



UNIVERZA V MARIBORU
FAKULTETA ZA STROJNIŠTVO
2000 Maribor, Smetanova ul. 17, P.P. 224

Datum: 26. maj 2004

TEMA ZA DOKTORSKO DISERTACIJO

Senat Fakultete za strojništvo je na svoji 13. redni seji dne 25. maja 2004 v skladu s statutom Univerze v Mariboru

sprejel

temo za doktorsko disertacijo

z naslovom:

**"PROGRAMIRANJE NUMERIČNO KRMILJENIH STROJEV
Z UPORABO EVOLUCIJSKIH METOD",**

ki jo je prijavil

Miha KOVAČIČ, univ. dipl. inž. str.

Doktorandovo delo spremlja:

Mentor: **red. prof. dr. Jože BALIČ**

Komentor: **Univ. Prof. Dr. Dipl.-Ing. Branko KATALINIČ, TU Wien, Avstrija**

Rok za oddajo doktorske disertacije v zadostnem številu vezanih izvodov je 4 leta.



Dekan:
red. prof. dr. Andrej POLAJNAR

PREJMEJO:
- prof. dr. Balič
- prof. dr. Katalinič
- Miha Kovačič
- Univerza v Mariboru
- arhiv



UNIVERZA V MARIBORU
FAKULTETA ZA STROJNIŠTVO
2000 Maribor, Smetanova ul. 17, P.P. 224

Datum: 30. junij 2004

Po sklepu 14. redne seje Senata fakultete za strojništvo z dne 29. junija 2004 in v skladu s statutom Univerze v Mariboru

i m e n u j e m
KOMISIJO ZA OCENO DOKTORSKE DISERTACIJE

z naslovom:
**"PROGRAMIRANJE NUMERIČNO KRMILJENIH STROJEV
Z UPORABO EVOLUCIJSKIH METOD"**

ki jo je predložil:
Miha KOVAČIČ, univ. dipl. inž. str.

v sestavi:

- doc. dr. Miran BREZOČNIK, predsednik
- red. prof. dr. Jože BALIČ, mentor - član
- Professor. Dr. sc. Branko KATALINIČ, Technische Universität Wien, Avstrija, komentor - član
- red. prof. dr. Janez KOPAČ, Fakulteta za strojništvo, Ljubljana, član

Komisija za oceno disertacije je dolžna praviloma v roku treh mesecev, najpozneje pa v roku šestih mesecev od imenovanja, pregledati disertacijo, podati pisno poročilo o oceni disertacije in ga s svojim predlogom predložiti senatu fakultete.



DEKAN:
red. prof. dr. Andej POLAJNAR

PREJMEJO:

- doc. dr. Brezočnik
- prof. dr. Balič
- Prof. Dr. Katalinič
- prof. dr. Kopač
- Miha Kovačič
- odloga

**UNIVERZA V MARIBORU
FAKULTETA ZA STROJNIŠTVO**

Miha Kovačič

**PROGRAMIRANJE
NUMERIČNO KRMILJENIH STROJEV
Z UPORABO EVOLUCIJSKIH METOD**

Doktorska disertacija

Maribor, september 2004

UNIVERZA V MARIBORU



FAKULTETA ZA STROJNIŠTVO



Miha Kovačič

**PROGRAMIRANJE
NUMERIČNO KRMILJENIH STROJEV
Z UPORABO EVOLUCIJSKIH METOD**

Doktorska disertacija

Maribor, september 2004

Avtor: Miha Kovačič, univ. dipl. inž. str.

Naslov: Programiranje numerično krmiljenih strojev z uporabo evolucijskih metod

UDK: 621.9.06-52:004.89(043.3)

Ključne besede: NC-stroji
NC-programiranje
evolucijsko računanje
genetski algoritem

Število izvodov: 10

Tisk: FS Maribor

Predgovor

Osnovno načelo v naravi je postopno spreminjanje k navadno boljšim, popolnejšim oblikam bivanja, ki mu s tujko rečemo evolucija.

V preteklosti si je človek lajšal življenje s pripravami, ki jim je nadel ime orodje. Kasneje je izdelal prve mehanske naprave - stroje, ki so samodejno izvajale določene dejavnosti. Naprave so postajale vse bolj dovršene in tako je človek izdelal samodejne stroje za namensko uporabo pri izdelovanju in uporabi dobrin; pravimo, da je izdelal krmiljeni obdelovalni stroj.

V dobi tehnološkega razcveta si človek ne more predstavljati proizvodnega procesa brez tako imenovanih numerično krmiljenih strojev, ki so krmiljeni s pomočjo navodil, kakor pove ime, sestavljenih iz številčnih znakov. Navodila se lahko v številčni obliki zapišejo tudi samodejno s pomočjo znanih podatkov o orodju, stroju, obdelovancu, tehnoloških podatkih in zelenem izdelku.

V delu, ki je pred nami, smo idejo o razvoju tako žive in nežive narave uporabili pri samodejni izdelavi navodil za krmiljenje obdelovalnih strojev. Navodila se samodejno izdelujejo postopoma, dokler niso izpolnjene osnovne zakonitosti proizvodnje. Merilo za »preživetje« proizvodnega sistema je tako na prvem mestu cena proizvodnje, kar drugače povedano pomeni zmanjšanje stroškov. Zmanjšanje stroškov pa pritegne vrsto, največkrat, nasprotujočih si zahtev kot so varnost, poraba materiala, časa in energije ter navsezadnje največkrat brezbrizen odnos do ekologije in okolja.

Delo je zasnovano tako, da ga lahko brez podrobnejšega predznanja prebira vsakdo, ki ga zanima področje obdelovalnih tehnologij, avtomatizacije in umetne inteligence.

Delo sem izdelal pod mentorstvom red. prof. dr. Jožeta Baliča in somentorstvom univ. prof. dipl.-ing. dr. tech. DDr. h.c. Branka Katalinića.

Delo je bilo financirano s strani Ministrstva za šolstvo, znanost in šport, v okviru usposabljanja mladih raziskovalcev.

Avtor

Zahvala

Zahvalil bi se tistim, ki so neposredno pripomogli k nastanku dela:

- *svojim staršem, Barbari, Janezu in svojemu bratu dvojčku Tinetu, ki mi celo življenje stojijo ob strani,*
- *svojemu dekletu Maši, za vso potrpežljivost in razumevanje,*
- *prijateljem Davorju, Jožeku in Petru za usmerjanje pri ključnih življenjskih odločitvah,*
- *Miranu za stalen vir navdiha,*
- *mentorju, red. prof. dr. Jožetu Baliču, za zagotavljanje pogojev dela v Laboratoriju za inteligentne obdelovalne sisteme,*
- *vsem ostalim sodelavcem Laboratorija za inteligentne obdelovalne sisteme (Ivo, Igor, Mirko, Boštjan) za nesebično pomoč in nasvete,*
- *Ministrstvu za šolstvo, znanost in šport, za financiranje.*

Povzetek

Ključne besede: NC-stroji, NC-programiranje, evolucijsko računanje, genetski algoritem

UDK: 621.9.06-52:004.89(043.3)

V nalogi predstavljen koncept posnema naravni razvoj, kjer v boju za naravne vire uspešni individuumi postopoma postajajo vse bolj dominantni, uspešni in prilagodljivi na okolje v katerem bivajo, manj uspešni pa so v naslednjih generacijah navzoči v manjšem številu. Pri predlaganem konceptu so prilagajanju podvrženi programi za numerično krmiljenje obdelovalnih strojev (NC-programi). Med simulirano evolucijo, iz znanih podatkov o orodju, stroju, obdelovancu, tehnoloških podatkih in zelenem izdelku, postopoma nastajajo, čedalje učinkovitejši NC-programi.

Koncept smo uporabili pri programiranju stružnice, frezalnega stroja in stroja za razrez pločevine. Zaradi robustnosti in univerzalnosti ga lahko uporabimo za programiranje kateregakoli numerično krmiljenega obdelovalnega stroja. Dobljene rešitve lahko primerjamo z rešitvami dobljenimi z obstoječimi sredstvi. Rezultate naloge lahko uporabimo v praksi.

Abstract

Key-words: *NC-machine tool, NC-programming, evolutionary computation, genetic algorithm*

UDK: 621.9.06-52:004.89(043.3)

The concept imitates the natural evolution of living organisms, where in the struggle for natural resources the successful individuals gradually become more and more dominant, and adaptable to the environment in which they live, whereas the less successful ones are present in the next generations rarely. In the proposed concept the NC-programs undergo adaptation. During the simulated evolution more and more successful organisms (the NC-programs) emerge on the basis of given data on tool, blank, and desired product.

Concept was used for NC-programming of turning, milling and metal sheet cutting. The proposed concept shows a high degree of universality, efficiency, and reliability and it can be simply adopted to other NC-machines. Presented systems and results can be compared with commercially available software packages. Results can be implemented in practice.

Vsebina

PREDGOVOR	I
ZAHVALA	II
POVZETEK	III
ABSTRACT	IV
VSEBINA	V
KAZALO SLIK	IX
UPORABLJENE OZNAKE	XII
UPORABLJENI SIMBOLI	XIII
1. UVOD	1
1.1 ZGODOVINA KRMILJENIH STROJEV	2
1.2 ZGODOVINA NUMERIČNO KRMILJENIH OBDELOVALNIH STROJEV	9
1.2.1 LETA NOVE TEHNOLOGIJE – PRVA GENERACIJA	10
1.2.2 UPORABA MIKRORAČUNALNIKOV – DRUGA GENERACIJA	12
1.2.3 UPORABA OSEBNEGA RAČUNALNIKA – TRETJA GENERACIJA	13
1.3 KRMILJENJE OBDELOVALNIH STROJEV	13
1.3.1 MEHANIČNO KRMILJENJE OBDELOVALNIH STROJEV	13
1.3.2 KOPIRNO KRMILJENJE OBDELOVALNIH STROJEV	14
1.3.3 NUMERIČNO KRMILJENJE OBDELOVALNIH STROJEV	14
2. PROGRAMIRANJE NUMERIČNO KRMILJENIH OBDELOVALNIH STROJEV	16

2.1	ZAMETKI SAMODEJNEGA PROGRAMIRANJA NC STROJEV	17
2.2	SODOBNI KONCEPT.....	20
3.	METODE EVOLUCIJSKEGA RAČUNANJA.....	23
3.1	EVOLUCIJA ŽIVIH BITIJ.....	23
3.1.1	NARAVNI IZBOR	24
3.1.2	GENETIKA	24
3.1.3	DNK – NOSILEC INFORMACIJ	25
3.1.4	GENI, GENOMI, MUTACIJE, KROMOSOM	26
3.2	SKUPNE LASTNOSTI METOD EVOLUCIJSKEGA RAČUNANJA	27
3.3	GENETSKI ALGORITMI.....	28
3.4	GENETSKO PROGRAMIRANJE	31
3.4.1	STRUKTURA PODVRŽENA ADAPTACIJI	31
3.4.2	PRIMARNE GENETSKE OPERACIJE	32
3.4.3	SEKUNDARNE GENETSKE OPERACIJE	34
3.4.4	EVOLUCIJA POPULACIJE	35
3.5	EVOLUCIJSKE STRATEGIJE	35
3.6	EVOLUCIJSKO PROGRAMIRANJE	36
4.	PROGRAMIRANJE NUMERIČNO KRMILJENIH STROJEV Z UPORABO EVOLUCIJSKIH METOD.....	38
5.	PROGRAMIRANJE STRUŽNICE IN EVOLUCIJA.....	40
5.1	PREDLAGAN KONCEPT	40
5.1.1	GLAVNI ALGORITEM	40
5.1.2	KODIRANJE NC-PROGRAMA	42
5.1.3	OVREDNOTENJE POPULACIJE	43
5.1.4	GENETSKE OPERACIJE	45
5.2	REZULTATI IN DISKUSIJA.....	46
5.2.1	PRIMER 1	46
5.2.2	PRIMER 2	47
5.2.3	PRIMER 3	49
5.3	ZAKLJUČEK.....	50
6.	PROGRAMIRANJE FREZALNEGA STROJA IN EVOLUCIJA.....	52

6.1 PREDLAGAN KONCEPT	52
6.1.1 ZGRADBA SISTEMA	53
6.1.2 KODIRANJE NC-PROGRAMA	54
6.1.3 OVREDNOTENJE POPULACIJE	55
6.1.4 GENETSKE OPERACIJE	57
6.2 REZULTATI IN DISKUSIJA	57
6.2.1 PRIMER 1	57
6.2.2 PRIMER 2	59
6.3 ZAKLJUČEK.....	64
7. PROGRAMIRANJE STROJA ZA RAZREZ PLOČEVINE IN EVOLUCIJA	65
7.1 ZGRADBA SISTEMA.....	65
7.1.1 RAZPOZNAVANJE ZNAČILNOSTI	66
7.1.2 DOLOČANJE VBODOV	68
7.1.3 KODIRANJE NC-PROGRAMA	70
7.1.4 OVREDNOTENJE POPULACIJE	72
7.1.5 GENETSKE OPERACIJE	72
7.2 PRIMER 1	73
7.3 PRIMER 2	76
7.4 ZAKLJUČEK.....	80
8. ZAKLJUČEK.....	82
KRATEK OPIS SISTEMA ZA SAMODEJNO PROGRAMIRANJE NC STROJEV Z GENETSKIMI ALGORITMI.....	84
PROGRAMSKI JEZIK LISP	85
JEDRO – GENETSKI ALGORITEM	86
LITERATURA	91
OSEBNA BIBLIOGRAFIJA ZA OBDOBJE OD 2000 DO 2004	97
ČLANKI IN DRUGI SESTAVNI DELI	97
1.01 IZVIRNI ZNANSTVENI ČLANEK	97
1.08 OBJAVLJENI ZNANSTVENI PRISPEVEK NA KONFERENCI	97

1.16 SAMOSTOJNI ZNANSTVENI SESTAVEK V MONOGRAFIJI	100
SEKUNDARNO AVTORSTVO.....	100
PREVAJALEC	100
ŽIVLJENJEPIS.....	101

Kazalo slik

<i>Slika 1: Mehanizem cerkvenega zvona z bodičastim valjem.....</i>	<i>2</i>
<i>Slika 2: Wilhelm Schickard: Originalna skica stroja.....</i>	<i>3</i>
<i>Slika 3: Pascalov stroj.....</i>	<i>4</i>
<i>Slika 4: Diferenčni stroj.....</i>	<i>5</i>
<i>Slika 5: Nastajanje vzorca na statvah.....</i>	<i>6</i>
<i>Slika 6: Število NC-strojov v ZDA (prirejeno po [44]).....</i>	<i>10</i>
<i>Slika 7: Število patentov v zvezi z NC-stroji (vir The European Patent Office [84]).....</i>	<i>10</i>
<i>Slika 8: Mehanično krmiljenje s krivoljno ploščo.....</i>	<i>13</i>
<i>Slika 9: Mehansko kopirno krmiljenje.....</i>	<i>14</i>
<i>Slika 10: Idejna zasnova APT programiranja.....</i>	<i>18</i>
<i>Slika 11: Zasnova part programming geometrije.....</i>	<i>18</i>
<i>Slika 12: Tok informacij pri APT programiranju s krivuljami.....</i>	<i>19</i>
<i>Slika 13: Preprost APT program.....</i>	<i>20</i>
<i>Slika 14: Princip sodobnega programiranja.....</i>	<i>21</i>
<i>Slika 15: Charles Darwin (1809-82).....</i>	<i>24</i>
<i>Slika 16: Gregor Mendel (1822-84).....</i>	<i>25</i>
<i>Slika 17: Molekula DNK.....</i>	<i>26</i>
<i>Slika 18: Splošna shema evolucijskih algoritmov.....</i>	<i>28</i>
<i>Slika 19: Problem trgovskega potnika.....</i>	<i>29</i>
<i>Slika 20: Populacija treh organizmov za primer trgovskega potnika.....</i>	<i>29</i>
<i>Slika 21: Reprodukcijski problem za problem trgovskega potnika.....</i>	<i>30</i>
<i>Slika 22: Križanje za problem trgovskega potnika.....</i>	<i>30</i>
<i>Slika 23: Popravljen potnik.....</i>	<i>31</i>
<i>Slika 24: Permutacija za problem trgovskega potnika.....</i>	<i>31</i>
<i>Slika 25: Različen nabor terminalov in funkcij.....</i>	<i>32</i>
<i>Slika 26: Križanje pri genetskem programiranju [19].....</i>	<i>34</i>
<i>Slika 27: Mutacija pri genetskem programiranju [19].....</i>	<i>34</i>

<i>Slika 28: Permutacija pri genetskem programiranju [19]</i>	35
<i>Slika 29: Končni avtomat za preverjanje parnosti</i>	37
<i>Slika 30: Končni avtomat in njegov potomec</i>	37
<i>Slika 31: Preprost obdelovanec in izdelek pri struženju</i>	40
<i>Slika 32: Surovec, izdelek, odpadek in načini obdelave</i>	41
<i>Slika 33: Genetski algoritem za programiranje NC-stroja v pseudo kodi</i>	42
<i>Slika 34: NC-program kot utežen graf</i>	42
<i>Slika 35: Obdelovanec kot dinamično okolje</i>	43
<i>Slika 36: Konsistentnost rezov in kolizija</i>	44
<i>Slika 37: Procedura ovrednotenje</i>	44
<i>Slika 38: Genetska operacija križanje NC programov</i>	45
<i>Slika 39: Poprava NC programa</i>	45
<i>Slika 40: Permutacija NC programa</i>	46
<i>Slika 41: Različne rešitve za enak izdelek</i>	47
<i>Slika 42: Preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5</i>	47
<i>Slika 43: Dolžina najboljših NC-programov v posamični generaciji</i>	48
<i>Slika 44: Enostaven izdelek in kompleksen surovec</i>	49
<i>Slika 45: Kompleksen izdelek in enostaven surovec</i>	50
<i>Slika 46: Surovec, izdelek, odpadek in načini obdelave</i>	54
<i>Slika 47: Podajalna giba različnih dolžin</i>	55
<i>Slika 48: Obdelovanec kot dinamično okolje</i>	56
<i>Slika 49: Konsistentnost rezov</i>	56
<i>Slika 50: Kolizija</i>	57
<i>Slika 51: Genetska operacija mutacija NC programa pri frezanju</i>	57
<i>Slika 52: Konvencionalna obdelava</i>	58
<i>Slika 53: Genetska obdelava</i>	59
<i>Slika 54: Prvo vpetje</i>	60
<i>Slika 55: Drugo vpetje - postopna obdelava</i>	60
<i>Slika 56: Drugo vpetje – genetska obdelava – generacija 0</i>	61
<i>Slika 57: Drugo vpetje – genetska obdelava – generacija 10</i>	62
<i>Slika 58: Drugo vpetje – genetska obdelava – generacija 45</i>	63
<i>Slika 59: Preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5</i>	64
<i>Slika 60: Izdelki in pločevina</i>	65
<i>Slika 61: Zunanje in notranje meje</i>	65

<i>Slika 62: Odprti in zaprti rez.....</i>	<i>66</i>
<i>Slika 63: Algoritem za programiranje NC-stroja v pseudo kodi</i>	<i>66</i>
<i>Slika 64: Gnezda rezov in globine gnezd</i>	<i>67</i>
<i>Slika 65: Postopno rezanje od rezov globljih globin gnezd do plitvin.....</i>	<i>67</i>
<i>Slika 66: Vbodi, zunanji in notranji rezi</i>	<i>68</i>
<i>Slika 67: Določanje položaja vboda</i>	<i>68</i>
<i>Slika 68: Določanje vbodov za notranje in zunanje reze</i>	<i>69</i>
<i>Slika 69: Določanje vbodov pri sosednjih rezih.....</i>	<i>69</i>
<i>Slika 70: Določanje vbodov vbočenih rezov</i>	<i>70</i>
<i>Slika 71: Položaj vboda, na katerem ni bilo moč določiti vboda.....</i>	<i>70</i>
<i>Slika 72: Različno utežene uteži od reza i k rezu $i+1$.....</i>	<i>71</i>
<i>Slika 73: Število preskočenih globin v gnezd.....</i>	<i>72</i>
<i>Slika 74: Primer 1</i>	<i>73</i>
<i>Slika 75: Najboljši NC-program v začetni generaciji 0.....</i>	<i>74</i>
<i>Slika 76: Najslabši NC-program v generaciji 0.....</i>	<i>74</i>
<i>Slika 77: Najboljši NC-program v generaciji 2</i>	<i>74</i>
<i>Slika 78: Najboljši NC-program civilizacije.....</i>	<i>75</i>
<i>Slika 79: Dolžina najboljših NC-programov v posamični generaciji.....</i>	<i>75</i>
<i>Slika 80: Preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5</i>	<i>76</i>
<i>Slika 81: Primer 2</i>	<i>77</i>
<i>Slika 82: Najboljši NC-program v generaciji 0</i>	<i>78</i>
<i>Slika 83: Najboljši NC-program v generaciji 7</i>	<i>78</i>
<i>Slika 84: Najboljši NC-program v generaciji 20</i>	<i>79</i>
<i>Slika 85: Najboljši NC-program v generaciji 50</i>	<i>79</i>
<i>Slika 86: Najboljši NC-program v generaciji 95</i>	<i>80</i>
<i>Slika 87: Dolžina podajalnih poti najboljših NC-programov v posamični generaciji</i>	<i>80</i>

Uporabljene oznake

A	adenozin
AI	Artificial Intelligence - umetna inteligenca
AIA	Aircraft industries Associations – združenje letalske industrije
APT	Automatically Programmed Tools- računalniški jezik za samodejno programiranje strojev
C	citozin
CAD	Computer Aided Design - računalniško podprto modeliranje in konstruiranje
CAM	Computer Aided Manufacturing – računalniško podprta proizvodnja
CIM	Computer Integrated Manufacturing - računalniško integrirana proizvodnja
CNC	Computer Numerical Control - numerično krmiljeni stroji
DNK	dezoksiribonukleinska kislina
DNC	Direct Numerical Control – direktno numerično krmiljenje
EMI	Electrical and Musical Instruments – tovarna električnih in glasbenih inštrumentov
FMS	Flexible Manufacturing Systems - prilagodljivi obdelovalni sistemi
G	gvanin
MIT	Massachusetts Institute of Technology – tehniški inštitut (Massachusetts)
NC	Numerical Control - numerično krmiljenje
PC	Personal Computer - osebni računalnik
RNK	ribonukleinska kislina
POS	prilagodljivi obdelovalni sistemi
UI	umetna inteligenca

Uporabljeni simboli

μ	število članov v generaciji
λ	število potomcev
$\mu+\lambda$	skupno število organizmov
a_i	terminal iz nabora \mathcal{T}
f	faktor vpliva
\mathcal{F}	nabor funkcijskih celic
f_i	funkcija iz nabora \mathcal{F}
i	indeks elementa
k	število argumentov
K	število kolizij, število preskočenih globin
M	število potomcev
n	število elementov
N_f	največje število funkcij v naboru funkcijskih celic \mathcal{F}
N_t	največje število terminalov v naboru terminalov \mathcal{T}
$P(t)$	populacija
\mathcal{R}	realna števila
r_i	dolžina reza i
S_i	stanje avtomata
t	čas
\mathcal{T}	nabor terminalskih celic
w_{ij}	utež na povezavi
x	neodvisna spremenljivka
X_t	x koordinata težišča
y	neodvisna spremenljivka
Y_t	y koordinata težišča

σ vektor standardne deviacije

\cup unija

1. UVOD

Eno izmed osnovnih načel obstoja je spreminjanje. Postopnemu spreminjanju s tujko pravimo evolucija. Navadno pod pojmom evolucija razumemo postopno spreminjanje k boljšim, popolnejšim oblikam [6].

Ko se je energija dovolj zgostila, je nastala materija [43][82]. Postopoma so se razvile oblike, sposobne rasti in razmnoževanja; najprej molekule spojin, ki so se bile sposobne deliti in kasneje zapletene celice z jedrom [77][89]. Takim oblikam pravimo živa bitja. Naslednjo stopnjo postopnega spreminjanja predstavljajo živa bitja, ki so ob sposobnosti razraščanja in razmnoževanja sposobne premikanja glede na prirojeno, nehoteno težnjo k določenemu ravnanju, ki mu pravimo nagon. Vrhunec razvoja živih bitij z njemu pripadajočim smiselnim premikanjem predstavlja človek.

Človek je živo bitje, ki se skuša utemeljeno, smiselno, razumsko prilagoditi okolju, v katerem biva, drugače rečeno, bori se za svoj obstoj, preživetje. Stalna težnja po prilagajanju človeka ves čas sili k ustvarjanju in odkrivanju, zato je opazoval sebe in predvsem okolje.

Tako je ugotovil, da nekatere pojave v okolju lahko oponaša s pomočjo pripomočkov, ki mu lajšajo obstoj. Pri opravih si je pomagal z orodji, kasneje pa so njegove telesne sposobnosti zamenjale učinkovitejše mehanske naprave, stroji. Tako je sčasoma izdelal celovitejše stroje, ki so združevali mehanski in umski pristop. Za take stroje ni bil več potreben človeški upravljalac.

Naloga je sestavljena tako, da v uvodu poda osnovne informacije o razvoju krmiljenih naprav ter razvoju samodejnih naprav zgrajenih v industrijske namene – obdelovalnih strojev. Posebej izstopa poglavje o programiranju samodejnih numerično krmiljenih obdelovalnih strojev (poglavje 2), ki jih na kratko imenujemo NC stroji. Kratica je okrajšava za Numerical Control, kar v širšem smislu pomeni »upravljanje s številskimi znaki«, nanaša pa se na navodila, s katerimi upravljamo tovrstne stroje.

Programiranje NC-strojov dandanes predstavlja pomemben člen v proizvodnem in finančnem razvoju sodobne industrije. Spoznanje, da lahko najdemo vzporednice med neživim in živim svetom nas pripelje do uporabe metod evolucijskega računanja, ki posnemajo načela evolucije: prilagajanje in preživetje. Evolucijske metode so na kratko povzete v poglavju 3.

Primarni cilj naloge, ki se počasi razkriva v poglavjih od 4 do 7, je izdelati enoten sistem za samodejno programiranje različnih NC strojev ob upoštevanju omejitev, ki temelji na uporabi metod evolucijskega računanja. Osnovna ideja samodejnega programiranja NC strojev z evolucijskimi metoda je predstavitev navodil NC stroja kot objekta, sposobnega spreminjanja

in prilagajanja. Navodila so sestavljena iz posamičnih ukazov, ki v različnem okolju dajejo različne rezultate. Spreminjamo navodila, spreminjamo izdelek. Uspešnejša navodila so seveda tista, ki izpolnjujejo dane pogoje, med katere vsekakor sodi na prvo mesto cena izdelka ob upoštevanju osnovnih ekoloških in varnostnih zahtev.

Ideja je prikazana najprej na preprostem, poenostavljenem, dvo-dimenzionalnem primeru struženja (poglavje 4). V poglavju 6 idejo razširimo na tro-dimenzionalno obdelavo freziranja, v poglavju 7 pa ideja dobi praktične razsežnosti uporabniško prijazno zasnovanega sistema za samodejno programiranje NC stroja za razrez pločevine.

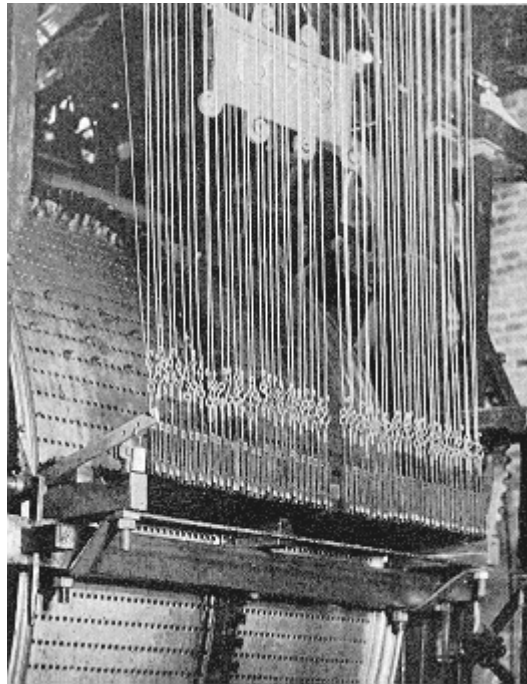
Sklepne ugotovitve naloge so podane v poglavju 8.

Najpomembnejše omejitve naloge so:

- nekatere predlagane in predstavljene rešitve naloge niso povsem praktične narave,
- raziskava ne preučuje kemičnih in fizikalnih lastnosti predstavljenih proizvodnih sistemov in
- ustvarjanje navodil NC strojev zahteva, zaradi časovne zahtevnosti računalniške obdelave, pravilno izbiro parametrov sistema za samodejno ustvarjanje navodil NC strojev.

1.1 Zgodovina krmiljenih strojev

Idejo o krmiljenju naprav po vnaprej določenih navodilih lahko zasledimo že v 14. stoletju, ko so zvonove krmilili s posebnimi valji z zatiči in na ta način dobili želeno melodijo, ki se je lahko poljubnokrat ponovila [10]. Slika 1 prikazuje mehanizem valja z zatiči izdelan leta 1663 na Nizozemskem (Nieuwe Kerk, Delft). Vzvodi zaradi zatičev na valju, ki se vrti, spreminjajo lego. Kladivca, ki so z vrvicami povezana z vzvodi udarjajo po zvonu.

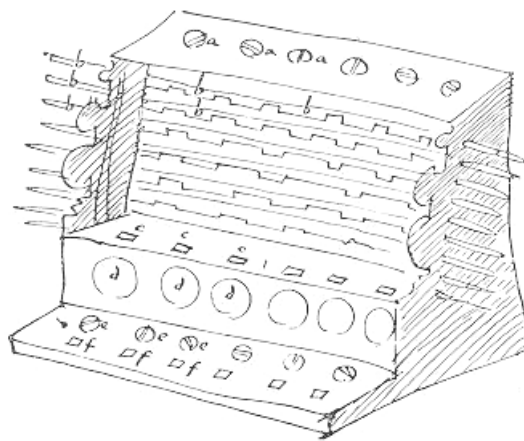


Slika 1: Mehanizem cerkvenega zvona z bodičastim valjem

Razvoj sistema numerično vodenih obdelovalnih strojev je s podobnimi izumi v tesni zvezi, na njegov nastanek in nezadržni razvoj pa močno vpliva razvoj elektronike in računalniške tehnike. V nadaljevanju je podanih nekaj ključnih zgodovinskih mejnikov razvoja računalništva, krmilne tehnike in numerično krmiljenih obdelovalnih strojev [10][11][73].

Stroji, ki so sposobni izvajati osnovne štiri aritmetične operacije (seštevanje, odštevanje, množenje in deljenje), so se prvič pojavili v začetku sedemnajstega stoletja v Evropi. Danes jih običajno označujemo z besedo kalkulatorji. S tem želimo poudariti, da je njihova sposobnost za programiranje majhna ali je pa sploh ni.

Najstarejši stroj za opravljanje osnovnih aritmetičnih operacij, za katerega vemo, je razvil in naredil leta 1623 Wilhelm Schickard (1592-1635), profesor matematike na univerzi v Tuebingenu (Slika 2). Schickardov stroj je bil v svojem času malo znan; verjetno tudi zato, ker je Schickard z družino vred umrl v epidemiji kuge leta 1635. Ohranjena pa je korespondenca med Schickardom in slavnim astronomom Keplerjem, v kateri je podroben opis stroja. Iz nje je razvidno, da je znal seštevati in odštevati, z nekoliko dodatnega ročnega dela pa tudi deliti in množiti. Dosti bolj kot Schickardov pa je znan podoben stroj, ki ga je leta 1642 naredil veliki francoski filozof, matematik in fizik Blaise Pascal (1623-1662).



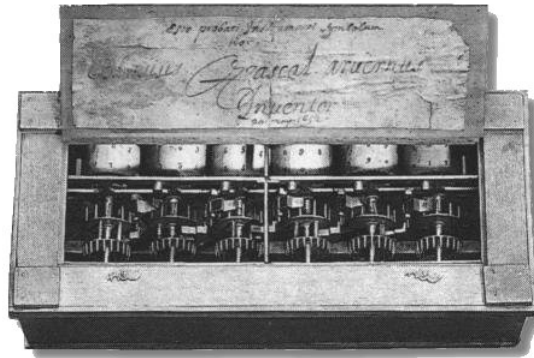
Slika 2: Wilhelm Schickard: Originalna skica stroja

Po zasnovi sta bila oba stroja zelo podobna, čeprav je Schickardov znal več. Oba sta za predstavljanje desetiških števil uporabljala zobata kolesa z desetimi zobmi in mehanizmom za prenos enote na naslednje kolo po vsakem obratu. To idejo srečujemo vsak dan, še danes pri mehanskih števcih, npr. v avtomobilu ali električnem števcu. V resnici pa je ideja zelo stara in je omenjena že v zapiskih Hera iz Aleksandrije iz drugega stoletja našega štetja. Pascalov stroj je imel dve skupini s po šestimi zobatimi kolesi. Ena skupina je služila kot pomnilnik, druga pa za vnašanje šestmestnega števila, ki ga je uporabnik želel prišteti ali odšteti od tistega v pomnilniku. Pogon stroja je bil seveda ročen s pomočjo posebne ročice.

Naslednje pomembno izboljšavo je naredil slavni nemški filozof in matematik Gotfried Leibniz (1646-1716), ki si med drugim skupaj z Newtonom deli čast za postavitev matematičnih osnov diferencialnega in integralnega računa. Leta 1671 je zasnoval stroj, ki je poleg seštevanja in odštevanja znal tudi množiti in deliti. Stroj je bil izdelan šele leta 1694 in je podoben Pascalovemu. Imel je še dve dodatni skupini koles za predstavitev multiplikanda in multiplikatorja oziroma dividenda in divizorja. Za množenje in deljenje je Leibniz izumil element, ki je danes znan kot Leibnizovo kolo, in ki se je v nekaterih strojih uporabljal še ne

tako dolgo nazaj. Leibnizov motiv za izdelavo stroja je bil razbremeniti od težaškega dela, ki ga je zahtevalo ročno izračunavanje pri astronomskih izračunih. S strojem je želel doseči, da bi se ljudje osredotočili na bolj pomembne vidike problemov.

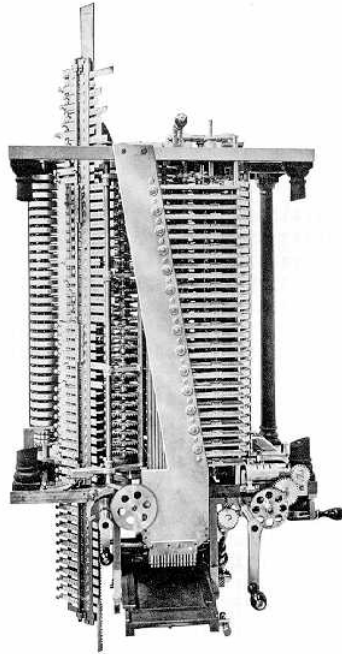
Čeprav je Leibnizov stroj pomenil rešitev za strojno izvajanje osnovnih štirih operacij, je bilo v praksi njegovo delovanje nezanesljivo. Isto velja tudi za Pascalov stroj. Kljub precejšnjim vloženim sredstvom tehnologija tistega časa ni mogla doseči natančnosti, ki je potrebna za zanesljivo delovanje. Resnično uporabni stroji so se pojavili šele v drugi polovici osemnajstega stoletja. Serijska proizvodnja in njihova splošna uporaba pa sta se razširila po letu 1850.



Slika 3: Pascalov stroj

Ena od najbolj izstopajočih oseb v zgodovini strojev za računanje je Anglež Charles Babbage (1792-1871). Babbage je izumil stroj, ki je po osnovi zelo podoben današnjim računalnikom. Čeprav je od angleške vlade uspel dobiti za tiste čase precej denarja za financiranje projekta, je državna podpora usahnila, ko se je pokazala zahtevnost problemov.

Diferenčni stroj (Slika 4) je prvi in preprostejši od obeh strojev za računanje, ki ju je izdelal. Namenjen je bil za računanje in tudi za avtomatično tiskanje matematičnih tabel. Od aritmetičnih operacij je uporabljal samo seštevanje, vendar v povezavi z metodo končnih diferenc, kar omogoča izračun poljubnega polinoma. Vse, kar je potrebno, je, da v registre diferenčnega stroja vstavimo začetne vrednosti diferenc in stroj bo, če ga poganja ustrezen motor (v Babbagevem času parni stroj), avtomatično računal in izpisoval zaporedne vrednosti funkcije.



Slika 4: Diferenčni stroj

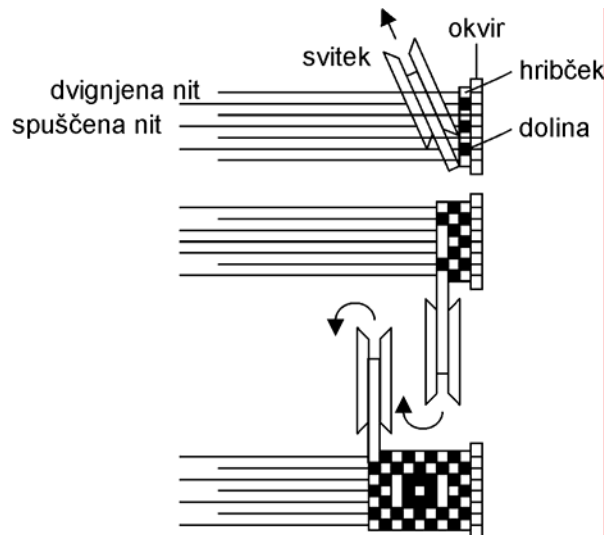
Babbage je najprej predlagal gradnjo diferenčnega stroja, ki bi obvladal polinome šeste stopnje z 20-mestno natančnostjo. Delo na gradnji se je pričelo leta 1823 in je trajalo s presledki do leta 1833, ko je bil projekt zaradi pomanjkanja denarja in sporov opuščen. Babbage ni nikoli dokončal tega stroja. Kasneje je, že med delom na analitičnem stroju, naredil še načrt za diferenčni stroj številka 2, ki je zmozel polinome sedme stopnje z 31-mestno natančnostjo. Ta stroj je zanimiv zato, ker so ga leta 1991 po originalnih Babbagevih načrtih zgradili v londonskem Science Museum [85]. Več kot tri tone težak stroj je stal približno 500.000 ameriških dolarjev in deluje brezhibno. Z njim so hoteli dokazati, da je tako zapleten stroj v resnici mogoče narediti samo z mehanskimi sredstvi.

Do realizacije precej skromnejše inačice diferenčnega stroja je prišlo že v Babbagevem času. Leta 1834 je eden od Babbagevih člankov prišel v roke Švedu Georgu Scheutz (1785-1873). Ta se je z določeno finančno pomoč švedske vlade lotil gradnje stroja, ki je zmozel polinome tretje stopnje s 15-mestno natančnostjo. Prvi model je deloval že leta 1837, končna verzija, ki je vsebovala tudi samodejno tiskanje rezultatov, pa je bila gotova leta 1853. Kasneje je podobne stroje zgradilo še več drugih ljudi. Ti stroji so bili v svojem času precejšnja atrakcija in so jih kazali na vseh pomembnih razstavah. Zanimivo pa je, da jih je bilo zgrajenih zelo malo. Potrebe po strojnem računanju v tistem času preprosto niso bile dovolj velike. Diferenčni stroj je bil namenjen za računanje in tiskanje tabel funkcij in ne za poljubne formule. Vendar pa do ideje o stroju za računanje poljubnih formul ni bilo daleč. Svojo prvo obliko je doživela v še bolj slavnem Babbagevem stroju, ki je znan pod imenom analitični stroj.

O stroju, ki bi bil splošnejši, je Babbage razmišljal že med delom na diferenčnem stroju. Po prekinitvi dela na njem je v letih 1834 in 1835 pričel razvijati idejo o stroju, ki bi lahko izvajal poljubno zaporedje aritmetičnih operacij. Kmalu je ugotovil, da je z izločitvijo aritmetičnega mehanizma v posebno centralno enoto mogoče doseči pomembne prihranke pri številu potrebnih delov. Skupaj z drugim mehanizmom, ki bi določal vrstni red operacij, bi bilo v principu mogoče rešiti poljuben problem (če zanemarimo omejite stroja in seveda

neizračunljive probleme). Najprej je za ta mehanizem uporabil bobn z zatiči. Leta 1836 se je odločil za luknjane kartice podobne tistim, ki jih je leta 1801 izumil J. M. Jacquard (1752-1834) za določanje vzorca na tkalskih strojih [58].

Pri Jacquardovih statvah so bile preluknjane kartice povezane v brezkončen trak in pričvrščene na vrhu statev. Vzorci na statvi nastajajo glede na položaj vodilnih niti. Luknjice na kartici so določale ali se vodilne niti spuščajo ali dvigajo. Princip nastajanja vzorca kaže slika 5.



Slika 5: Nastajanje vzorca na statvah

V primerjavi z bobni so luknjane kartice veliko preprostejše za uporabo in shranjevanje.

Po Babbagevem lastnem opisu lahko računski del analitičnega stroja razdelimo na mlin (ang. mill), v katerem se izvajajo vse operacije in pomnilnik (ang. store), v katerem so shranjeni operandi, na katerih se izvajajo operacije in v katerega se shranjujejo rezultati operacij.

Mlin je znal izvajati štiri operacije: seštevanje, odštevanje, množenje in deljenje. Vrstni red operacij so določale luknjane kartice, ki jih je bilo potrebno pripraviti pred pričetkom dela. Te kartice so bile razdeljene v dve ločeni skupini, vsaka s svojim mehanizmom za branje: ukazne in operandne kartice.

Babbage je razlikoval dve vrsti ukaznih kartic. Prva vrsta so operacijske kartice, ki določajo, katera od operacij se naj izvede v določenem koraku. Druga vrsta pa so tako imenovane kombinatorične kartice, ki so, kadar je bil izpolnjen določen pogoj, sprožile mehanizem za ponavljanje. S pomočjo tega mehanizma so se ukazne in/ali podatkovne kartice vrnile nazaj - ukazi so se zato lahko ponovili. Pogoj, pri katerem se je to zgodilo, je bil predznak števila. Povedano v današnji terminologiji: kombinatorične kartice so omogočale pogojne skoke. To sposobnost analitičnega stroja imamo danes za enega od najpomembnejših dosežkov Charlesa Babbagea.

Tudi pri operandnih karticah Babbage razlikoval dve vrsti. Prva vrsta so kartice spremenljivk, ki določajo, katere pomnilniške lokacije naj se uporabljajo v dani operaciji. Drugače povedano, določale so kje so operandi in kam naj gre rezultat. Za vsako od operacij je bilo

točno določeno, koliko kartic jo mora spremljati. Druga vrsta so kartice števil. Vsaka kartica te vrste vsebuje predznačeno število, ki se prenese v določeno pomnilniško lokacijo.

Ukazne in operandne kartice skupaj predstavljajo tisto, čemur danes pravimo program. Babbage se je zelo dobro zavedal razlike med fizičnim in programskim pogledom na stroj. Predvideval je na primer, da se kartice po uporabi lahko shranijo in ponovno uporabijo z istimi ali spremenjenimi vhodnimi podatki. S tem v zvezi je omenjal tudi knjižnico za analitični stroj, v kateri bi bile shranjene kartice za najpomembnejše probleme.

1863 je M. Fournaux patentiral avtomatski klavir, znan pod imenom Pianola. Jedro klavirja poleg običajnih klavirskih sestavnih delov, vsebuje širok papirnati trak z luknjicami, skozi katere stisnjen zrak krmili mehanično tipkovnico in igra melodije zapisane na papirnatem traku. Klavir so kasneje razvili do te mere, da je bilo mogoče regulirati tudi zvok, jakost in hitrost igranja, kar je omogočilo natančnejšo izvedbo klavirskih skladb [11].

Prav preprostost in zanesljivost sta bila glavna razloga za komercialno uspešnost naprav na osnovi luknjanih kartic. Z njimi je povezano ime Američana Hermana Holleritha (1860-1929), ki jih je prvič uspešno uporabil leta 1887 pri statistični obdelavi podatkov o smrtnosti v Baltimoru. Njegov največji uspeh pa je bila odločitev, da se njegove naprave uporabijo pri obdelavi podatkov ljudskega štetja v ZDA leta 1890. Obdelava, ki je zahtevala luknjanje kakih 56 milijonov kartic, je zelo uspela in naprave te vrste so se hitro razširile po ZDA in v svetu. Podjetje Tabulating Machine Company, ki ga je leta 1896 ustanovil Hollerith, se je leta 1911 združila z dvema drugima v novo podjetje, to pa se je leta 1924 preimenovalo v International Business Machines Corporation ali IBM.

Razvoj elektromehanike je odprl nove možnosti pri realizaciji strojev za računanje. Ena od prvih uporab je bila uvedba elektromotorjev za pogon mehanskih kalkulatorjev. Ti stroji so se v začetku dvajsetega stoletja zelo razširili. Drugo pomembno področje uporabe predstavljajo naprave na osnovi luknjanih kartic, s katerimi je bilo mogoče sortirati in tabelirati velike količine podatkov.

Prvi uspešni poskusi nadaljevanja Babbagevega dela so se pojavili šele konec 1930-tih let neodvisno v Nemčiji in ZDA. Glavni razlog za to, da se ti poskusi niso zgodili bolj zgodaj, je v majhnih potrebah po povsem avtomatskem strojnem računanju. Po tem, kar vemo danes, je prvi delujoči stroj Babbageve vrste naredil Nemeč Konrad Zuse. S problemom gradnje stroja se je pričel ukvarjati leta 1934 kot študent gradbeništva. Zuse ni vedel za Babbagevo delo in je do načrtov za svoje stroje prišel samostojno. V nasprotju z večino prejšnjih poskusov se je odločil za binarno in ne za desetiško aritmetiko. Poleg tega je uporabil predstavitev števila tako imenovani plavajoči vejici v podobni obliki, kot jo poznamo danes. Čeprav vemo, da je binarno aritmetiko predlagal Leibniz, plavajočo vejico Torres in programsko vodenje stroja Babbage, je bil Zuse prvi, ki je te ideje združil v delujočem stroju.

Zuse je imel veliko težav s financiranjem svojih projektov. Njegovi stroji so zato po velikosti veliko skromnejši od približno v istem času nastajajočih ameriških. Leta 1938 je dokončal stroj Z1, ki je bil povsem mehanski in pri delovanju nezanesljiv. Nato je pričel z gradnjo stroja Z2, pri katerem je bila aritmetična enota zgrajena iz telefonskih relejev, za pomnilnik pa je uporabil mehanski pomnilnik od Z1. Po Zusevem pripovedovanju je aritmetična enota delovala, ko je bil mobiliziran leta 1939 in stroj je ostal nedokončan. Po približno enem letu sta Zuse in njegov sodelavec Schreyer uspela prepričati oblast o pomembnosti njunega dela. Zuseja so izpustili iz vojske ter mu celo dali finančno podporo. Tako je bil leta 1941

dokončan elektromehanski stroj Z3, za katerega danes mislimo, da je bil prvi delujoči programsko vodeni računalnik za splošne namene. Zgrajen je bil iz približno 2600 telefonskih relejev in je imel relejski pomnilnik velikosti 64 22-bitnih besed. Vsi ukazi so bili 8-bitni in so bili shranjeni na luknjanem traku.

Za vnos podatkov je služila tipkovnica, rezultat pa je bil prikazan v numerični obliki z žarnicami na posebnem zaslonu.

Konec vojne je dočkala samo izpopolnjena verzija Z3, ki je bila pod oznako Z4 narejena leta 1945. Z4 je imel 32 bitne besede in se je uporabljal še osem let po koncu vojne na ETH v Zuerichu. Zuse je že leta 1941 nameraval zgraditi elektronsko verzijo s 1500 elektronkami, vendar je oblast njegov predlog zavrnila. Zuse ni vedel za delo, ki je v tem času potekalo v ZDA, podobno kot tudi Američani niso vedeli za Zuseja. Zato ni presenetljivo, da so verjeli, da je bil njihov Harvard Mark I prvi delujoči programsko vodeni računalnik za splošne namene.

Duhovni oče računalnika Harvard Mark I je bil Howard Aiken (1900-1973), fizik s Harvardske univerze. Kasneje so bili pod njegovim vodstvom zgrajeni še Mark II, Mark III in Mark IV. Zadnja dva sta bila elektronska, vendar ne s shranjenim programom. Drugače kot Zuse je Aiken poznal Babbageovo delo. Gradnjo elektromehanskega programsko vodenega računalnika za splošne namene je predlagal leta 1937, delo pa se je pričelo leta 1939. Harvard Mark I je v resnici proizvod firme IBM. Podobno, kot analitični stroj je Mark I uporabljal za pomnilnik desetiška števna kolesa, ki pa so bila realizirana elektromehansko. Njegov pomnilnik je obsegal 72 23-mestnih desetiških števil, ukaze pa je dobival preko luknjanega traku. Za primerjavo povejmo, da je seštevanje in odštevanje trajalo 0.3 sekunde, množenje 6 in deljenje 12 sekund.

Že pred letom 1940 je bilo znano, da je logične funkcije, ki so potrebne pri računalnikih, mogoče realizirati tako z elektromehanskimi releji, kot z vakumskimi elektronkami. Leta 1906 odkrita trioda, ki so ji sledile druge vrste elektronk ter z njimi narejena logična in pomnilniška vezja, kot je npr. Eccles-Jordanov flip-flop iz leta 1919, so predstavljala osnovo, na kateri je bilo mogoče razmišljati o elektronskem računalniku. vendar so bili praktični problemi pri gradnji stroja z nekaj tisoč elektronkami tako veliki, da ljudje dolgo niso verjeli v možnost uspešne realizacije.

Prvi poskus gradnje računalnika z uporabo vakumskih elektronk je naredil John Atanasoff. Njegov stroj je bil elektronski analogni računalnik za reševanje sistema linearnih diferencialnih enačb. Zaradi hudih omejitev analognega načina se je leta 1935 preusmeril na digitalnega. V letih od 1936 do 1938 mu je uspelo razviti in preizkusiti vrsto elektronsko logičnih elementov, na osnovi katerih se je leta 1939 lotil gradnje razmeroma velikega sistema, ki bi lahko reševal sisteme do 30 linearnih enačb. Velik del sistema je bil zgrajen in preizkušen, vendar je bilo delo leta 1942 prekinjeno in stroj nikoli v celoti končan.

Na University of Pennsylvania je bil na Moore School of Electrical Engineering razvit in narejen najprej ENIAC in nato še EDVAC. Delo na ENIACu (Electronic Numerical Integrator and Calculator, elektronsko numerični integrator in kalkulator) se je pričelo leta 1943 pod vodstvom raziskovalcev Johna Mauchly in J. Presper Eckerta. Denar je zagotovila ameriška vojska in eden od glavnih motivov je bila potreba po računanju in tiskanju balističnih tabel za razne vrste topov in bombnih merilcev v letalih. Količina računanja, ki je

potreben pri pripravi teh tabel, je tako velika, da so bili elektromehanski stroji dosti prepočasni. ENIAC je bil dokončan leta 1945, uradno pa se je delo na njem začelo leta 1946.

Programiranje ENIACa je bilo ročno s postavljanjem stikal in povezovanjem kablov. Med ukazi so bili tudi pogojni skoki in ni uporabljal traka z ukazi. Med programiranjem je stroj stal in ni bil uporaben za reševanje drugih problemov. S stališča uporabe je bil to korak nazaj, ki ga pa ni težko razumeti, če upoštevamo, da so bile operacije na ENIACu približno 1000-krat hitrejše, kot pa na elektromehanskih računalnikih in tudi veliko hitrejše od čitalnikov luknjanega traku ali kartic.

Ideja o računalniku s shranjenim programom se danes povezuje z imenom znanega ameriškega matematika madžarskega porekla Johna von Neumanna (1903-1957), ki je bil svetovalec pri ENIACu. Prvič jo je Johna von Neumann objavil junija 1945 v predlogu za nov elektronski računalnik EDVAC (Electronic Discrete Variable Computer, elektronski računalnik za diskretne spremenljivke). V članku sta bila prvič omenjena tudi pojma centralne procesne enote in glavnega pomnilnika. Vsi današnji računalniki so praktično von Neumannovi. Leta 1949 je izdelal računalnik EDSAC.

Po vrnitvi v Princeton je leta 1946 von Neumann skupaj s kolegi pričel delati na razvoju novega računalnika s shranjenim programom, ki je kasneje postal znan pod imenom IAS računalnik (kratica za Institute for Advanced Study, kjer je delo potekalo). Velik pomen, ki ga ima IAS v zgodovini razvoja računalnikov, je predvsem posledica prostega dostopa do vseh informacij v zvezi z njim. Večina ostalih projektov je potekala v precejšnji tajnosti. Nasprotno pa je IAS služil kot neke vrste šola za vse, ki so jih računalniki zanimali. Veliko računalnikov prve generacije, npr. znane serije IBM 700 in 7000 ter UNIVAC I, je bilo zato podobnih IAS. Ta podobnost se je ohranila še dolgo potem.

IAS je pričel delovati poleti 1951, čeprav je bila uradna otvoritev šele leta 1952. Podobno kot EDVAC in EDSAC je bil tudi IAS binarni stroj. IAS je bil približno 10-krat hitrejši od ENIAC-a.

V tem časovnem obdobju so se pojavili prvi programabilni obdelovani stroji zato je vmesno, da se jim na tej prelomnici povsem posvetimo.

1.2 Zgodovina numerično krmiljenih obdelovalnih strojev

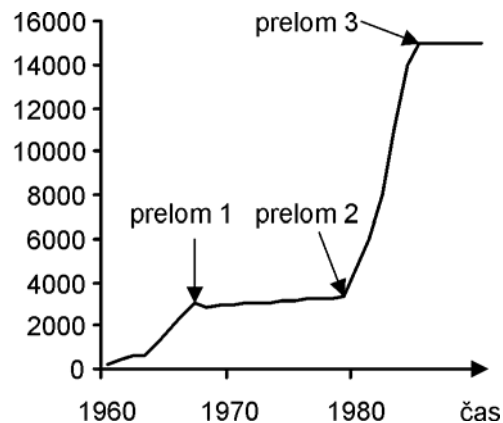
V razmahu zadnjih petdeset let lahko razvoj numerično krmiljenih obdelovalnih strojev strnemo v tri razvojne stopnje - generacije, na katere so vplivale predvsem tri tehnološke inovacije, tri korenite spremembe, ki so pretresle obstoječo industrijo [5][44][32][26].

V prvo razvojno stopnjo lahko uvrstimo bistvene spremembe v zgradbi proizvodnje z uvajanjem numerično krmiljenih obdelovalnih strojev v letih 1950 do 1960.

Naslednjo razvojno stopnjo v letih 1970 in 1980 je moč določiti z uvajanjem mikroročunalnikov za numerična krmilja. Razlika v ceni krmilja za NC-stroj med letoma 1963 in 1973 je padla za 50%, v letu 1978 pa že za 75%. Prodaja NC-strojev je naraščala od 30-40% letno v letih 1978 do 1984. Čas druge razvojne stopnje sovпада s pojavom japonskih proizvajalcev NC-strojev na svetovnem trgu.

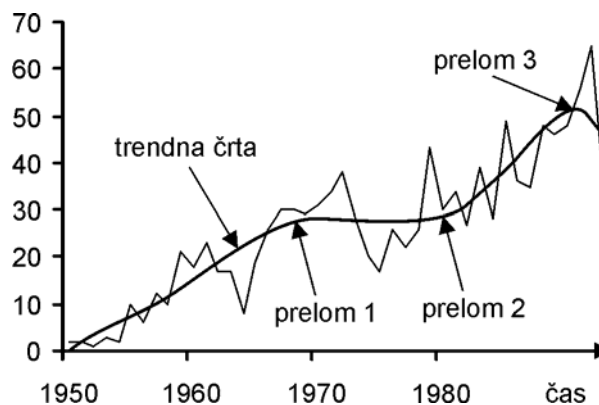
Tretjo stopnjo razvoja je sprožila množična uporaba računalnikov pri načrtovanju (CAD – computer aided manufacturing, računalniško podprto konstruiranje) in proizvodnji (CAM – computer aided manufacturing). Cilj tretje razvojne stopnje je bil povezovanje in avtomatizacija proizvodnje s pomočjo računalnikov (CIM – computer integrated manufacturing, računalniško integrirana proizvodnja).

Slika 6 prikazuje prodajo NC strojev na enega izmed največjih trgov na svetu, v ZDA. Preloma 1 (1976) in 2 (1978) na krivulji sovpadata s tremi razvojnimi stopnjami, prelom 3 (1984) pa predstavlja ustaljeno prodajo NC strojev, ki traja še dandanes.



Slika 6: Število NC-strojov v ZDA (prirejeno po [44])

Oblika trendne črte s tremi pregibi na sliki 7 spominja na pregibe slike 6. Prikazuje namreč število patentnih prijav v času od leta 1950 do danes, ki se tičejo numerično krmiljenih obdelovalnih strojev.



Slika 7: Število patentov v zvezi z NC-stroji (vir The European Patent Office [84])

1.2.1 Leta nove tehnologije – prva generacija

Začetek sodobnega razvoja NC-tehnologije datira nekako v letih 1948-1949. začelo se je v ZDA, kjer so v družbi United States Air Force želeli izdelovati zahtevne izdelke hitreje, natančneje in ceneje kot s konvencionalnimi postopki. Iskali so tehnološko rešitev za samodejno izdelavo izdelkov po načrtu. Pod vodstvom Johna Parsona, podpredsednika John Parsons Company (Traverse City, Michigan City), in laboratorija Servomechanics Laboratory iz MIT (Massachusetts Institute of Technology) so raziskovalci leta 1951 razvili prve frezalne stroje in jih preizkusili v tovarnah.

John T. Parsons se je domislil osnovne ideje, da mora biti samodejni stroj podatkovni sistem. Pomladi leta 1949 je kontaktiral s profesorjem Gordonom S. Brownom iz laboratorija Servomechanics Laboratory iz MIT, ki so med vojno razvijali sisteme za krmiljenje topov na kupolah vojaških vozil in so po vojni iskali povezavo z industrijo, predvsem v smeri obdelovalnih strojev. Parsons je projekt predstavil kot razvoj »Cardamatic Milling Machines I and II« (Frezalni stroj I in II) pod taktirko združenih MIT in Parsons Corporation. Parsons je uspel prepričati Air Force, da so projekt finančno podprli. Parsons Corporation je tako prejela 200.000 dolarjev, od katerih je Servomechanics Laboratory prejel 80.000 dolarjev. MIT je za raziskave porabila kar 360.000 dolarjev, zato je račun za preostalih 280.000 dolarjev naslovila Air Forceu. Za osnovo raziskav so uporabili konvencionalni 3-osni frezalni stroj podjetja Cincinnati Milling Machine Company. Septembra 1952 so preizkusne stroje predstavili javnosti. Po finančni podpori so pri Air Force upali, da bodo v industriji toplo sprejeli novo tehnologijo, toda pričakovanja so bila daleč od resničnosti. MIT in Air Force sta v letih 1953 in 1954 združno izvajala program obveščanja in seznanjanja NC tehnologije.

Podjetje Giddings & Lewis Machine Tool Company je bilo eno izmed redkih, ki je skušalo izboljšati natančnost svojih obdelovalnih strojev z numeričnim krmiljenjem. Zmanjšanje velikosti in zahtevnosti kontrolne enote NC-strojev je pripeljalo do sodelovanja pri novem projektu med MIT in General Electric, kjer so se jim pridružili Giddings & Lewis in Concord Control. Na razstavi izdelovalcev obdelovalnih strojev National Machine Tool Builders Association Show leta 1955 je podjetje Giddings & Lewis predstavilo prvi tržni model pet-osnega NC-frezalnega stroja.

Cincinnati Milling so se odločili za sodelovanje z britanskim podjetjem EMI (Electrical and Musical Instruments) za izdelavo kontrolnih enot. Sodelovanje se je izjalovilo, zato so jih pri Cincinnati Milling začeli izdelovati samostojno. Eden prvih je bil Cincinnati Hydrotel z navpičnim vretenom, ki so jih izdelali s sodelovanjem z MIT. Krmilje je bilo izvedeno z elektronkami, omogočalo je hkratno premikanje treh osi in dobivalo podatke preko dvojiško kodiranih luknjanih trakov. S temi stroji je bilo možno izdelati vedno zahtevnejše integralne dele za potrebe letalske industrije. Šlo je za izdelke, ki jih je bilo možno natančno opisati z malo matematičnimi podatki, vendar zelo težko ročno izdelati.

Leta 1956 Kearney & Trecker izdela NC krmiljeni frezalni stroj in ga škandalozno imenujejo »prvi komercialni NC frezalni stroj v ZDA«. Najbolj poznan model tega časa je zagotovo »Milwaukeeomatic«. Podjetje sodeluje s podjetjem Bendix (Detroit) in skupaj dobavljata stroje družbi Aircraft Manufacturer Northrop Company. Bendixov oddelek za raziskave se je začel poglobljati v numerično krmiljenje pod vodstvom gojenca iz Servomechanics Laboratory iz MIT. Med prvimi kupci strojev podjetja Kearney & Trecker so bili Hughes Aircraft Company (Los Angeles). Takrat se je vizionarski Howard Hughes, ki je poleg filmske družbe 20th Century Fox, vodil Hughes Aircraft Company, v kateri je bilo zaposlenih 30.000 delavcev, odločil, da bo razvil nov proizvod – kontrolno enoto, ki jo bo prodajal vključno z obdelovalnim strojem. Prvi NC obdelovalni stroj je sicer kupil od Kearney & Trecker, toda izkoristil je nizek tečaj nemške marke proti dolarju (1:4.20) in zato ni čudno, da je prve stroje izdelalo nemško podjetje Burkhart&Weber (Reutlingen), ki so stroje konstruirali in izdelovali, Hughesovo podjetje Hughes Tool Company pa jim je dostavljalo kontrolne enote. Prodajo sta si razdelila na zgodno Evropo brez Švedske (Burkhart&Weber) in preostali svet (Hughes). Serije strojev so bile imenovane MT3 in MT3 A. Stroji so bili izdelani na tak način, da so klasičnim vrtalnim strojem dodali kontrolno enoto. Burkhart&Weber so vzeli zadevo resno in ustanovili posebno organizacijsko enoto za NC stroje, ki je predstavljala 30% vseh zaposlenih. Stroje MT3 A so najprej prodali Francozom (Snecma, Paris), kasneje pa še 150

strojev Američanom v sredini 70-ih let. 1966 so začeli z lastnim razvojem in izdelali MC 4 z popolnoma novo zasnovo. Dobiček podjetja Burkhart&Weber ni nikoli prerasel stroškov velikih projektov, ki so se jih lotevali. 1982 jih je kupilo švicarsko podjetje Georg Fisher in jih 1994 prodalo. 1992 in 2001 so razpisali stečaj in so trenutno v lasti Italijanov (Riello).

Ker nova tehnologija, glede na slabo prodajo, v ZDA ni obrodila pričakovanih sadov, je Japonsko podjetje Yamazaki Mazak uspelo pridobiti licenco od Hughes Aircraft (prej so poskušali pridobiti licenco pri Burkhart&Weber) za stroje MT3 A. Prej so izdelovali le stružnice. Februarja 1966 so pri Yamazaki Mazak začeli z raziskovanjem NC obdelovalnih strojev. Na podlagi licence so zgradili stroj MT3 A in ga podrobno preučili. Jeseni 1968 so predstavili svoje stroje, stružnice 800R, 1000M in 1500R ter frezalni stroj BTC5 in se tako zavihтели na sam vrh na trgu proizvajalcev obdelovalnih strojev.

V Nemčiji je leta 1957 podjetje Schiess AG predstavilo prvi frezalni in vrtalni stroj, ki ga je krmilil Brown, Bowerie and Cie. krmilnik. Podjetje se je leta 1992 združilo z Dorries Scharmman in dve leti zatem bankrotiralo. Leta 1959 je podjetje Pittler AG izdelalo prvo NC stružnico, leta 1959 pa je bil predstavljen že prvi večnamenski obdelovalni stroj.

Na razstavi IMTE leta 1960 v Chicagu je bilo predstavljenih že 60 različic NC strojev.

1.2.2 Uporaba mikroročunalnikov – druga generacija

Tehnološki razvoj je razdelil trg na dva dela, na visoko produktivne stroje in stroje novih tehnologij (velike hitrosti obdelave, velika natančnost). V prvem delu so prednjačili Japonci. V isto kategorijo lahko uvrstimo tudi Tajvance, Korejce in Kitajce, ki so skušali slediti Japoncem. Nemci so bili znani predvsem po strojih izrednih kakovosti, kar jim je omogočilo, da so postali izvoznik NC obdelovalnih strojev številka ena v 70-ih letih. Več švicarskih podjetij, vezanih na urarsko industrijo, se je ukvarjalo z natančnostjo strojev; razvili so stroje sposobne izdelati zahtevne izdelke z izjemno natančnostjo.

Če si izberemo za izhodišče Nemčijo v začetku 50-ih let s 100 točkami, ima Švica s svojimi reprezentančnimi podjetji kar 120 točk, kar se tiče natančnosti obdelovalnih strojev. Podjetja v Veliki Britaniji, ZDA in Italiji so precej v ozadju s svojimi 80 točkami. Merilni in prilagodljivi sistemi na numerično krmiljenih obdelovalnih strojih so Japonce povsem približali z natančnostjo Švicarjev in Nemcev.

Znižanje cen mikroprocesorjev in elektronskega spomina je povzročilo padec cen krmilnih enot numerično krmiljenih strojev, kar je omogočilo Japoncem, ki so že v začetku razvoja vgrajevali mikroprocesorje v numerično krmiljene stroje, da so s konkurenčno ceno povsem prehiteli Nemčijo kot največjo izvoznico numerično krmiljenih strojev.

Prvotno so Japonci izdelovali obdelovalne stroje pod licenco tujih podjetij; npr. pri Mitsubishi Heavy Industries so od švicarskega podjetja Oerlikon kupili tehnologijo za izdelavo kopirnih stružnic, tehnologijo za vrtalne in frezalne stroje so kupili od italijanskega podjetja Innocenti, tehnologijo za stroje za obdelavo zobnikov pa so odkupili od Nemcev (Lorenz).

Cena je postala prevladujoč faktor pri izbiri obdelovalnega stroja. Med 1970 in 1980 je povprečna cena japonskega NC stroja znašala polovico ali celo tretjino cene ameriških strojev, ki so ceno NC strojev zmanjševale na podlagi nižjih plač.

Za primer omenimo uspeh podjetja Fanuc. Podjetje Fujitsu je 1972 ustanovilo neodvisno podjetje Fujitsu-Fanuc s svojo raziskovalno skupino. Do leta 1980 so postali največji svetovni proizvajalec s 50% svetovnim tržnim deležem in z 80% tržnim deležev na Japonskem.

1.2.3 Uporaba osebnega računalnika – tretja generacija

Uporaba osebnega računalnika (PC – personal computer), ki je začela 1980 silovit pohod po obdobju vztrajnih raziskav, je pomenila poenoteno, cenejšo in enostavnejšo obdelavo podatkov. Prvi računalniško krmiljeni NC stroj se je pojavil v letu 1990.

Podatki o obdelavi so potovali v eni smeri: iz osebnega računalnika do krmilne enote NC stroja in od krmilne enote do stroja. Nekatera podjetja (npr. Yamazaki Mazak) so v krmilne enote vgrajevali operacijske sisteme podobne operacijskim sistemom v osebnih računalnikih. Kasneje so osebne računalnike začeli vgrajevati v krmilno enoto, tako je osebni računalnik postal krmilna enota, ki hrani podatke o planu dela, navodilih, delu in strojni diagnostiki.

Obdobje tretje generacije traja še danes.

1.3 Krmiljenje obdelovalnih strojev

Ob zgodovinskem pregledu razvoja obdelovalnih strojev in numerično krmiljenih obdelovalnih strojev lahko krmiljene obdelovalne stroje razdelimo v naslednje skupine:

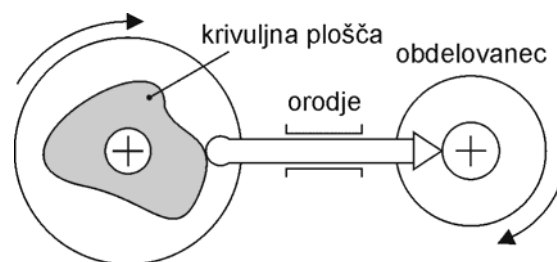
- mehanično krmiljeni obdelovalni stroji,
- kopirno krmiljeni obdelovalni stroji in
- numerično krmiljeni obdelovalni stroji [9].

1.3.1 Mehanično krmiljenje obdelovalnih strojev

Krmiljenje mehanično krmiljenih obdelovalnih strojev se navadno izvaja s pomočjo krivulj:

- na ploščah ali
- bobnih.

Slika 8 prikazuje, kako se rotacijsko gibanje krivuljne plošče neposredno pretvori v premočrtno gibanje orodja. Pretvorba v premočrtno gibanje se lahko izvaja tudi posredno, preko mehanizma.



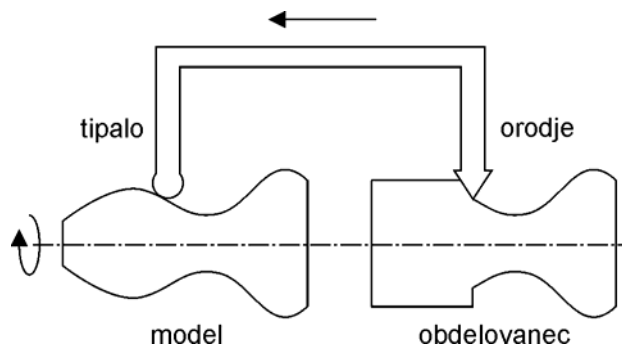
Slika 8: Mehanično krmiljenje s krivuljno ploščo

Mehansko krmiljeni stroji so namenjeni pretežno izdelavi velikih serij izdelkov, zato prihaja, navkljub kvalitetnim izvedbam vrtljivih in gibljivih mehanizmov, do njihove obrabe, kar pa povzroča nenatančnosti in napake pri izdelavi. Naslednji, mehničnemu krmiljenju sorodni, način krmiljenja je kopirno krmiljenje.

1.3.2 Kopirno krmiljenje obdelovalnih strojev

Razvoj kopiranja se je začel v 18. stoletju z ročno krmiljenimi napravami in se nadaljeval z njihovo avtomatizacijo v začetku 20. stoletja. Kopirno krmiljenje je namenjeno predvsem obdelavi z odrezovanjem geometrijsko zahtevnih obdelovancev.

Naloga kopirnega sistema obdelovalnega stroja je, da podatke o obdelavi z modela (mehanični pomnilnik) neposredno ali posredno prenese na obdelovanec. Podatki se lahko ojačajo ali ne. Slika 9 prikazuje poenostavljen mehanski kopirni sistem na stružnici. Gibanje tipala po modelu se prenaša na orodje, ki obdeluje obdelovanec. Glavna pomanjkljivost mehaničnih kopirnih sistemov je, da kopirnih podatkov, ki se z modela prenašajo preko vzvodov, ni mogoče ojačati. Zaradi tega deluje povratna rezalna sila na model, ki se hitro obrabi.



Slika 9: Mehansko kopirno krmiljenje

Pri električnem, hidravličnem in pnevmatskem prenosu kopirnih podatkov iz modela na obdelovanec, pa le-te lahko ojačamo. Sistem za tipanje in naprava za premikanje orodja sta pri takih sistemih ločena in prinašata naslednje prednosti:

- sile, ki delujejo na tipalo, so minimalne,
- zaradi majhnih mas in vztrajnosti tipala je kopiranje natančno in
- naprava za premikanje orodja je lahko močna in toga, da lahko premaguje velike torne in reakcijske sile nastale na podlagi ojačane tipalne sile.

Primer neposrednega krmiljenja kopiranja predstavlja numerično kopiranje. Pri posrednih kopirnih strojih so podatki odčitani s tipalom in obdelovalni podatki časovno in prostorsko povezani, kar se odraža na naslednjih ovirah in pomanjkljivostih:

- ne moremo spreminjati odtipanih podatkov,
- nihanje in tresenje stroja slabo vpliva na tipanje modela,
- model moramo skladiščiti dalj časa kot je čas proizvodnje (rezervni deli) in
- uporaba je omejena le na izdelavo izdelkov, katerih lastnosti ne poznamo.

Zato se pri numeričnem kopiranju podatki, odčitani z modela, shranijo v računalniku kot številčni podatki, ki jih lahko po želji spreminjamo. Tipala so lahko mehanska ali nemehanska. Preden bi numerično kopiranje do potankosti spoznali, bomo raje spoznali računalniško krmiljenje stroje.

1.3.3 Numerično krmiljenje obdelovalnih strojev

Numerično krmiljenim strojem pravimo kar NC-stroji. Beseda NC izhaja iz angleških besed »numerical control«, kar pomeni številsko, numerično upravljanje. Obdelovalnemu stroju se

posredujejo podatki v numerični obliki. Poseben del stroja, t.i. krmilje, pa te podatke obdela in jih posreduje izvršnim elementom stroja (npr. elektromotorjem).

Za izdelavo izdelka potrebujemo podatke o:

- geometriji obdelovanca in izdelka,
- obdelovalnih pogojih,
- orodjih,
- stroju in
- obdelovalnem poteku.

Izvor podatkov so torej risba izdelka, obdelovanec, stroj, izbor orodij in delovni načrt. Na osnovi zbranih podatkov se izdelajo navodila – program, ki predstavljajo relativno gibanje med orodjem in obdelovancem. Program se lahko izdelava ročno ali samodejno. Na kratko je ta tematika opisana v naslednjem poglavju.

2. PROGRAMIRANJE NUMERIČNO KRMILJENIH OBDELOVALNIH STROJEV

Z razvojem numerično obdelovalnih strojev se je postopoma pojavila potreba po digitalnih računalnikih, ki bi bili sposobni krmiljenja celotnega obdelovalnega stroja s senzorji in krmilniki v realnem času. Prihodom digitalnih računalnikov pa pomeni korenito spremembo v zasnovi programiranja NC obdelovalnih strojev.

Nastali so prvi računalniški jeziki, v katerih so bili ukazi zapisani v angleščini za razliko od ukazov s kodiranimi ali kriptogramskimi znaki. Tako je bilo možno ukaze spreminjati in dopolnjevati [32].

Večina industrijskih držav so začele z nacionalnimi programi, da bi pospešile razvoj nove tehnologije.

V Nemčiji je pet univerz sodelovalo z industrijo: Berlin (Prof. Spur), Stuttgart (Prof. Ehrhardt, Prof. Stute), Aachen (Prof. Opitz), Karlsruhe (Prof. Warnecke) in Hannover (Prof. Kienzle, Prof. Osenberg). Ukvarjali so se s struženjem, frezanjem, vlivanjem, računalniškim krmiljenjem in računalniškim konstruiranjem. Kot rezultat sodelovanja je bilo izdelanih več direktno krmiljenih sistemov (DNC – direct numbering control) in odmeven programski sistem EXAPT. Pri direktnem krmiljenju so računalnik povezali s krmilnimi enotami več NC obdelovalnih strojev.

Podobne korake so ubrali tudi na Norveškem, kjer so štiri najboljša podjetja sodelovala z norveško univerzo (University of Norway, Central Institute of Industrial Research) pri izdelavi nacionalnega programskega sistema za programiranje NC obdelovalnih strojev.

V Franciji se je v sodelovanju združilo 50 največjih proizvajalcev.

Programski sistem za avtomatizacijo proizvodnega procesa so razvijali tudi na Nizozemskem.

Velika Britanija se razen z izjemo »600 group« v osvajanju nove tehnologije ni posebej izkazala.

V Ameriki je Air Force s tremi milijoni dolarjev finančno podprl projekt na MIT z imenom Investigations in Computer-Aided Design for Numerically Controlled Manufacturing Processes (Raziskave računalniško podprtega konstruiranja za numerično krmiljene proizvodne procese) [76]. Projekt, ki ga je vodil Douglas Taylor Ross, je trajal od leta 1959 do leta 1967. Projekt je logično nadaljevanje razvoja APT (Automatically Programmed Tools) jezika – jezika za samodejno programiranje strojev [5], zato bomo naslednje poglavje

posvetili opisu izvirnim idejam modernega programiranja numerično krmiljenih strojev. S sočasnim razvojem računalniško podprtega konstruiranja in računalniško podprtih orodij za samodejno programiranje obdelovalnih strojev je nastal leta 1954 prvi dvo dimenzionalni grafični programski vmesnik za prostoročno risanje in kasneje leta 1955 prva ineraktivna tipkovnica [75][32].

2.1 Zametki samodejnega programiranja NC strojev

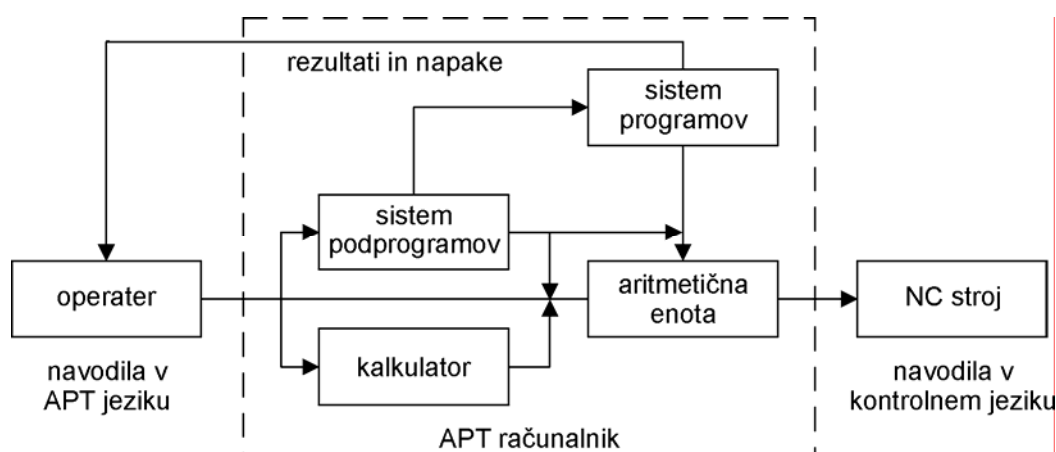
28. marca 1959 so v časopisu New Yorker Magazine zapisali, da so na MIT izdelali stroj, ki je sposoben sprejemati navodila v angleščini. Predstavljen je bil pepelnik-spominek izdelan s prvim APT sistemom v treh dimenzijah. Prej so znanstveniki predstavili enak APT sistem, 25. januarja 1959, na konferenci na MIT-ju sponzorirano s strani Aircraft Industries Association (AIA) [75].

Idejni oče APT jezika, Douglas Taylor Ross, je pri snovanju jezika za samodejno programiranje numerično krmiljenih strojev predpostavil sledeče:

- Ker je celotno področje samodejnega programiranja numerično krmiljenih strojev precej novo, bo potrebno največ truda vložiti v semantiko (pomenoslovje) jezika, da bo prišlo do njegove splošnejše uporabe.
- Jezik naj bo grajen modularno (iz več delov), z možnostjo spreminjanja in dopolnjevanja.
- Sintaktični (sestavljanje programskih stavkov) in semantični (pomen programskih stavkov) del sistema – prevajanje in izvrševanje, ne smeta biti ločena.
- Sistem mora biti neodvisen od geometrije površin, podpirati mora različne tipov obdelovalnih strojev in njihovih krmilnih enot.

Air Force je 1956 začel s sponzoriranjem projekta, za katerega jih je navdušil Arnold Siegel iz Digital Computer Laboratory s prototipom prvega jezika za samodejno programiranje strojev. Demonstracijo je izvedel na dvo dimenzionalnem primeru, katerega osnovni gradniki so bile ravne linije in krogi.

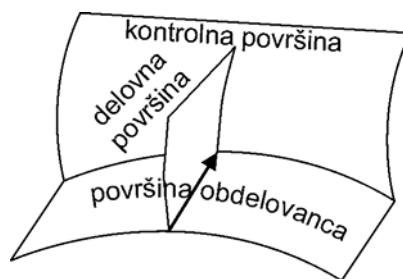
Koncept je bil zasnovan glede na računalnikovo strukturo. Program za obdelavo vhodnih pseudo navodil bi navodila prevedel, v krmilni enoti, poznana navodila, ki bi izbirala različne načine obnašanja aritmetične enote. Aritmetična enota bi posredovala rezalne vektorje (opis poti orodja) v različnih oblikah za različne numerično krmiljene stroje. Slika 10 prikazuje idejno zasnovo APT programiranja Douglas Taylor Rossa. Izpopolnjen, delujoči koncept je prikazan na sliki 10.



Slika 10: Idejna zasnova APT programiranja

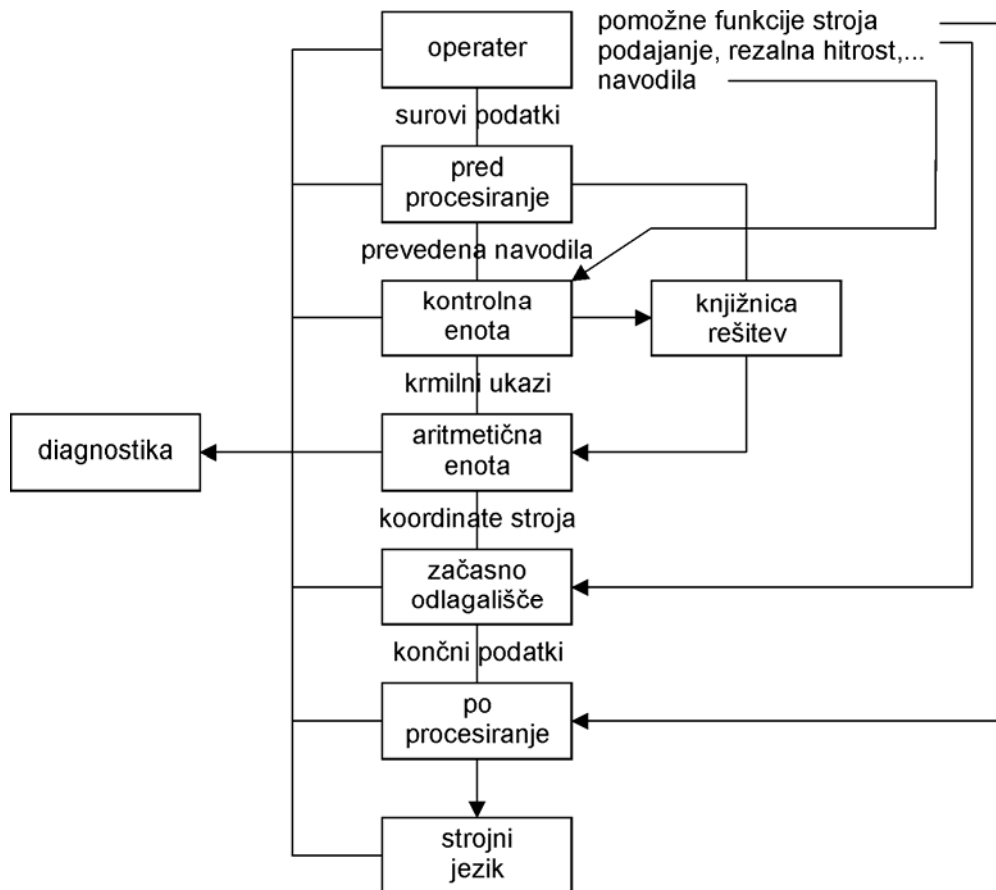
Snovalci novega jezika so se zavedali, da bodo težko ločili računalniško podprto konstruiranje in samodejno programiranje NC strojev, kar pomeni, da bo težko samodejno programirati stroj, da izdelava izdelek, brez da bi prej natančno opisali njegove geometrijske značilnosti. Osnove jezika torej morajo izhajati iz geometrije izdelka in obdelovanca, zato so leta 1956 prvič uporabili besedno zvezo part programming (part-obdelovanec, programming-programiranje).

Slika 11 prikazuje zasnovo part programming. Orodje se giblje v delovni površini (angl. driving surface) nad površino obdelovanca (angl. part surface) proti kontrolni površini (angl. check surface). V začetnih stopnjah razvoja jezika APT so se odločili, da bo orodje predstavljeno kot točka, ki se odsekoma giblje po presečišču površine obdelovanca in delovne površine. Presečišče obeh ravnin sestavljajo ravni segmenti. V naslednjih, izboljšanih stopnjah part programminga, so želeli ravne odseke zamenjati s krivuljami, kasneje pa bi idejo o samodejnem programiranju razširili še z uporabo regij.



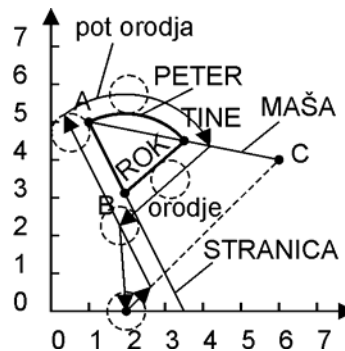
Slika 11: Zasnova part programming geometrije

Tok informacij pri APT programiranju s krivuljami, ki se bistveno ne razlikuje od toka informacij sodobnih sistemov za samodejno programiranje NC strojev, prikazuje slika 12.



Slika 12: Tok informacij pri APT programiranju s krivuljami

Za lažjo predstavitev in razumevanje koncepta jezika APT prikazujemo enostaven primer programiranja poti orodja NC frezalnega stroja. Program (Slika 13) vsebuje geometrijske podatke, na katerih so osnovani gibi orodja. Če se vrnemo na opis slike 12, se program, ko ga zapiše operater pred procesira, predela v računalniku razumljivo obliko. Kontrolna enota poskrbi za klicanje in uporabo podprogramov iz knjižnice obstoječih rešitev, ki jih nato prenese aritmetični enoti, ki poskrbi za izračun poti orodja. Podatki se shranijo v začasno odlagališče, da jih lahko po potrebi po procesiramo in prilagodimo strojnemu jeziku posamičnega stroja.



```

A = POINT / 1, 5
B = POINT / 2, 3
C = POINT / 6, 4
TL DIA / +1, MM
FEDRAT / 500, MPM

START = FROM, POINT / 2, 0 IN DIR, POINT / C

STRANICA = GO TO, LINE / THRU, A, AND, B
           WITH, TL LFT, GO LFT, ALONG / STRANICA

PETER GO RGT, ALONG, CIRCLE/WITH, CTR AT, B, THRU, A

MAŠA LINE / THRU, A, AND, C
TINE POINT / X LARGE, INT OF, MAŠA, WITH, PETER

ROK LINE / THRU, TINE, AND, B
GO RGT, ALONG / ROK, UNTIL, TOOL, PAST, STRANICA

GO TO / START
STOP, END, FINI

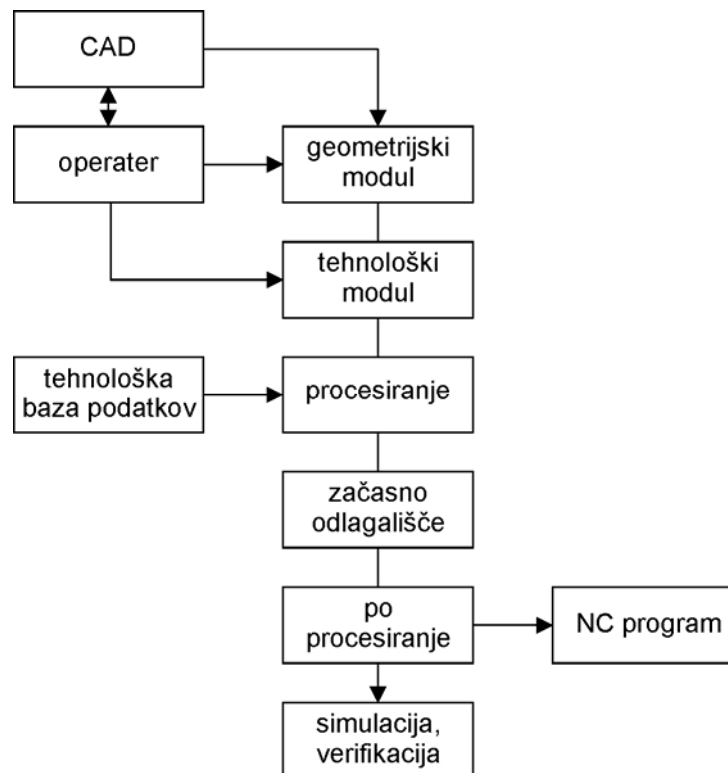
```

»točka A(1,5)«
 »točka B(2,3)«
 »točka C(6,4)«
 »premer orodja je 1mm«
 »hitrost podajanja je 500mm/min«
 »orodje potuje od točke START(2,0) v smeri točke C«
 »orodje potuje po ravni črti od točke A do B«
 »orodje potuje po levi vzdolž STRANICA«
 »orodje potuje vzdolž kroga, s središčem v točki B, do točke A«
 »ravna črta od A do C«
 »presečišče s krogom PETER in ravno črto MAŠA, kjer je X koordinata večja (na desni strani kroga)«
 »ravna črta od TINE do B«
 »orodje potuje po desni ravne črte ROK, dokler ne pride do STRANICA«
 »orodje potuje do točke START«
 »konec programa«

Slika 13: Preprost APT program

2.2 Sodobni koncept

V svetu je danes razvitih veliko sistemov za samodejno projektiranje tehnologije, ki predstavljajo različne stopnje avtomatizacije procesa programiranja [10][9][15][11]. V splošnem pa se ti sistemi ne razlikujejo od prvotno zasnovanih sistemov v 50-ih letih 20. stoletja. Zgrajeni so kot prikazuje slika 14.



Slika 14: Princip sodobnega programiranja

Vhodne informacije se lahko oblikujejo s posebnimi programskimi jeziki (npr. APT) ali pa s pomočjo CAD sistemi, pri katerem lahko uporabimo geometrijo preko ustreznega standardnega formata. Več o standardih lahko najdete v [11].

Tehnološki modul predstavlja predvsem banka tehnoloških podatkov, ki omogoča samodejno izbiro tehnologije obdelave glede na geometrijo obdelovanca, izdelka, izbrano orodje in stroj.

Processor je sestavljen iz programov, ki določa pot orodja na podlagi izbire tehnološkega modula po vnaprej programirani strategiji obdelave. Strategijo obdelave sistem samodejno izdela na podlagi različnih kriterijev, ki so lahko minimalni čas obdelave, minimalni stroški, kvaliteta obdelave ipd. Večina obstoječih procesorjev nima modula za določanje optimalne obdelave glede na izbrane kriterije.

Naloga poprocesorja je, da s procesorjem obdelane podatke predela v obliko namenjeno za točno določen NC stroj in NC krmiljem.

Navadno lahko poprocesirana navodila za krmiljenje NC strojev uporabnik simulira in preveri njihovo delovanje na računalniku.

Razvoj sistemov za samodejno programiranje NC strojev se širi v:

- razvoj obstoječih sistemov,
- povezovanje sistemov za samodejno programiranje NC strojev z načrtovanjem tehnologije in
- razvoj novih sistemov.

Razvoj obstoječih sistemov zajema višanje stopnje avtomatizacije, nižanje količine vhodnih podatkov, razvoj modulov za optimiranje obdelave, ločitev poprocesorja od sistema za samodejno programiranje in modularno gradnjo programskih sistemov.

Pri povezavi sistemov za samodejno programiranje in načrtovanja proizvodnje bi pripeljalo do skupnega koriščenja geometrijskih in tehnoloških bank podatkov, kar bi lajšalo količino in vnos potrebnih vhodnih podatkov.

Novi programski sistemi se razvijajo na tistih področjih, kjer se NC krmiljenje šele začnja uveljavljati. To so:

- obdelava brez odvzemanja materiala: izsekovanje, kovanje, upogibanje,
- področja, kjer to terja razvoj NC strojev in krmilij: stroji z več prostostnimi stopnjami, interpolacije višjega reda in
- področja, kjer konvencionalni programski sistemi ne zadostujejo.

Kot smo že uvodoma omenili je evolucija splošno načelo, kar pomeni, da je prisotno tudi pri obdelovalnih strojih in programiranju le-teh. V daljšem časovnem obdobju je človek razvil samodejno krmiljene stroje, ki jih uporablja še danes. Vmesno je na tem mestu omeniti idejo, da lahko umetno ustvarjeni evoluciji podvržemo tudi NC program, ki skrbi za krmiljenje obdelovalnega stroja. Evolucija poskrbi za spreminjanje in izbiranje le tistih navodil, ki bolje zadostijo zahtevam po natančnosti, hitrosti, ekonomičnosti in varnosti obdelave. Umetno evolucijo lahko ustvarimo s pomočjo vse bolj uveljavljenih metod evolucijskega računanja, ki jih bomo na kratko opisali v sledečem poglavju.

3. METODE EVOLUCIJSKEGA RAČUNANJA

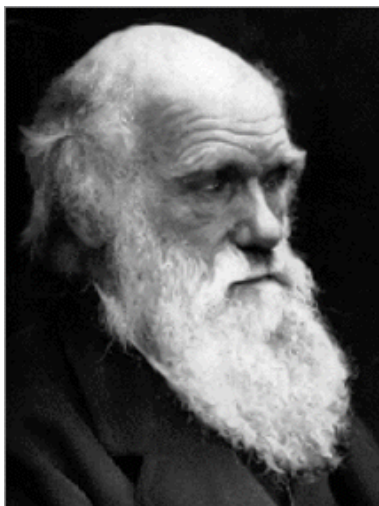
V tem poglavju bomo napravili kratek pregled metod evolucijskega računanja. Metode oponašajo evolucijo živih organizmov in temeljijo na Darwinističnih principih genetskega kombiniranja in izboru (preživetju) najsposobnejših članov. Nekateri avtorji imenujejo postopke, ki posnemajo evolucijo živih organizmov tudi genetske metode.

3.1 Evolucija živih bitij

Do konca 18. stoletja je vladalo med naravoslovci prepričanje, da je pestrost življenja na svetu rezultat enkratne stvaritve, kakršna je opisana v Bibliji. Odkritja orjaških okamnelih kosti pa so francoskemu naravoslovcu in baronu Georgesu Cuvieru (1779-1832) vzbudila prve dvome o tem. Dokazal je, da fosilne kosti ne pripadajo nobeni danes živeči živalski vrsti, če pa so živali z orjaškimi kostmi že izumrle, potem svet ni nespremenljiv. V začetku 19. stoletja so mnogi misleci že priznavali razvoj življenja na Zemlji, ki so ga zasledovali s preučevanjem fosilnih ostankov v različno starih geoloških skladih. Rojevala se je ideja o evoluciji oziroma postopnem spreminjanju vrst. Večina naravoslovcev je bila že prepričana o evoluciji, pripravili so se le še o možnih načinih spreminjanja rastlinskih in živalskih vrst [77][89].

V prvi polovici 19. stoletja sta med znanstveniki prevladovali predvsem dve razvojni domnevi. Francoski naravoslovec Jean-Baptiste Lamarck (1744-1829) je zastopal teorijo o postopnem spreminjanju v evoluciji organizmov [61]. Njegov najhujši nasprotnik je bil že omenjeni baron Cuvier [27], ki je bil prepričan, da je evolucija potekala zaradi mnogih zaporednih katastrof in nenadnih zamenjav enih organizmov z drugimi. Lamarck je 1809 razložil svojo hipotezo o razvoju s pomočjo adaptacij, po kateri naj bi se pri organizmih skozi vse življenje razvijali določeni organi, s pomočjo katerih so se lažje prilagajali življenjskim razmeram. Med razmnoževanjem so osebki tako pridobljene prilagoditve prenašali na svoje potomce. Po tej hipotezi se je žirafam postopoma razvijal vse daljši vrat zato, ker so se stegovale za hrano v visokih drevesnih krošnjah.

Angleški naravoslovec Charles Darwin (Slika 15) je razvil teorijo o evolucijskih spremembah [28], ki je popolnoma nasprotovala Lamarckovi ideji o dedovanju med življenjem pridobljenih značilnosti. Darwin je prve ideje o zgodovinskem razvoju živih bitij dobil na znanstvenem popotovanju okrog sveta (1831-36), ki se ga je udeležil kot naravoslovec na kraljevi ladji Beagle. Svojo teorijo je izoblikoval šele 1858 in jo skupaj z Alfredom Russelom Wallaceom (1823-1913) [81] objavil v znanstvenem članku, naslednje leto pa v knjigi z naslovom Nastanek vrst in naravni izbor. Zaradi zanikanja v Svetem pismu opisane stvaritve življenja na Zemlji sta Darwinova teorija in tudi njen avtor doživljala hude napade Cerkve pa tudi mnogih naravoslovcev. Darwinova evolucijska teorija je postala širše priznana šele potem, ko so jo podprli T. H. Huxley (1825-95) in drugi znani naravoslovci.



Slika 15: Charles Darwin (1809-82)

3.1.1 Naravni izbor

Ob začetku svojega dolgega popotovanja po svetu je bil tudi Darwin prepričan o enkratnem nastanku in poznejši nespremenljivosti rastlinskih in živalskih vrst. Prve dvome so mu zasejala šele opazovanja življenjskih oblik v patagonskih pampah. Tam je namreč odkril fosilne ostanke raznih izumrlih sesalcev, ki so bili po obliki zelo podobni nekaterim še živečim vrstam. Med obiskom na otočju Galapagosu ga je britanski guverner opozoril, da lahko po vzorcih na oklepih orjaških kopenskih želv zanesljivo ugotovi, s katerega otoka so. To je bil za Darwina eden najzanesljivejših dokazov, da se posamezni osebki neke vrste zaradi izolacije postopoma spreminjajo in da se lahko iz ene vrste postopoma razvije več novih [77][89].

Darwin je seveda s svojimi sodobniki delil mnenje, da fosilni ostanki izumrlih organizmov kažejo zgodovino sprememb rastlinskih in živalskih oblik, vendar pa se še ni prav zavedal pomena izumrlih vrst za evolucijo. Darwinova naravoslovna nadarjenost je bila v tem, da je pokazal na mehanizme, ki bi lahko povzročali opazovane spremembe pri vrstah. Naravoslovcem je bilo dobro znano, da pri razmnoževanju skoraj vedno nastane veliko več jajčec in da se razvije več potomcev, kot pa jih potem doseže spolno zrelost. Darwin je tudi poznal osnove izbiranja dobrih in preprečevanja nezaželenih lastnosti pri gojenju domačih živali in rastlin. Med svojim popotovanjem je spoznal, da podobno izbiranje ali selekcija poteka tudi v naravi, le da jo namesto rejcev opravljajo razni naravni mehanizmi. Zaradi drobnih razlik v dednih lastnostih so posamezni organizmi neke vrste po naključju bolje prilagojeni na svoje življenjsko okolje od svojih vrstnikov. Zato imajo večje možnosti za preživetje, svoje »dobre« lastnosti pa prenašajo naprej na generacije svojih potomcev. V naravi preživijo samo najsposobnejši osebki, zaradi ohranjanja in nabiranja »koristnih« dednih sprememb skozi mnogo generacij pa v tisočletjih nastajajo nove vrste.

Ena največjih pomanjkljivosti Darwinove teorije o naravnem izboru je bila v tem, da ni razložila nastajanja dednih sprememb pri organizmih. Pojav novih dednih lastnosti je postal razumljiv šele po odkritju pomena molekul DNK (dezoksiribonukleinska kislina) za dedovanje.

3.1.2 Genetika

Za razvoj eksperimentalne genetike so najpomembnejša dela češkega meniha Gregorja Mendla (Slika 16). Na svojem vrtu v brnskem samostanu je križal razne sorte graha in nato

skozi več generacij opazoval obliko semen in še razne druge lastnosti ter prešteval potomce s podedovanimi lastnostmi. Svoja opazovanja je usmeril samo na nekaj lastnosti in ko je določal delež potomcev, pri katerih so se iz generacije v generacijo pojavljale opazovane lastnosti, je ugotovil posebne načine in pravila prenašanja dednih lastnosti. Šele v tem stoletju so dokazali, da sta Mendlovi načeli o cepitvi in o neodvisni razporeditvi dednih lastnosti uporabni za preučevanje dednosti pri skoraj vseh rastlinskih in živalskih vrstah [77][89].

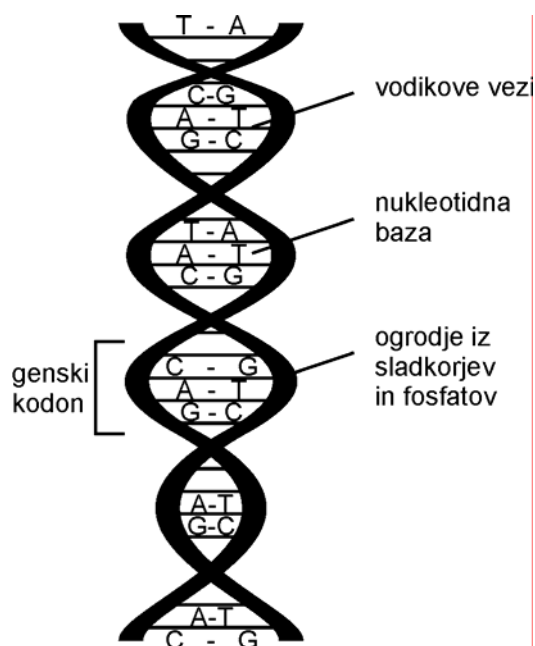


Slika 16: Gregor Mendel (1822-84)

3.1.3 DNK – nosilec informacij

Že na začetku 20. stoletja je bilo znano, da organizmi od svojih staršev ne podedujejo vseh lastnosti, ampak se lastnosti staršev v potomcih na razne načine prerazporedijo in kombinirajo, tako da se posamezne lastnosti dedujejo večinoma neodvisno ena od druge. Tedanji genetiki so domnevali, da podatke o dednih lastnostih prenaša neka organska spojina, vendar pa prave prenašalke še niso poznali. Vedeli so, da mora biti prenašalska molekula zelo spremenljiva in dovolj velika za shranjevanje velike količine informacij o vseh dednih lastnostih nekega organizma. Mnogi so to vlogo pripisovali posebnim beljakovinskim molekulam, dokler ni 1944 ameriški mikrobiolog Oswald T. Avery (1877-1955) s poskusom dokazal, da se lahko dedne lastnosti bakterij spremenijo, kadar vanje od zunaj pride tuja dezoksiribonukleinska kislina (DNK) [48].

DNK (Slika 17) je osnovna dedna snov skoraj vseh živih bitij. Velike molekule so sestavljene iz ogromnega števila sestavnih delov, vendar je njihova zgradba v osnovi vseeno presenetljivo preprosta. Posamezna molekula DNK je iz dveh ločenih nizov, ki se prečno povezujeta in spiralasto ovijata drug okrog drugega ter na ta način oblikujeta dvojno spiralasto vijačnico. Vsak niz je iz velikega števila štirih različnih vrst nukleotidov. Vse vrste nukleotidov imajo enako osnovno zgradbo in so sestavljeni iz molekule sladkorja (dezoksiriboza) in fosfatnega ostanka. Molekula fosfatnega ostanka se naveže na sladkorjevo molekulo drugega nukleotida in na ta način nastaneta stranska niza ali ogrodje molekule DNK. Štiri vrste nukleotidov se med seboj razlikujejo samo po posebnih dušikovih bazah (nukleotidne baze), ki so vezane na sladkorjevo molekulo. Med obema stranskima nizoma molekule DNK so štiri različne baze - adenin (A), citozin (C), gvanin (G) in timin (T). Baze enega stranskega niza se s pomočjo vodikovih vezi povezujejo z ustreznimi komplementarnimi bazami drugega niza.



Slika 17: Molekula DNK

S to enostavno zgradbo molekule DNK lahko zadovoljivo razložimo sposobnost samopodvajanja (reduplikacije) teh molekul in način branja šifriranih dednih zapisov za izdelavo aminokislin in beljakovin v molekuli DNK. Iz ene molekule DNK nastaneta dve popolnoma enaki molekuli tako, da se oba z nukleotidnimi bazami povezana stranska niza ločita, nato pa se na nukleotide vsake polovice navezujejo novi nukleotidi s komplementarnimi bazami.

3.1.4 Geni, genomi, mutacije, kromosom

Že Mendel je ugotovil, da se dedne lastnosti prenašajo iz ene generacije v drugo v obliki posameznih in nepovezanih enot. Po odkritju zgradbe molekule DNK so spoznali tudi obliko in načine prenašanja teh enot, ki so jih imenovali geni. Gen je del dvojne vijačnice DNK z mnogimi, v nizu razporejenimi nukleotidnimi pari. S številom in razporedom teh parov so dana vsa navodila za izdelavo nekega encima ali molekule ribonukleinske kisline (RNK), ki prenese informacije v citoplazmo. Vse dedne informacije nekega organizma oziroma vse gene skupaj imenujemo genom [77][89].

Mutacija je dedna sprememba, ki nastane zaradi spremenjenega zaporedja nukleotidnih baz na nekem genu. Naravne mutacije nastajajo predvsem med podvajanjem molekul DNK, tako da se v molekulo vključi dodatna nukleotidna baza, lahko pa se posamezna baza tudi izgubi ali pa zamenja z drugo, neustrezno nukleotidno bazo. Obseg nastajanja teh sprememb se lahko zelo poveča zaradi učinkovanja raznih kemikalij in ionizirajočih žarčenj (povzročene ali inducirane mutacije). Obsežne mutacije so večinoma škodljive in lahko motijo ali celo prekinajo življenjske funkcije prizadetega organizma. Naravne mutacije so lahko za organizme celo koristne, vsekakor pa so nujno potrebne za delovanje evolucijskih procesov.

Aleli so različni geni za isto dedno lastnost ali gen. Vsak diploiden organizem ima za vsak gen vsaj dva različna alela, od katerih je enega podedoval od matere, drugega pa od očeta. V okviru populacije in vrste imajo osebki za vsak gen večinoma mnogo različnih alelov, kar tudi zagotavlja potrebno variabilnost in delovanje evolucijskih mehanizmov.

Pri uporabi v molekuli DNK zapisanih dednih informacij mora celica kopirati zaporedje nukleotidnih baz iz gena na posebno molekulo RNK (gensko prepisovanje ali transkripcija). Te kopije imenujemo obveščevalne RNK. Večinoma nastajajo na vzdolžno razcepljenih delih molekul DNK v celičnem jedru (kjer je večina dednih informacij), od koder potem prenesejo podatke do ribosomov. To so majhni celični organeli v citoplazmi, na katerih poteka sinteza beljakovin. Ribosomi povezujejo molekule aminokislin v zaporedju, ki ga določajo trojčki nukleotidnih baz na obveščevalni RNK. Beljakovinske molekule so lahko encimi in kot katalizatorji omogočajo potek vseh biokemijskih reakcij v telesu.

Kromosomi so sestavljeni iz mnogih različnih vrst beljakovin, ki obdajajo in se tesno povezujejo z eno samo molekulo DNK; ta je med delitvijo jedra podvojena. Na vsakem kromosomu je mnogo genov. Vse celice nekega organizma imajo enako število kromosomov, ki pa so lahko različnih oblik. Razne rastlinske in živalske vrste imajo zelo različna kromosomska števila. Človeške celice imajo 23 različnih kromosomov. Ker pa so vsi kromosomi v parih, ima vsaka človekova celica po 46 kromosomov.

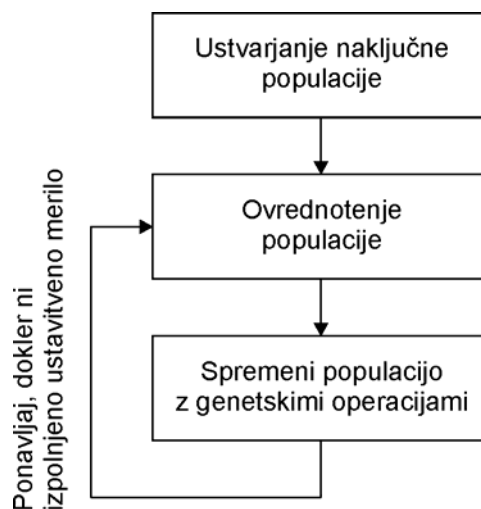
3.2 Skupne lastnosti metod evolucijskega računanja

Optimizacija je postopek, pri katerem se v prostoru možnih rešitev ciljne funkcije, ki opisuje problem, išče najboljša rešitev. Tradicionalne optimizacijske metode so se za te namene uspešno uveljavile, vendar predvsem pri optimiranju preprostih sistemov. Ker uporabljajo deterministične operatorje, jih spremljajo številne pomanjkljivosti kot so hiter zdrs v lokalni optimum, ciljna funkcija mora biti zvezna, odvedljiva in brez šuma ter neučinkovitost pri preučevanju zapletenih sistemov. Zaradi teh slabosti se je v zadnjih desetletjih za preučevanje, načrtovanje in napovedovanje obnašanja zapletenih sistemov uveljavilo precej novih optimizacijskih postopkov, ki temeljijo na zgledih iz narave. Za razliko od klasičnih optimizacijskih metod sodobni pristopi uporabljajo verjetnostna načela, zato nobena od zgornjih omejitev za njih ne velja. Nekateri verjetnostni računski postopki posnemajo zakonitosti iz neživega, drugi pa iz živega sveta. Najbolj znan primer iz neživega sveta je algoritem simulirano ohlajanje, ki oponaša model ohlajanja snovi. Primeri iz živega sveta so predvsem nevronske mreže in evolucijsko računanje. Metode evolucijskega računanja posnemajo načela biološke evolucije kot sta genetsko kombiniranje in naravni izbor najsposobnejših organizmov [19][18].

Glavni značilnosti evolucijskih metod sta, da se rešitve ne iščejo po vnaprej postavljenih (determinističnih) poteh in da se sočasno obravnava množica enostavnih objektov. Struktura rešitev je prepuščena evolucijskemu procesu. Ker smo pri reševanju problemov z evolucijskimi metodami zvesti biološkemu izhodiščem postopka, pravimo rešitvam organizmi ali tudi kromosomi. Zaradi verjetnostne narave metod evolucijskega računanja ni nobenega zagotovila, da prav vsaka evolucija pripelje do zadovoljivega izida.

Slika 18 prikazuje splošno shemo evolucijskih algoritmov. Reševanje problema pričnemo z (običajno) naključnim stvarjenjem rešitev (točk). Vsaka točka predstavlja bolj ali manj natančno rešitev problema. Nato organizme ovrednotimo in tistim, ki bolje rešijo dani problem (t.j. tistim, ki so bolj prilagojeni okolju), podelimo večjo verjetnost, da sodelujejo pri operacijah selekcije in spreminjanja. S selekcijo zagotovimo preživetje bolj uspešnih članov populacije in njihovo napredovanje v nespremenjeni obliki v naslednjo iteracijo, ki ji pravimo tudi generacija. S spreminjanjem vplivamo na enega ali več organizmov in iz njih ustvarjamo njihove potomce. Po opravljeni selekciji in spreminjanju dobimo novo generacijo,

ki jo spet ovrednotimo. Postopek ponavljamo tako dolgo, da je izpolnjeno ustavitveno merilo postopka. To je lahko največje predpisano število generacij ali pa zadostna kakovost rešitev.



Slika 18: Splošna shema evolucijskih algoritmov

Obstaja več različnih metod evolucijskega računanja. Najbolj znane so:

- genetski algoritmi,
- genetsko programiranje,
- evolucijske strategije in
- evolucijsko programiranje.

Razlikujejo se predvsem po vrsti struktur, ki so podvržene adaptaciji in po načinu tvorjenja novih rešitev. Čeprav so nekatere metode (npr. genetsko programiranje) stare le nekaj let, so bila teoretična izhodišča postavljena že v šestdesetih letih.

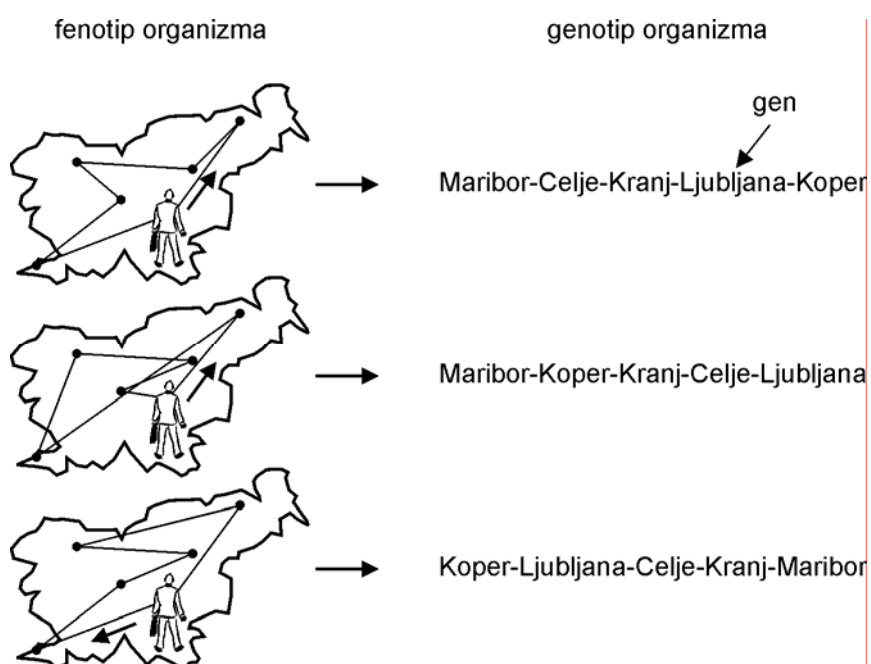
3.3 Genetski algoritmi

Med vsemi metodami evolucijskega računanja so se doslej najbolj uveljavili genetski algoritmi. Njihov tvorec je J. Holland [46], njihovo praktično uporabo na številnih področjih pa so veliko prispevali še D.E. Goldberg [38][39] in mnogi drugi avtorji [22][23][24][37][66].

Genetski algoritmi so poseben primer metod evolucijskega računanja, pri kateri so organizmi sestavljeni na razmeroma preprost način. Za primer vzemimo dobro znani problem trgovskega potnika [35]. Trgovski potnik mora obiskati vsa mesta v takšnem vrstnem redu, da se bo čim hitreje vrnil domov. Organizmi pri genetskih algoritmi so točke v prostoru rešitev, v našem primeru zaporedja mest, ki jih mora trgovski potnik obiskati. Predstavitev organizma z nizom imenujemo genotip, dejansko rešitev pa fenotip. Organizem ima koordinate, ki jim pravimo geni. Na sliki 20 je prikazana populacija treh organizmov (genotip) in njihovih predstavitev v obliki nizov (fenotip). Trgovski potnik mora obiskati Koper, Kranj, Ljubljano, Celje in Maribor in se vrniti domov. Ker je število mest, ki jih mora trgovski potnik obiskati, enako pet, je tudi posamezni osebek sestavljen iz petih genov.



Slika 19: Problem trgovskega potnika

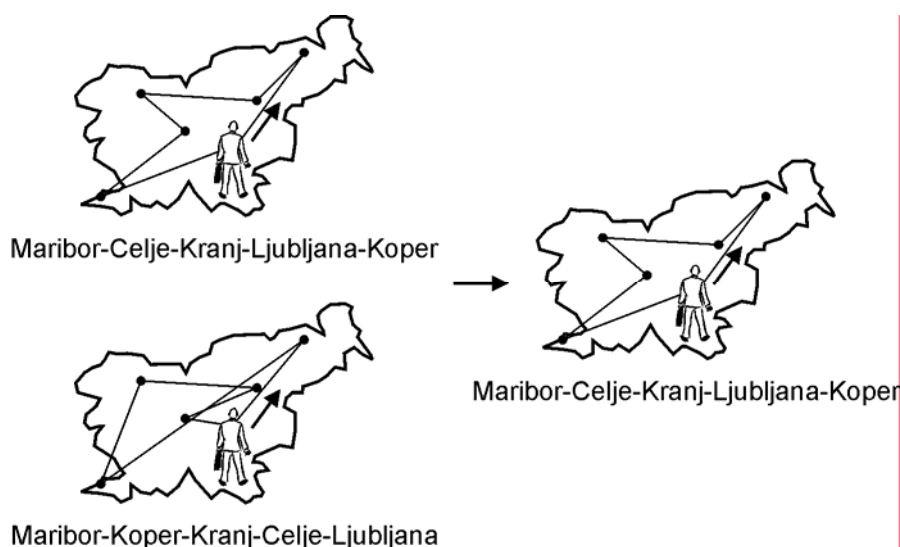


Slika 20: Populacija treh organizmov za primer trgovskega potnika

Po stvarjenju naključne populacije organizmov sledi njihovo ovrednotenje, ki je gonilna sila evolutivskega postopka. Kakovost posameznega organizma ugotavljamo glede na povprečno kvaliteto vseh organizmov v populaciji. Boljšim organizmom (rešitvam) predpišemo večjo verjetnost za sodelovanje pri osnovnih genetskih operacijah. Organizmi, ki so sposobnejši torej pogosteje prenesajo svoj genetski material v naslednjo generacijo, slabi pa počasi odmirajo iz populacije.

Osnovne genetske operacije konvencionalnih genetskih algoritmov so reprodukcija (selekcija), križanje, in mutacija. Oglejmo si jih nekoliko pobližje.

Slika 21 prikazuje operacijo reprodukcije. Reprodukcijska daje večjo verjetnost izbora boljšim organizmom, ki jih nato nespremenjene prenesemo v naslednjo generacijo.



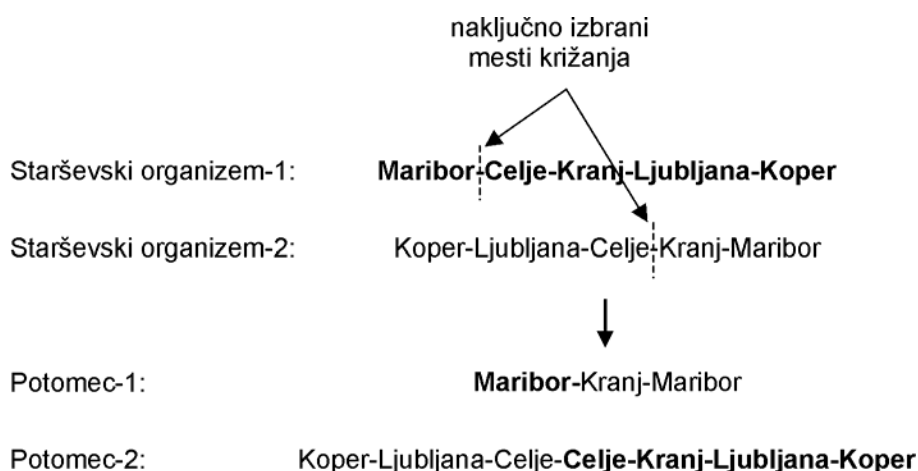
Slika 21: Reprodukcijski proces za problem trgovskega potnika

Metod za izbor organizmov, ki sodeluje v genetskih operacijah je več; najpogosteje se uporabljata metoda turnirja in proporcionalna metoda.

Pri turnirski izbiri organizma se najprej iz populacije po naključju izbere manjša skupina (npr. od dva do deset) članov, nato pa se izbere najboljši med njimi, ki je zmagovalec turnirja.

Bistvo proporcionalne metode lahko ponazorimo z igralniško ruleto. Predstavljajmo si, da posamezni krožni izsek rulete predstavlja sposobnost organizmov. Bolj so organizmi sposobni večji so izseki. Vsakokratni spust kroglice na ruleto predstavlja en izbor organizma. Jasno je, da se po mnogih zaporednih spustih kroglica pogosteje ujame v izseke, ki so širši, in s tem pogosteje izbere boljše organizme [18].

Slika 22 prikazuje operacijo križanja. Križanje omogoča izmenjavo genetskega materiala med organizmi. Iz dveh staršev nastaneta dva potomca. Izbira mest križanja, ki je nakazana s črtkano črto, je naključna. Obstaja mnogo različnih načinov izvedbe križanja [68][78]. Zaradi nazornosti so prikazani le genotipi organizmov.



Slika 22: Križanje za problem trgovskega potnika

$$\mathcal{F} = \{f_1, f_2, \dots, f_{N_f}\},$$

kjer je N_f največje število funkcij, in nabora terminalov:

$$\mathcal{T} = \{a_1, a_2, \dots, a_{N_t}\},$$

kjer je N_t največje število terminalov. Vsaka posamezna funkcija f_i iz nabora \mathcal{F} ima določeno število argumentov $z(f_i)$. Ustrezno število argumentov za funkcije iz nabora \mathcal{F} , je potemtakem določeno s seznamom $\{z(f_1), z(f_2), \dots, z(f_{N_f})\}$.

Nabor funkcij lahko vključuje:

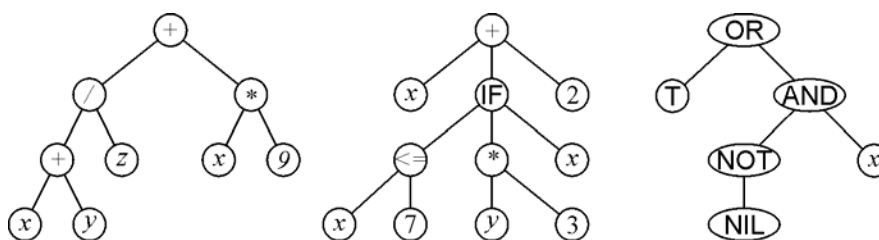
- osnovne računske operacije (npr. +, -, *, /),
- ostale matematične funkcije (npr. exp, sin, cos, ln),
- Boolove operacije (npr. AND, OR, NOT),
- relacijske funkcije (npr. ≤ =),
- funkcije za nadzor izvajanja programa (npr. IF),
- funkcije za iteracijo,
- funkcije za rekurzijo,
- funkcije, ki so določene glede na preučevano problemsko področje.

Nabor terminalov \mathcal{T} lahko vključuje številske konstante (npr. 1, 3.4, -2.1), logične konstante (npr. T, NIL), spremenljivke (npr. x, y) itd. Terminale lahko obravnavamo kot funkcije brez argumentov.

Po združitvi funkcij in terminalov v enoten nabor \mathcal{U} , dobimo:

$$\mathcal{U} = \mathcal{F} \cup \mathcal{T} = \{f_1, f_2, \dots, f_{N_f}, a_1, a_2, \dots, a_{N_t}\}$$

Raznolikost organizmov, sestavljenih iz različnega nabora terminalov in funkcij prikazuje slika 25.



Slika 25: Različni nabori terminalov in funkcij

3.4.2 Primarne genetske operacije

Primarne genetske operacije pri genetskem programiranju so reprodukcija (selekcija), križanje in mutacija. Omenimo lahko še permutacijo in poenostavljanje.

Reprodukcija je nesporna operacija, ki vselej, ko je izbrana, vzame samo en starševski organizem in izdelava samo enega potomca. Reprodukcija je izvedena v dveh korakih.

Najprej se iz populacije računalniških programov na osnovi prilagojenosti, z eno izmed metod za izbor, izbere računalniški program. Nato se ta program brez sprememb prenese v naslednjo generacijo.

Križanje je spolna operacija, ki vnaša spremembe v populacijo na ta način, da izdelava dva potomca, katerih genetski material sestoji iz dveh starševskih računalniških programov. Ker se križanje odvija med računalniškimi programi, ki so zapletene strukture, je posebno pozornost potrebno posvetiti ohranitvi sintaktične brezhibnosti računalniških programov.

Slika 26 prikazuje križanje dveh računalniških programov. Oba starševska programa za križanje, starša 1 in starša 2, izberemo z eno izmed metod, ki se uporablja tudi za izbor člana za reprodukcijo. Starša sta matematična izraza $(x + y)/z + xz$ in $x(1 - yz)$.

Z naključno izbiro točke križanja starša 1 in točke križanja starša 2, dobimo ostanek starša 1 ter ostanek starša 2, in odlomek starša 1 ter odlomek starša 2. Potomca 1 ustvarimo z vstavljanjem odlomka starša 2 na mesto točke križanja starša 1. Potomca 2 dobimo na podoben način. Potomca sta matematična izraza $(1 - xz)/z + xz$ in $x(x + y)$. Vidi se, da oba vključujeta genetske lastnosti svojih staršev.

Če sta za obe točki križanja izbrani terminalski vozlišči, se terminala obeh staršev med seboj preprosto zamenjata. Učinek takšnega križanja je podoben točkovni mutaciji.

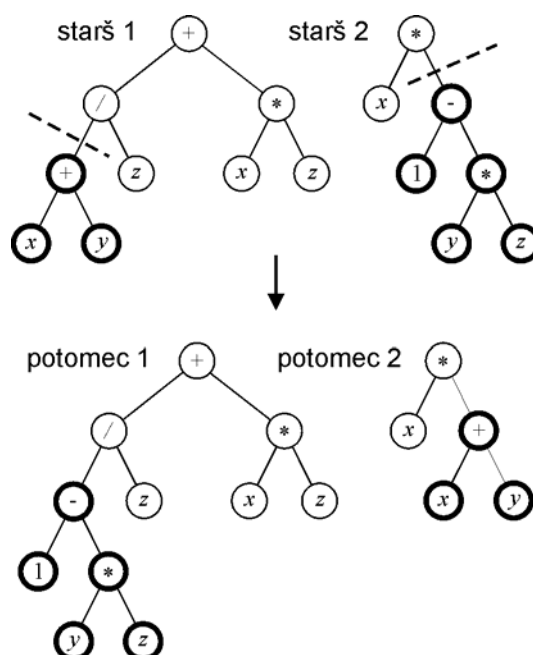
Če je za točko križanja enega izmed staršev izbrano korensko vozlišče, se celotno drevo tega starša vstavi na mesto točke križanja drugega starša. Dobljen potomec ima lahko precejšnjo globino. Drugi potomec je enak odlomku drugega starša.

V izjemno redkem primeru, ko je za točko enega starša izbrano korensko vozlišče, za točko križanja drugega starša pa terminalsko vozlišče, postane prvi starš prvi potomec, drugi potomec pa je kar terminal.

Če sta za točki križanja obeh staršev izbrani korenski vozlišči, je križanje enakovredno reprodukciji teh dveh staršev.

Tudi tedaj, ko sta oba starša enaka, je malo možnosti, da dobimo enaka potomca, saj sta točki križanja pri obeh starših v splošnem različni. Pri konvencionalnem genetskem algoritmu s fiksno dolžino organizmov, pa križanje dveh enakih staršev vedno vodi do enakih naslednikov, kar zmanjšuje genetsko raznolikost populacije.

Po križanju lahko dobimo računalniške programe s precej velikimi globinami. Da ne obremenjujemo po nepotrebnem računalniških zmogljivosti, je smiselno omejiti največjo globino dreves na razumno mejo. Če križanje dveh starševskih programov izdelava potomca, ki je globlji od največje dopustne globine, se v naslednjo generacijo prenese prvi njegov starš, hkrati z njim pa potomec, ki ni preglobok. Če sta oba potomca globlja od največje dopustne globine, se v naslednjo generacijo preneseta starša. V tem primeru je križanje enakovredno reprodukciji obeh staršev.



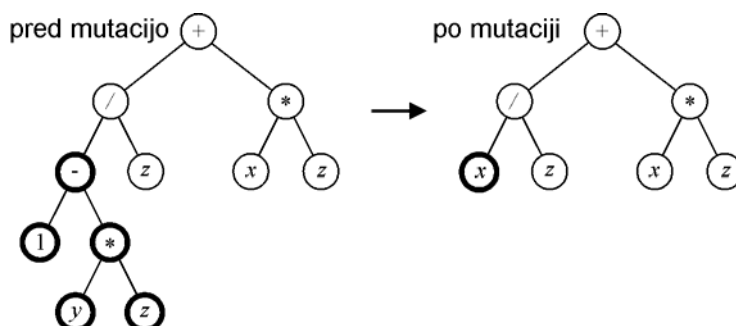
Slika 26: Križanje pri genetskem programiranju [19]

3.4.3 Sekundarne genetske operacije

Poleg primarnih genetskih operacij se lahko pri genetskem programiranju uporabljajo še:

- mutacija,
- permutacija in
- poenostavljanje.

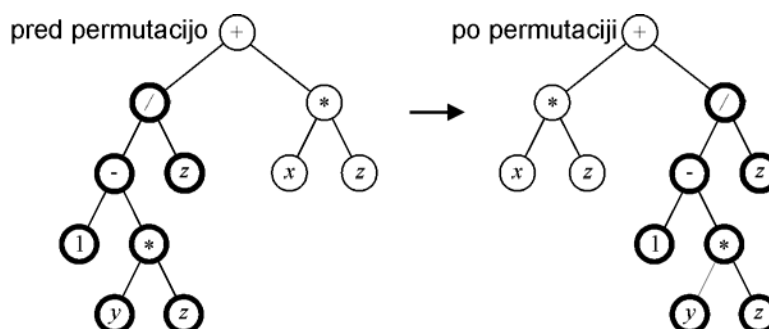
Mutacija je nespolna operacija, ki vnaša naključne spremembe v populacijo računalniških programov. Učinkuje na samo enega starša in izdela samo enega potomca. Slika 27 prikazuje operacijo mutacije pri genetskem programiranju. Proces mutacije se začne s selekcijo (izvedena je na enak način kot pri reprodukciji ali križanju) člana za mutacijo. Nato se izbere naključno izbere točka mutacije v računalniškem programu. Točka je lahko notranja (funkcijska) ali zunanja (terminalska). Vse kar leži na tej točki in pod njo se odstrani in nadomesti z naključno ustvarjenim programskim drevesom.



Slika 27: Mutacija pri genetskem programiranju [19]

Permutacija je nespolna operacija, ki učinkuje na samo enega starša in izdela samo enega potomca. Slika 28 prikazuje operacijo permutacije. Za operacijo permutacije je potrebno najprej izbrati računalniški program. Nato se izbere naključna notranja (funkcijska) točka

drevesa (točka 3). Če ima funkcija ob tej točki k argumentov, se najprej na naključen način izbere ena izmed $k!$ mogočih permutacij argumentov, nato pa se permutacija tudi izvrši.



Slika 28: Permutacija pri genetskem programiranju [19]

Poenostavljanje je operacija, ki poenostavlja računalniške programe med izvajanjem genetskega procesa. Poenostavljanje je nespolna operacija, ki učinkuje le na en starševski program in izdelava samo enega potomca. Uporablja se predvsem pri računalniških programih, ki so po značaju Boolovi izrazi. V tem primeru se poenostavljanje vrši na osnovi DeMorganovih zakonov. Primer: $(OR\ x\ x)$ se poenostavi v x , $(NOT(NOT\ x))$ v x .

3.4.4 Evolucija populacije

Po končanem prvem krogu, ki vključuje:

1. stvarjenje začetne populacije,
2. ovrednotenje populacije in
3. genetsko spreminjanje vsebine programov, sledi iterativno ponavljanje korakov 1 in 2.

Evolucija populacije je enaka kot pri genetskih algoritmih, razlika je edino v tem, da pri genetskem programiranju mutacija ponavadi ni vključena kot samostojna operacija. Tudi permutacija in poenostavljanje v večini primerov nista potrebni.

Po določenem številu generacij so računalniški programi praviloma vedno bolj prilagojeni na okolje. Naloga je rešena, ko vsaj en program v populaciji doseže postavljeno merilo o potrebni kakovosti rešitve. Ker imamo opravka s stohastičnim postopkom nas ta ne pripelje do zadovoljivih rešitev v vsaki civilizaciji. Da lahko računamo na zadostno verjetnost rešitve, je potrebno problem obdelati v več neodvisnih civilizacijah. Število civilizacij, potrebnih za rešitev, je odvisno od težavnosti problema. Tudi konvencionalni genetski algoritmi ne postrežejo z dovolj kakovostno rešitvijo v vsaki civilizaciji, vendar je to tam manj očitno, saj so naloge, ki jih z njimi rešujemo, običajno preprostejše kot pri genetskem programiranju.

3.5 Evolucijske strategije

Evolucijske strategije se uporabljajo predvsem za optimiranje realnih funkcij, zato so prevladujoče oblike rešitev vektorji realnih števil. Tvorec prvinske oblike evolucijskih strategij je I. Rechenberg [74].

Zgodnje oblike evolucijskih strategij so temeljile na populaciji s samo enim članom in na uporabi samo ene genetske operacije–mutacije. Posameznik je sestavljen iz para realnih vektorjev, npr. $v = (x, \sigma)$. Vektor x predstavlja točko v iskalnem prostoru, vektor σ pa je vektor standardne deviacije. Pri mutaciji se vektor x nadomesti z vektorjem:

$$x^{t+1} = x^t + N(0, \sigma), \quad (1)$$

pri čemer je $N(0, \sigma)$ vektor naključno izbranih Gausovih števil s srednjo vrednostjo nič in standardno deviacijo σ . Postopek temelji na dejstvu, da so manjše spremembe pogostejše kot velike. Starševski organizem nadomesti potomec, če ima slednji boljšo prilagojenost, in če so hkrati izpolnjene omejitve (npr. nova točka mora biti znotraj preučevanega področja).

Naj bo f stroškovna funkcija. Če omejitev ni, nadomesti potomec (x^{t+1}, σ) starša (x, σ) , če je $f(x^{t+1}) < f(x^t)$. V nasprotnem primeru potomec umre, in populacija ostane nespremenjena. Čeprav je v populaciji samo en član, se opisan postopek imenuje evolucijska strategija z dvema članoma, ali krajše model vrste $(1 + 1)$. Vzrok za to je v tem, da potomec pravzaprav tekmuje s svojim staršem, in na nivoju tekmovanja sta (začasno) v populaciji dva člana.

Evolucijske strategije je razširil H. P. Schwefel, ki je predlagal populacije z več člani in kombiniranje (križanje) rešitev. Sprva so bile evolucijske strategije z več člani vrste $(\mu+1)$, kjer je μ velikost populacije. Kasneje so jih razvili in dopolnili še z modeloma vrst $(\mu+\lambda)$ in (μ, λ) , kjer je λ število potomcev.

Pri modelu vrste $(\mu+\lambda)$ ustvari μ organizmov λ potomcev, izmed vseh $\mu+\lambda$ organizmov pa nato izberemo μ najboljših za naslednjo generacijo.

Pri modelu vrste (μ, λ) prav tako μ osebkov ustvari λ potomcev, vendar μ najboljših potomcev nadomesti svoje starše. Tako je življenje posameznika omejeno na eno generacijo. Viri navajajo, da se zato model vrste (μ, λ) še posebej dobro odreže pri problemih, kjer se optimum s časom spreminja.

3.6 Evolucijsko programiranje

Originalno metodo evolucijskega programiranja je razvil L. Fogel. Njen osnovni namen je razvoj inteligentnih sistemov (napovedovanje sprememb v okolju) s pomočjo simulirane evolucije. Metoda izhaja iz avtomatskega načrtovanja končnih avtomatov, vendar se danes uporablja kot večnamenska optimizacijska metoda [18].

Okolje predstavlja vhodno zaporedje simbolov iz abecede s končnim številom znakov, (neznani) algoritem pa mora na osnovi vhodnega zaporedja napovedati nov izhodni simbol. Izhod vpliva na stroškovno funkcijo, ki meri točnost napovedi.

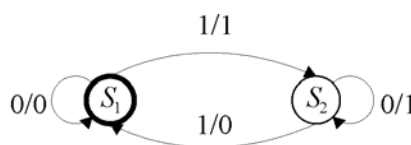
Predpostavimo zaporedje dogodkov, ki jih označimo s simboli a_1, a_2, \dots ; algoritem mora napovedati naslednji (neznani) simbol, npr. a_{n+1} na osnovi prejšnjih (znanih) simbolov a_1, a_2, \dots, a_n . Namen evolucijskega programiranja je razviti tak algoritem.

Struktura, ki je podvržena adaptaciji, je populacija končnih avtomatov [83]. Slika 29 prikazuje diagram prehoda stanj preprostega končnega avtomata za preverjanja parnosti. Diagram prehoda stanj je usmerjen graf, kjer so vozlišča posamezna stanja (S_1 pomeni sodo stanje, S_2 pa liho stanje), povezave pa nakazujejo prehod iz enega stanja k drugemu. Zapis a/b ob povezavi od S_1 do S_2 pomeni, da vhodna vrednost a , ko je avtomat v stanju S_1 , povzroči izhod b , in prehod na naslednjo stanje S_2 . Začetno stanje je S_1 .

Z evolucijskim programiranjem variramo populacijo končnih avtomatov. Posamezni organizem predstavlja mogočo rešitev problema oziroma vedenjski vzorec. Da dobimo oceno

prilagojenosti, moramo vsak končni avtomat ovrednotiti. To storimo tako, da ga izpostavimo okolju, kjer avtomat pregleda vse prejšnje simbole. Za vsako zaporedje, npr. a_1, a_2, \dots, a_i , ustvari končni avtomat izhod a'_{i+1} , ki se primerja z naslednjim opazovanim simbolom a_{i+1} . Če je bilo do nekega trenutka pregledanih npr. n simbolov, naredi končni avtomat n napovedi (eno za vsako podzaporedje a_1, a_1, a_2 , in tako naprej do a_n). Prilagojenost končnega avtomata je odvisna od natančnosti vseh n napovedi.

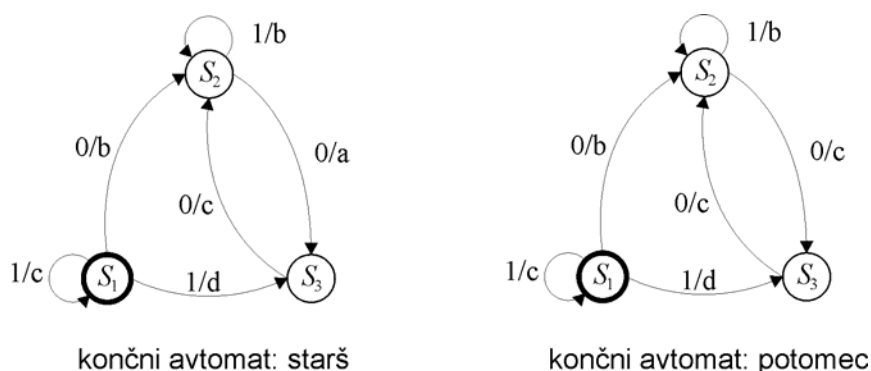
Podobno kot pri evlucijskih strategijah, tudi pri evlucijskem programiranju ustvarimo najprej potomce, nato pa izberemo posameznike za naslednjo generacijo. Variiranje rešitev se izvaja samo z mutacijo. Z mutiranjem populacije dobimo M potomcev; vsak starš prispeva po enega. Potomce ovrednotimo, nato pa izmed $2M$ članov izberemo M posameznikov za naslednjo generacijo.



Slika 29: Končni avtomat za preverjanje parnosti

Slika 30 prikazuje operacijo mutacije končnega avtomata. Z mutacijo lahko spremenimo izhodni simbol, način prehajanja enega stanja v drugega in začetno stanje. Poleg tega lahko dodamo novo stanje ali pa obstoječe odvezamemo. Največje in najmanjše število možnih stanj je omejeno. Katero od zgornjih mutacij izberemo je odvisno od verjetnostne porazdelitve, ki pa se lahko med evlucijskim procesom spreminja. Hkrati je mogoče na posamezniku izvesti tudi več mutacij. Število mutacij posameznika je odvisno od dodatne verjetnostne porazdelitve.

Evlucijsko programiranje je podobno evlucijski strategiji tipa $(\mu+\lambda)$ z lastnostjo $\mu=\lambda$, ki uporablja samo mutacijo.



Slika 30: Končni avtomat in njegov potomec

4. PROGRAMIRANJE NUMERIČNO KRMILJENIH STROJEV Z UPORABO EVOLUCIJSKIH METOD

V današnjem času je čedalje večja potreba za racionalnost proizvodnih procesov in sistemov, za čim manjše proizvodne stroške in za visoko kakovost dobrin. Hkrati pa je vedno bolj v ospredju tudi skrb za energetske varčne tehnološke procese, kar z drugimi besedami pomeni, da naj bodo proizvodni sistemi in tudi izdelki čim bolj prilagojeni ekosistemu. Za uresničitev zastavljenih ciljev je bilo doslej razvitih veliko različnih pristopov za optimiranje in modeliranje proizvodnih sistemov in procesov. Tipične optimizacijske naloge v proizvodnji so npr.: optimiranje razmestitev strojev v proizvodnem obratu, planiranje in optimiranje transportnih poti, optimiranje zasedenosti kapacitet proizvodnih strojev in naprav, izbira optimalne obdelave, optimiranje tehnoloških parametrov za različne vrste obdelave ipd. [69][70][71] Tudi programiranje numerično krmiljenih strojev (NC-stroji; ang. numerizing control machines) lahko opredelimo kot proizvodni proces, ki mora biti izveden na čim bolj optimalen način. Ker so NC-stroji navzoči domala že v vseh obdelovalnih sistemih, se je v zadnjih desetletjih avtomatično programiranje NC-strojev zelo razmahnilo. Tako so dandanes na voljo številne komercialne programske rešitve, ki vključujejo tudi avtomatično programiranje NC-strojev.

Obstoječe rešitve se razlikujejo po zanesljivosti, učinkovitosti, fleksibilnosti in univerzalnosti. Doslej ni bilo moč zaslediti univerzalne rešitve za optimizacijske probleme, ki so navzoči pri obdelavi. Poleg tega pa se še vedno velika večina raziskav s področja optimiranja programiranja sodobnih NC-strojev (in tudi številnih drugih procesov in sistemov v sodobni industriji) poslužuje predvsem konvencionalnih, nedeterminističnih postopkov optimiranja. Ker pa imamo v sodobni proizvodnji pogosto opravka s kombinatorično eksplozijo različnih proizvodnih scenarijev in rešitev, je možno s konvencionalnimi postopki dobiti v večini primerov zgolj sub-optimalne rešitve problemov, ki jih rešujemo.

Na mnogih področjih znanosti in tehnologije je moč zaznati uveljavljanje nedeterminističnih inteligentnih sistemov, sposobnih učenja [51][67] in odzivanja na kompleksno, nepredvidljivo in spremenljivo okolje. Več uspešnih optimizacij različnih tehnoloških sistemov je bilo izvedenih s pomočjo metod evolucijskega računanja [9][31][37][18][52].

V nalogi smo za programiranje NC-strojev uporabili genetski algoritem. Čeprav je predlagan koncept namenjen za programiranje NC stružnic, frezalnih strojev in strojev za razrez pločevine, ga lahko priredimo tudi za druge NC-stroje. Pristop posnema naravno evolucijo živih organizmov, kjer v boju za naravne vire uspešni individuumi postopoma postajajo vse bolj dominantni, uspešni in prilagodljivi na okolje v katerem bivajo, manj uspešni pa so v naslednjih generacijah navzoči v manjšem številu. Pri predlaganem konceptu so adaptaciji podvrženi NC-programi, ki so predstavljeni kot uteženi grafi. Med simulirano evolucijo, po

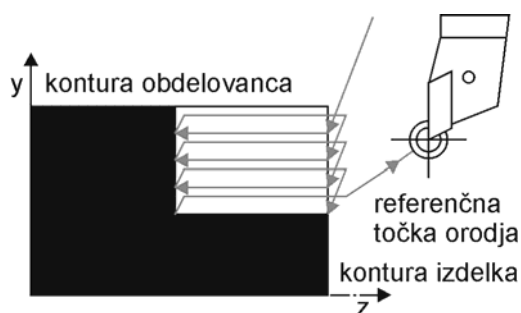
analogiji iz živega sveta, iz znanih podatkov o orodju, stroju, obdelovancu, tehnoloških podatkih in želenem izdelku, izdeluje postopoma, iz generacije v generacijo, čedalje učinkovitejše organizme – NC-programe.

V naslednjih poglavjih bomo predstavili sledeče sisteme za samodejno programiranje s pomočjo genetskih algoritmov:

- sistem za samodejno programiranje NC-stružnice,
- sistem za samodejno programiranje NC-frezalnega stroja in
- sistem za samodejno programiranje NC-stroja za razrez pločevine.

5. PROGRAMIRANJE STRUŽNICE IN EVOLUCIJA

Struženje je postopek obdelave z odrezavanjem, ki rabi v glavnem za izdelavo valjastih predmetov, pri čemer je moč obdelovati tudi ravne površine [13]. Slika 31 prikazuje konturi preprostega surovca in želene končne oblike izdelka. Cilj struženja je zagotoviti relativno gibanje orodja z ozirom na obdelovanec. Posledica relativnega gibanja je premikanje referenčne točke orodja in s tem nastajanje želene oblike izdelka. Gibanje orodja je sestavljeno iz delovnega giba in podajalnega giba. Odrezavanje materiala poteka v več rezih, katerih optimalen vrstni red je v splošnem znan le za zelo preproste izdelke [20][45]. Vsak delovni gib orodja odvzame določeno količino materiala in tako ustvarja odrezke.

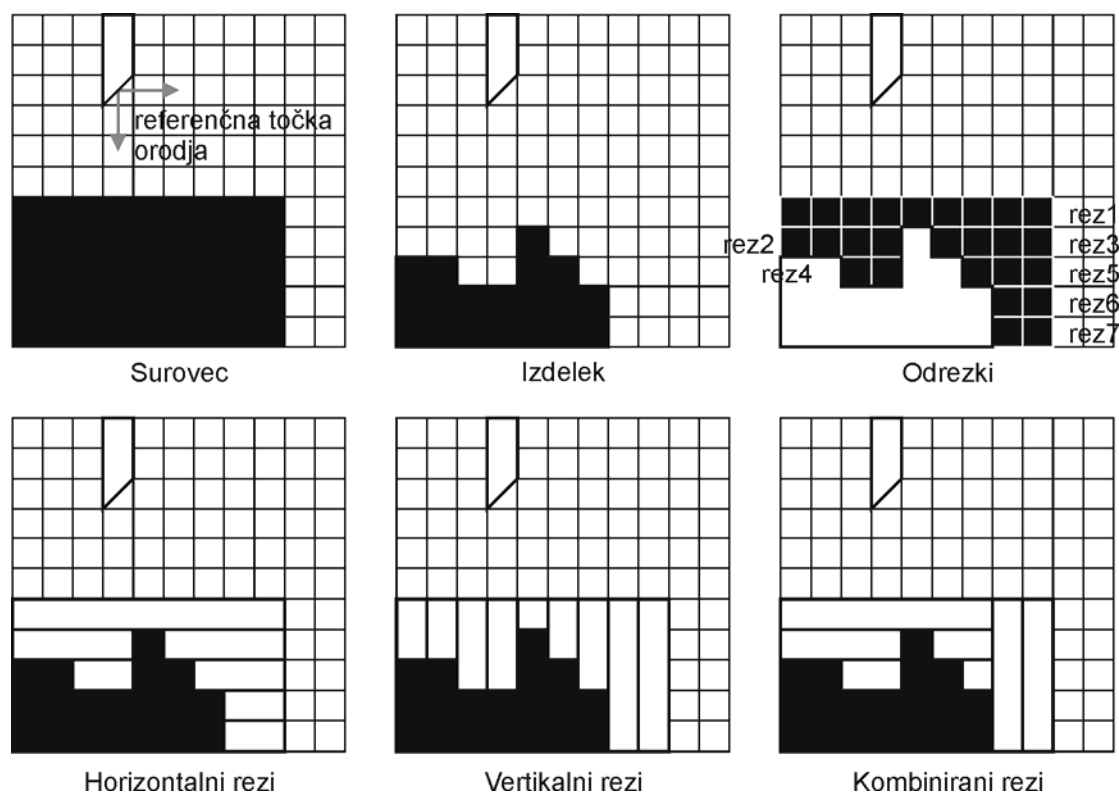


Slika 31: Preprost obdelovanec in izdelek pri struženju

5.1 Predlagan koncept

5.1.1 Glavni algoritem

Osnovno idejo predlaganega koncepta bomo prikazali na preprostem primeru. Na sliki 32 zgoraj so prikazani surovec, izdelek in odgovarjajoči odrezki. Območje možnega premikanja orodja (noža) je razdeljeno (diskretizirano) na 11x11 kvadratov. Orodje debeline en kvadrat, se lahko giblje diskretno v smereh gor, dol, levo in desno, reže pa le v desno ali navzdol. Zaradi diskretne zasnove obdelovalnega polja, je referenčna točka orodja kar na sredini rezalnega roba rezalne ploščice. Material, ki ga je potrebno odvzeti, razdelimo na več rezov, ki so sestavljeni iz odrezkov (Slika 32, zgoraj desno). Za primer, rez *rez1* je sestavljen iz 9-ih odrezkov, reza *rez2* in *rez3* vsak iz 4 odrezkov, rez *rez4* iz 2 odrezkov itd. Na sliki 32 spodaj so prikazani načini obdelave. Zaradi diskretne narave sistema je možno izbrati horizontalne, vertikalne ali pa kombinirane reze.



Slika 32: Surovec, izdelek, odpadki in načini obdelave

Slika 33 prikazuje glavni algoritem za genetsko programiranje NC-stroja (tudi NC stružnice) v pseudo kodi. Najprej je treba vnesti podatke o surovcu in izdelku. Pri predstavljenem konceptu je vnos izveden avtomatično, neposredno iz CAD modelov surovca in izdelka. Sistem nato razdeli obdelovalno polje na želeno število kvadratov. Na koncu podatkovnega vnosa je treba podati še startno in končno točko premikanja orodja in referenčno točko orodja. Odslej je sistem sposoben evlucijskega ustvarjanja in optimiranja NC-programov skozi več generacij t .

Boolova spremenljivka `poznamo_NC_program` odloča o tem ali za začetno populacijo NC-programov $P(t)$ uporabimo rešitve od predhodno izvršene obdelave ali pa ustvarimo začetno populacijo povsem naključno. Vsak kromosom (organizem) v populaciji predstavlja NC-program, s kolizijami ali brez. NC-program je sestavljen iz podajalnega giba med začetno točko in začetnim rezom, iz podajalnih in delovnih gibov med rezi, in iz podajalnega giba med končnim rezom in končno točko orodja. Evlucijsko razvijanje rešitev se začne, ko kromosome ovrednotimo. Tistim rešitvam, ki bolje rešijo problem, priredimo večjo verjetnost, da sodelujejo pri operacijah selekcije in spreminjanja. S selekcijo zagotovimo preživetje bolj uspešnih članov populacije in njihovo napredovanje v nespremenjeni obliki v naslednjo generacijo. S spreminjanjem vplivamo na enega ali več organizmov in iz njih ustvarjamo njihove potomce. Po opravljeni selekciji in spreminjanju dobimo novo generacijo, ki jo spet ovrednotimo. Postopek ponavljamo tako dolgo, dokler ni izpolnjen ustavitveni pogoj postopka. To je lahko največje predpisano število generacij ali pa zadostna kakovost rešitev.

```

začni
  vnos_podatkov o surovcu, izdelku, startni, končni in referenčni točke orodja
  diskretizacija obdelovalnega polja
  začni
    t←0
    če poznamo_NC_program
      vstavi P(t)
      drugače ustvari P(t)
    ovrednoti P(t)
    dokler (ni izpolnjen_ustavitveni_pogoj)
      začni
        t←t+1
        spreminjaj P(t)
        ovrednoti P(t)
      končaj
    končaj
  izpis rezultatov
končaj

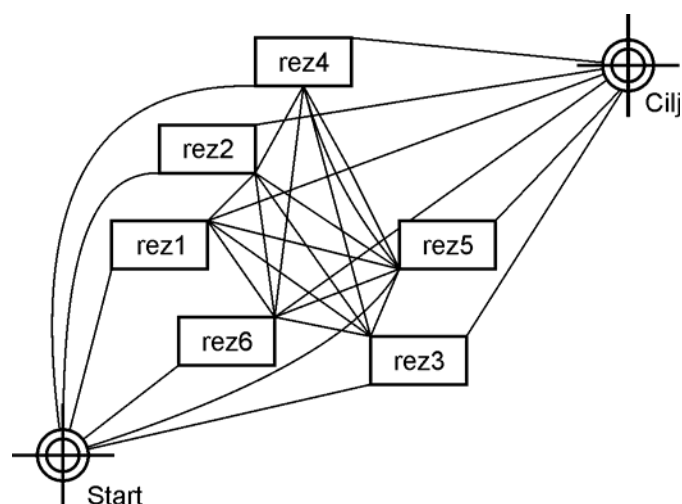
```

Slika 33: Genetski algoritem za programiranje NC-stroja v pseudo kodi

5.1.2 Kodiranje NC-programa

Naključno ustvarjene NC-programe lahko predstavimo z uteženim grafom (Slika 34) [83]. Točke grafa so delovni gibi (rezi), povezave med točkami pa podajalni gibi. Vsaka točka grafa je povezana z vsemi ostalimi točkami grafa. Točke in povezave grafa so utežene.

Zaporedje genov (rezov) predstavlja zaporedno gibanje noža od reza do reza. Prvi in zadnji gen seznama predstavljata začetno in končno točko gibanja noža. Vmesni členi ponazarjajo zaporedje rezov, ki so po vrstnem redu podvrženi obdelavi. Nož se najprej s podajalnim gibom giblje od začetne točke do prvega reza v seznamu. Nato izvede delovni gib do prvega reza, ga obdela, nadaljuje s podajalnim gibom do naslednjega reza, ga obdela, in tako naprej dokler ne obdela vseh rezov. Na koncu s podajalnim gibom doseže končno točko. Naključno ustvarjeni NC programi se med seboj razlikujejo po številu potrebnih korakov orodja in številu kolizij med orodjem in obdelovancem.



Slika 34: NC-program kot utežen graf

5.1.3 Ovrednotenje populacije

Pred nadaljevanjem najprej definirajmo dva pomembna pojma: konsistentnost rezov in konsistentnost odrezkov. Rez je konsistenten, če se vsaj en odrezek v rezu drži obdelovanca. Odrezek je konsistenten, če se še drži obdelovanca (konsistentnega reza). Ugotavljanje konsistentnosti rezov in odrezkov je zelo pomembno, saj je treba zaradi hitro spreminjajočega se okolja (obdelovanca) neprestano preverjati, katera področja obdelovanca so že obdelana in katera še ne.

Cilj kombinatorične naloge je najti takšno pot skozi graf (z začetno točko Start in končno točko Cilj), ki bo vsebovala vse točke grafa in bo vsota uteži na povezavah in točkah ter uteženega števila kolizij, minimalna. To z drugimi besedami pomeni, da je treba poiskati optimalno varen NC-program. Vsota uteži na povezavah in točkah je tako:

$$\text{Ovrednoti}_{\text{NC-program}} = \sum W_{\text{NCx}} = W_{s1} + W_{rez1} + W_{12} + W_{rez2} + W_{23} + W_{rez3} + W_{34} +$$
 (2)

$$W_{rez4} + W_{45} + W_{rez5} + W_{56} + W_{rez6} + W_{6c} + f \cdot K,$$

kjer so:

NC_x x -ti NC program,

w_{si} dolžina podajalnega giba od začetne točke S do prvega odrezka $reza-i$,

w_{rezi} dolžina delovnega giba konsistentnega $reza-i$,

w_{ij} dolžina podajalnega giba med konsistentnim $rezom-i$ in konsistentnim $rezom-j$,

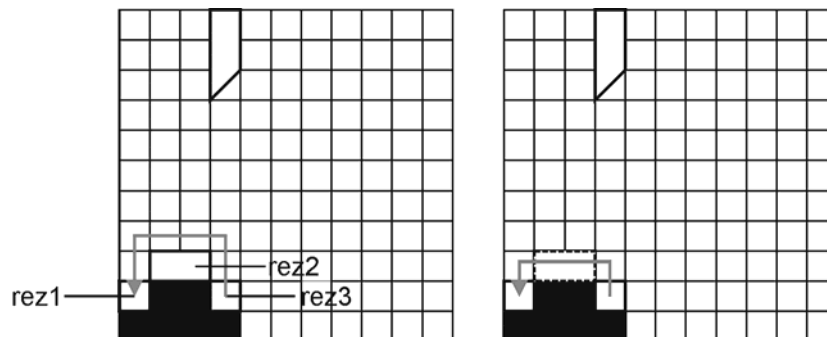
w_{ic} dolžina delovnega giba od zadnjega odrezka zadnjega konsistentnega $reza-i$ do končne točke,

f faktor vpliva,

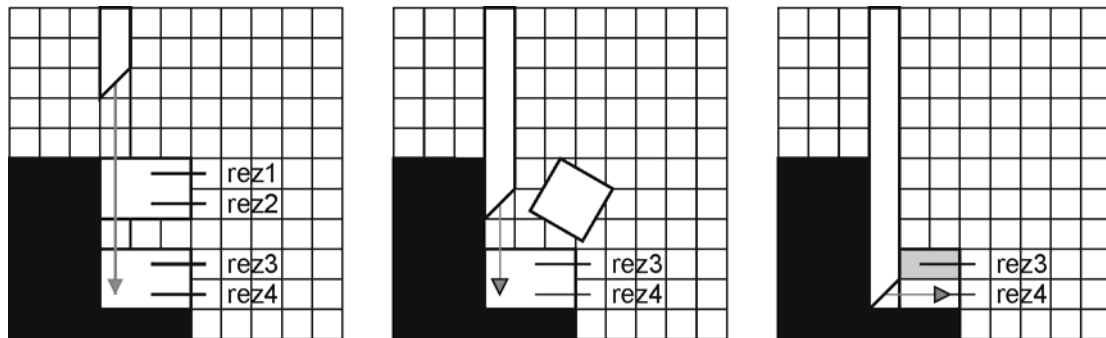
K število kolizij.

Največjo zahtevnost pri ovrednotenju NC-programa predstavljajo ugotavljanja dolžine poti orodja, konsistentnosti rezov in odrezkov ter število kolizij.

Zaradi spreminjanja oblike obdelovanca med obdelavo se posledično spreminja tudi dolžina poti orodja. Za primer pogledjmo sliko 35 na levi. Pri podajalnem gibu orodja od reza $rez3$ do reza $rez1$ potrebnih 7 korakov orodja, medtem ko je na sliki 5 desno po obdelavi reza $rez2$, za isto akcijo potrebnih le še 5 korakov. Med podajalnimi in delovnimi gibi orodja sistem budno spremlja gibanje noža, sproti preverja konsistentnost rezov in sešteva možne kolizije.



Slika 35: Obdelovanec kot dinamično okolje



Slika 36: Konsistentnost rezov in kolizija.

Na sliki 36 levo se nož giblje vertikalno k prvemu odrezku reza rez4. Na sredini slike 36 nož med gibanjem odreže rez1 in rez2 od obdelovanca. Ta dva reza pri takem zaporedju izvajanja rezov torej nista več konsistentna (nista več povezana z obdelovancem). Na sliki 36 desno je prikazana situacija, ko nož obdela prvi odrezek reza rez4 in se začne gibati v desno. Z držalom udari v odrezek reza rez3, ki se nahaja nad rezom rez4. Pri obdelavi reza rez4 nož zadane dva (preostala) odrezka reza rez3. Tako je število kolizij K za primer takšnega zaporedja izvajanja rezov enako 2.

Slika 37 opisuje proceduro ovrednotenje, ki največ prispeva h kompleksnosti reševanja problema planiranja in optimiranja NC-programov z genetskimi algoritmi. V proceduro so vključene štiri najpomembnejše funkcije: preverjaj_konsistenco, preverjaj_kolizijo, premikaj in reži. Funkcija preverjaj_kolizijo skrbi za ugotavljanje nezaželenih trkov orodja z obdelovancem. Funkcija preverjaj_konsistenco nadzira ločevanje odrezkov od obdelovanca in s tem preverja konsistentnost rezov. Funkciji premikaj in reži sta namenjeni za simulacijo delovnega in podajalnega giba orodja.

začni

```

preverjaj_konsistenco vsak_korak
preverjaj_kolizijo vsak_korak
premik od start do prvi_odrezek_reza_1
reži rez_1 dokler obstajajo_odrezki_reza_1
za i=2 do i=število_rezov ponavljaj
začni
  dokler (ni naslednji_konsistentni_rez)
  izberi naslednji_konsistentni_rez
  premik od zadnji_konsistentni_odrezek_zadnjega_konsistentnega_reza
  do prvi_konsistentni_odrezek_naslednjega_konsistentnega_reza
  dokler obstajajo_konsistentni_odrezki_naslednjega_reza
  reži naslednji_konsistentni_rez

```

končaj

```

premik od zadnji_konsistentni_odrezek_zadnjega_konsistentnega_reza
do cilj

```

```

vrni število_kolizij, NC_program, dolžina_NC_programa

```

končaj

Slika 37: Procedura ovrednotenje

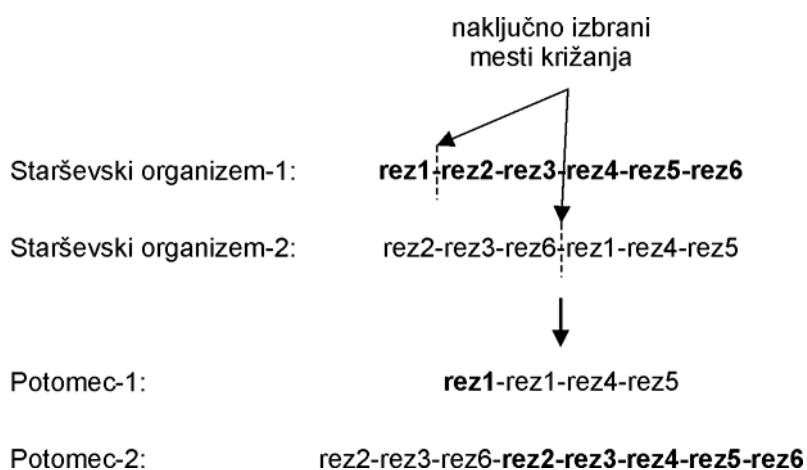
Procedura ovrednotenje se začne izvajati s klicem funkcij preverjaj_kolizijo in preverjaj_konsistenco, ki ažurno spremljata vsak gib noža vsak_korak in nadzorujeta ločevanje odrezkov od obdelovanca in ugotavljata nedovoljene položaje noža. Sledi ustvarjanje poti od začetne (podane) pozicije noža start do prvega odrezka prvi_odrezek_reza_1 v prvem rezu iz seznama rezov. Obdelava traja toliko časa dokler so odrezki prvega reza iz seznama rezov konsistentni. Postopek se nadaljuje z izbiro naslednjega

konsistentnega reza naslednji_konsistentni_rez. Ponovno se izvede podajalni gib, tokrat od zadnjega konsistentnega odrezka zadnji_konsistentni_odrezek_zadnjega_konsistentnega_reza zadnjega konsistentnega reza do prvega konsistentnega odrezka prvi_konsistentni_odrezek_naslednjega_konsistentnega_reza naslednjega reza v zaporedju rezov. Posamezen rez obdelujemo toliko časa dokler so odrezki konsistentni, celotno obdelavo obdelovanca pa izvajamo tako dolgo, da ne pregledamo vseh rezov število_rezov v seznamu rezov. Število rezov število_rezov v seznamu rezov je odvisno od oblike izdelka in je za vsak posamezen obdelovanec enako. Procedura ovrednotenje vrne (funkcija vrni) število kolizij število_kolizij (K), NC-program NC_program (tj. potovanje noža) in število korakov noža dolžina_NC_programa.

5.1.4 Genetske operacije

Uporabili smo genetske operatorje reprodukcije (selekcije), križanja in permutacije. Za izbiro kromosomov, ki sodelujejo v operacijah izbiranja in spreminjanja, smo uporabili turnirsko izbiro.

Pri reprodukciji najprej naključno izberemo en kromosom iz populacije in ga nespremenjenega prenesemo v naslednjo generacijo. Pri križanju naključno izberemo dva kromosoma iz populacije. Tudi točka križanja je izbrana naključno. Dele izbranih kromosomov medsebojno zamenjamo. Pri križanju je treba zagotoviti, da potomca vsebujeta vse reze iz seznama rezov, zato je potrebno kromosoma po križanju popraviti (Slika 39).



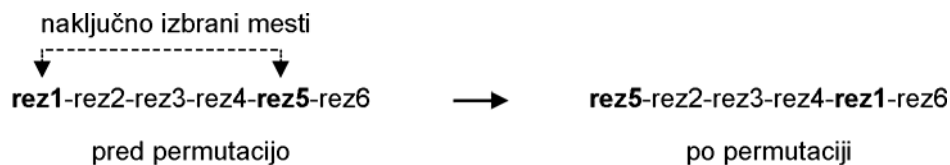
Slika 38: Genetska operacija križanje NC programov

Potomec-1: rez1-rez4-rez5-rez2-rez3-rez6

Potomec-2: rez2-rez3-rez6-rez1-rez4-rez5

Slika 39: Poprava NC programa

Pri permutaciji (Slika 40) najprej po naključju izberemo kromosom, nato pa dva naključno izbrana gena (reza) medsebojno zamenjamo.



Slika 40: Permutacija NC programa

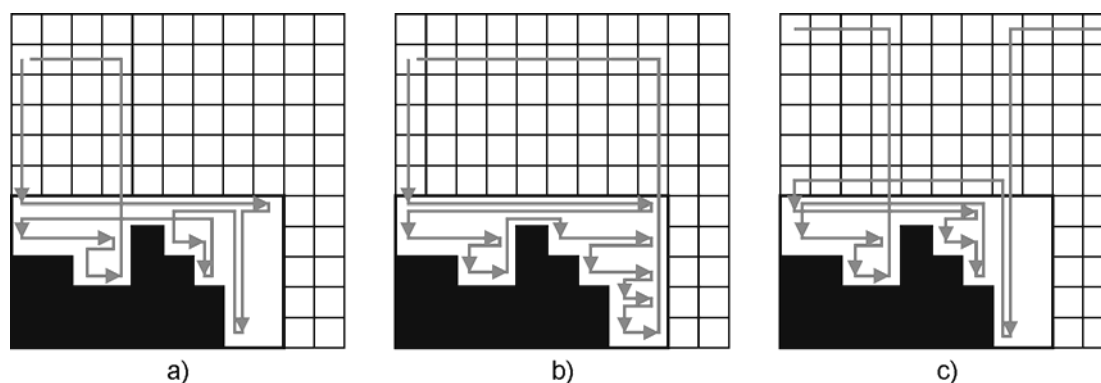
5.2 Rezultati in diskusija

Delovanje in učinkovitost sistema sta prikazani na izbranih testnih primerih. Najprej prikazujemo primer izdelave preprostega izdelka iz preprostega surovca (Podpoglavje 5.2.1). Za podrobnejši prikaz sposobnosti sistema in detajlne analize delovanja sistema pa podajamo še primer izdelave preprostega izdelka iz oblikovno zahtevnega obdelovanca (Podpoglavje 5.2.2) ter izdelavo zahtevnega izdelka iz preprostega obdelovanca (Podpoglavje 5.2.3). Pri vseh treh primerih želimo še posebej naglasiti splošnost, inteligentnost in avtonomnost predlaganega koncepta, adaptacijske sposobnosti sistema na dinamično okolje (spreminjanje oblike obdelovanca) in samoorganizacijsko napredovanje k čedalje uspešnejšim rešitvam.

Pri vseh zagonih sistema so bili uporabljeni enaki parametri evolucije in sicer velikost populacije 50, maksimalno število generacij 50, verjetnost reprodukcije 0.2, križanja 0.6 in permutacije 0.2. Uporabljena je bila turnirska izbira organizmov z velikostjo turnirja 7.

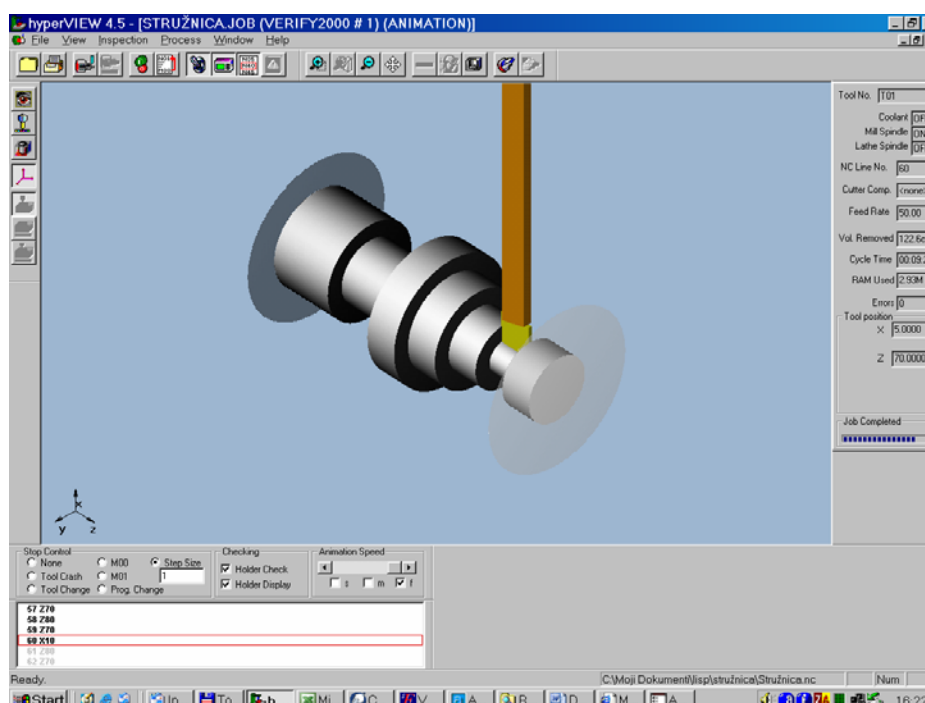
5.2.1 Primer 1

V tem primeru je bil cilj izdelati razmeroma preprost izdelek iz razmeroma enostavnega surovca (slika 41). Podali smo še startno in končno točko gibanja orodja. Slika 41 prikazuje rezultate treh neodvisnih zagonov sistema. Prikazani so samo najboljši organizmi. Na sliki 41a je sistem inteligentno skrajšal obdelavo z vertikalnim rezom na desni strani obdelovanca. S tem je bila zagotovljena krajša obdelava in zmanjšana obraba orodja. S praktičnega stališča je takšna globina rezanja morda vprašljiva, saj nož prodre v material kar za 4 korake. Vendar pa je treba poudariti, da v tej fazi raziskave gre predvsem za fleksibilnost in univerzalnost sistema, kjer so dovoljena le preprosta pravila obnašanja in sicer: (1) orodje se lahko giblje korakoma levo, desno, navzgor in navzdol, (2) orodje reže le navzdol in desno, (3) poškodbe noža in stroja niso dovoljene in (4) izdelek naj bo izdelan čim hitreje, s tem da se upoštevajo prva tri pravila. Število korakov noža, ki so potrebni, da se izdelek v celoti izdela, znaša v tem primeru 51, število korakov, ko je orodje v kontaktu z obdelovancem pa je 22. Slika 41b prikazuje še eno rešitev (organizem) enakega problema. Vidi se, da je tokrat sistem končal obdelavo na povsem drugem mestu na obdelovancu kot pri prejšnjem primeru. Za izdelavo izdelka je bilo sedaj potrebnih 63 korakov noža, od tega pa je bilo orodje v kontaktu z obdelovancem 26 korakov. Dolžina NC-programa in obremenitev orodja sta torej v tem primeru 19.05% in 15.38% večji kot prej. Na sliki 41c je prikazana izdelava enakega izdelka iz enakega surovca, le da sta začetna in končna točka gibanja orodja različni kot v prejšnjih dveh primerih. Število korakov za obdelavo izdelka znaša 60, orodje pa je v kontaktu z obdelovancem 21 korakov.



Slika 41: Različne rešitve za enak izdelek

Slika 42 prikazuje preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5.



Slika 42: Preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5

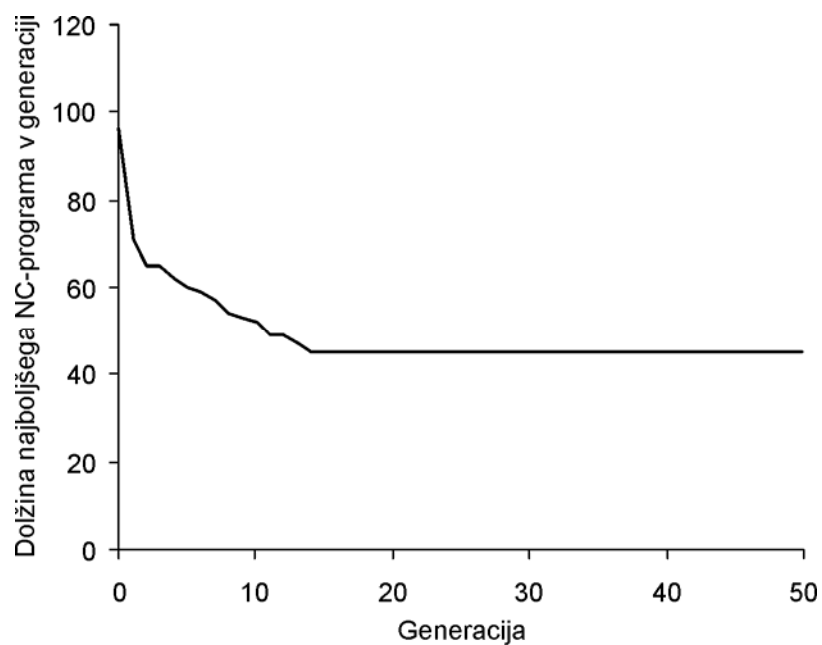
5.2.2 Primer 2

S tem primerom želimo pokazati inteligentnost in univerzalnost sistema v primeru, ko gre za oblikovno zahteven obdelovanec (unija črnih kvadratov in s črno, odebeljeno črto obrobjenih kvadratov), izdelek pa je razmeroma preprost (črni kvadrati). Začetna in končna točka gibanja orodja sta na istem kvadratu. Uporabljeni so kombinirani rezi.

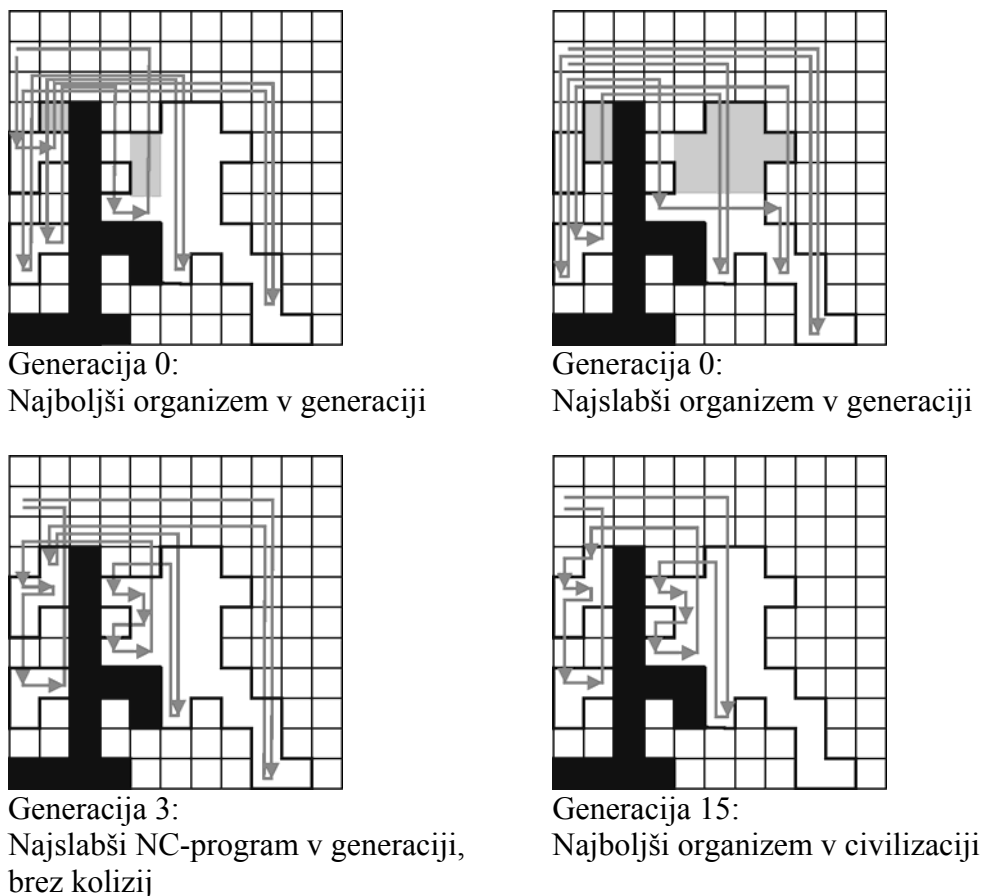
Slika 44 prikazuje nekatere zanimive rešitve med evolucijo NC-programov. Rezultat naključnega ustvarjanja NC-programov je seveda slab. Najboljši organizem v začetni generaciji 0, ima dolžino 96 in število kolizij $K=3$. Kolizije držala orodja z obdelovancem so označene sivo. Za primerjavo podajmo še najslabši NC-program v generaciji 0. Ta program ima dolžino 106 korakov in povzroči kar $K=11$ poškodb orodja ali pa obdelovanca. Najslabši NC-program brez kolizij je nastal v generaciji 3 in ima dolžino 82. V generaciji 15 razvije

evolucija najboljši in najvarnejši NC-program z dolžino 48, ki je v tem zagonu tudi najboljši NC-program.

Slika 43 prikazuje postopni razvoj najboljših rešitev glede na njihovo dolžino. Iz slike je razvidno, da se je najkrajši NC-program razvil že v generaciji 15 in se ohranil vse do zadnje generacije 50.



Slika 43: Dolžina najboljših NC-programov v posamični generaciji

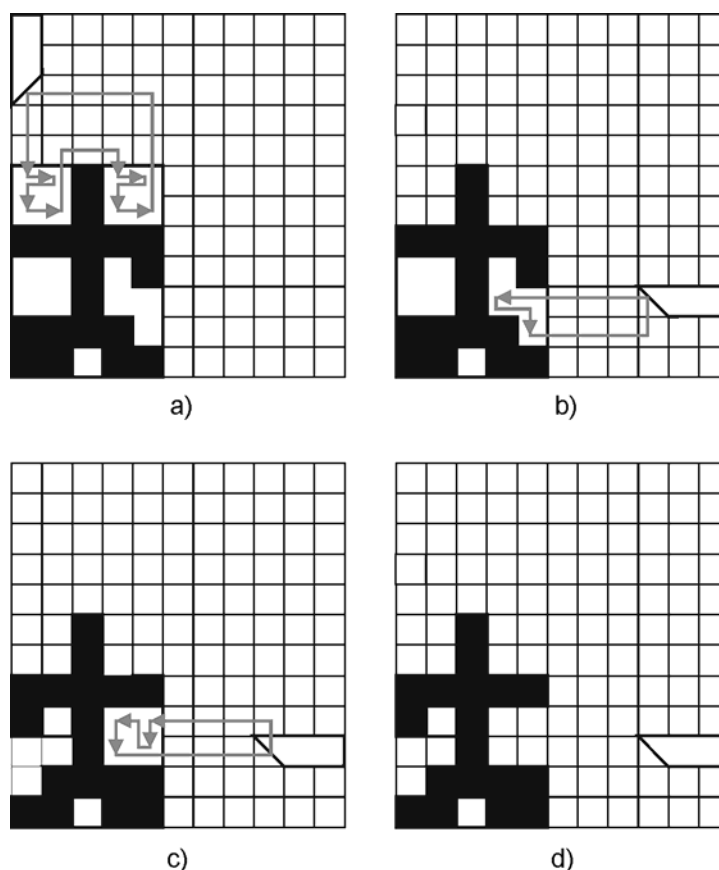


Slika 44: Enostaven izdelek in kompleksen surovec

5.2.3 Primer 3

Izdelkov v splošnem ni mogoče izdelati z enim vpetjem noža ali obdelovanca. Predlagan sistem avtomatično ponudi možnosti ponovnega vpetja noža ali obdelovanca, kjer obdelava z enim vpetjem ni mogoča in tako avtonomno izvede tehnologijo obdelave. Sistem prepozna mesta na obdelovancu, ki jih je nemogoče obdelati, jih v tekoči obdelavi pusti neobdelana, prepne nož in ponovi algoritem na sliki 2. Sistem torej avtonomno sestavi zaporedje vpetij noža in obdelovanca, planira in optimira obdelavo za posamično vpetje ter opozori na napake pri zasnovi izdelka in izbiri obdelovanca. Sistem torej opozori, če izdelka na danem stroju ni mogoče v celoti izdelati.

Slika 10a prikazuje obdelovanca, ki je vpet na način Vpetje-1. Lega noža je pravokotna glede na os vrtenja obdelovanca. Sistem za to vpetje v razmeroma kratkem času izdelava NC-program dolžine 24 korakov. Ker izdelek še ni izdelan, sistem izvede prevpetje noža Vpetje-2, tako da je le-ta sedaj vzporeden z osjo vrtenja glavnega vretena stružnice (Slika 14b). Sistem izdelava NC-program dolžine 12 korakov. Izdelek še vedno ni v celoti izdelan, zato sistem obrne obdelovanec za 180° in obdelava dostopna mesta obdelovanca z NC-programom dolžine 14 korakov (Slika 14c). Na koncu sistem zazna, da izdelek še nima predpisanih dimenzij (Slika 14d), zato označi neobdelana mesta (odebeljeno obrobljeni kvadrati). Izdelek lahko naknadno izdelamo z drugimi obdelovalnimi postopki ali pa spremenimo izdelek ali obdelovanec.



Slika 45: Kompleksen izdelek in enostaven surovec

5.3 Zaključek

V tem poglavju smo prikazali koncept avtomatskega programiranja NC-stružnic z metodo genetskih algoritmov. Raziskava je pokazala, da je predlagan sistem evolucijskega iskanja optimalnih NC-programov učinkovit in univerzalen, zato ga lahko v nadaljevanju uporabimo tudi za programiranje drugih vrst NC-strojov.

Predlagan inteligentni sistem nalogo obdelave najprej hierarhično razdeli na manjše naloge: vpetja, reze in odrezke. Nato določi zaporedje izvajanja vpetij, in inteligentno izdelava NC-programe za posamezno vpetje noža in obdelovanca. Strategijo obdelave izdelava sistem s pomočjo preprostih pravil o gibanju orodja in dovoljenih in nedovoljenih dogodkih.

Zaradi kompleksnosti problematike in velikosti iskalnega prostora rešitev smo uporabili metodo genetski algoritmov, ki posnema biološko evolucijo živih bitij. Inteligentne rešitve in s tem optimalni NC-programi so nastajali postopoma, iz generacije v generacijo, kot posledica interakcij med orodjem in dinamičnim okoljem, ki ga predstavlja obdelovanec.

Poseben poudarek smo namenili ovrednotenju obdelovalnih strategij. Sistem je sposoben avtonomno planirati tehnologijo obdelave, zaznavati obdelana in neobdelana področja obdelovanca, planirati in optimirati delovne in podajalne gibe orodja, zaznati kolizije in ugotoviti, če izdelka na danem stroju ni mogoče v celoti obdelati.

Slaba lastnost sistema je njegova diskretizirana zasnova, ki precej prispeva k časovni zahtevnosti iskanja rešitev. Zato bi bilo vmesno prihodnje raziskave usmeriti v zasnovo

izpopolnjenega sistema z nediskretiziranim obdelovalnim poljem, različnimi načini obdelave (groba obdelava, fina obdelava), različnimi vpetji obdelovanca in orodja, različnimi oblikami orodja ter rezalnimi parametri (globina reza, podajanje, hitrost vrtenja vretena).

6. PROGRAMIRANJE FREZALNEGA STROJA IN EVOLUCIJA

Pri frezanju je posledica relativnega gibanja orodja in obdelovanca, podobno kot pri struženju, premikanje referenčne točke orodja in s tem nastajanje želene oblike izdelka. Freziranje je poleg struženja eden izmed najpomembnejših in najpogosteje uporabljenih proizvodnih procesov v sodobni proizvodnji [42][7][13]. Aktualne raziskave s področja freziranja lahko v grobem razdelimo na: ustvarjanje poti orodja in optimizacijo rezalnih parametrov [62][30]. Pri frezanju optimiramo predvsem: rezalne sile [50] in kvaliteto površine [1][2][34][72]. Raziskave ustvarjanja pot orodja se glede na obdelavo delijo na grobo freziranje in fino freziranje, kjer je glavna problematika ugotavljanje položaja orodja glede na površino obdelovanca [41][63]. Načeloma ustvarjanje poti za grobo freziranje zajema freziranje žepov [42], 2D in 2,5D freziranje [86][62] ter več-osno freziranje [62]. Pristopov za samodejno ustvarjanje poti je več [14][86]. Skupno vsem metodam za samodejno ustvarjanje poti je, da se na določen »inteligenten« način vnaprej ustvari pot orodja, po kateri se orodje potem tudi natančno giblje. Seveda pa pri ustvarjanju poti orodja ne smemo pozabiti na preverjanje in simuliranje, ki omogočata zaznavanje kolizij in simulacijo postopka odrezavanja [29][7]. V literaturi konceptov, ki združujejo samodejno ustvarjanje poti, simulacijo in preverjanje pri postopku freziranja skorajda ni zaslediti [7].

6.1 Predlagan koncept

Ideja predlaganega koncepta za samodejno ustvarjanje NC programov za freziranje je obdelava materiala z naključnimi, navideznimi gibi orodja, ki se tekom simulirane evolucije, na podlagi preprostih pravil, razvijejo v smiselne in optimalne obdelovalne strategije. Pristop ustvarjanja se bistveno razlikuje od konvencionalnega pristopa ustvarjanja NC programov. Za delovanje konvencionalnih sistemov je v začetku potrebno vnesti podatke o geometriji orodja, ter CAD modele surovca in izdelka. Nato se na podlagi izbrane strategije ustvari pot – NC kodo, po kateri se orodje brezpogojno giblje. Preden NC program po-procesiramo – prilagodimo za izvajanje na različnih izvedbah strojev – ga lahko simuliramo ali preverimo. Stroj izdelava izdelek na podlagi NC kode.

Predlagani koncept sestavljata dve stopnji. V sistem je potrebno vnesti podatke o geometriji orodja, surovca, izdelka in stroja. V prvi stopnji koncepta poteka avtomatsko določanje položaja vpetja in število vpetij. Sistem je sposoben zaznati tudi mesta, ki jih strojno ne moremo obdelati in omogoča spremembo CAD modela.

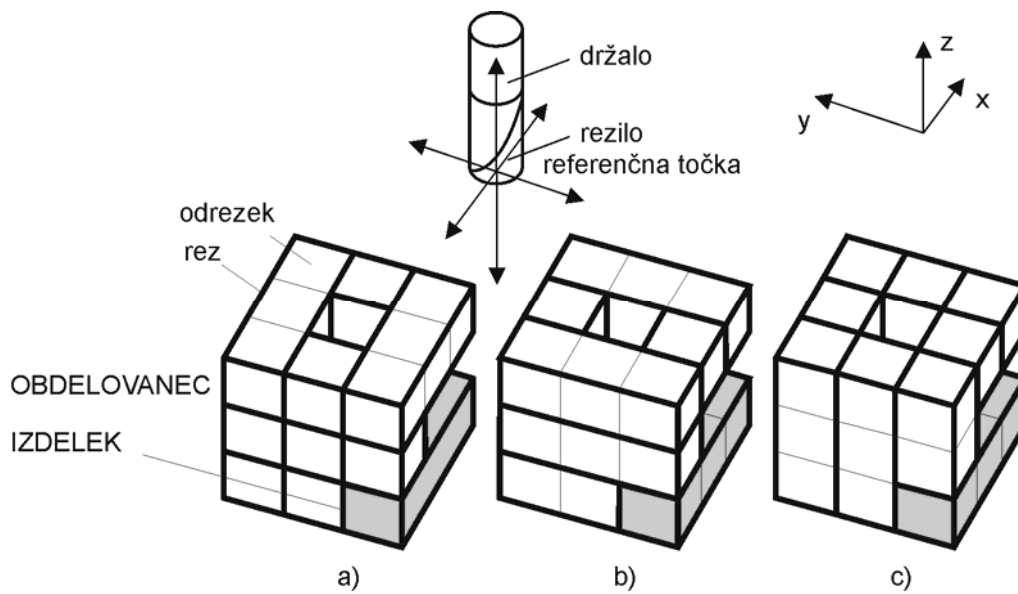
Nastajanje poti orodja predlaganega koncepta združuje 3 stopnje konvencionalnega pristopa generiranja NC code: ustvarjanje poti orodja, simulacija in preverjanje ter poprocesiranje v eno. Razlika pri predlaganem konceptu je, da se pot orodja v splošnem ne ustvari na podlagi v

naprej izdelanih strategij [11][52] in da le-te nastajajo sproti, na podlagi vnešenih vhodnih podatkov ter, še pomembneje, na podlagi spreminjajočih se pogojev med samo (navidezno) obdelavo.

Izdelava izdelka – spreminjanje izdelka med obdelavo in ustvarjanje obdelovalnih strategij je torej interaktivni proces, ki zahteva manj toge, inteligentnejše in tako ekonomične in s strani uporabnika prijaznejše in varnejše rešitve. Obdelovalne strategije predlaganega koncepta zato zajemajo več kompleksnih podatkov, kot so samo ustvarjena pot orodja: informacije o nedovoljenih položajih orodja, tehnologiji obdelave (določanje zaporedij vpetja), konsistentnosti rezov in možnosti zasnove novega izdelka. Zasnovani sistem tako omogoča samodejno določanje zaporedja vpetja in samodejno spreminjanje CAD modela na osnovi izdelane obdelovalne strategije (Podpoglavje 6.2.2), skrbi za ekonomičnost in hkrati omogoča varno delo z izbranim orodjem in strojem. Na podlagi obdelovalne strategije se izdela NC koda, s katero krmilimo stroj med izdelavo izdelka.

6.1.1 Zgradba sistema

Surovec lahko grobo obdelamo s frezanjem na več načinov. Zaradi, v splošnem, velikega števila obdelovalnih strategij, smo sistem za programiranje NC programov za frezanje poenostavili in tako diskretizirali orodje (stroj), obdelovanec in izdelek. Slika 46 prikazuje surovec, izdelek in odgovarjajoče odrezke. Orodje debeline stranice volumskega elementa (kocka), se lahko giblje diskretno v vseh smereh. Volumski element je odvisen od materiala obdelovanca, orodja, tehnoloških parametrov (hitrosti vrtenja, podajanja, globine rezanja). Globina maksimalnega reza je odvisna od velikosti rezila frezala in držala. Referenčna točka orodja je na spodnjem delu orodja. Material, ki ga je potrebno obdelati, razdelimo na več rezov, ki so sestavljeni iz odrezkov. Slika 46 prikazuje načine obdelave (obdelovalne strategije). Zaradi diskretne narave sistema bi lahko izbirali med obdelavo horizontalnih (Slika 46a), vertikalnih (Slika 46b) ali pa kombiniranih rezov (Slika 46c). Relativno gibanje orodja pri predlaganem konceptu ni odvisno od izbire obdelovalne strategije. Način obdelave (združevanje odrezkov v reze) izbiramo zgolj da zmanjšamo število možnih obdelovalnih strategij. Manjši je volumen odrezka, več je rezov. Več je rezov, večje je število možnih obdelovalnih strategij diskretiziranega sistema.

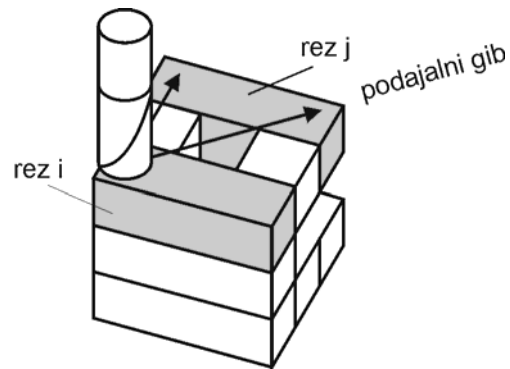


Slika 46: Surovec, izdelek, odpadki in načini obdelave

Slika 33 prikazuje glavni algoritem za genetsko programiranje NC strojev v pseudo kodi. Podobno kot pri struženju je na začetku treba vnesti podatke o surovcu in izdelku. Vnos podatkov je izveden avtomatično, neposredno iz CAD modelov surovca in izdelka. Sistem nato razdeli obdelovalno polje na želeno število enakih volumskih delov (v našem primeru so to kocke). Na koncu podatkovnega vnosa je treba podati še startno in končno točko premikanja orodja in referenčno točko orodja. Odslej je sistem sposoben evolucijskega ustvarjanja in optimiranja NC-programov skozi več generacij. Delovanje algoritma poznamo že iz poglavja 5.

6.1.2 Kodiranje NC-programa

Naključno ustvarjene NC-programe lahko ponovno predstavimo z uteženim grafom. Točke grafa so delovni gibi (rezi), povezave med točkami pa podajalni gibi. Vsaka točka grafa je povezana z vsemi ostalimi točkami grafa. Točke in povezave grafa so utežene. Razlika med uteženim grafom NC-programa pri frezanju in struženju (Slika 34) je le v tem, da so povezave v različnih smereh, različno utežene. Praktično pomeni, da lahko podajalno gibanje od reza i k rezu j traja različno dolgo. Na sliki sta prikazani obe možnosti gibanja orodja od reza i do reza j . Rez j lahko začnemo obdelovati na dveh koncih, zato sta tudi podajalna giba različnih dolžin.



Slika 47: Podajalna giba različnih dolžin

Zaporedje genov (rezov) predstavlja zaporedno gibanje noža od reza do reza. Prvi in zadnji gen seznama predstavljata začetno in končno točko gibanja noža. Vmesni členi ponazarjajo zaporedje rezov, ki so po vrstnem redu podvrženi obdelavi. Nož se najprej s podajalnim gibom giblje od začetne točke do prvega reza v seznamu. Nato izvede delovni gib do prvega reza, ga obdela, nadaljuje s podajalnim gibom do naslednjega reza, ga obdela, in tako naprej dokler ne obdela vseh rezov. Na koncu s podajalnim gibom doseže končno točko. Naključno ustvarjeni NC programi se med seboj razlikujejo po številu potrebnih korakov orodja in številu kolizij med orodjem in obdelovancem.

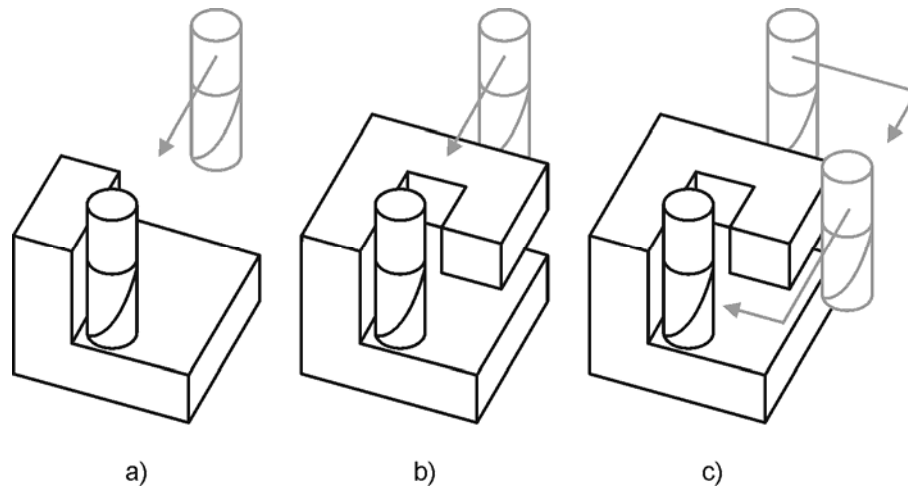
6.1.3 Ovrednotenje populacije

NC program ovrednotimo tako, da obdelovanec navidezno obdelamo (simulacija). Sistem med navidezno obdelavo avtonomno zazna predele surovca, ki jih ni mogoče obdelati, kolizije orodja (preverjanje) z obdelovancem in strojem, konsistentnost rezov ter volumen odvzetega materiala (kateri predeli obdelovanca so bili že obdelani). Število korakov orodja je odvisno od oblike obdelovanca in izdelka. Na obliko obdelovanca med obdelavo vpliva gibanje orodja. Na gibanje orodja vplivata konsistentnost rezov in volumen odvzetega materiala.

Kot smo že omenili pri struženju (Poglavje 5), je rez konsistenten, ko se vsaj en odrezek reza drži obdelovanca.

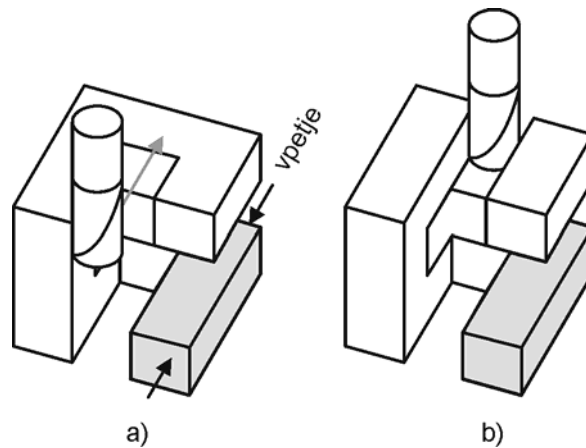
NC-program ovrednotimo podobno kot pri struženju (Poglavje 5).

Glede na spreminjanje obdelovanca med obdelavo se spreminja pot in dolžina poti. Gibanje orodja za razliko od struženja (ravnina) poteka v prostoru. Pri gibanju orodja iz začetne točke v končno točko se material v primeru 48b obdeluje, v primeru 48a pa ne, kar je potrebno pri ovrednotenju obdelovalne strategije upoštevati.



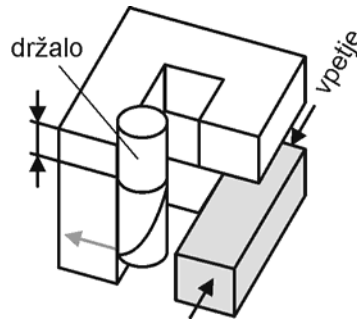
Slika 48: Obdelovanec kot dinamično okolje

Med podajalnimi in delovnimi gibi orodja sistem budno spremlja gibanje noža, sproti preverja konsistentnost rezov in sešteva možne kolizije. Slika 49 prikazuje, kako se bo sivo obarvani material, pri gibanju orodja, od obdelovanca ločil. Temu materialu pravimo, da je nekonsistenten. Dobrodošla lastnost sistema za samodejno ustvarjanje NC programov je, da je sposoben zaznati nekonsistenten material, ker je to material, ki ga naknadno ni potrebno obdelati.



Slika 49: Konsistentnost rezov

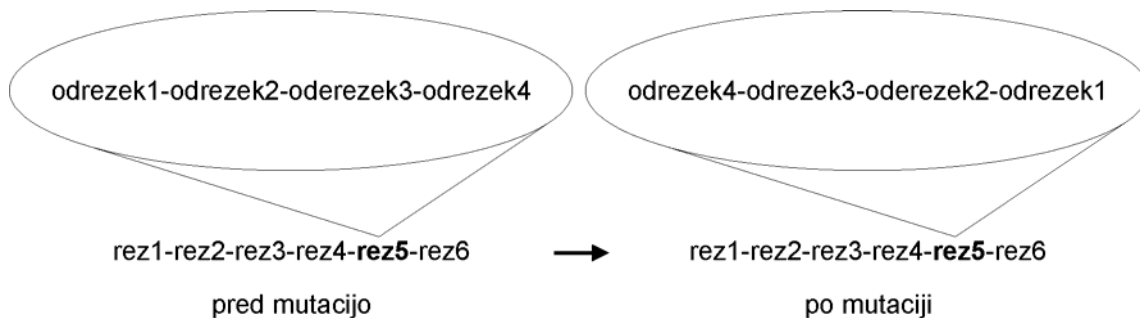
Najpomembnejša lastnost sistema je detekcija kolizij orodja (držala orodja) z obdelovancem, kar prikazuje slika 50. Pri gibanju orodja z držalom zadane v obdelovanec. Čim več je držala v stiku z obdelovancem (Slika 50), težja in nevarnejša je kolizija. Algoritem za preverjanje konsistentnosti in zaznavanje kolizij je predstavljen v poglavju 5.



Slika 50: Kolizija

6.1.4 Genetske operacije

Uporabili smo genetske operatorje reprodukcije (selekcije), križanja, permutacije in mutacije. Operacije reprodukcije, križanja in permutacije so opisane v poglavju 5. Pri operaciji mutacija pa se na naključno izbranemu rezu zamenja smer obdelave (Slika 51). Namesto obdelave odrezkov naključno izbranega reza od *odrezka1* do *odrezka2*, poteka po mutaciji obdelava v smeri od *odrezka2* do *odrezka1*. Za izbiro kromosomov, ki sodelujejo v operacijah izbiranja in spreminjanja, smo uporabili turnirsko izbiro.



Slika 51: Genetska operacija mutacija NC programa pri frezanju

6.2 Rezultati in diskusija

Delovanje in učinkovitost sistema sta prikazani na izbranih testnih primerih. Najprej prikazujemo primer izdelave preprostega izdelka iz relativno zapletenega surovca (poglavje 6.2.1). Za podrobnejši prikaz sposobnosti sistema in detajlne analize delovanja sistema pa podajamo še primer izdelave relativno zapletenega izdelka iz oblikovno preprostega obdelovanca (poglavje 6.2.2).

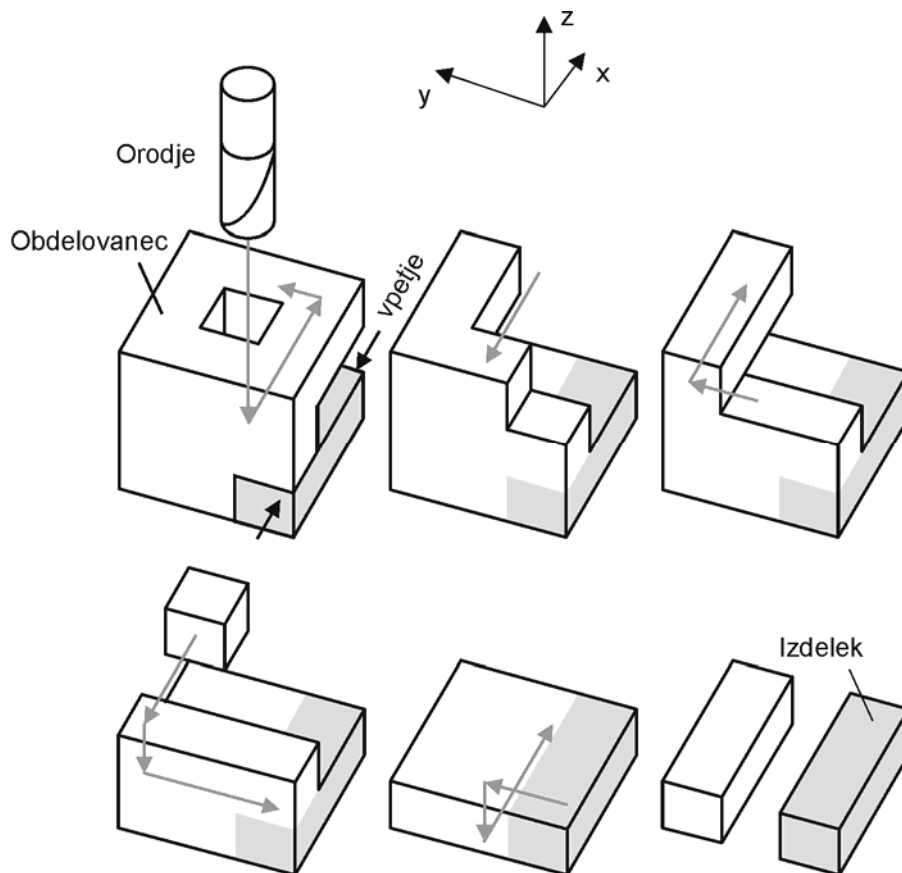
Pri vseh zagonih sistema so bili uporabljeni enaki parametri evolucije in sicer velikost populacije 500, maksimalno število generacij 50, verjetnost reprodukcije 0.1, križanja 0.6, permutacije 0.2 in mutacije 0.1. Uporabljena je bila turnirska izbira organizmov z velikostjo turnirja 7.

6.2.1 Primer 1

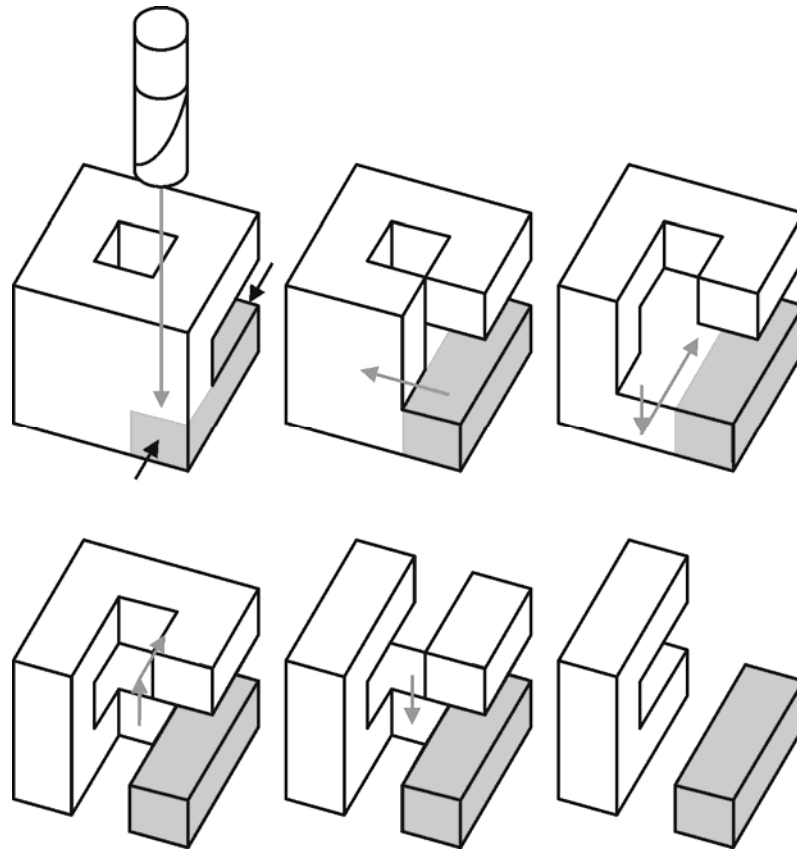
V tem primeru je prikazana izdelava preprostega izdelka iz relativno zapletenega surovca. Velikost izdelka je 3 volumnske kocke, obdelovanca pa 20. Izbrali smo horizontalni način obdelave v smeri x osi, kar pomeni da odrezke grupiramo v horizontalne reze, vzporedne z x osjo (Slika 46a). V uvodnem delu poglavja smo omenili, da obdelava pri predlaganem konceptu ni odvisna od izbire obdelovalne strategije. Razlika med konvencionalno in genetsko obdelavo za dani primer je prikazana na slikah 52 in 53.

Pri konvencionalnem postopku predlagana strategija obdelave vsiljuje obdelavo izdelka na ta način, da se orodje skuša gibati večino časa obdelave v ravnini vzporedno s horizontalno ravnino (xy ravnino). Pri konvencionalnem načinu obdelave (Slika 52) je orodje izdelalo izdelek s 41-imi koraki. Orodje je bilo v stiku kar s 13 odrezki.

Pri novem konceptu pogledimo razvoj ene izmed 20-ih neodvisno razvitih civilizacij. V začetni generaciji 0 je bil rezultat slepega iskanja relativno slab. Najboljša strategija dolžine 41 je povzročila lom orodja samo enkrat v primerjavi z najslabšo strategijo začetne generacije dolžine 49, ki je zlomila orodje (ali trčila v obdelovanec) kar 5 krat. Prva collision-free strategija dolžine 45 se je pojavila že v generaciji 1. Najboljša strategija dolžine 33 se je pojavila že v generaciji 7 (od 50-ih) in je prikazana na sliki 53. Orodje je bilo v stiku le s 7 odrezki. Konvencionalna obdelava se od genetske obdelave glede na dolžino poti razlikuje za 19.51%, glede na obrabo orodja pa kar za 46.15%.



Slika 52: Konvencionalna obdelava



Slika 53: Genetska obdelava

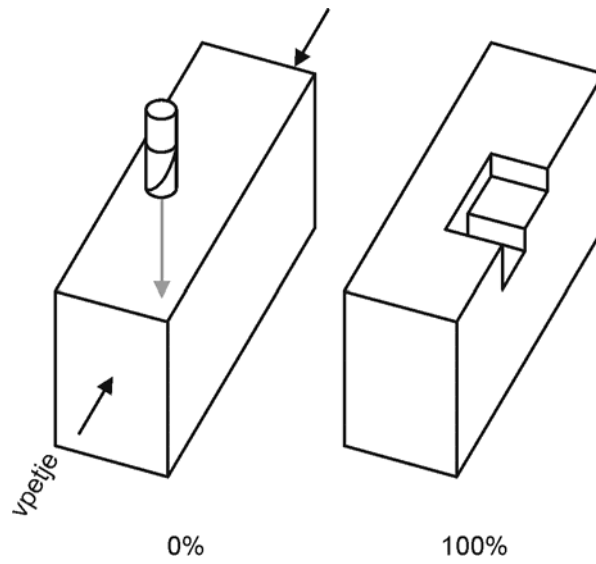
6.2.2 Primer 2

Tokrat je potrebno izdelati relativno kompliciran izdelek (Slika 55e) iz enostavnega obdelovanca 54a. Sistem samodejno zazna, mesta, kjer obdelava ni mogoča z enim vpetjem. Sistem bo moral enkrat prevpeti obdelovanec. Sprva bo obdelal vdolbino na boku obdelovanca (Slika 54), kasneje pa še njegov zgornji del (Slika 55).

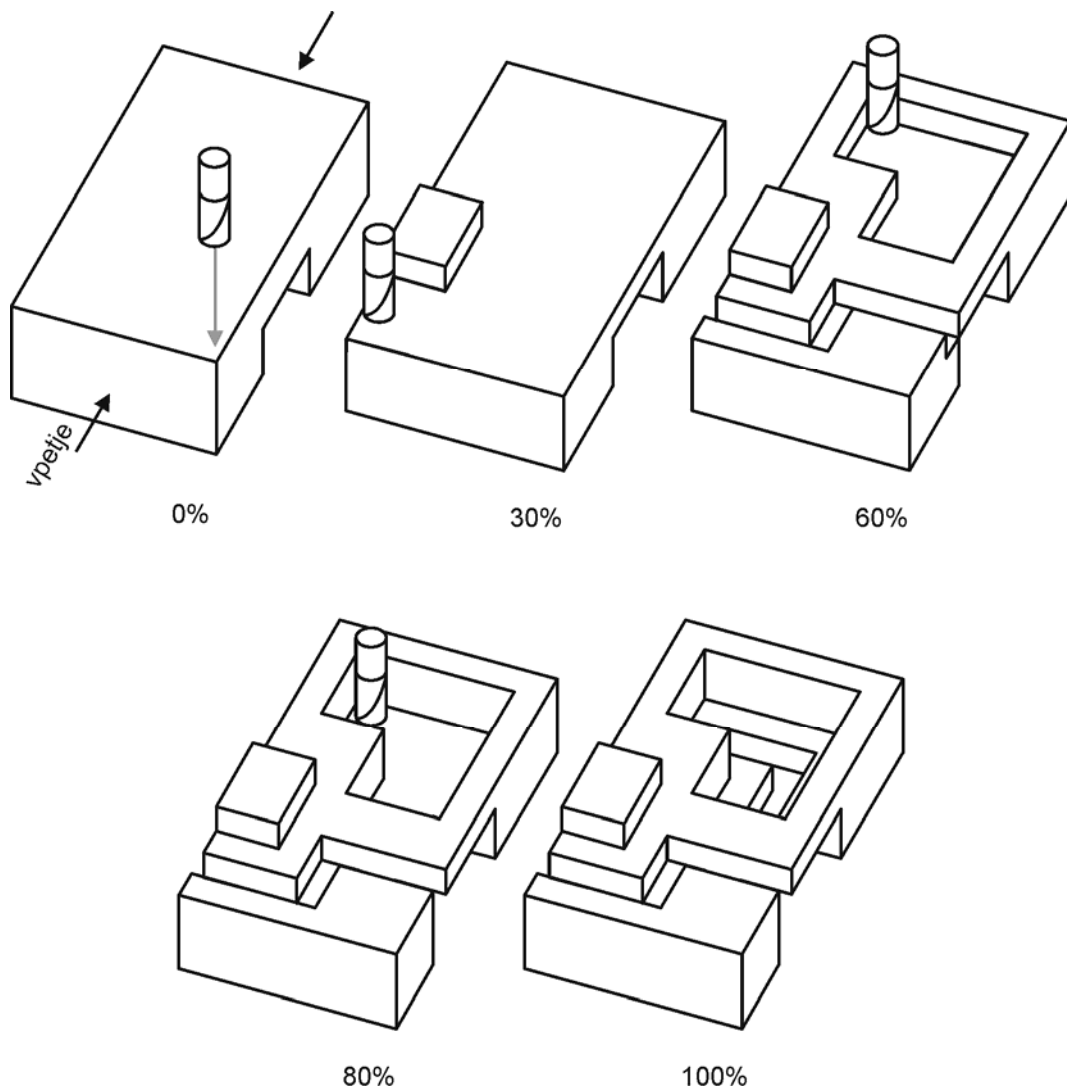
Sistem z obdelavo na boku ni imel težav. S 33-imi koraki orodja je izdelal vdolbino na boku (Slika 54). Sistem je nato prevpel obdelovanec.

Slika 55 prikazuje potek pri konvencionalnem, postopnem odrezavanju materiala, ki smo ga izvedli za primerjavo s predlaganim konceptom. Material se odrezuje postopoma, plast za plastjo. Obdelava je prikazana v odvisnosti od dolžine (procenti dolžine obdelovalne poti) prepotovane poti orodja. Število korakov, ki ga orodje potrebuje, da med gibanjem med začetno in končno točko izdelata izdelek, je v prvem primeru 263. Z vidika konsistentnosti rezov sistem ni imel težav, ker je obdelovanec preproste oblike.

Na slikah 56 do 58 bomo prikazali razvoj civilizacije NC-programov po prevpetju obdelovanca. NC-programi se izdelujejo in izboljšujejo postopoma, iz generacije v generacijo.

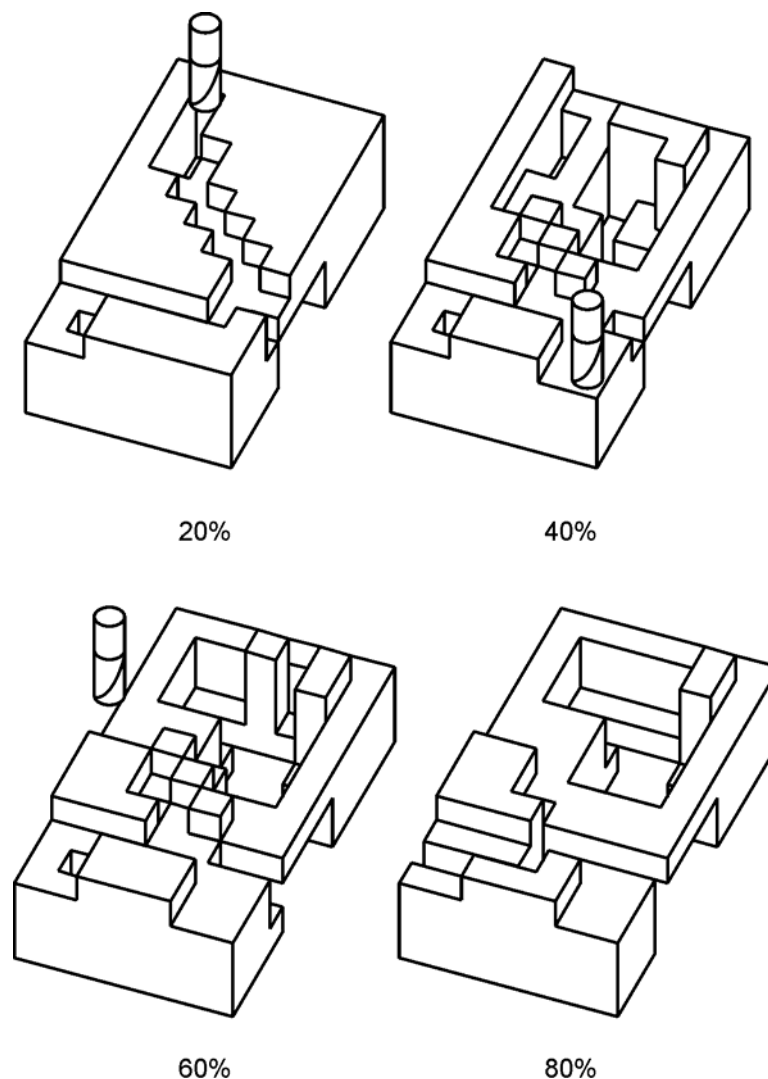


Slika 54: Prvo vpetje



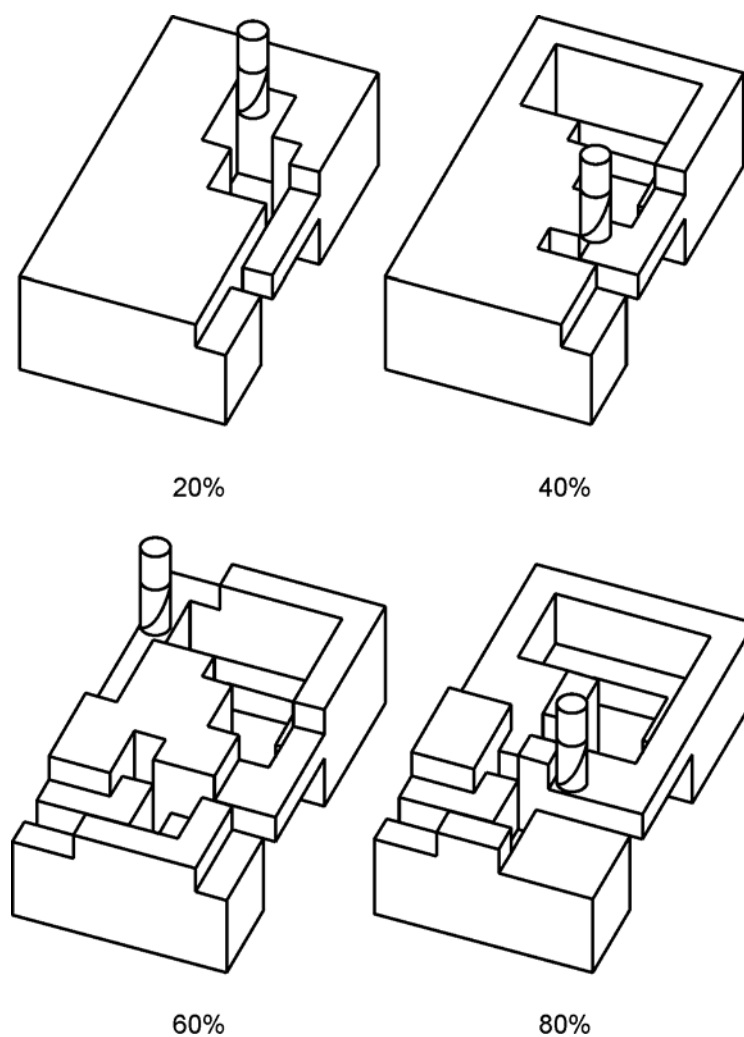
Slika 55: Drugo vpetje - postopna obdelava

V začetni generaciji – generaciji 0 se naključno ustvarijo NC-programi. Slika 56 prikazuje obdelavo (procenti dolžine obdelovalne poti) po najboljšem naključno ustvarjenem NC-programu v generaciji 0. Dolžina poti najboljšega naključno ustvarjanja NC-programa v generaciji 0 znaša 257 korakov, kar je manj kot dolžina poti orodja pri konvencionalnem načinu obdelave. Z držalom orodja nismo zadeli v obdelovanec. Na videz kaotično gibanje orodja na podlagi naključno ustvarjenega NC-programa ne daje vtisa po zmanjševanju dolžine obdelave. V naprej ustvarjene poti, ki jih dobimo iz geometrijskih podatkov o izdelku, obdelovancu in orodju, pri predlaganem konceptu ne predstavljajo natančno določenih poti orodja, temveč so le smernice med katerimi orodje izbira in se posledično inteligentno, upoštevajoč preprosti navodili: čim hitreje izdelaj izdelek in ne poškoduj sebe in okolice, optimalno giblje. Pri enaki obrabi orodja, kot pri konvencionalnem načinu obdelave, se je obdelovalnost povečala za 2.28%.



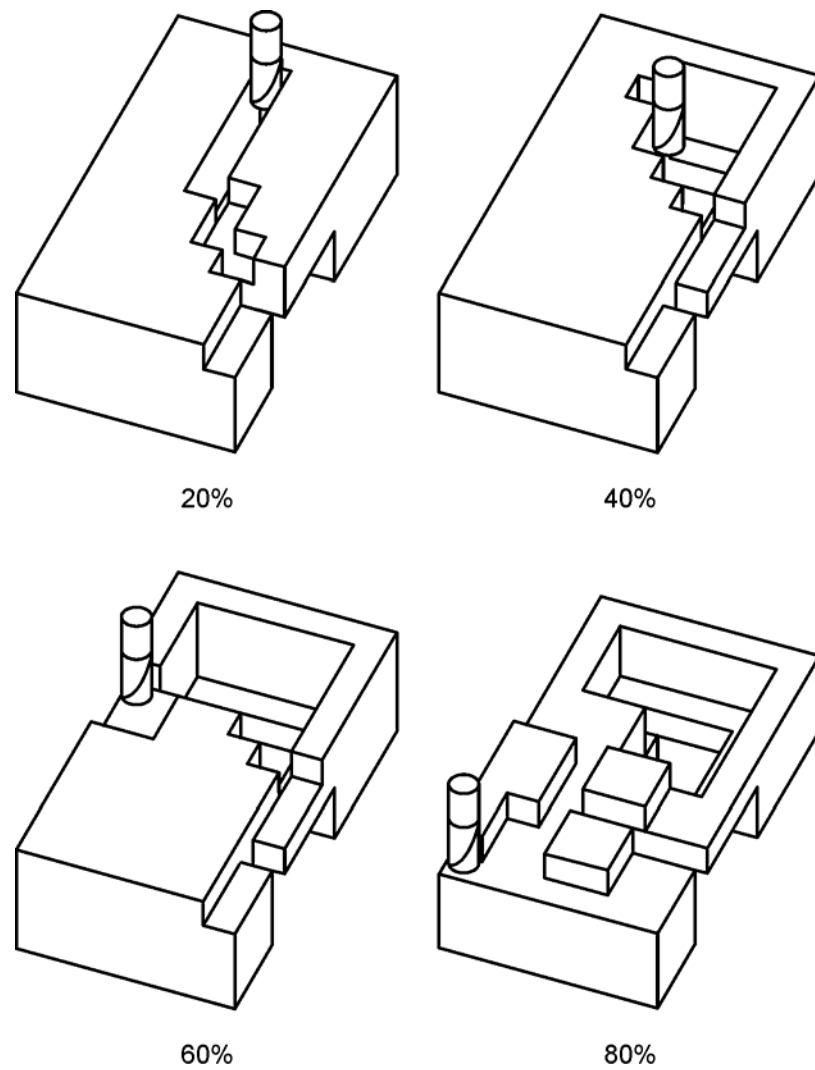
Slika 56: Drugo vpetje – genetska obdelava – generacija 0

Slika 57 prikazuje potovanje orodja glede na najboljši NC-program generacije 10. Dolžina poti najboljšega naključno ustvarjenega NC-programa v generaciji 10 znaša 183 korakov, kar pomeni, da smo, glede na konvencionalni način obdelave, obdelovalnost povečali za 30.41%. Z držalom orodja nismo zadeli v obdelovanec.



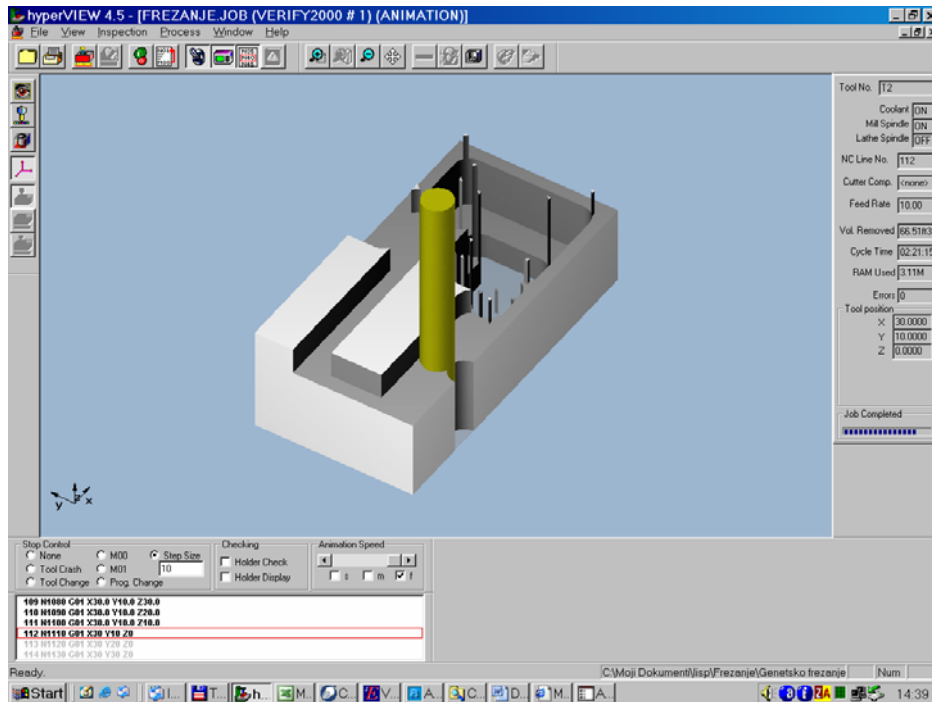
Slika 57: Drugo vpetje – genetska obdelava – generacija 10

Slika 58 prikazuje potovanje orodja glede na najboljši NC-program generacije 45, kjer se je pojavil najboljši NC-program civilizacije. Dolžina poti najboljšega NC-programa v civilizaciji znaša 137 korakov, kar pomeni, da smo, glede na konvencionalni način obdelave, obdelovalnost povečali kar za 47.90%. Z držalom orodja nismo zadeli v obdelovanec.



Slika 58: Drugo vpetje – genetska obdelava – generacija 45

Slika 58 prikazuje preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5.



Slika 59: Preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5

6.3 Zaključek

V poglavju smo prikazali koncept samodejnega programiranja NC-frezalnega stroja, ki spominja na zasnovani koncept programiranja NC-stružnice, le da je sistem postavljen v tro-dimenzionalno, namesto v tro-dimenzionalno okolje.

Slaba lastnost sistema je ponovno njegova diskretizirana zasnova in z zasnovo povezana časovna zahtevnost iskanja rešitev. Prihodnje raziskave bi ponovno lahko smiselno usmerili v zasnovo izpopolnjenega sistema z nediskretiziranim obdelovalnim poljem, različnimi načini obdelave (groba obdelava, fina obdelava), različnimi vpetji obdelovanca in orodja, različnimi oblikami orodja ter rezalnimi parametri (globina reza, podajanje, hitrost vrtenja vretena).

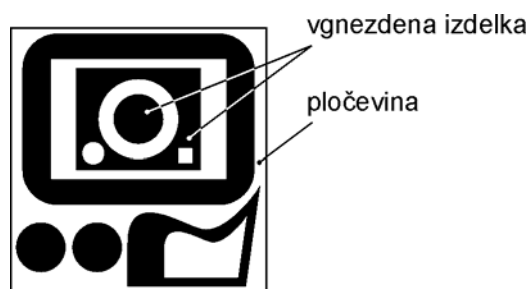
V naslednjem poglavju bo prikazan sistem za samodejno programiranje NC-stroja za rezanje pločevine, ki že skorajda v celoti vsebuje nediskretizirane elemente zasnove sistema. Zaradi časovnih omejitev so diskretizirani elementi zasnove sistema ohranjeni v manjši meri.

7. PROGRAMIRANJE STROJA ZA RAZREZ PLOČEVINE IN EVOLUCIJA

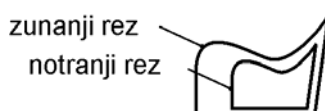
Lasersko rezanje omogoča brezkontaktni, hitrejši razrez tako mehkega kot trdega materiala z ožjimi zarezi, zato ga v sodobni industriji težko pogrešamo. Sočasno pa se s povečano uporabo laserja ter ostalih NC strojev povečuje potreba po ustreznih, enostavnih, preprostih, zanesljivih in uporabniku prijaznih programskih paketov za samodejno ustvarjanje NC-programov [47]. Samodejno ustvarjanje NC-programov pri laserskem razrezu materiala združuje CAD koncepte, razpoznavanje značilnosti in ustvarjanje in optimiranje NC-programov [52]. V tem poglavju se bomo omejili na samodejno ustvarjanje NC-programov za dvo-dimenzionalni razrez materiala.

7.1 Zgradba sistema

V splošnem sistemi za samodejni razrez pločevine izdelajo NC-program na podlagi predhodno izdelane razporeditve izdelkov po pločevini [52]. Dvo-dimenzionalni izdelki so v predlaganem sistemu lahko poljubne velikosti in oblik. Lahko so vgnezdjeni (Slika 60), zato lahko vsebujejo notranje in zunanje meje, ki jih bomo imenovali rezi (Slika 61).



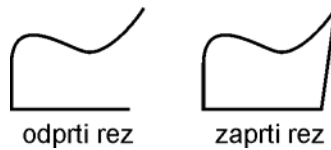
Slika 60: Izdelki in pločevina



Slika 61: Zunanje in notranje meje

Na začetku sistem razpozna značilnosti na podlagi CAD risbe. Samodejno se določijo odnosi med rezi in nato se določijo še vbodi. Rezi so lahko ugnezdjeni ali neugnezdeni. Če gnezdo vsebuje le en rez, je le-ta neugnezden. Rezi so lahko odprti ali zaprti (Slika 62). Vbodi so mesta na pločevini, pri katerih pride do vžiga laserja. Ponavadi je rez pri vžigu laserja precej nenatančen [3], zato se rezanje po določenem rezu izvede naknadno. Vbodi se določijo

le pri zaprtih rezih. Nadalje sistem ustvarja in optimira NC-programe za razrez pločevine. Ustvarjanje in optimiranje NC-programov je izvedeno na podlagi že opisanega genetskega algoritma (Slika 33). Slika 63 prikazuje algoritem sistema za programiranje stroja za razrez pločevine v pseudo kodi.



Slika 62: Odprti in zaprti rez

```

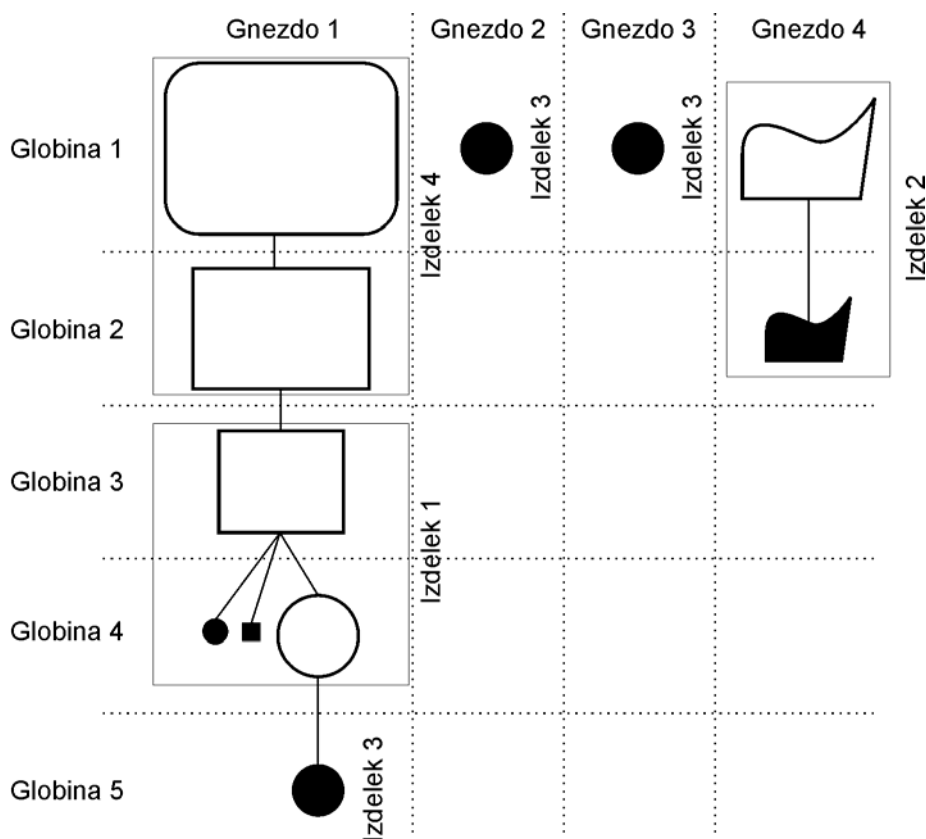
začni
  vnos_podatkov o surovcu, izdelkih, startni, končni točki
  razpoznavanje_značilnosti
  določanje_vbodov
  začni
    t←0
    če poznamo_NC_program
      vstavi P(t)
      drugače ustvari P(t)
    ovrednoti P(t)
    dokler (ni izpolnjen_ustavitveni_pogoj)
      začni
        t←t+1
        spreminjaj P(t)
        ovrednoti P(t)
      končaj
  končaj
  izpis rezultatov
končaj

```

Slika 63: Algoritem za programiranje NC-stroja v pseudo kodi

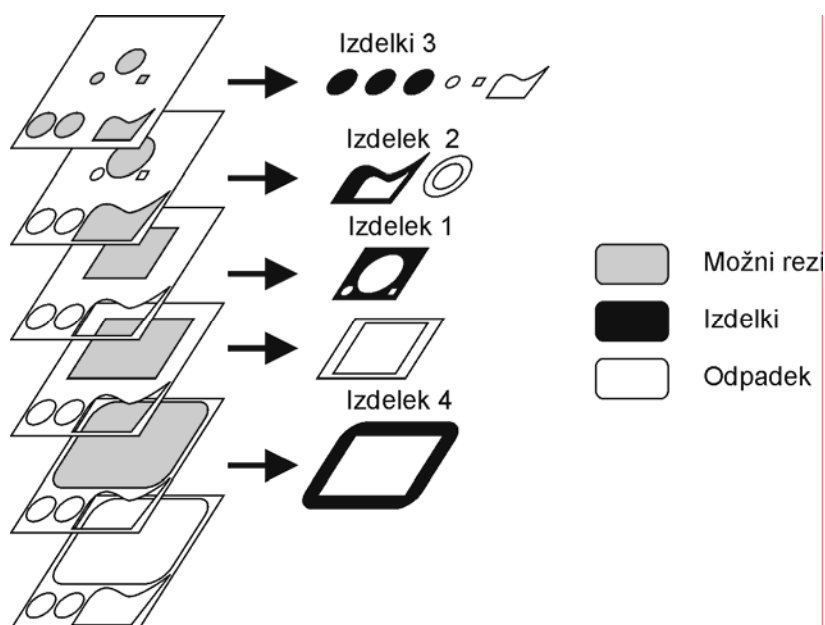
7.1.1 Razpoznavanje značilnosti

Razporeditev izdelkov je sestavljena iz ugnezdenih in nevgnezdenih izdelkov. Kot so lahko ugnezdeni izdelki so lahko ugnezdeni tudi rezi. Na sliki 64 so prikazana gnezda in njihove globine za primer slike 60. Gnezdo 2 in 3 sta sestavljena samo iz rezov izdelka 3, zato v tem primeru rečemo, da sta reza izdelka 3 nevgnezdena. Gnezdo 1 je sestavljeno iz rezov izdelka 4, 1 in 3. Izdelki 4, 1 in 3 so ugnezdeni, prav tako kot njihovi rezi. V primeru gnezda 4 pa izdelek 2, v nasprotju z njegovimi rezi, ni ugnezden. Gnezdrom lahko ugotovimo tudi globino. Gnezdo 1 ima globino 5, gnezdi 2 in 3 sta globine 1, gnezdo 4 pa je globine 2. Sistem za razpoznavanje značilnosti je sposoben samodejno razpoznati vgnezdenost izdelkov in rezov ter določiti globino gnezd.



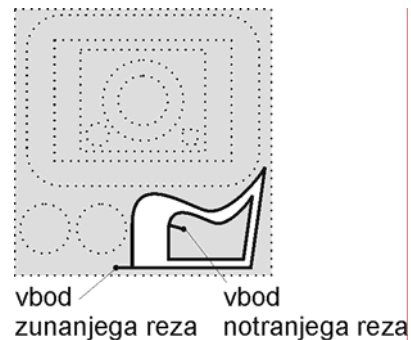
Slika 64: Gnezda rezov in globine gnezd

Zaradi natančnosti izdelave je potrebno upoštevati, da v gnezdu najprej režemo globlje ugnezdene reze. Na sliki 64 lahko na začetku rezanja režemo le črno označene reze izdelkov; izbiramo lahko torej med 6 možnostmi. Režemo torej reze od globljih globin gnezd vse do plitvin (globina gnezda 1). Za boljšo predstavljenost kaže slika 65 postopno rezanje od rezov globljih globin do plitvine. Nazadnje se izreže rez na globini 1 gnezda 1.



Slika 65: Postopno rezanje od rezov globljih globin gnezd do plitvin

Na podlagi zunanjih in notranjih mej izdelkov se določijo notranji in zunanji rezi ter pripadajoči vbodi (Slika 66).

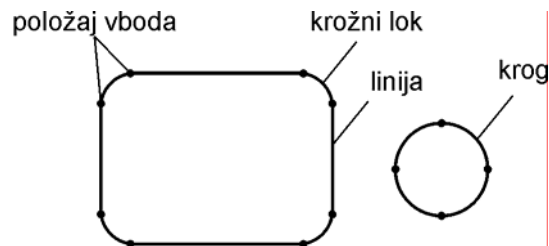


Slika 66: Vbodi, zunanji in notranji rezi

Na sliki 64 lahko enostavno ugotovimo, kateri rezi so notranji in kateri zunanji. Notranji rezi se vedno nahajajo na parnih globinah gnezda (2, 4, 6,...), zunanji pa na neparnih globinah gnezda (1, 3, 5,...). V skladu s to ugotovitvijo se določajo vbodi zunanjih in notranjih rezov.

7.1.2 Določanje vbodov

Posamični rezi so lahko sestavljeni iz linij, krogov in krožnih lokov. Pri odprtih rezih, kot smo že omenili se vbodi ne določajo. Notranji in zunanji vbodi se določajo na enak način. Slika 67 prikazuje določanje položaja vboda. Zaradi časovne zahtevnosti izvajanja predlaganega sistema, položaja vbodov ne moremo poljubno določiti. Položaji vboda se določijo le na obeh koncih linij in krožnih lokov, ter na vsakih 90° kroga.



Slika 67: Določanje položaja vboda

Naknadno se določi orientacija vboda, ki je za rez enake oblike različna glede na to ali je rez notranji ali zunanji. Na podlagi določenih položajev vbodov se izračunata koordinati težišča položajev vbodov (X_t , Y_t) po splošno znanih enačbah [57]:

$$X_t = \frac{\sum_{i=1}^n x_i}{n} \quad (3)$$

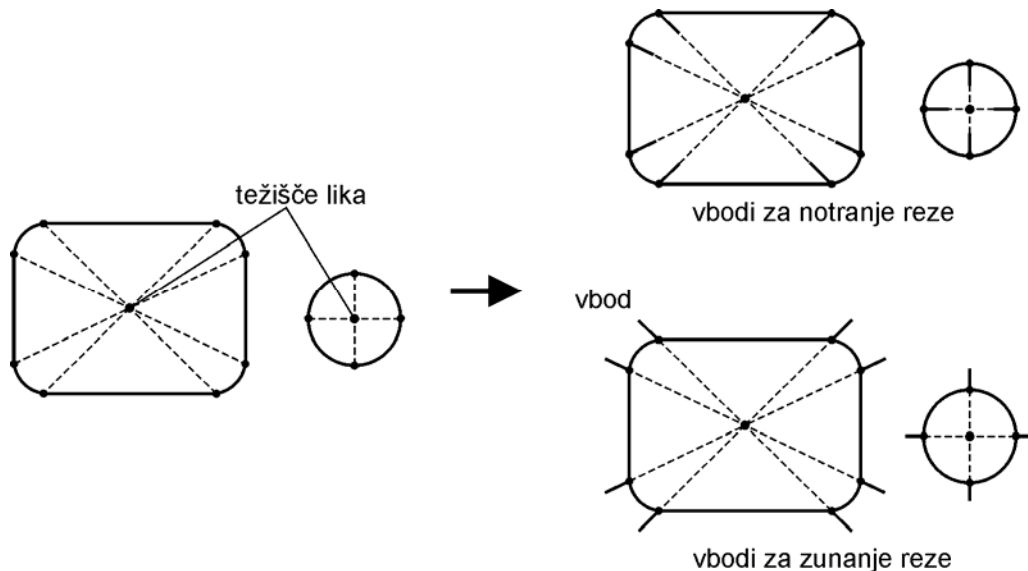
in

$$Y_t = \frac{\sum_{i=1}^n y_i}{n}, \quad (4)$$

kjer so:

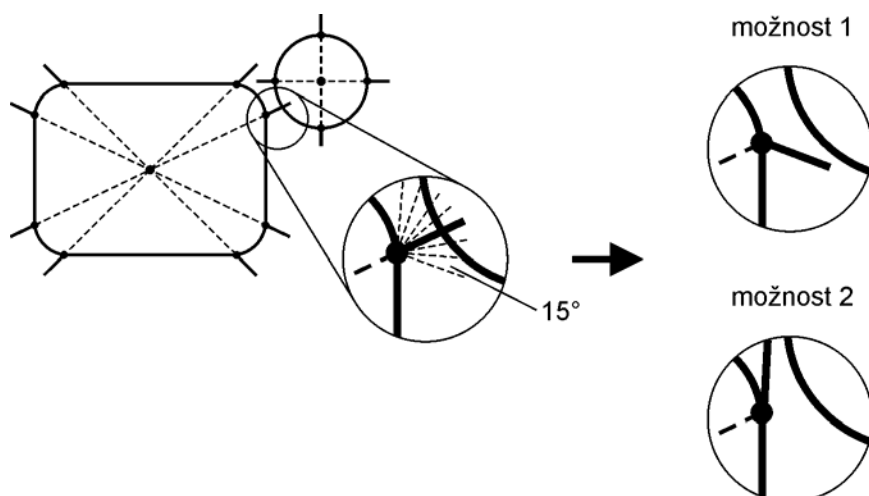
- n število vbodov,
- i indeks vboda,
- x_i x koordinata položaja vboda in
- y_i y koordinata položaja vboda.

Na podlagi izračunanega težišča položajev vbodov se izračunajo vbodi, ki so pri notranjih izbočenih rezih v obrnjeni proti težišču, pri zunanjih rezih pa so obrnjeni v stran od težišča. Na sliki 68 je prikazano določanje vbodov za notranje in zunanje reze.



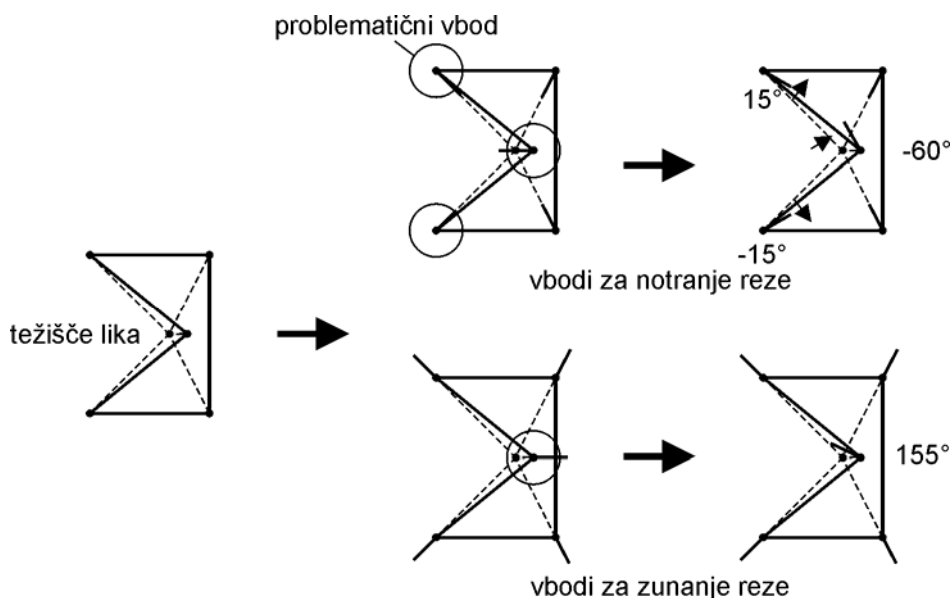
Slika 68: Določanje vbodov za notranje in zunanje reze

Včasih se zgodi, da se vboda ne da enostavno ali celo ne da določiti. Na sliki 69 je prikazano določanje zunanjih vbodov, kjer so rezi preveč skupaj. Vbod na sliki, ki je prikazan povečan, je zaradi bližine krožnega reza potrebno zavrteti okrog položaja vboda za določen kot. Sistem samodejno ustvarja vbode vsakih 15° od prvotnega položaja vboda. Izbere prvi možni vbod. V primeru na sliki 69 sta možna celo dva vboda. Sistem naključno izbere samo enega.



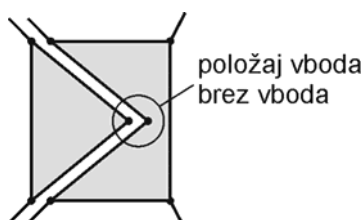
Slika 69: Določanje vbodov pri sosednjih rezih

Slika 70 prikazuje določanje vbodov notranjih in zunanjih vbočenih rezov. Vbode, ki so problematični je potrebno zavrteti okrog položaja vboda za določen kot. Na sliki 70 desno zgoraj je bilo potrebno en vboč zavrteti za 15° v protiučni smeri okrog položaja vboda, drug za 60° v urni smeri in tretji za 15° v urni smeri. Na sliki 70 desno spodaj pa je bilo potrebno problematični vboč zavrteti za 155° v protiučni smeri. Problematični vbodi so na sliki označeni s krogom.



Slika 70: Določanje vbodov vbočenih rezov

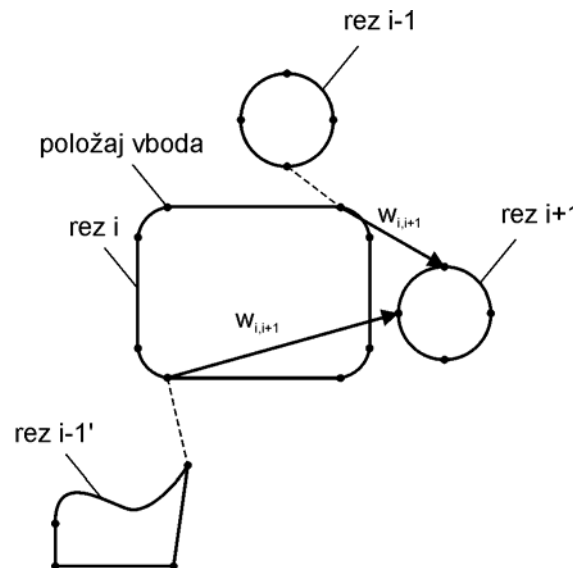
Slika 71 prikazuje primer, pri katerem ni bilo moč določiti vboda na s krogom označenih položajih vboda. Na kakršenkoli način je sistem zavrtel vboč, orientacija vboda ni bila možna, zato sistem vboda ni določil.



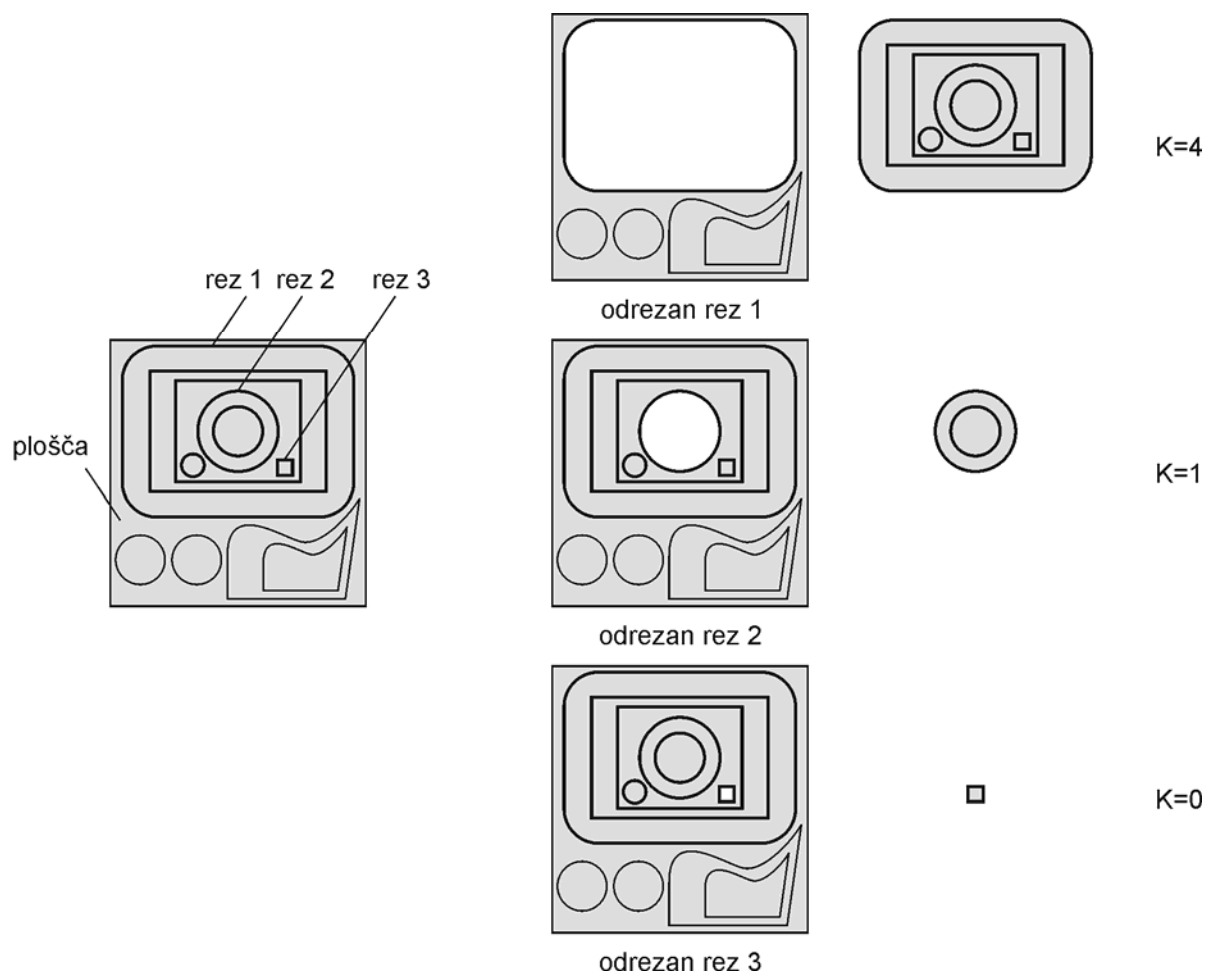
Slika 71: Položaj vboda, na katerem ni bilo moč določiti vboda

7.1.3 Kodiranje NC-programa

Naključno ustvarjene NC-programe lahko ponovno predstavimo z uteženim grafom. Točke grafa so delovni gibi (rezi), povezave med točkami pa podajalni gibi. Vsaka točka grafa je povezana z vsemi ostalimi točkami grafa. Točke in povezave grafa so utežene. Uteži na povezavah od reza i k rezu $i+1$ so lahko različno utežene. Vrednosti uteži med rezoma i in $i+1$ je odvisna od predhodno rezanega reza $i-1$ ($i-1'$). Slika prikazuje spremembo dolžine poti med rezoma i in $i+1$, v odvisnosti od predhodno rezanega reza. Pri gibanju rezalnika od reza i k rezu $i+1$ sistem izbere najbližji vboč reza $i+1$; torej se rezalnik giblje od vboda reza i , ki je najbližji vboču reza $i-1$ ($i-1'$), do najbližjega vboda rezu $i+1$. Na sliki vbodi zaradi nazornosti niso prikazani. Prikazani so le položaji vbodov.

Slika 72: Različno utežene uteži od reza i k rezu $i+1$

Zaporedje genov (rezov) ponovno predstavlja zaporedno gibanje rezalnika od reza do reza. Prvi in zadnji gen seznama predstavljata začetno in končno točko gibanja rezalnika. Vmesni členi ponazarjajo zaporedje rezov, ki so po vrstnem redu podvrženi rezanju. Nož se najprej s podajalnim gibom giblje od začetne točke do prvega reza v seznamu. Nato izvede delovni gib do prvega reza, ga izreže, nadaljuje s podajalnim gibom do naslednjega reza, ga izreže, in tako naprej dokler ne izreže vseh rezov. Na koncu s podajalnim gibom doseže končno točko. Naključno ustvarjeni NC programi se med seboj razlikujejo po dolžini opravljene poti rezalnika in številu preskočenih globin v gnezdu K . Slika 73 prikazuje različna števila preskočenih globin gnezda 1 iz slike 64. Za primer, v gnezdu 1 izberemo 3 reze: rez 1, ki je na globini 1, rez 2, ki je na globini 4 in rez 3, ki je na globini 5. Globina gnezda 1 je 5. Če iz cele plošče, ki je na stroju vpeta, izrežemo rez 1, bo plošča po odstranitvi izrezanega kosa, zgedala kot je prikazano na sliki 73 desno zgoraj. Iz vpete plošče sedaj ne moremo nadaljevati z rezanjem reza 2 in reza 3. Število preskočenih globin gnezda znaša 5 (globina gnezda 1) zmanjšano za 1 (globina reza 1 v gnezdu 1); torej $K=5-1=4$. Če iz cele vpete plošče izrežemo rez 2 (Slika 73 desno sredina) bo število preskočenih globin gnezda 1 $5-4=1$, če pa iz cele vpete plošče izrežemo rez 3 (Slika 73 desno spodaj), pa bo število preskočenih globin gnezda 1 enako 0.



Slika 73: Število preskočenih globin v gnezdu

7.1.4 Ovrednotenje populacije

NC-program ovrednotimo podobno kot pri struženju (Poglavje 5) in frezanju (Poglavje 6), le da namesto kolizij v enačbi nastopa število preskočenih globin K . Da osvežimo spomin, bomo enačbo 2 (Poglavje 5) ponovno navedli v rahlo spremenjeni obliki:

$$\text{Ovrednoti}_{\text{NC-program}} = w_{1,s} + \sum_{i=1}^{n-1} w_{i,i+1} + \sum_{i=1}^n r_i + w_{n,c} + f \cdot K, \quad (5)$$

kjer so:

- $w_{i,i+1}$ dolžina podajalnega med rezoma i in $i+1$, v odvisnosti od predhodnega reza $i-1$,
- $w_{s,1}$ dolžina podajalnega giba med začetno točko s in prvim rezom,
- $w_{n,c}$ dolžina podajalnega giba od zadnjega reza n do končne točke c ,
- r_i dolžina reza i ,
- f faktor vpliva in
- K število preskočenih globin.

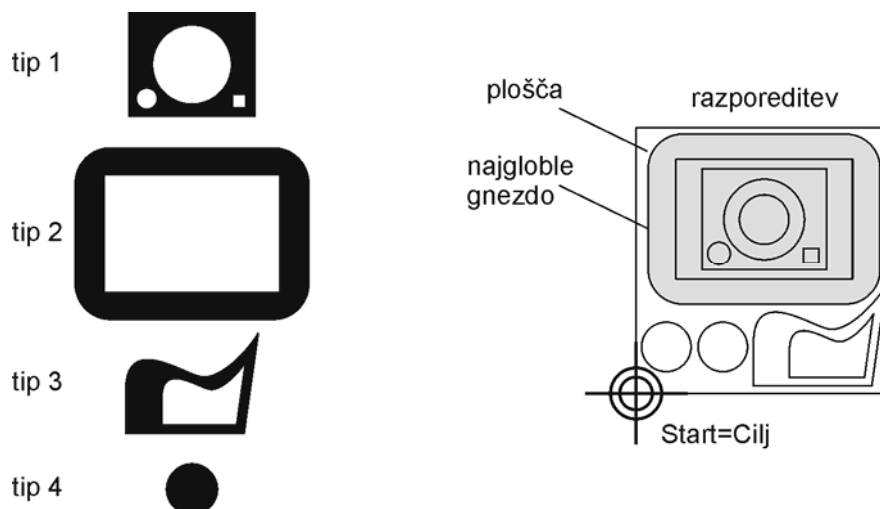
7.1.5 Genetske operacije

Uporabili smo genetske operatorje reprodukcije (selekcije), križanja in permutacije (Podpoglavje 5.1.4). Za izbiro kromosomov, ki sodelujejo v operacijah izbiranja in spreminjanja, smo uporabili turnirsko izbiro.

7.2 Primer 1

Za prikaz delovanja sistema za programiranje stroja za razrez pločevine smo si izbrali dva primera; prvi je prikazan na sliki 74.

Plošča je dimenzij 1000×1000 mm. V razporeditvi je 6 izdelkov štirih različnih tipov. Imamo po 1 izdelek tipa 1, 2 in 3, ter 3 izdelke tipa 4. Največja globina rezov posamičnega gnezda je 5. Na sliki je najgloblje gnezdo označeno sivo. Najgloblji rez najglobljega gnezda je zunanji rez izdelka tipa 4. Izbrali smo tudi začetno in končno točko rezalnika. Skupna dolžina vseh rezov je 14007.5 mm.



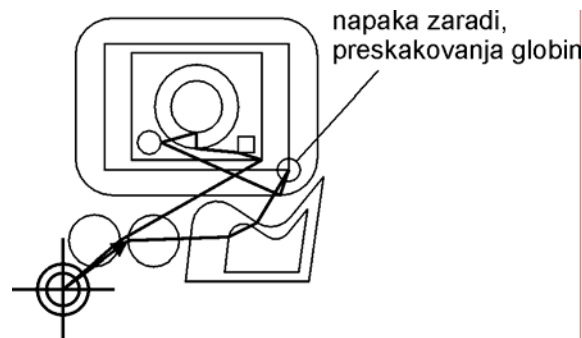
Slika 74: Primer 1

Sistem za programiranje stroja za razrez pločevine smo pognali 100-krat. Pri vseh zagonih so bili uporabljeni enaki parametri evolucije in sicer velikost populacije 1000, maksimalno število generacij 30, verjetnost reprodukcije 0.1, križanja 0.6 in permutacije 0.3. Uporabljena je bila turnirska izbira organizmov z velikostjo turnirja 7.

Povprečna dolžina najboljših poti vsakega zagona je znašala 16542.18 mm. Najboljša pot izmed 100-ih zagonov sistema je dolžine 16093.99 mm.

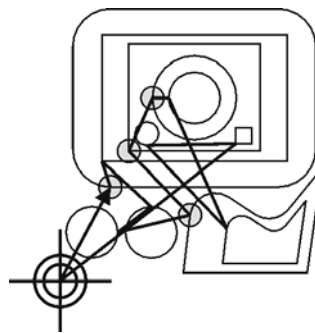
Kot zanimivost bomo prikazali enega izmed neodvisnih zagonov sistema.

Rezultat naključnega ustvarjanja NC-programov stroja za razrez pločevine je seveda slab. Najboljši organizem v začetni generaciji 0, ima dolžino 17180.6 mm. Dolžina podajalnih poti rezalnika glede na skupno dolžino znaša 18.46%. Položaj vboda reza, kjer se pojavijo napake zaradi preskakovanja globin rezov v gnezdu je obkrožen sivo.



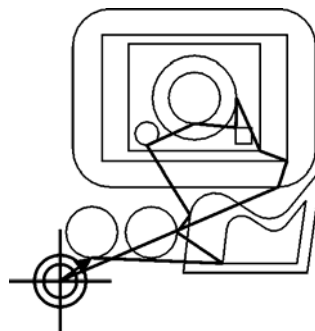
Slika 75: Najboljši NC-program v začetni generaciji 0

Za prikaz raznolikosti organizmov v začetni generaciji podajmo še najslabši NC-program v generaciji 0 (Slika 76). Ta program ima dolžino 4200.43 mm in povzroči kar $K=8$ preskokov globin. Število napačno izrezanih rezov je 4. Dolžina podajalnih poti rezalnika glede na skupno dolžino znaša 23.07%.



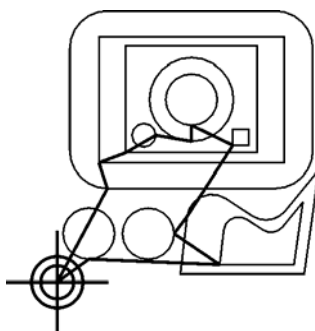
Slika 76: Najslabši NC-program v generaciji 0

V generaciji 2 se pojavi prvi NC-program brez preskokov globin (Slika 77). NC-program ima dolžino 16515.01 mm. Dolžina podajalnih poti glede na skupno dolžino poti znaša 15.18%.



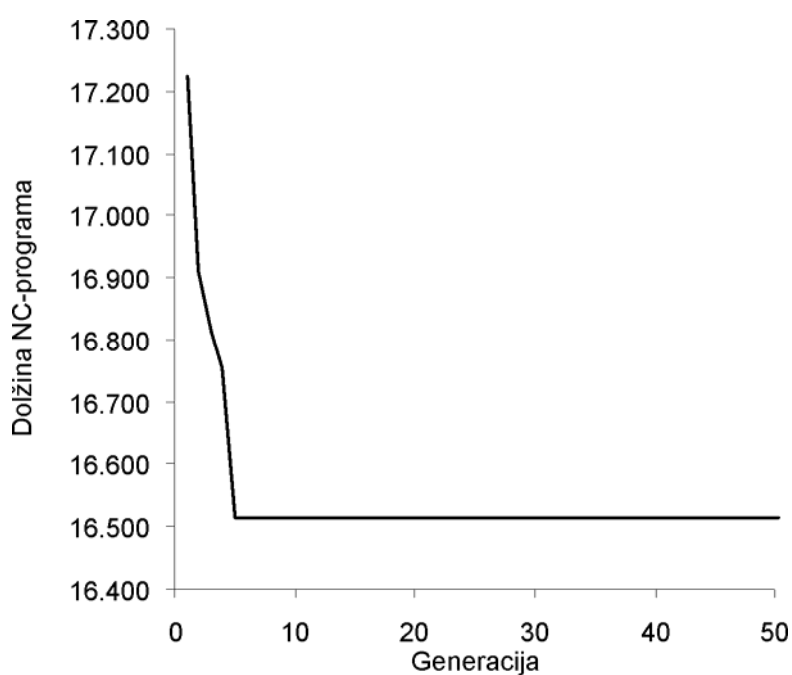
Slika 77: Najboljši NC-program v generaciji 2

Že generaciji 5 razvije evolucija najboljši NC-program civilizacije (zagona sistema) z dolžino 16093.99 mm. Dolžina podajalnih poti glede na skupno dolžino poti znaša 12.96%.



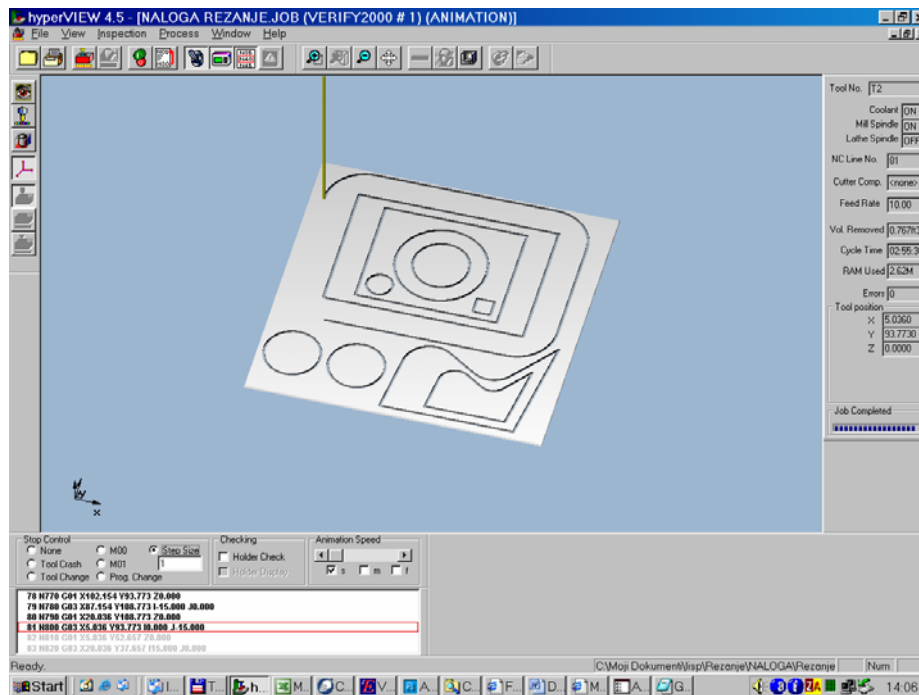
Slika 78: Najboljši NC-program civilizacije

Slika 79 prikazuje postopni razvoj najboljših rešitev skozi generacije glede na njihovo dolžino. Iz slike je razvidno, da se je najkrajši NC-program razvil že v generaciji 5 in se ohranil vse do zadnje generacije 50.



Slika 79: Dolžina najboljših NC-programov v posamični generaciji

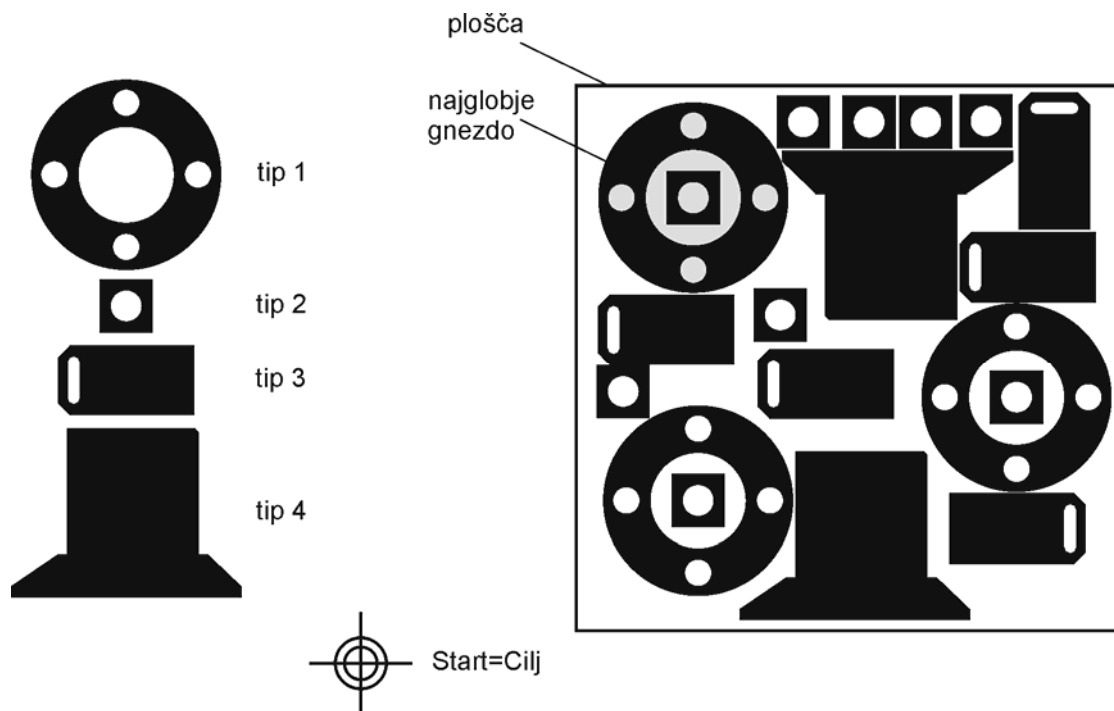
Dobljeni najboljši program smo preverili v programskem okolju HyperVIEW 4.5 (Slika 80).



Slika 80: Preverjanje najboljšega NC-programa civilizacije v programskem okolju HyperVIEW 4.5

7.3 Primer 2

Za primer 2 smo izbrali razporeditev z večjim številom izdelkov (Slika 81). Plošča je dimenzij 1600×1600 mm. V razporeditvi (Slika 81) je 19 izdelkov štirih različnih tipov. Vseh notranjih in zunanjih rezov je 48. Imamo po 3 izdelke tipa 1, 9 izdelkov tipa 2, 5 izdelkov tipa 3 ter 2 izdelka tipa 4. Največja globina rezov posamičnega gnezda je 4. Na sliki je najgloblje gnezdo označeno sivo. Najgloblji rez najglobljega gnezda je notranji rez izdelka tipa 4. Izbrali smo tudi začetno in končno točko rezalnika. Skupna dolžina vseh rezov je 32167.4 mm.

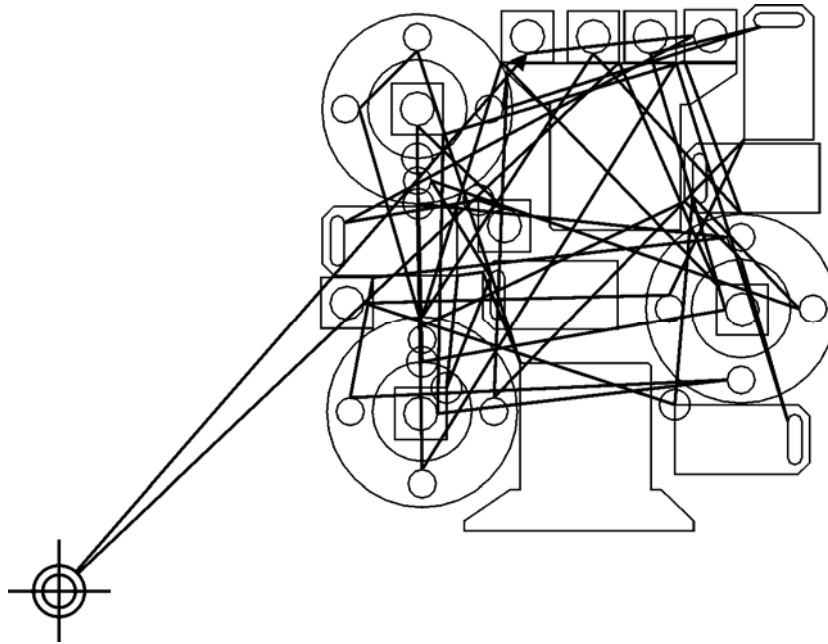


Slika 81: Primer 2

Sistem za programiranje stroja za razrez pločevine smo ponovno pognali 100-krat. Pri vseh zagonih so bili uporabljeni enaki parametri evolucije in sicer velikost populacije 1000, maksimalno število generacij 100, verjetnost reprodukcije 0.1, križanja 0.6 in permutacije 0.3. Uporabljena je bila turnirska izbira organizmov z velikostjo turnirja 7.

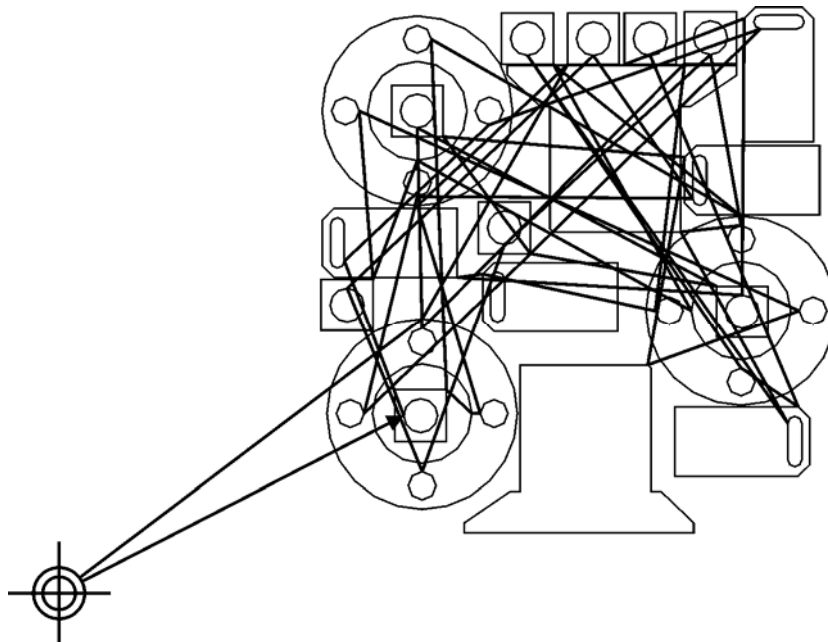
Tokrat bomo prikazali razvoj NC-programa enega neodvisnega zagona. Prikazali bomo le dolžino delovnih gibov NC-programa in ne celotne dolžine.

Rezultat naključnega ustvarjanja NC-programov stroja za razrez pločevine na začetku razvoje ene neodvisne civilizacije je seveda slab. Najboljši organizem v začetni generaciji 0 (Slika 82), ima dolžino podajalnih poti 36495.4 mm, kar predstavlja 113.45% rezalnih poti NC-programa in kar 68.02% celotne poti rezalnika. Položaj vboda reza, kjer se pojavijo napake zaradi preskakovanja globin rezov v gnezdu je obkrožen sivo. Število preskočenih globin $K=9$, število napačno izrezanih rezov pa je 6.



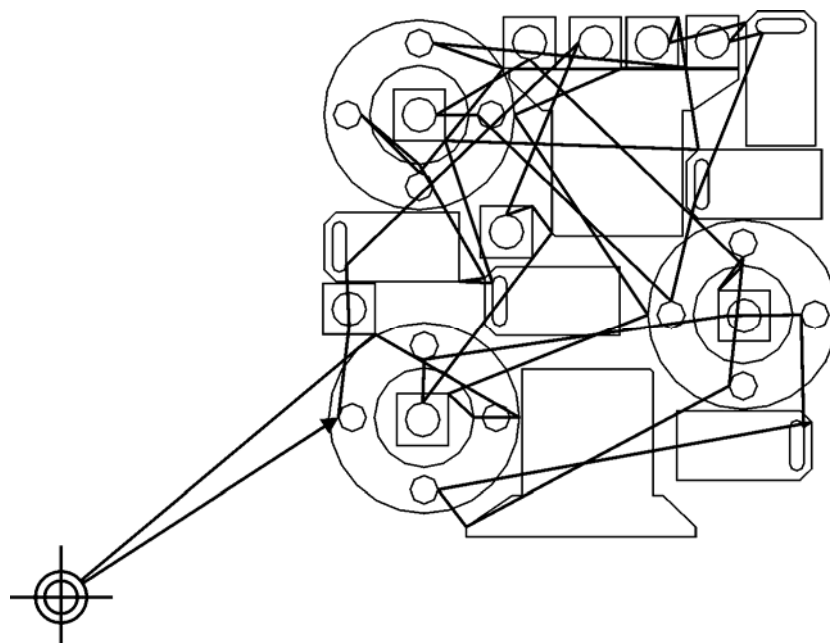
Slika 82: Najboljši NC-program v generaciji 0

V generaciji 7 se pojavi prvi NC-program brez preskokov globin (Slika 83). Dolžina podajalnih poti znaša 35179.8 mm, kar znaša 109.36% glede na rezalno pot in 52.23% glede na celotno pot rezalnika.



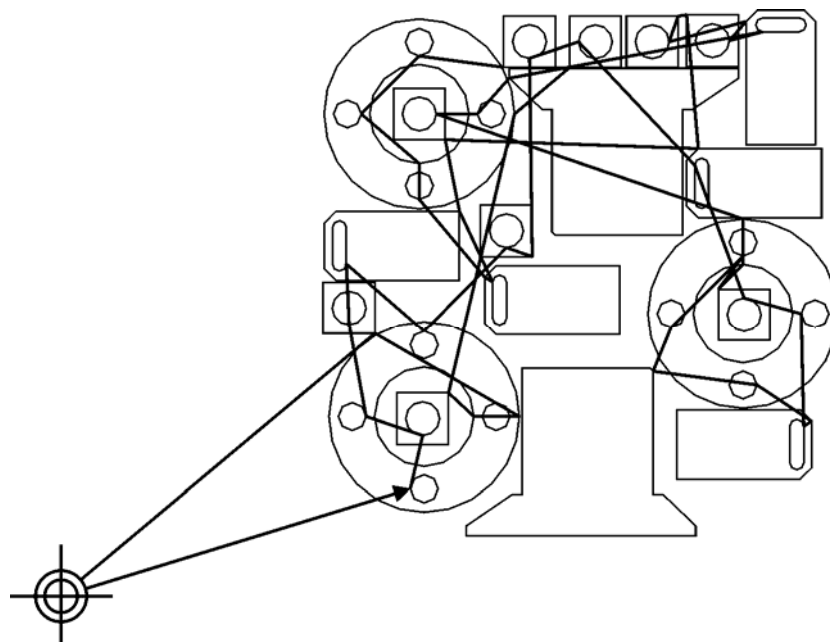
Slika 83: Najboljši NC-program v generaciji 7

Najboljši program generacije 20 (Slika 84) ima dolžino podajalnih poti 21489.7 mm, kar znaša 66.80% rezalne poti in 40.05% celotne poti rezalnika.



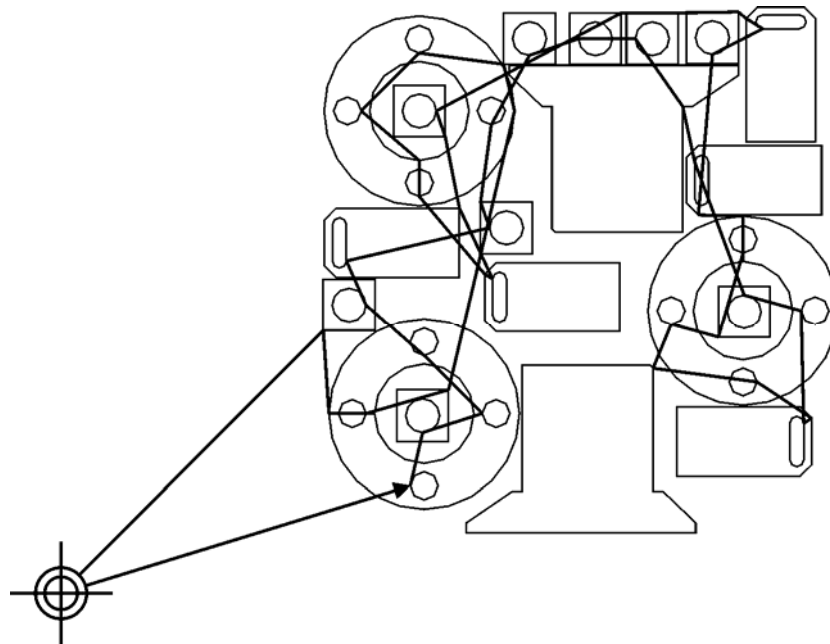
Slika 84: Najboljši NC-program v generaciji 20

Videz poti NC-programov razvitih v posamičnih generacijah se skozi evolucijo občutno spreminja. Poti so vedno manj prepletene. Občuten napredek v primerjavi z NC-programi v prej prikazanih generacijah 0, 7 in 20, je viden pri NC-programu, ki je nastal v generaciji 50 (Slika 85). Dolžina podajalnih poti je, 14885.5 mm kar znaša 46.28% glede na rezalno pot in 31.36% glede na celotno pot rezalnika.



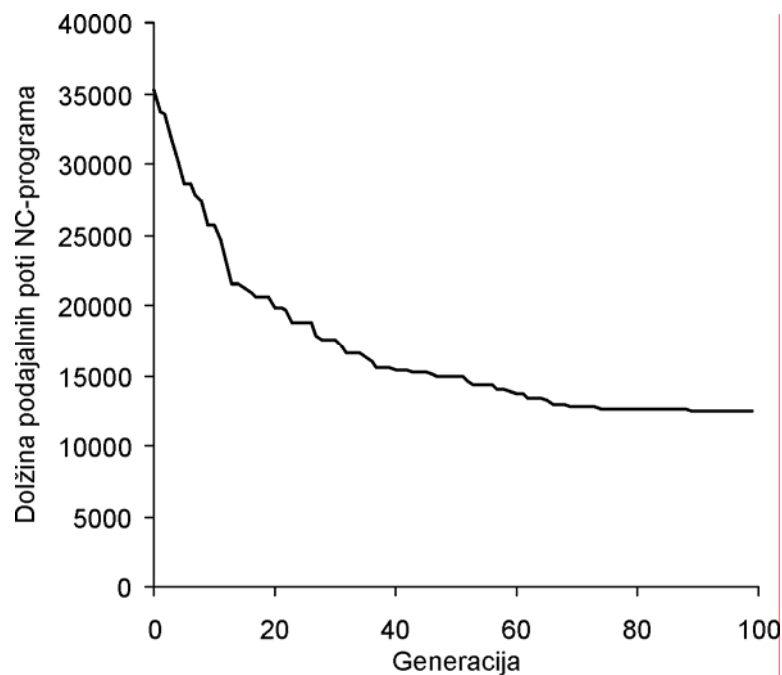
Slika 85: Najboljši NC-program v generaciji 50

V generaciji 95 je nastal najboljši NC-program civilizacije (neodvisnega zagona sistema) (Slika 86). Dolžina najkrajše poti podajalne poti je 12453.2 mm kar znaša 38.71% rezalne poti in 27.90% celotne poti rezalnika.



Slika 86: Najboljši NC-program v generaciji 95

Slika 87 prikazuje postopni razvoj najboljših NC-programov skozi generacije glede na njihovo dolžino podajalnih poti.



Slika 87: Dolžina podajalnih poti najboljših NC-programov v posamični generaciji

7.4 Zaključek

V tem poglavju smo prikazali koncept avtomatskega programiranja NC-stroja za razrez pločevine z metodo genetskih algoritmov. Predhodno opisana sistema za programiranje NC-stručnice in NC-frezalnega stroja sta s svojo diskretizirano zasnovo omogočila le predstavitev koncepta, ki se je praktično nadgradil šele pri zasnovi sistema za samodejno programiranje

NC-stroja za razrez pločevine. Pri izdelanem sistemu izdelke sestavljajo notranji in zunanji rezi. Rezi so poljubnih oblik, sestavljeni iz linij, polkrogov in krogov.

Izdelani inteligentni sistem je sposoben samodejno prepoznati vgnezenost izdelkov v razporeditvi, določiti vbode in zaporedja rezanja rezov, ki sestavljajo razporejene izdelke. Položaj vbodov se določi le na zato določenih mestih na rezu.

Z metodo genetskih algoritmov je sistem sposoben najti najkrajšo pot med posamičnimi rezi in izpisati NC-program. Sistem je možno z manjšimi uporabniku prirejenimi popravki povsem praktično uporabiti.

Prihodnje raziskave bi bilo smiselno usmeriti v zasnovo izpopolnjenega sistema za tri-dimenzionalno rezanje, s poljubnim določanjem položajev vbodov, s sposobnostjo zaznavanja kolizij in optimizacijo hitrosti ter pospeševanja pri rezanju.

8. ZAKLJUČEK

V nalogi je predstavljeno samodejno programiranje numerično krmiljenih obdelovalnih strojev z genetskim algoritmom. Naloga je praktično široko zastavljena in podprta s teoretičnimi izhodišči.

Jedro naloge predstavlja zamisel, da se s časom spreminjata prav tako živa in neživa narava. Kot so se skozi tisočletja spreminjali pripomočki, ki so lajšali človeku naporno in monotono delo, tako lahko spreminjanju podvržemo programe, ki krmilijo obdelovalni stroj. Pri tem je načeloma vseeno za kateri tip stroja gre. Program, ki obdelovalni stroj krmili ima pri tem vlogo povezovalca vhodnih in izhodnih virov informacij in materiala. Dejansko je potrebno poznati stanje na začetku-vhodno stanje in stanje na koncu-izhodno stanje; pogoj, da začetno stanje preide v končno pa predstavlja program. V splošnem lahko govorimo o surovcu, kot začetnemu stanju sistema in izdelku kot končnemu stanju sistema. Numerično krmiljeni obdelovalni stroj iz surovca po programu izdeluje izdelek. Ker bi iz surovca, brez programa za numerično krmiljeni obdelovalni stroj, težko izdelali izdelek je ta segment upravičil nastanek dela.

V svetu obstaja množica obstoječih bolj ali manj učinkovitih sistemov za samodejno ustvarjanje programov za numerično krmiljene stroje kot so stružnice, frezalni, merilni, vrtni stroji, roboti, manipulatorji, avtonomna vozila in podobno. V nalogi smo lastnosti numerično krmiljenih obdelovalnih strojev skušali postaviti na skupni imenovalec, z vpeljavo evolucijskega potenciala in jih programirati z enotnim postopkom kot je genetski algoritem. Evolucijski potencial omogoča ohranjanje in razvijanje lastnosti, ki omogoča preživetje v danem okolju. V proizvodnem okolju predstavlja evolucijski potencial množica nasprotujočih si zahtev kot so npr. varnost, poraba materiala, časa in energije ter navsezadnje največkrat brezbrizen odnos do okolja. Pomeni, da bo sistem za samodejno ustvarjanje programov ohranil in izboljševal program, ki se bo v danem proizvodnem okolju najbolje prilagodil.

Jedro sistema za samodejno ustvarjanje programov predstavlja genetski algoritem. Delovanje algoritma je idejno razmeroma preprosto. V začetku se ustvari naključna populacija možnih programov. Nad populacijo se izvajajo evolucijske operacije, ki so nam znane iz evolucije živih bitij: reprodukcija, križanje in mutacija. Najboljša rešitev – najuspešnejši organizem – najboljši program – program z največjim evolucijskim potencialom je tisti, ki najbolje izpolnjuje zadane pogoje uspešnosti. Programi nastajajo postopoma iz generacije v generacijo.

Izdelani so bili sledeči sistemi za samodejno ustvarjanje programov za krmiljenje:

- stružnice,
- frezalnega stroja in
- stroja za razrez pločevine.

Vsak sistem vsebuje tri podsisteme:

- obdelava in vnos vhodnih podatkov (CAD model surovca, izdelka, orodja in stroja),
- planiranje in optimiranje potovanja orodja s pomočjo genetskih algoritmov ter
- obdelava in preverjanje rezultatov.

Zaradi nazornosti so v nalogi najprej predstavljeni enostavnejši, idealizirani sistemi in primeri, ki postajajo, v skladu z idejo o nenehnem spreminjanju k boljšim, zahtevnejšim oblikam, zahtevnejši in praktično bolj uporabni. Zadnji sistem, sistem za samodejno ustvarjanje programov stroja za razrez pločevine predstavlja praktično aplikacijo, ki jo je možno, z manjšimi izboljšavami, postaviti ob bok obstoječim, komercialnim sistemom.

V delu smo pokazali, da lahko različne obdelovalne stroje programiramo na povsem enak način. Čedalje hitrejši razvoj numerično krmiljenih obdelovalnih strojev omogoča povsem praktično uporabo sodobnih nekonvencionalnih rešitev, ki se jih z obstoječimi klasičnimi postopki težko ali celo nemogoče dokopljemo. Čeprav izvirne, nam v nalogi pridobljene rešitve omogočajo, da jim bomo v prihodnosti namenili še precej pozornosti.

KRATEK OPIS SISTEMA ZA SAMODEJNO PROGRAMIRANJE NC STROJEV Z GENETSKIMI ALGORITMI

Sistem za samodejno programiranje NC strojev s pomočjo genetskih algoritmov smo v celoti programirali v jeziku autolisp vgrajenega v grafično okolje AutoCAD 2002, ki omogoča interaktivno delo z objekti na risbi. Najpomembnejši del sistema je genetski algoritem; jedro, ki ga lahko z zanemarljivimi spremembami prilagodimo vsakemu obdelovalnemu stroju. Jedro vsebuje približno 300 učinkovitih programskih vrstic izvorne kode, razporejene v približno 12 večjih enot (glede na predstavitev surovca in izdelka). Jedro smo vgradili v tri neodvisne sisteme za samodejno programiranje:

- sistem za programiranje NC stružnice,
- sistem za programiranje NC frezalnega stroja in
- sistem za programiranje NC stroja za razrez pločevine.

Sistema za programiranje NC stružnice in NC frezalnega stroja sta sestavljena vsak iz 35 (12 izmed 35-ih enot predstavlja jedro) večjih enot, sistem za programiranje NC stroja za razrez pločevine pa je sestavljen iz 41 enot. Vsak posamičen sistem v povprečju vsebuje 1200 lastnoročno zapisanih učinkovitih programskih vrstic. Za primerjavo: celotna disertacija je zapisana v malo več kot 3400 vrsticah.

Vsaka programska enota je bila skrbno testirana in nemalokrat programirana v večih različicah. Ne moremo prezreti, da je vsak, še tako majhen prihranek časa, zelo pomemben. Nekateri deli programa se med evolucijo civilizacije izvedejo več 10.000- ali 100.000-krat.

Vsak sistem za samodejno programiranje sestavljajo 4 vsebinski sklopi:

- vnos geometrijskih, tehnoloških in podatkov za izpis rezultatov ter parametrov za nadzor evolucije,
- ustvarjanje NC programa s pomočjo simulirane evolucije in
- izpis ali izris rezultatov.

V podrobnejši vpogled posamičnih sklopov se zaradi preobsežnosti obravnavane problematike in nazornosti ne bomo spuščali. V nadaljevanju bomo podali le osnovne funkcije jedra, ki so gonilna sila simulirane evolucije.

V naslednjem podpoglavju bomo na kratko predstavili programski jezik LISP, katerega različica, autolisp je vgrajena v grafično okolje AutoCAD.

Programski jezik LISP

Za programiranje lahko uporabimo domala katerikoli programski jezik (npr. LISP, FORTRAN, BASIC, PASCAL, C, FORTH, ...), ki je sposoben ovrednotiti hierarhične računalniške programe. Zaradi nekaterih prednosti jezika LISP, ki jih bomo podrobneje osvetlili v nadaljevanju, so uporabljeni sistemi za samodejno programiranje NC strojev v celoti zgrajeni v tem jeziku. Navkljub tej odločitvi, pri programiranju nikakor ne prevladuje LISP in podobni jeziki, pač pa je povsem splošna metoda, ki se lahko z malo več ali manj programerskega napora izvede s pomočjo poljubnega računalniškega jezika [17].

LISP je eden izmed najstarejših programskih jezikov, saj ga je razvil John McCarthy za potrebe razvoja umetne inteligence že v sredini petdesetih let [65]. LISP je jezik, ki je namenjen za obdelavo seznamov (LISP = LISt Processing). Ima mnogo narečij, kot so MACLISP, INTERLISP, ZETALISP in COMMON LISP. LISP se lahko uporablja kot tolmač (interpreter) ali kot prevajalnik (translator).

Osnovna in edina struktura v LISP-u je seznam, ki je sestavljen iz členov. Členi so najmanjši elementi v LISP-u. Lahko so števila, nizi, sezname, funkcije ali katerikoli drugi dovoljeni podatkovni tipi. Nekateri členi imajo v LISP-u poseben pomen; najbolj pomembna sta logični konstanti `T` (True) in `NIL` (False). Celotna LISP-ova sintaksa in semantika je implementacija matematične teorije rekurzivnih funkcij, kar preprosto pomeni uporaba vpoklicanih (vgrajenih ali na novo zasnovanih) funkcij. Seznam lahko definiramo na rekurziven način takole [18][79]:

- Prazen seznam je `NIL`; za prazen seznam se uporablja tudi oznaka `» () «`.
- Če so $\check{c}_1, \check{c}_2, \dots, \check{c}_n$ členi ali sezname, potem je tudi $(\check{c}_1 \check{c}_2 \dots \check{c}_n)$ seznam.

Primeri seznamov so: `(+ 1 2)`, `(cos a)` in `(* a (+ s 1.34) c)`.

Med programi in podatki (oboje so sezname) v LISP-u ni sintaktične razlike, zato jih LISP obravnava na enak način. Program v LISP-u katerikoli del lastne kode razume kot podatke, ki ga po potrebi spremeni in spremenjenega tudi izvrši. V tej preprosti strukturi temeljijo nekatere izredne lastnosti, s katerimi se LISP ponaša že vse od nastanka.

Operatorjev v LISP-u ni, zato so vse osnovne računske operacije (podobno kot vse ostale funkcije) le funkcije z ustreznim številom argumentov.

Seštevanje dveh števil izvedemo z `(+ 1 2)`. V zgornjem seznamu sta funkciji za seštevanje `+`, pridružena dva argumenta, konstanti `1` in `2`. Prvi element v seznamu obravnava LISP vedno kot funkcijo, ostale elemente pa kot argumente. Značilnost LISP-a je, da so izrazi podani v prefiksni obliki. Najprej je podana funkcija, nato pa ustrezni argumenti. Za primerjavo povejmo, da je v FORTRAN-u, ki uporablja infiksni zapis, oblika zgornjega izraza enaka `1+2`, v jeziku FORTH, ki uporablja postfiksni zapis, pa `1 2 +`.

Izrazi v LISP-u so ponavadi sestavljeni iz več funkcij. Naslednji izraz, ki ga lahko obravnavamo tudi kot program, je sestavljen iz dveh funkcij: `(* (+ 1 2) 4)`. Najprej se ovrednoti ugnezen izraz `(+ 1 2)`, ki vrne vrednost `3`, nato pa še celotni izraz, ki vrne vrednost `12`.

Naslednja dobrota LISP-a je, da v programih ni potrebno navesti tipe spremenljivk in funkcij. Še celo več – spremenljivkam z istim imenom se lahko med izvajanjem programa prireja

podatke različnih tipov. Spremenljivke v LISP-u so torej zelo prilagodljive, saj dinamično zavzamejo takšen tip kot je tip podatka. V LISP-u ima poseben pomen funkcija `QUOTE`, ki pusti izraz neovrednoten. Tako z `(QUOTE (+ 1 2))`, ne ovrednotimo argumenta funkcije `QUOTE`.

Nasprotni učinek od `QUOTE` ima funkcija `EVAL`. Funkcija `EVAL` ovrednoti izraze, ki smo jih prej predznačili s funkcijo `QUOTE`. S kančkom domišljije lahko zaslutimo zanimive možnosti, saj je argument funkcije `EVAL` lahko katerikoli izraz. Če spremenljivki `A` priredimo izraz (t.j. vrednost) `(QUOTE (+ 1 2))`, se ta ovrednoti šele z vnosom `(EVAL A)`. Rezultat ovrednotenja je konstanta 3.

Jedro – genetski algoritem

Osnova za uspešno delovanje genetskega algoritma je generator naključnih števil, ki omogoča ustvarjanje čim bolj raznovrstnih rešitev. Po skrbni analizi smo se odločili za Park-Millerjev generator naključnih števil. V autolispu zgleda takole:

```
(defun generator (spodaj zgoraj / a b)
  (setq a 2147483647.0
        b 16807.0
        seme (rem (* seme b) a))
  (- 1.0 (/ seme a)))
); defun
```

; a in b sta lokalni
; spremenljivki/konstanti
; določimo velikost konstant
; a in b
; seme je globalna
; spremenljivka, kar pomeni,
; da po zagonu funkcije
; generator ostane znana
; funkcija rem vrne ostanek
; pri deljenju dveh števil
; funkcija generator vrne
; vrednost med 0 in 1

Če spremenljivki `seme` določimo vrednost 1.0 in funkcijo `generator` poženemo 10.000-krat, je vrednost spremenljivke `seme` po zadnjem zagonu enaka 1.04362E+9. Generator smo testirali tako, da smo 100.000.000-krat zagnali funkcijo `generator` (začetna vrednost semena je bila 1.0). 50.004.241-krat smo dobili vrednost manjšo od 0.5 in 49.995.759-krat smo dobili vrednost večjo od 0.5. ob predpostavki, da je idealno razmerje števil manjših in večjih od nič enako 1.0, se dobljeni izidi od idealnih razlikujejo za 0.00848%.

Funkcijo `generator` lahko brez težav predelamo v generator naključnih realnih števil:

```
(defun gen-realnih-št (spodaj zgoraj / a b)
  (setq a 2147483647.0
        b 16807.0
        seme (rem (* seme b) a))
  (+ spodaj (* (- zgoraj spodaj) (- 1.0 (/ seme t))))
); defun
```

; za zagon funkcije gen-
; realnih-št potrebujemo
; spodnjo in zgornjo mejo
; intervala
; funkcija gen-realnih-št
; vrne vrednost na intervalu
; med spodaj in zgoraj

Z generatorjem lahko ustvarjamo skoraj naključne organizme, možne rešitve sistema. Skoraj naključne zato, ker števila niso povsem naključna, saj jih izračunamo po natančno določenih

pravilih. To ima poleg slabosti tudi dobro plat, da lahko naključna števila do potankosti poustvarimo.

V nadaljevanju bo predstavljena funkcija, ki ustvarja rešitve za problem trgovskega potnika, predstavljenem v podglavju 3.3. Določiti je potrebno nabor mest:

```
(setq mesta `(CELJE MARIBOR LJUBLJANA KRANJ KOPER))
```

Funkcijo, ki ustvarja naključne možne poti trgovskega potnika, lahko zapišemo takole:

```
(defun ustvarjanje (število-rešitev rešitve) ; argument je število rešitev,
; ki jih želimo imeti
(repeat število-rešitev
  (setq rešitve
    (cons
      (ustvari-rešitev) ; vsakič ustvari rešitev in
      rešitve ; jo doda v seznam rešitve
    ); cons
  ); setq
); repeat
rešitve ; vrni rešitve
); defun
```

Pri tem zapišimo še funkcijo ustvari-rešitev:

```
(defun ustvari-rešitev (/ rešitev)
(repeat (length mesta) ; ponavljaš tolikokrat, kot je
; število mest
  (setq rešitev ; rešitev nastaja z dodajanjem
    (cons ; naključno izbranih mest
      (nth ; funkcija nth izbere člen iz
        ; seznamu
        (fix (generator-reálnih-št 0 (length mesta)))
        mesta ; funkcija fix vrne celo
        ; število
      ) ; nth
      rešitev
    ); cons
  ); setq
); repeat
rešitev ; vrni rešitev
); defun
```

Želimo imeti populacijo petih organizmov. Semenu določimo vrednost 1.0. Pri enkratnem zagonu funkcije `ustvarjanje` (ukaz v `lispu` (`ustvarjanje 5`)) dobimo populacijo (seznam) petih organizmov:

```
(
  (CELJE CELJE LJUBLJANA MARIBOR LJUBLJANA)
  (KOPER KRANJ KOPER MARIBOR LJUBLJANA)
  (KOPER KOPER CELJE LJUBLJANA KRANJ)
  (CELJE MARIBOR MARIBOR KOPER KRANJ)
  (LJUBLJANA LJUBLJANA MARIBOR KOPER KOPER)
)
```

Takoj je moč opaziti, da vsi osebki vsebujejo podvojena mesta (gene). Potrebno bi bilo ugotoviti, katerih genov organizem ne vsebuje in zamenjati s podvojenimi geni. Zaradi nazornosti funkcije `poprava` ne bomo prikazali. Pri popravi pregledamo organizem in ugotovimo, katera mesta so podvojena in katerih mest ni v organizmu. Ko v organizmu drugič

najdemo podvojeno mesto, ga nadomestimo s prvim manjkajočim mestom iz nabora mest mesta. Po popravi organizmi zglejajo takole:

```
(
  (CELJE KRANJ LJUBLJANA MARIBOR KOPER)
  (KOPER KRANJ CELJE MARIBOR LJUBLJANA)
  (KOPER MARIBOR CELJE LJUBLJANA KRANJ)
  (CELJE MARIBOR LJUBLJANA KOPER KRANJ)
  (LJUBLJANA CELJE MARIBOR KOPER KRANJ)
)
```

Zamenjani geni so označeni odebeljeno.

Organizme lahko spreminjamo s pomočjo genetske operacije permutacije in križanja. Pri permutaciji se v organizmu zamenjata dve naključno izbrani mesti. Pri križanju potomci vsebujejo genetski material obeh starševskih organizmov. Operacija permutacija je prikazana na naslednjem zgledu. Pri funkciji permutacija se naključno izbereta mesti, ki jih bomo v organizmu zamenjali. Permutant (organizem po permutaciji) nastane z dodajanjem genov prvotnega organizma. Na prvo, naključno izbrano mesto vstavimo drugo, naključno izbrano mesto in obratno.

```
(defun permutacija (organizem / i seznam mesto-1 mesto-2 permutant)
  ; argument je seznam mest,
  ; i, seznam, mesto-1, mesto-2
  ; in permutant so lokalne
  ; spremenljivke
  (setq mesto-1
    (fix (gen-realnih-št 0.0 (length organizem))) ; izberemo mesto-1 in mesto-2
  (setq mesto-2
    (fix (gen-realnih-št 0.0 (length organizem)))
  (setq i 0) ; števec i dobi vrednost 0
  (repeat (length organizem)
    (if
      ; če je i enak mestu-1 vstavi
      ; mesto-2
      (= i mesto-1)
      (setq permutant (cons (nth mesto-2 organizem) permutant))
      ; če je i enak mestu-2 vstavi
      ; mesto-1, če ne vstavi i-ti
      ; člen organizma
      (= i mesto-2)
      (setq permutant (cons (nth mesto-1 seznam) permutant))
      (setq permutant (cons (nth i seznam) permutant))
    ); if
  ); if
  (setq i (1+ i)) ; povečamo števec zanke za 1
); repeat
(reverse permutant) ; obrni permutanta
); defun
```

Semenu določimo vrednost 1.0. Pri enkratnem zagonu funkcije permutacija, pri čemer želimo spremeniti (permutirati) organizem (CELJE **KRANJ** LJUBLJANA MARIBOR **KOPER**) (ukaz v lispu (permutacija '(CELJE **KRANJ** LJUBLJANA MARIBOR **KOPER**))) dobimo permutant:

```
(CELJE KRANJ LJUBLJANA MARIBOR KOPER).
```

Organizem ni permutiral; izbrani mesti za zamenjavo sta bili **KOPER** in **KOPER**. Funkcijo permutacija zaženemo drugič. Dobimo permutant:

```
(CELJE LJUBLJANA KRANJ MARIBOR KOPER).
```

Izbrani mesti za zamenjavo sta bili KRANJ in LJUBLJANA in sta v zgornjem organizmu natisnjena odebeljeno.

Podobno kot funkcija `permutacija`, je zasnovana funkcija `križanje`. Naključno se izbereta mesti križanja v vsakem starševskem organizmu. Prvi potomec nastane z dodajanjem vseh mest, pred naključno izbranim prvim mestom iz prvega starševskega organizma, in nato še z dodajanjem vseh mest za naključno izbranim drugim mestom drugega starševskega organizma. Drugi potomec nastane podobno. Omeniti je potrebno, da potomca lahko vsebujeta podvojena mesta, ali je celo brez vseh mest, zato je potrebno, da se nad potomcema izvede operacija poprave; zamenjajo ali izbrišejo se podvojena mesta in dodajo mesta iz nabora mest.

```
(defun križanje (starš-1 starš-2 / i potomec-1 potomec-2 mesto-1 mesto-2)
  (setq mesto-1 (fix (gen 0.0 (length starš-1)))           ; naključno izberi
        mesto-2 (fix (gen 0.0 (length starš-2)))           ; mesto-1 in mesto-2
        i 0
  ); setq
  (repeat (1+ mesto-1)                                     ; potomec-1 nastane z
    (setq                                         ; dodajanjem starša-1
      potomec-1 (cons (nth i starš-1) potomec-1)
      i (1+ i)
    ); setq
  ); repeat
  (setq i mesto-2)
  (repeat (- (length starš-2) mesto-2)                 ; in nato še z
    (setq                                         ; dodajanjem starša-2
      potomec-1 (cons (nth i starš-2) potomec-1)
      i (1+ i)
    ); setq
  ); repeat
  (setq i 0)
  (repeat (1+ mesto-2)                                   ; potomec 2 nastane z
    (setq                                         ; dodajanjem starša-2
      potomec-2 (cons (nth i starš-2) potomec-2)
      i (1+ i)
    ); setq
  ); repeat
  (setq i mesto-1)
  (repeat (- (length starš-1) mesto-1)                 ; in nato še z
    (setq                                         ; dodajanjem starša-1
      potomec-2 (cons (nth i starš-1) potomec-2)
      i (1+ i)
    ); setq
  ); repeat
  (list (reverse potomec-1) (reverse potomec-2))       ; funkcija vrne pravilno
); defun                                                ; obrnjena potomca
```

Če izvedemo operacijo `križanje` nad oseboma (CELJE KRANJ LJUBLJANA MARIBOR KOPER) in (KOPER KRANJ CELJE MARIBOR LJUBLJANA), pri čemer ima spremenljivka `seme` začetno vrednost 2, dobimo rezultat dva seznama mest: (CELJE KRANJ LJUBLJANA MARIBOR KOPER MARIBOR LJUBLJANA) in (KOPER KRANJ CELJE MARIBOR KOPER). Prvemu potomcu z operacijo poprave odstranimo zadnje, podvojeno mesto Ljubljano, drugemu pa zadnje, podvojeno mesto Koper zamenjamo z Ljubljano. Potomca po operaciji poprave izgledata enako kot starševska organizma. V tem primeru z operacijo križanja ni prišlo do sprememb genetskega materiala.

Po spreminjanju organizmov je populacijo možnih rešitev treba ovrednotiti. Uspešnost organizmov v primeru trgovskega potnika predstavlja najkrajša prepotovana razdalja, ki jo

potrebuje trgovski potnik, ko obiskuje vsa mesta iz nabora mest. Funkcija za ovrednotenje preprosto zglada takole:

```
(defun ovrednoti (mesta / i dolžina-poti)           ; argument je seznam mest
  (setq i 0)
  (setq dolžina-poti (razdalja (nth i mesta) START)) ; izračunaj dolžino poti
  (repeat (1- (length mesta))                     ; med izhodiščem in prvim
    (setq dolžina-poti                             ; mestom v seznamu
      (+ dolžina-poti                               ; seštevaj razdalje med mesti
        (razdalja
          (nth i mesta)
          (nth (1+ i) mesta)
        )
      )
    ); setq
  ); repeat
  (setq i (1+ i))
); repeat
(+ dolžina-poti (razdalja START (last mesta)))    ; vrni vsoto vseh razdalj in
); defun                                           ; razdalje med zadnjim mestom
                                                    ; v seznamu in izhodiščem
```

Funkcija `razdalja` meri razdaljo med dvema mestoma.

LITERATURA

- [1] Alauddin, H., El Baradie, M. A., Hashmi, M. S. J., (1995), Computer-aided analysis of a surface-roughness model for end milling, *Journal of Materials Processing Technology*, 55(2), 123-127
- [2] Alauddin, H., El Baradie, M. A., Hashmi, M. S. J., (1996), Optimisation of surface finish in end milling Inconel 718, *Journal of Materials Processing Technology*, 56(1-4), 54-65
- [3] Alope, R., Girish, V., Scrutton, R.F., Mollian, P.A., (1997), A model for prediction of dimensional tolerances of laser cut holes in mild steel thin plates, *International Journal of Machine Tools and Manufacture*, 37(8), 1069-1078
- [4] Altintas Yusuf, (2000), *Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC design*, Cambridge University Press, Cambridge, England
- [5] *APT part programming*, (1967), New York, McGraw-Hill
- [6] Bajec, A., (1980), *Slovar slovenskega knjižnega jezika*, Državna založba Slovenije, Ljubljana
- [7] Balasubramaniam, M., Sarma, S. E., Marciniak, K., Collision-free finishing toolpaths from visibility data, *Computer-Aided Design*, In Press, Uncorrected Proof, Available online 14 February 2002
- [8] Balič, J., (1986), *Model univerzalnog kompleksnog postprocesora u sistemu interaktivnog računarskog programiranja numerički upravljanih strojeva za bušenje i glodanje*, Fakultet strojarstva i brodogradnje, Zagreb
- [9] Balič, J., (1999), *Contribution to Integrated Manufacturing*, (DAAAM publishing series, Manufacturing technology), DAAAM International, Vienna
- [10] Balič, J., (2000), *Prilagodljivi obdelovalni sistemi*, Fakulteta za strojništvo, Maribor
- [11] Balič, J., (2001), *Računalniška integracija proizvodnje*, Fakulteta za strojništvo, Maribor
- [12] Balič, J., (2003), *Intelligent manufacturing systems: programming and control*, Fakulteta za strojništvo, Maribor, Department of Production Engineering and Management, Crete

- [13] Balič, J., (2003), *Proizvodne tehnologije*, Fakulteta za strojništvo, Maribor
- [14] Balič, J., Korošec, M., Intelligent tool path generation for milling of free surfaces using neural networks, *International Journal of Machine Tools and Manufacture*, In Press, Uncorrected Proof, Available online 31 May 2002
- [15] Balič, J., Živec, Z., Čuš, F., (1995), Model of Universal Manufacturing Interface in CIM, *Journal of Materials Processing Technology*, 52(1), 102-114
- [16] Boole, G., (1954), *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*, Dover
- [17] Bratko, I., (1990), *Prolog programming for artificial intelligence – 2nd ed.*, Addison-Wesley Publishing Company, Wokingham, England
- [18] Brezočnik, M., (1998), *Modeliranje tehnoloških sistemov z uporabo genetskih metod*, doktorska disertacija Fakulteta za strojništvo, Maribor
- [19] Brezočnik, M., (2000), *Uporaba genetskega programiranja v inteligentnih proizvodnih sistemih*, Fakulteta za strojništvo, Maribor
- [20] Chao-ton, Mu-Chen, C., (1999), Computer-aided optimization of multi-pass turning operations for continuous forms on CNC lathes, *IIE Transactions*, 31, 583-596
- [21] Chen, N., (1990), *Feature-based numerical control program generation from computer aided design models*, University of California, Los Angeles, California
- [22] Cheng, R., Gen, M., (1993), On film copy deliverer problem, ed. Zheng, W., *Proceedings of the Second International Conference on Systems Science and System Engineering*, Beijing
- [23] Cheng, R., Gen, M., (1994), Evolution program for resource constrained project scheduling problem, *Proceedings of ICEC'94*
- [24] Cheng, R., Gen, M., Tozawa, T., (1995), Vehicle Routing Problem with Fuzzy Due-time, *Using Genetic Algorithms. Japanese Journal of Fuzzy Theory and Systems* 7, 665-679
- [25] Chiou, C. J., Lee, Y. S., (2002), A machining potential field approach to tool path generation for multi-axis sculptured surface machining, *Computer-Aided Design* 34 (5), 357-371
- [26] Crandell, T. M., (2003), *CNC Machining and Programming: An Introduction*, Industrial Press Inc., New York
- [27] Cuvier, G., (1995), *Historical portrait of the progress of ichthyology, from its origins to our own time*, The Hopkins university press, Baltimore, Maryland, London
- [28] Darwin, C., (1859), *On the origin of species by means of natural selection or the preservation of favored races in the struggle for life*, Murray, London

- [29] Ding, X. M., Fuh, J. Y. H., Lee, K. S., (2001) Interference detection for 3-axis mold machining, *Computer-Aided Design*, 33(8), 561-569
- [30] Dragomatz, D., Mann, S., (1997), A classified bibliography of literature on NC milling path generation, *Computer-Aided Design*, 29(3), 239-247
- [31] Drstvenšek, I., (1998), Model tehnološke baze obdelovalnih operacij v postopkih optimiranja rezalnih pogojev z uporabo genetskih algoritmov, *Doktorska disertacija*, Fakulteta za strojništvo, Maribor
- [32] Francis Reintjes, J., Marcus, R. S., (1968), *Computer-aided processing of the news*, Status Report ESL-SR-348, Cambridge, Massachusetts
- [33] Freeman, J., Grefenstette, J., Gopal, R., Rosmaita, B., & Van Gucht D., (1985), Genetic algorithms for the traveling salesman problem, J. J. Grefenstette (ed), *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, 160-168
- [34] Fuh, K. H., Wu, C. F., (1995), A proposed statistical model for surface quality prediction in end-milling of Al alloy, *International Journal of Machine Tools and Manufacture*, 35(8), 1187-1200
- [35] Garey, M.R., Johnson, D.S, (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York
- [36] Gates, W., Mefferd, W., *Automated laser tool*, Patent Number: US3736402, 1973
- [37] Gen, M., Cheng, R., (1997), *Genetic Algorithms And Engineering Design*, J. Wiley & Sons, Inc., New York
- [38] Goldberg, D. E., (1989), *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley
- [39] Goldberg, D., Lingle, R., (1985), Alleles, loci and the traveling salesman problem, *proc. 1st Conference on Genetic Algorithms*, 154-159
- [40] Grefenstette, J., Gopal, R., Rosmaita, B., and van Gucht, D., (1985), Genetic algorithms for the traveling salesman problem, *Proceedings of an International Conference on Genetic Algorithms and their Applications*
- [41] Hansen, A. (1989). *Tool positioning and path generation algorithms for computer-aided manufacturing*, University of southern California, Los Angeles, California
- [42] Hatna, A., Grieve, R. J., Broomhead, P., (1998), Automatic CNC milling of pockets: geometric and technological issues, *Computer Integrated Manufacturing Systems*, 11(4) 309-330
- [43] Hawking, S. W., (2002), *Vesolje v orehovi lupini*, Učila International, Tržič

- [44] Heinrich, A., (2001), The recent history of the machine tool industry and the effects of technological change, Munich
- [45] Hodolič, J., (1979), Automatizacija projektovanja optimalne putanje alata kod numerički upravljanih mašina alatki za obradu struganjem, Novi Sad, Yugoslavia
- [46] Holland, J., (1975), Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor
- [47] Jackson, S. D., Mittal, O. R., (1993), Automatic generation of 2-axis laser-cutter NC machine program and path planning from CAD, Computers in Industry, 21(2), 223
- [48] Jerman, I., (1997), Evolucija s teoretično biologijo in temelji molekularne evolucije, Študentska organizacija Univerze, Ljubljana
- [49] Katalinić, B., (1990), Industrieroboter und Flexible Fertigungssysteme für Drehteile, VDI Verlag, Düsseldorf
- [50] Kim, G. M., Cho, P. J., Chu, C. N., (2000), Cutting force prediction of sculptured surface ball-end milling using Z-map, International Journal of Machine Tools and Manufacture, 40(2), 277-291
- [51] Kononenko, I., (1997), Strojno učenje, Fakulteta za računalništvo in informatiko, Ljubljana
- [52] Kovačič, M., Balič, J., (2003), Evolutionary programming of a CNC cutting machine. Int. j. adv. manuf. technol., 22(1/2), 118-124
- [53] Koza, J. R., (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, Massachusetts
- [54] Koza, J. R., (1994), Genetic programming II: automatic discovery of reusable programs, The MIT Press, Cambridge, London
- [55] Koza, J. R., (1999), Genetic programming III: darwinian invention and problem solving, Morgan Kaufmann, San Francisco
- [56] Koza, J. R., (2003), Genetic programming IV: routine human-competitive machine intelligence, Kluwer Academic Publishers, Norwell, Dordrecht
- [57] Kraut, B., (1987), Strojniški priročnik, Tehnička knjiga, Zagreb
- [58] Krušič, M., (1982), Stroji: Velika ilustrirana enciklopedija, Mladinska knjiga, Ljubljana
- [59] Kusiak, A., (1990), Intelligent Manufacturing Systems, Prentice-Hall, Inc., New Jersey
- [60] Kyung, S. P., Soung, H. K., (1998), Artificial intelligence approaches to determination of CNC machining parameters in manufacturing: a review, Artificial Intelligence in Engineering 12(1-2), 127-134

- [61] Lamarck, J.B., (1815), *Zoological Philosophy: An Exposition with Regard to the Natural History of Animals*, University of Chicago Press, Chicago
- [62] Licari, R., Lo Valvo, E. Piacentini, M., (2001), Part program automatic check for three axis CNC machines, *Journal of Materials Processing Technology*, 109(3), 290-293
- [63] Lin, Y., (1999), An adaptive tool path generation algorithm for precision surface machining, *Computer-aided design*, 4, 237-247
- [64] Makhanov, S. S., Batanov, D., Bohez, E., Sonthipaumpoon, K., Anotaipaiboon, W., Tabucanon, M., (2002), On the tool-path optimization of a milling robot, *Computers & Industrial Engineering*, 43(3), 455-472
- [65] McCarthy, J., (1960), *The Lisp Programmer's Manual*, M.I.T, Computation Center
- [66] Michalewicz, Z., (1994), *Genetic Algorithm + Data Structure = Evolutional Programs*, 2nd ed., Springer-Verlag, New York
- [67] Mitchell, T. M., (1997), *Machine Learning*, The McGraw-Hill Companies, Inc., New York
- [68] Oliver, I., Smith, D. in Holland, J., (1988), A study of permutation crossover operators on the traveling salesman problem, *proc. 2nd International Conference on Genetic Algorithms*, pp. 224-230
- [69] Polajnar, A., (1997), *Proizvodni management*, Fakulteta za strojništvo, Maribor
- [70] Polajnar, A., (1998), *Priprava proizvodnje*, Fakulteta za strojništvo, Maribor
- [71] Polajnar, A., Buchmeister, B., Leber, M., (2001), *Proizvodni menedžment*, Fakulteta za strojništvo, Maribor
- [72] Raj, H. K., Sharma, S. R., Srivastava, S., Patvardhan, C., (2000), Modeling of manufacturing processes with ANNs for intelligent manufacturing, *International Journal of Machine Tools and Manufacture*, 40(6), 851-868
- [73] Ralston, A., (2000), *Encyclopedia of Computer Science*, Nature, London
- [74] Rechenberg, I., (1974), *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog Verlag, Stuttgart
- [75] Ross, D. T., (1978), *Origins of the APT Language for Automatically Programmed Tools*, *ACM SIGPLAN Notices*, 13(8)
- [76] Ross, D. T., Ward, J. E., (1968), *Investigations in Computer-Aided Design for Numerically Controlled Manufacturing Processes*, Final Technical Report, Cambridge, Massachusetts
- [77] Stušek, P., (2001), *Biologija človeka*, 1. izd., 1. natis, DZS, Ljubljana

- [78] Syswerda, G., (1989), Uniform crossover in genetic algorithms, Schaffer, 2-9
- [79] Špiler, J., (1992), Autocad 12 Basic, Ljubljana
- [80] Vosniakos, G., (1998), An intelligent software system for the automatic generation of NC programs from wireframe models of 2-1/2D mechanical parts, Computer integrated manufacturing systems, (1-2), 53-65
- [81] Wallace, A. R., (1912), Darwinism: An exposition of the theory of natural selection with some of its applications: with a portrait of the author, map and illustrations, Macmillan, London
- [82] Weinberg, S., (1982), Prve tri minute: sodobni pogled na nastanek vesolja, Društvo matematikov, fizikov in astronomov, Ljubljana
- [83] Wilson, R. J., (1997), Uvod v teorijo grafov, Društvo matematikov, fizikov in astronomov Slovenije, Ljubljana
- [84] www.espacenet.com, 23.05.2004
- [85] www.sciencemuseum.org.uk, 23.05.2004
- [86] Yan, S., Shuilai, W., Shuiguang, T., (2000), Uneven offset method of NC tool path generation for free-form pocket machining, Computers in Industry, 43(1), 97-103
- [87] Yoonhwan, W., (1999), Automatic generation of optimal process plans for machined parts from 3D solid models, Colorado state university, Fort Collins, Colorado
- [88] Zhang, D., (1992), Nc-lathe: An interactive environment for numerical control lathe training, University of Houston, Houston, Texas
- [89] Žnideršič, M., (1995), Družinska enciklopedija Guinness, Slovenska knjiga, Ljubljana

OSEBNA BIBLIOGRAFIJA ZA OBDOBJE OD 2000 DO 2004

ČLANKI IN DRUGI SESTAVNI DELI

1.01 Izvirni znanstveni članek

1. KOVAČIČ, Miha, BALIČ, Jože. Evolutionary programming of a CNC cutting machine. *Int. j. adv. manuf. technol.*, September 2003, vol. 22, no. 1/2, str. 118-124.
2. BREZOČNIK, Miran, KOVAČIČ, Miha. Integrated genetic programming and genetic algorithm approach to predict surface roughness. *Mater. manuf. process.*, May 2003, vol. 18, iss. 3, str. 475-491.
3. PODBREGAR, Matej, KOVAČIČ, Miha, PODBREGAR-MARŠ, Aleksandra, BREZOČNIK, Miran. Predicting defibrillation success by "genetic" programming in patients with out-of-hospital cardiac arrest. *Resuscitation*. [Print ed.], 2003, vol. 57, iss. 2, str. 153-159.
4. KOVAČIČ, Miha, BREZOČNIK, Miran. Genetic programming approach for surface quality prediction. *Teh. vjesn. - Stroj. fak.*, 2003, vol. 10, no. 1, str. 19-24.
5. FICKO, Mirko, KOVAČIČ, Miha, BREZOČNIK, Miran. Genetic algorithms : a useful optimization method for manufacturing problems. *Acad. J. Manuf. Eng.*, 2004, vol. 2, no 1, str. 21-26.
6. BREZOČNIK, Miran, KOVAČIČ, Miha, FICKO, Mirko. Intelligent systems for next-generation manufacturing. *Acad. J. Manuf. Eng.*, 2004, vol. 2, no 1, str. 34-37.
7. KOVAČIČ, Miha, BALIČ, Jože, BREZOČNIK, Miran. Evolutionary approach for cutting forces prediction in milling. *J. mater. process. technol.*. [Print ed.], Available online 8 June 2004, 6 str.

1.08 Objavljeni znanstveni prispevek na konferenci

8. KOVAČIČ, Miha, BREZOČNIK, Miran. Evolutionary transport system. V: KATALINIĆ, Branko (ur.). *Annals of DAAAM for 2001 & Proceedings of the 12th International DAAAM Symposium "Intelligent Manufacturing & Automation: Focus on Precision Engineering"* : Jena University of Applied Sciences, 24-27 October 2001, Jena, Germany. Vienna: DAAAM International, 2001, str. 253-254.

9. KOVAČIČ, Miha, BALIČ, Jože. Genetic algorithm method for cutting optimisation of Monel alloy. V: DOBRZANSKI, Leszek A. (ur.). Proceedings of the 10th Jubilee International Scientific Conference Achievements in Mechanical & Materials Engineering AMME'2001, Gliwice-Cracow-Zakopane, Poland, December 9-13, 2001. Gliwice: Silesian University of Technology, Institute of Engineering Materials and Biomaterials, cop. 2001, str. 305-316.
10. BREZOČNIK, Miran, KOVAČIČ, Miha. Survey of the evolutionary computation and its application in manufacturing systems. V: JURKOVIĆ, Milan (ur.), KARABEGOVIĆ, Isak (ur.). RIM 2001, 3rd International Conference on Revitalization and Modernization of Production, Bihać, Bosnia and Herzegovina, September 27-28, 2001. Proceedings. Bihać: Tehnički fakultet, 2001, str. 501-508.
11. KOVAČIČ, Miha, BREZOČNIK, Miran. Genetic approximation method for surface quality prediction. V: KATALINIĆ, Branko (ur.). Annals of DAAAM for 2002 & Proceedings of the 13th International DAAAM Symposium "Intelligent Manufacturing & Automation: Learning from nature, Wiena, University of Technology, 23-26th October, 2002. Vienna: DAAAM International, 2002, str. 285-286.
12. BREZOČNIK, Miran, KOVAČIČ, Miha. Prediction of surface roughness with genetic programming. V: DOBRZANSKI, Leszek A. (ur.). Proceedings of the 11th International scientific conference Achievements in mechanical & materials engineering, AMME'2002, Gliwice-Zakopane, Poland, December 15-18, 2002. Gliwice: Organising committee of the International scientific conference, Institute of Engineering Materials and Biomaterials of the Silesian University of Technology, 2002, str. 23-26.
13. KOVAČIČ, Miha, BREZOČNIK, Miran. Genetic programming approach for surface quality prediction. V: KATALINIĆ, Branko (ur.), KLJAJIN, Milan (ur.). Proceedings of the 1st DAAAM international conference on Advanced technologies for developing countries, september 12-14, Slavonski Brod. Osijek: University of Josip Juraj Strossmayer; Slavonski Brod: Mechanical engineering faculty, 2002, str. 405-410.
14. BREZOČNIK, Miran, KOVAČIČ, Miha. Integrated evolutionary computation environment for optimizing and modeling of manufacturing processes. V: BRDAREVIĆ, Safet (ur.), EKINOVIĆ, Sabahudin (ur.), COMPAMYS PASCUAL, Ramon (ur.), CALVET VIVANCOS, Joan. TMT 2002, 6th International Research/Expert Conference Trends in the Development Machinery and Associated Technology, Neum, BiH, 18-22 September 2002. Proceedings. Zenica: Faculty of mechanical engineering, 2002, str. 107-110.
15. KOVAČIČ, Miha, BREZOČNIK, Miran. Genetic programming approach for cutting forces prediction in milling. V: KATALINIĆ, Branko (ur.). Annals of DAAAM for 2003 & proceedings of the 14th International DAAAM symposium "Intelligent manufacturing & Automation: focus on reconstruction and development" 22-25th October 2003, Sarajevo, Bosnia and Herzegovina. Vienna: DAAAM International, 2003, str. 234-235.
16. KOVAČIČ, Miha, BREZOČNIK, Miran, BALIČ, Jože. Genetic programming approach for surface quality prediction of monel alloy. V: KUZMAN, Karl (ur.). 4th International Conference on Industrial Tools ICIT 2003, Bled, Celje, Slovenia, April 8-12, 2003. Conference proceedings. Celje: Tecos, Slovenian Tool and Die Development Centre, 2003, str. 385-388.

- 17.** KOVAČIČ, Miha, BREZOČNIK, Miran. Genetic programming planner for mobile robots. V: BOHANEC, Marko (ur.), FILIPIČ, Bogdan (ur.), GAMS, Matjaž (ur.), TRČEK, Denis (ur.), LIKAR, Borut (ur.). Inteligentni in računalniški sistemi. Kompleksni sistemi v e-poslovanju : zbornik A 6. mednarodne multi-konference Informacijska družba IS 2003, 13. do 17. oktober 2003 = proceedings A of the 6th International Multi-Conference Information Society IS 2003, 13-17th October, Ljubljana, Slovenia. Ljubljana: Institut "Jožef Stefan", 2003, a, str. 85-88.
- 18.** KOVAČIČ, Miha, BREZOČNIK, Miran. Evolutionary computation approach for autonomous vehicles. V: KATALINIČ, Branko (ur.), TUFEKČIĆ, Džemo (ur.), ŠELO, Ramiz (ur.). Proceedings of the 2nd DAAAM International conference on Advanced technologies for developing countries, June 25-28, 2003, Tuzla, Bosnia and Herzegovina. Tuzla: Faculty of Mechanical Engineering; Vienna: DAAAM International, 2003, str. 339-344.
- 19.** KOVAČIČ, Miha, BALIČ, Jože, BREZOČNIK, Miran. Evolutionary approach for cutting forces prediction in milling. V: OLABI, Abdul G. (ur.), HASHMI, M.S.J (ur.). Proceedings of the International conference on advances in materials and processing technologies (AMPT 2003), 8-11 July 2003, Dublin. Dublin: Dublin City University, cop. 2003, vol. 1, str. 852-855.
- 20.** KOVAČIČ, Miha, BREZOČNIK, Miran. Genetic algorithm method for computer aided quality control. V: BRDAREVIĆ, Safet (ur.). 3rd Research/Expert Conference with International Participation, Quality 2003, 13th and 14th November 2003, Zenica, Bosnia and Herzegovina. Proceedings. Zenica: Faculty of Mechanical Engineering in Zenica, 2003, str. 125-130.
- 21.** KOVAČIČ, Miha, BALIČ, Jože, BREZOČNIK, Miran. Genetic algorithm approach for autonomous vehicles. V: International conference on industrial technology [also] IEEE ICIT 2003, Hotel Habakuk Maribor, Slovenia, December 10-12, 2003. Proceedings. Piscataway: IEEE, 2003, str. 974-978.
- 22.** BREZOČNIK, Miran, KOVAČIČ, Miha, FICKO, Mirko. Genetic-based approach to predict surface roughness in end milling. V: VIVANCOS CALVET, Joan (ur.), PUERTA SALES, Ferran (ur.), EKINOVIĆ, Sabahudin (ur.), BRDAREVIĆ, Safet (ur.). TMT 2003, 7th International Research/Expert Conference Trends in the Development Machinery and Associated Technology, Lloret de Mar, Barcelona, Spain, 15-17 September 2003. Proceedings. Zenica: Faculty of mechanical engineering, 2003, str. 529-532.
- 23.** PODBREGAR, Matej, KOVAČIČ, Miha, PODBREGAR-MARŠ, Aleksandra, BREZOČNIK, Miran. Napovedovanje uspešnosti defibrilacije s pomočjo "genetskega" programiranja pri bolnikih z zunajbolnišničnim srčnim zastojem. V: BRUČAN, Andrej (ur.), GRIČAR, Marko (ur.), VAJD, Rajko (ur.), FINK, Andrej (ur.). Deseti mednarodni simpozij o urgentni medicini, Portorož, Slovenija, 11.-14. junij 2003. Urgentna medicina : izbrana poglavja 2003 : zbornik : selected topics [2003] : proceedings. Ljubljana: Slovensko združenje za urgentno medicino: = Slovenian Society for Emergency Medicine, 2003, str. 163-165.

24. KOVAČIČ, Miha, BALIČ, Jože. Genetic algorithm approach for CNC cutting machine. V: MicroCAD 2004. Section J, Production engineering and manufacturing systems. Miskolc: University of Miskolc, Innovation and Technology Transfer Centre, 2004, str. 97-102.
25. KOVAČIČ, Miha, BREZOČNIK, Miran. Evolution and transport systems. V: BUDIN, Leo (ur.), RIBARIĆ, Slobodan (ur.). Computers in technical systems. Intelligent systems : proceedings of the joint conferences CTS+CIS : zbornik radova zajedničkih savjetovanja CTS+CIS. Rijeka: Hrvatska udruga za mikroprocesorske, procesne i informacijske sustave, mikroelektroniku i elektroniku - MIPRO HU, 2004, str. 41-44.
26. KOVAČIČ, Miha, BALIČ, Jože, BREZOČNIK, Miran. Evolutionary approach to manufacturing control. V: WANG, Lihui (ur.), XI, Jeff (ur.), SULLIVAN, William G. (ur.), AHMAD, Munir (ur.). Proceedings of the 14th International conference on flexible automation & intelligent manufacturing, June 12-14, 2004, Toronto, Canada. Ottawa, Ontario, Canada: National research council of Canada, 2004, str. 915-921.
27. KOVAČIČ, Miha, BREZOČNIK, Miran, BALIČ, Jože, ČUŠ, Franc. Programming of coordinate measuring machine using genetic algorithm. V: MURAVYOV, Sergey V. (ur.). 10th IMEKO TC7 International Symposium on Advances of Measurement Science, June 30 - July 2, 2004, Saint-Petersburg, Russia. Proceedings. Tomsk: Polytechnic University, 2004, str. 251-255.
28. KOVAČIČ, Miha. Uporaba trirazsežnega tiskanja v tekstilstvu = 3D printing in textile industry. V: DRSTVENŠEK, Igor (ur.). Slojevite tehnologije. Maribor: Fakulteta za strojništvo, 2004, str. 118-121.

1.16 Samostojni znanstveni sestavek v monografiji

29. KOVAČIČ, Miha, BREZOČNIK, Miran. Model of evolutionary transport system. V: KATALINIĆ, Branko (ur.). *DAAAM International scientific book 2002*. Vienna: DAAAM International, 2002, str. 323-328.
30. BREZOČNIK, Miran, KOVAČIČ, Miha. Modelling of intelligent mobility for next-generation manufacturing systems. V: KATALINIĆ, Branko (ur.). *DAAAM International scientific book 2003*, (DAAAM International scientific book). Vienna: DAAAM International, 2003, str. 095-102.
31. KOVAČIČ, Miha, BREZOČNIK, Miran. Evolutionary method for surface quality prediction. V: KATALINIĆ, Branko (ur.). *DAAAM International scientific book 2003*, (DAAAM International scientific book). Vienna: DAAAM International, 2003, str. 331-338.

SEKUNDARNO AVTORSTVO

Prevajalec

32. DRSTVENŠEK, Igor (ur.). *Slojevite tehnologije = Layered technologies*. Maribor: Fakulteta za strojništvo, 2004. 133 str., ilustr.

ŽIVLJENJEPIS

Osební podatki:	Miha KOVAČIČ, univerzitetni diplomirani inženir strojništva rojen 21. oktobra 1974 v Celju
Osnovna šola:	1981-1989 III. Osnovna šola v Celju
Srednja šola:	1989-1993 Poklicna in tehniška strojna šola Celje
Dodiplomski študij:	1993-1995 Univerza v Mariboru, Fakulteta za strojništvo, višješolski program (Celje), konstrukterstvo in gradnja strojev 1995-2000 Fakulteta za strojništvo, univerzitetni program, proizvodno strojništvo
Podiplomski študij	2002 Doktorski izpit
Zaposlitev:	Od 1. novembra 2000 zaposlen kot mladi raziskovalec na Fakulteti za strojništvo v Laboratoriju za inteligentne obdelovalne sisteme.