljubljani in diplomiral 1988 na smeri Industrijska elektronika. Prvo službo je nastopil pri svojem štipenditorju, Iskri Avtomatiki, na Razvojnem inštitutu. Jeseni 1989 se je kot asistent stažist zaposlil na Fakulteti za elektrotehniko in računalništvo v Ljubljani in vpisal podiplomski študij. Zdaj kot docent predava več predmetov na študijski smeri Elektronika.