Рачунарска интелигенција

Генетски алгоритми

Александар Картељ

kartelj@matf.bg.ac.rs

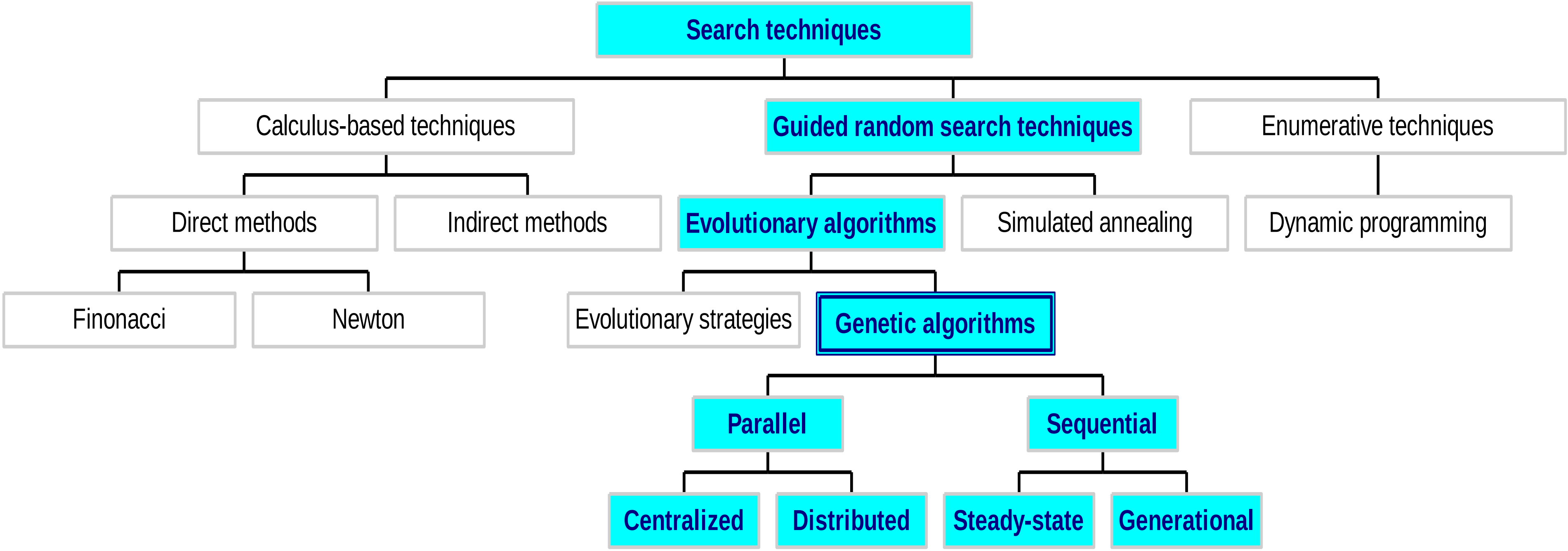
Ови слајдови представљају прилагођење слајдова:
A.E. Eiben, J.E. Smith, Introduction to Evolutionary computing: Genetic algorithms

Датум последње измене: 11.12.2019.

Генетски алгоритми

Уводни концепти и једноставни (канонски) генетски алгоритам

Технике претраге



Генетски алгоритам (GA)

- Развијен у Америци 1970-их
- Кључни аутори: J. Holland, K. DeJong, D. Goldberg
- Главне примене:
  - Проблеми у дискретном домену
- Карактеристике:
  - Није претерано брз – као и већина популационих метахеуристика
  - Добра хеуристика за решавање комбинаторних проблема
  - Доста варијанти, нпр. различити механизми укрштања, мутације, итд.

Једноставни генетски алгоритам (SGA)

- Оригинални генетски алгоритам (GA) који је развио John Holland се назива још и једноставни (канонски) GA или **SGA** (енг. Simple GA)
- Други GA се разликују у:
  - Репрезентацијама (кодирањима и декодирањима)
  - Мутацијама
  - Укрштању
  - Селекцији

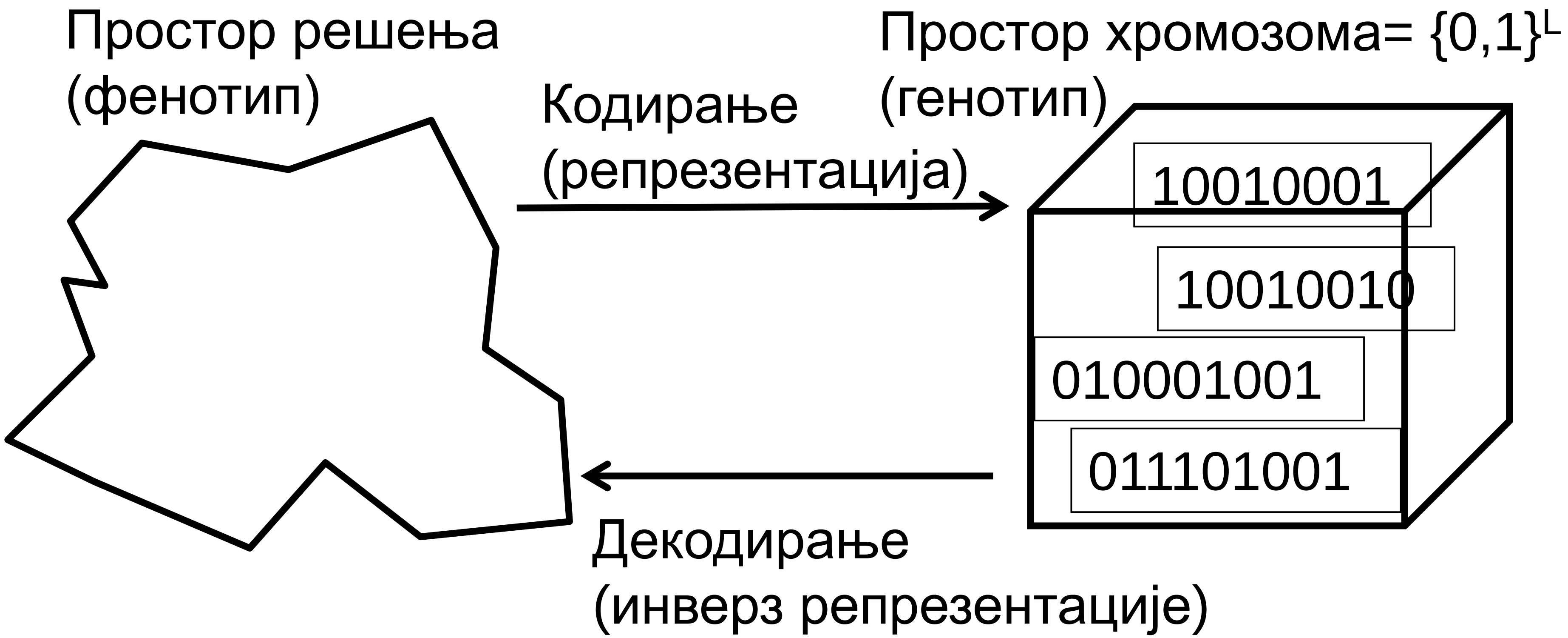
SGA псеудокод

```
Иницијализуј популацију;
Евалуирај популацију;  //израчунавање фитнеса хромозома
while Није задовољен услов за завршетак
{
    Одабери родитеље за укрштање;
    Изврши укрштање и мутацију;
    Евалуирај популацију;
}
```

SGA елементи

Карактеристика GA	Имплементација у оквиру SGA
Репрезентација	Низ битова
Укрштање	n-позиционо или равномерно укрштање
Мутација	Извртање битова са фиксном вероватноћом
Селекција родитеља	Фитнес-сразмерна
Селекција преживелих	Родитељи се потпуно замењују децом
Специјалност	Фокус је на укрштању

SGA Репрезентација (кодирање и декодирање)



SGA Укрштање

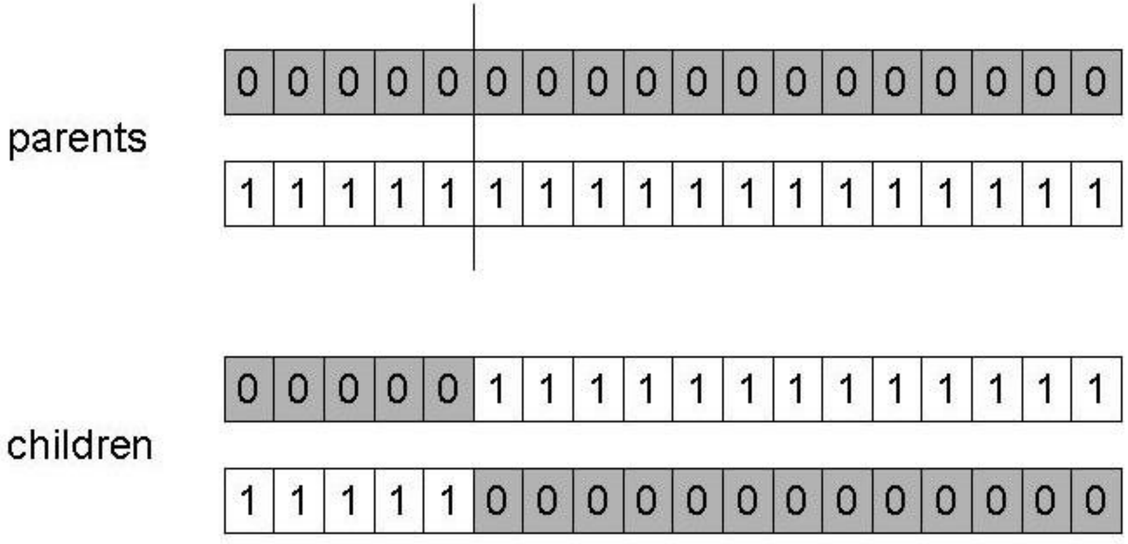
1. Одабери родитеље у скуп за укрштање

    (величина скупа за укрштање = величина популације)

1. Разбацај (енг. Shuffle) скуп за укрштање
2. За сваки пар узастопних хромозома
   примењује се укрштање са вероватноћом p_c,
   а ако се не примени, онда се копирају родитељи
3. За свако дете примењује се мутација са вероватноћом p_m по сваком биту независно
4. Замени целу популацију
   са новодобијеном популацијом деце

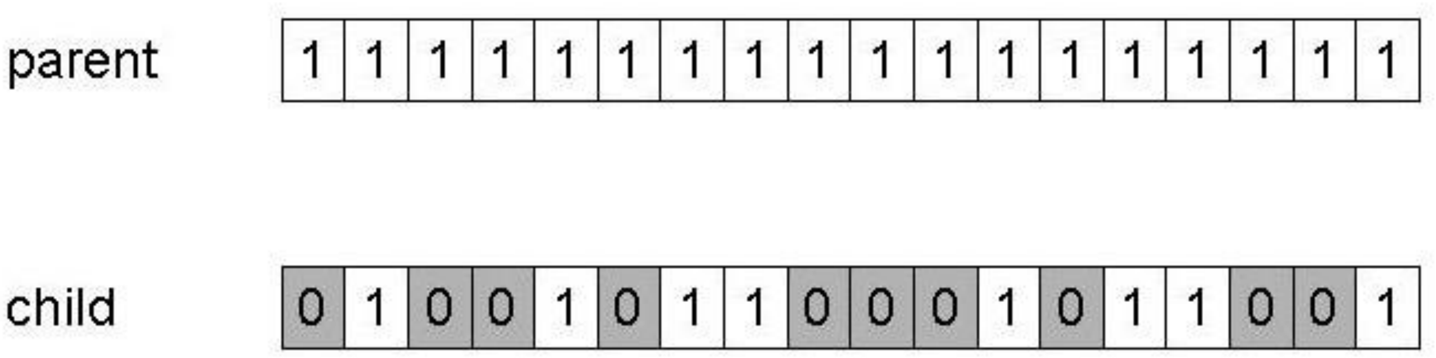
SGA оператор укрштања са једном тачком

- Одабери случајну позицију (мању од броја гена)
- Раздвоји сваког родитеља по овој позицији на два дела
- Креирај децу разменом делова између родитеља
- p_c је обично из интервала [0.6, 0.9]



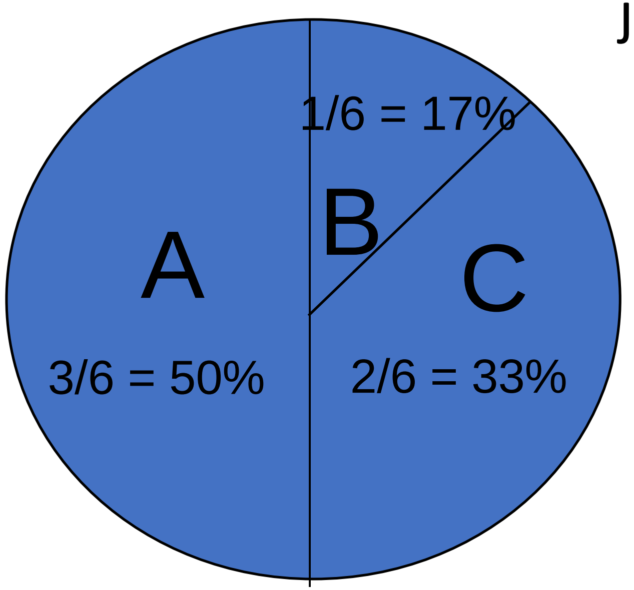
SGA оператор мутације

- Сваки ген (бит) са вероватноћом p_m
- p_m се назива стопа мутације
  - Типично има вредност између (1/величина популације) и (1/ дужина хромозома)



SGA оператор селекције

- Основна идеја: боље јединке имају већу шансу
  - Шансе су сразмерне фитнесу
  - Имплементација: рулетски точак
    - Додели свакој јединки исечак точка
    - Окрени точак n пута за одабир n јединки



fitness(A) = 3

fitness(B) = 1

fitness(C) = 2

Пример

- Једноставан проблем: max x^2 на скупу $\{0,1,\ldots,31\}$
- GA приступ:
  - Репрезентација: бинарни код, нпр. 01101 $\leftrightarrow$ 13
  - Величина популације: 4
  - Једнопозиционо укрштање, мутација по битовима
  - Рулетска селекција
  - Случајна иницијализација популације

Пример - селекција

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

Пример - укрштање

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

Пример - мутација

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

Закључак

- SGA је и даље тема многих студија
  - Релевантан метод за поређење (енг. benchmark) са другим GA
- Многа ограничења:
  - Репрезентација је превише рестриктивна
  - Мутација и укрштање примењиви само за битовску или целобројну репрезентацију
  - Селекција осетљива на случај када популација конвергира (фитнес вредности блиске)
  - Генерисање популације се може унапредити техником експлицитног преживљавања

Генетски алгоритми

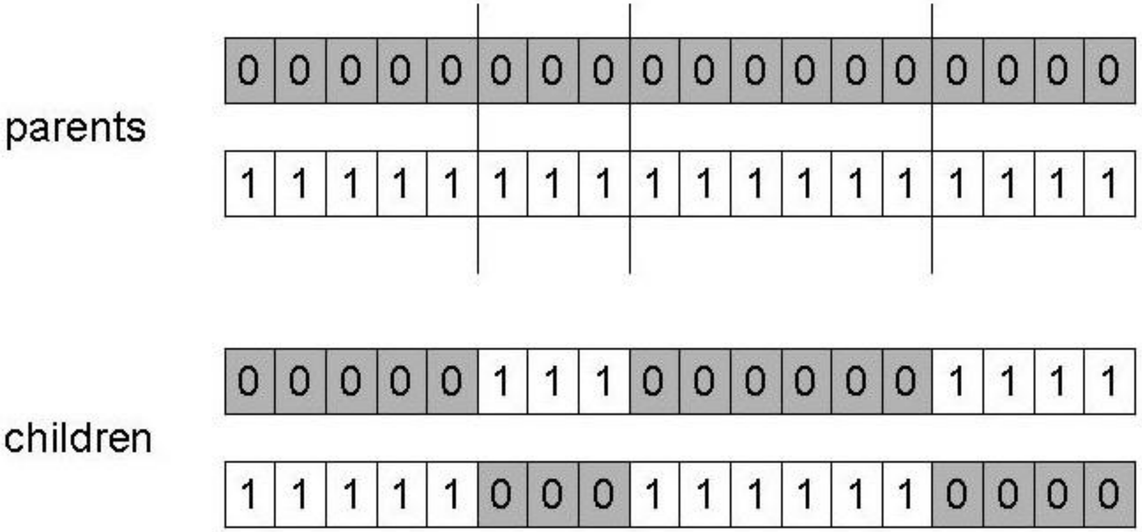
Остали оператори укрштања

Други оператори укрштања

- Квалитет једнопозиционог укрштања
  зависи од редоследа променљивих у репрезентацији решења
  - Већа је шанса да ће гени који су близу бити задржани у потомству
  - Такође, гени који су на различитим крајевима хромозома
    се не могу наћи у истом потомку
  - Ово се зове *позициона пристрасност*
  - Може бити корисна уколико знамо структуру проблема,
    међутим, у општем случају је непожељна

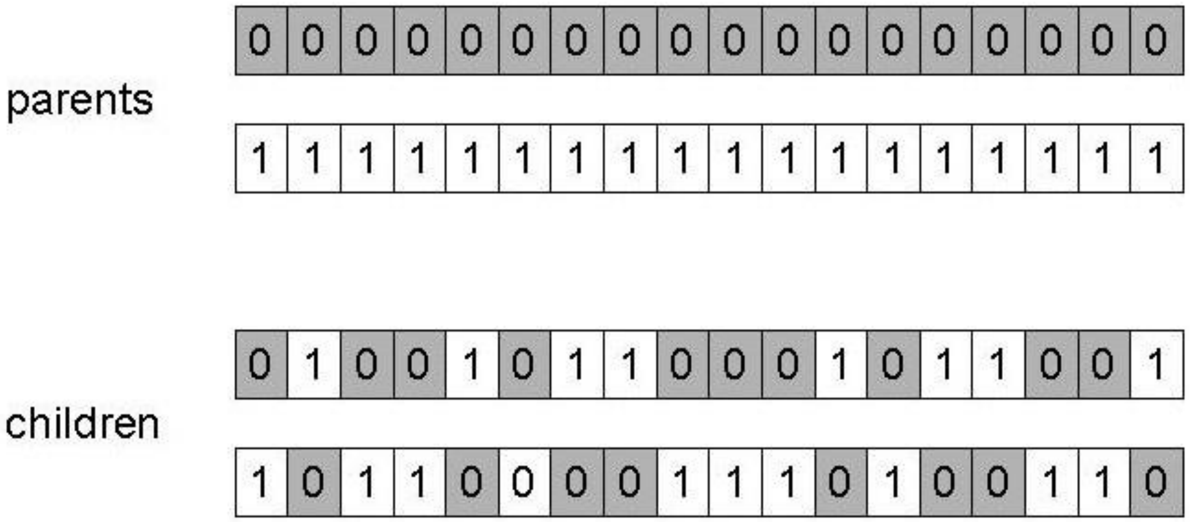
n-позиционо укрштање

- Бира се n случајних позиција
- Раздваја се по тим позицијама
- Спајају се алтернирајући делови
- Ово је уопштење једнопозиционог укрштања,
  у којем и даље постоји *позициона пристрасност*



Равномерно укрштање

- Као код бацама новчића, додељују се ’главе’ једном родитељу, ’писмо’ другом
- Баца се новчић за сваки ген првог детета и узима ген из одговарајућег родитеља
- Друго дете је инверз првог
- Наслеђивање је стога независно од позиције



Укрштање или мутација?

- Дебата дуга неколико деценија

- Одговор (или бар опште прихваћенији аргумент):
- Зависи од проблема, али
- Најбоље је да постоје оба пошто имају различите улоге
- Само мутацијски ЕА су могући,
  док Само укрштајући ЕА не би радили

Укрштање или мутација? (2)

- Експлорација:
  откривање нових области у простору претраге
- Експлоатација:
  оптимизација у оквиру постојећих области (комбиновање решења)
- Постоји кооперација и конкуренција између њих
- Укрштање ради експлоатацију,
  прави комбинације „између“ родитељских хромозома
  → ако неки алел потребан за глобални оптимум не постоји,
  онда глобално решење никад неће бити достигнуто
- Мутација је доминантно експлоративна,
  пошто уводи нову информацију и тиме проширује простор претраге
  → мутација врши и експлоатацију, јер мења локалну околину тренутног решења

Укрштање или мутација? (3)

- Само укрштање може да комбинује информације два родитеља
- Само мутација може да уведе нове информације (генски алели)
- Укрштање не мења фреквенцију генских алела у оквиру популације (на пример: 50% нула на првом биту, ?% после извођења *n* укрштања)
- Да би се погодио оптимум, обично је потребна „срећна“ мутација

Генетски алгоритми

Реалне и пермутацијске репрезентације и оператори

Друге репрезентације

- Грејово кодирање целих бројева (и даље бинарни хромозоми)
  - Грејово кодирање је понекад погодније, јер малим променама у генотипу се праве и мале промене у фенотиуп (за разлику од стандардног бинарног кода)
  - “Глаткије” генотип-фенотип пресликавање може да побољша рад GA
- Данас је, међутим, опште прихваћено да је боље кодирати нумеричке вредности директно као:
  - Целе бројеве
  - Реални бројеви у фиксном зарезу
  - Ово захтева да и оператори не буду дизајнирани да раде са бинарним бројевима, већ са одговарајућим типом целим/реалним бројевима

Директна целобројна репрезентација

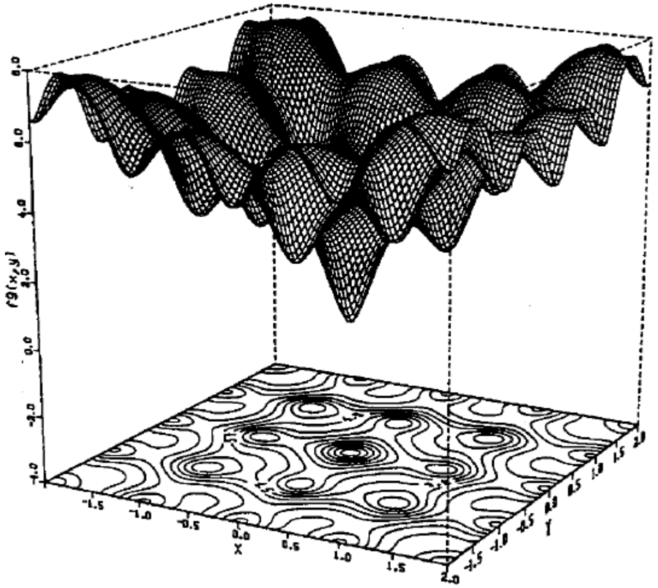
- Неки проблеми природно имају целобројну репрезентацију решења, нпр. вредности параметара у процесирању слика
- Неки други могу имати категоричке вредности из фиксираног скупа, нпр. {плаво, зелено, жуто, розе}
- n-позиционо / равномерно укрштање ради у овим ситуацијама
- Бинарна мутација се мора проширити (не може бити само извртање битова)
  - Мутирање у блиске (сличне) вредности
  - Мутирање у насумичне вредности (типично код категоричних промењивих)

Проблеми у реалном домену

- Шта ако проблем има решење са реалном репрезентацијом, нпр. у проблемима глобалне оптимизације $f : \Re^n \rightarrow \Re$
- Типичан тест пример: Ackley-јева функција

$$f(\overline{x}) = -c_1 \cdot exp\left(-c_2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n} \cdot \sum_{i=1}^{n} cos(c_3 \cdot x_i)\right) + c_1 + 1$$

$$c_1 = 20,\ c_2 = 0.2,\ c_3 = 2\pi$$



Пресликавање реалних вредности на низове битова

$z \in [x,y] \subseteq \Re$ представљени као низ битова $\{a_1,\ldots,a_L\} \in \{0,1\}^L$

- $[x,y] \rightarrow \{0,1\}^L$ мора бити инверзно (један фенотип за сваки генотип)
- Γ: $\{0,1\}^L \rightarrow [x,y]$ дефинише репрезентацију

$$\Gamma(a_1,\ldots,a_L) = x + \frac{y-x}{2^L - 1} \cdot \left(\sum_{j=0}^{L-1} a_{L-j} \cdot 2^j\right) \in [x,y]$$

- Само 2^L вредности од могућих бесконачно је могуће кодирати
- L детерминиште прецизност решења
- Велика прецизност → дугачки хромозоми (спора еволуција)
- Алтернативно, кодирање може бити директно уз дораду оператора

Мутација за директно реално кодирање

Општа шема за бројеве у фиксном зарезу

$$\overline{x} = \left\langle x_1,\ \ldots, x_l \right\rangle \rightarrow \overline{x}' = \left\langle x_1', \ldots, x_l' \right\rangle$$
$$x_i, x_i' \in \left[LB_i, UB_i \right]$$

- Равномерна мутација:

  x_i' се бира равномерно из $[LB_i, UB_i]$
- Аналогно извртању битова (бинарни код)
  или насумичном мутирању (код целих бројева)

Мутација за директно реално кодирање (2)

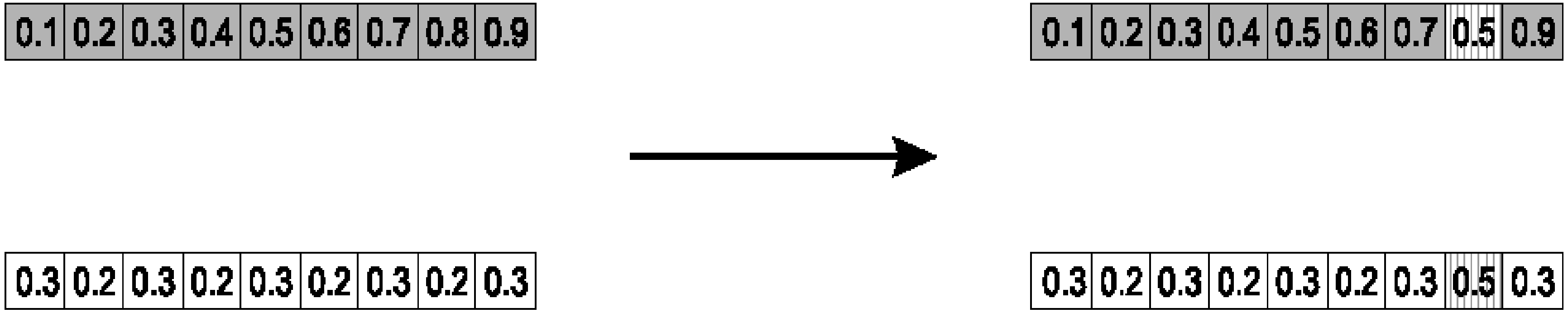
- Неравномерне мутације:
  - Постоје мутације чија се вероватноћа мења са временом, позицијом, итд.
  - Стандардни приступ је додељивање случајне девијације свакој промењљивој, а потом извлачење промењљивих ис $N(0, \sigma)$
  - Стандардна девијација σ контролише удео промена (2/3 девијација ће се налазити у опсегу ($-\sigma$ to $+\sigma$)

Укрштање за директно реално кодирање

- Код дискретног домена (бинарни или целобројни):
    - Сваки алел детета *z* је директно наслеђен од неког од родитеља *(x,y)* са једнаком вероватноћом: $z_i = x_i$ or y_i
- Овде нема смисла користити n-позиционо или равномерно
- Скица решења:
    - Формирање деце који су “између” родитеља (тзв. Аритметичко укрштање)
    - $z_i = \alpha x_i + (1 - \alpha) y_i$ где је $\alpha : 0 \leq \alpha \leq 1$.
    - Параметар α може бити:
        - константа: равномерно аритметичко укрштање
        - променљива (нпр. зависи од старости популације)
        - одабран случајно сваки пут

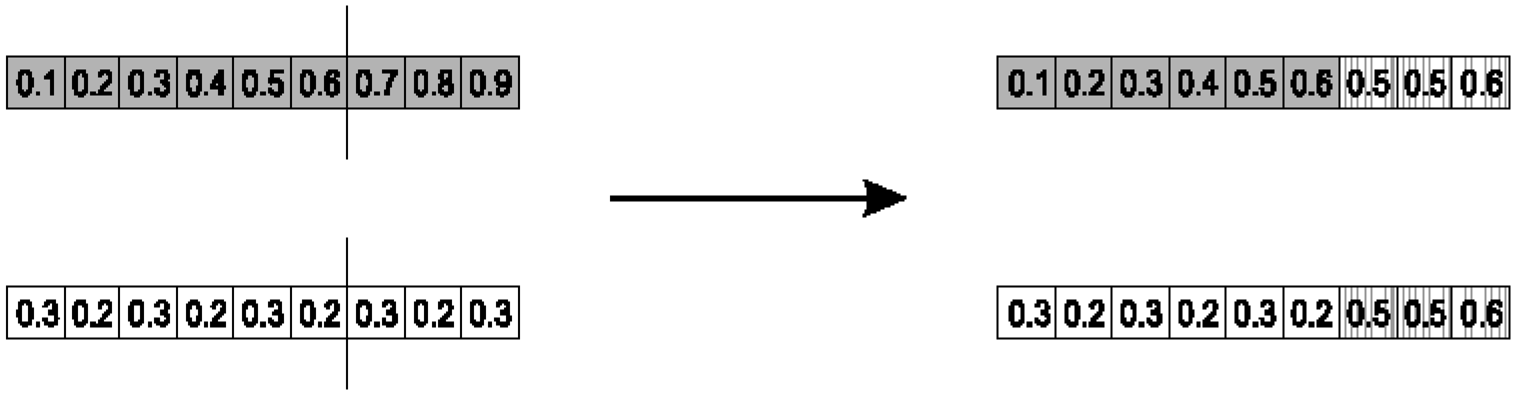
Једноструко аритметичко укрштање

- Родитељи: $\langle x_1,\ldots,x_n \rangle$ and $\langle y_1,\ldots,y_n \rangle$
- Одабери један ген (k) случајно,
- Нпр. за $\alpha = 0.5$ добијамо:



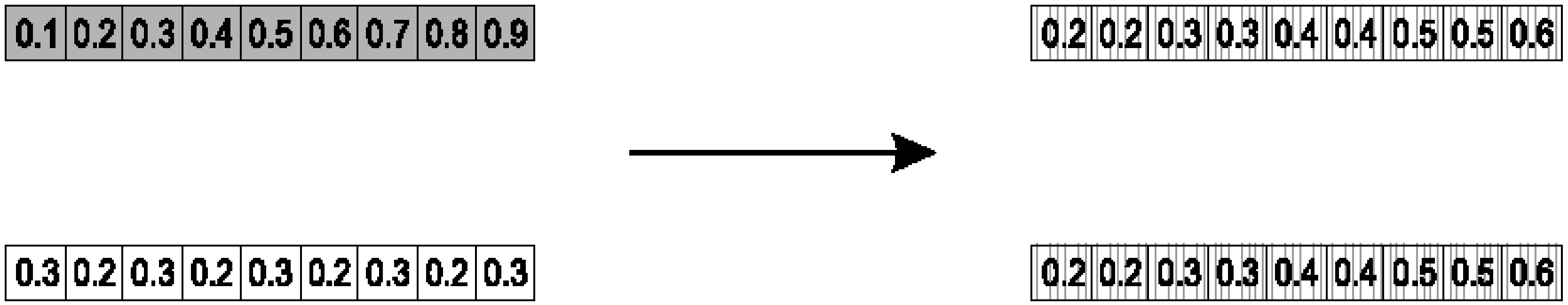
Једноставно аритметичко укрштање

- Родитељи: $\langle x_1,\ldots,x_n \rangle$ and $\langle y_1,\ldots,y_n \rangle$
- Одабери случајни ген *(k)* који одређује позицију
- Нпр. за $\alpha = 0.5$ добијамо:



Целовито аритметичко укрштање

- Најчешће коришћено (задржавање 1 детета – дупло више укрштања)
- Родитељи: $\langle x_1,\ldots,x_n \rangle$ and $\langle y_1,\ldots,y_n \rangle$
- Нпр. за α = 0.5 добија се:



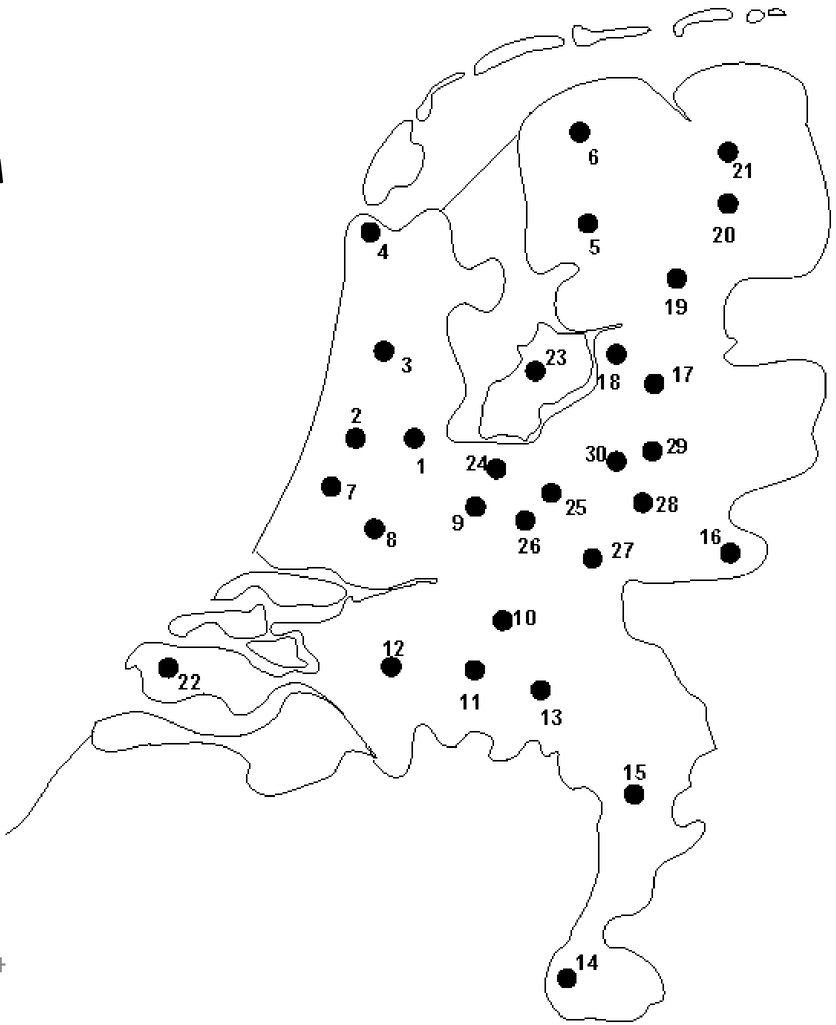
Проблеми засновани на пермутацијама

- Постоје многи проблеми који за решење имају уређену структуру: линеарну, квадратну, хијерархијску, итд.
- Задатак је организовати објекте у одговарајућем редоследу:
  - Пример: проблем сортирања
  - Пример: проблем трговачног путника (TSP)
- Овакви проблеми се генерално изражавају посредством пермутација:
  - Ако постоји *n* променљивих, онда је репрезентација сачињена од *n* целих бројева, таквих да се сваки појављује тачно једном

Пример - TSP

- Проблем:
  - Нека је дато n градова
  - Пронаћи руту са минималном дужином
- Кодирање:
  - Означи градове са 1, 2, … , *n*
  - Једна комплетна рута је једна пермутација (нпр. за n =4 [1,2,3,4], [3,4,2,1] су у реду)
- Простор претраге је ВЕЛИК:

за 30 градова,
30! $\approx 10^{32}$ могућих рута



Мутација над пермутацијама

- Нормални оператори мутације доводе до недопустивих решења
  - Нпр. оператор који гену *i* са вредношћу *j*
  - Мења у неку вредност *k* би значило да се вредност *k* појављује више пута, док се вредност *j* више не налази у решењу
- Стога, се морају мењати вредности бар двема променљивама
- Вероватноћа мутације сада описује вредност примене оператора над целим решењем, а не над појединачним позицијама

Мутација заснована на уметању

- Изаберу се две вредности (два алела) на случајан начин
- Други се умеђе тако да буде после првог,
  при чему се сви остали померају уколико је потребно
- Приметити да ово задржава већи део уређења
  односно информације о претходном суседству
  - То је добро, јер мутација не треба да изазива драматичне промене

Мутација заснована на замени

- Изаберу се две вредности на случајан начин и замене се
- Задржава већину уређења

Мутација заснована на инверзији

- Изаберу се две вредности на случајан начин,
  а потом се обрне редослед вредности између њих
- Мало интензивнија промена уређења од претходна два приступа

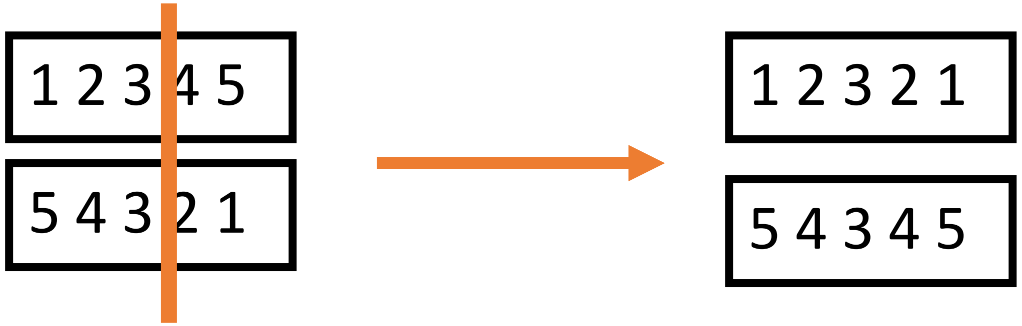
Мутација заснована на мешању

- Изабере се подскуп позиција на случајан начин
- Случајно се реорганизују вредности на тим позицијама

(позиције не морају да буду узастопне као на слици)

Укрштање у пермутационим проблемима

- “Нормални” оператори укрштања доводе до недопустивих решења



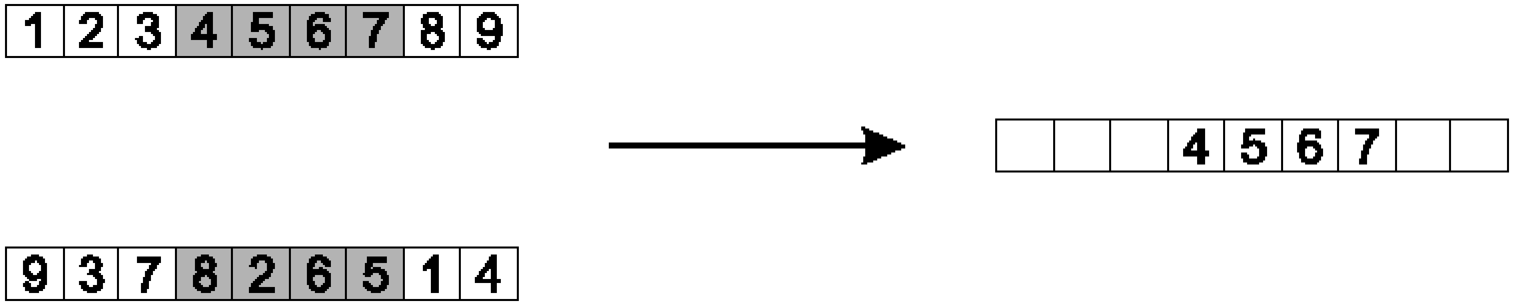
- Предложени су многи специјализовани оператори у зависности од интензитета комбиновања родитељских алела

Укрштање првог реда

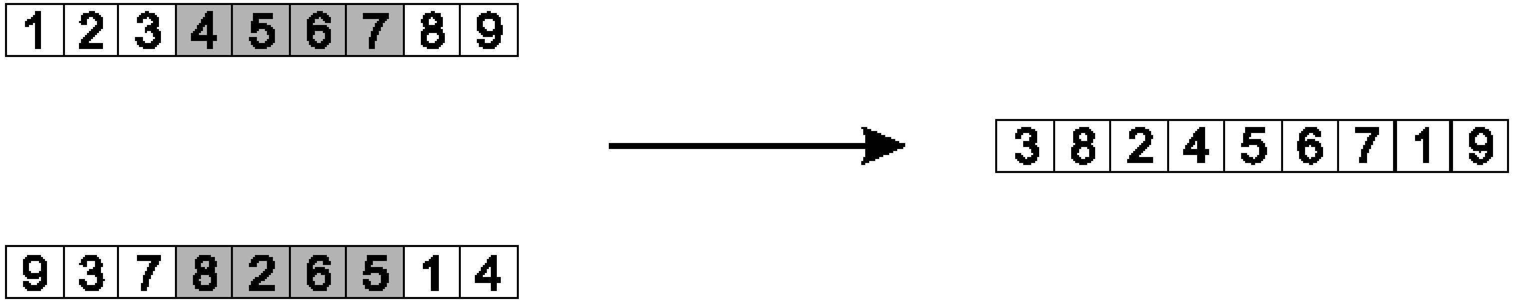
- Идеја је да се задржи релативно уређење
- Општа шема:
  1. Одабрати сегмент хромозома првог родитеља
  2. Ископирати овај сегмент у прво дете
  3. Ископирати преостале вредности (бројеве) тако да:
     - Копирање почиње десно од копираног сегмента
     - Коришћењем **редоследа** датог другим родитељем
  4. Слично се ради и за друго дете

Укрштање првог реда - пример

- Копирање првог сегмента првог родитеља



- Копирање преосталих вредности у редоследу другог 1,9,3,8,2



Делимично укрштање (PMX)

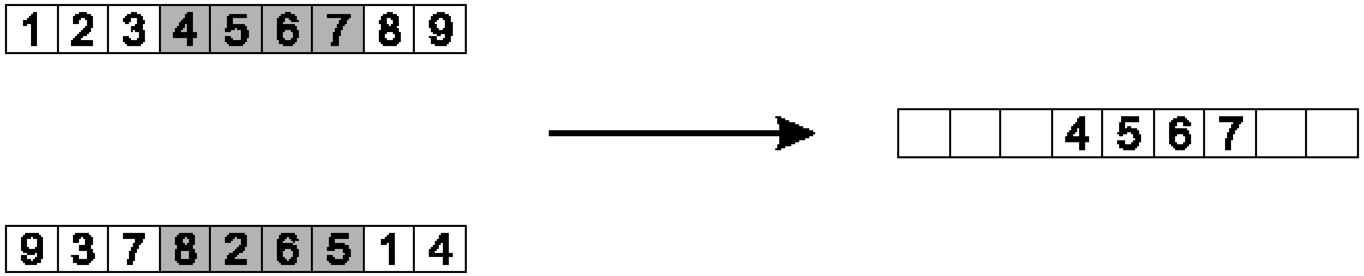
Општа шема за родитеље P1 и P2:

1. Одабрати случајни сегмент и копирати га од P1
2. Почев од позиције почетка сегмента,
   тражити елементе у том сегменту за P2 који нису били копирани
3. За сваки од ових *i* пронађи пронађи вредност *j* из *P1*
   која је копирана на његово место
4. Постави *i* на позицију заузету са *j* у P2,
   пошто знамо сигурно да *j* неће бити тамо (она је већ у детету)
5. Ако је место на којем се налази *j* у P2 већ заузето вредношћу *k,*
   онда постави *i* на позицију коју заузима *k* у P2
6. На крају се преостали елементи само ископирају из P2.

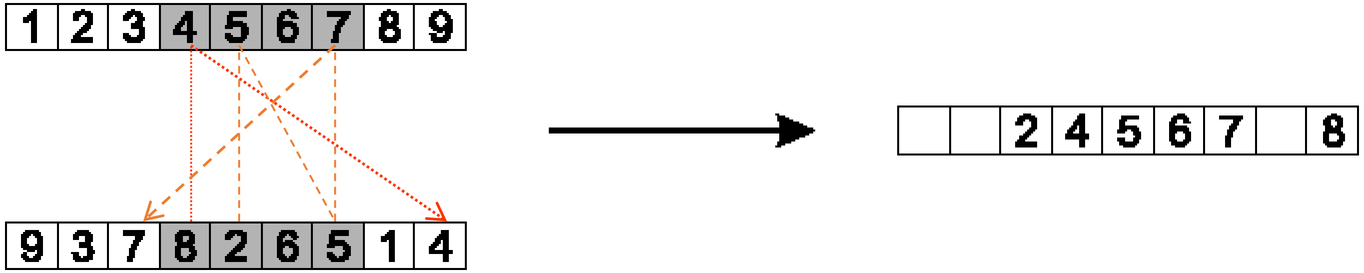
Друго дете се креира аналогно

PMX - пример

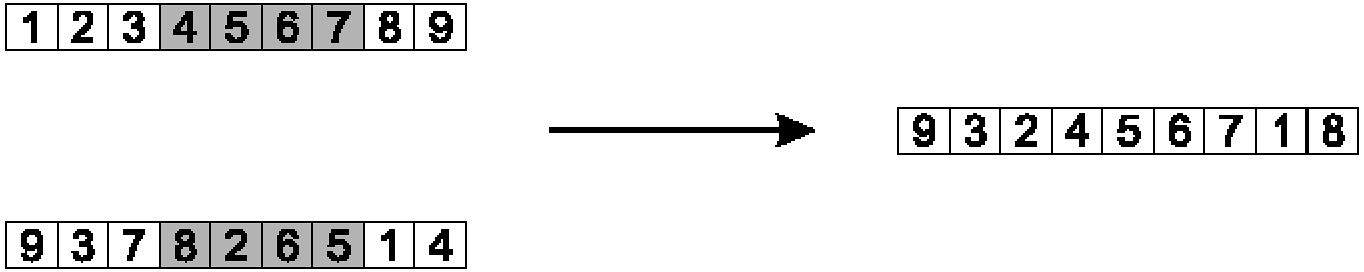
- Корак 1



- Корак 2



- Корак 3



Генетски алгоритми

Популациони модели и селекција

Популациони модели

- SGA користи тзв. Генерацијски модел (Generation GA – GGA):
  - Свака јединка преживи тачно једну генерацију
  - Цео скуп родитеља је замењен својим потомцима
- Са друге стране, постоји и тзв. модел са Стабилним стањем (Steady-State GA SSGA) :
  - Једно дете се генерише по генерацији,
  - Један члан популације бива замењен њиме,
- Генерацијски јаз
  - Удео популације која се мења
  - 1.0 за GGA, 1/величина популације за SSGA

Такмичење засновано на фитнесу

- Селекција се може јавити у два наврата:
  - Селекција родитеља за укрштање
  - Селекција преживелих - бирање из скупа родитељи + деца оних који ће прећи у наредну генерацију
- Разлике међу селекцијама се праве на основу:
  - оператора: дефинишу различите вероватноће
  - алгоритми: дефинишу како су вероватноће имплементиране

Пример селекције: SGA

- Очекивани број копија јединке *i*

$$E(n_i) = \mu \bullet f(i)/ \langle f \rangle$$

(μ = величина популације, f(i) = фитнес јединке i, ⟨f⟩ просечан фитнес популације)

- Рулетска селекција:
    - За дату расподелу вероватноћа, окрени рулетски точак *n* пута
    - Нема гарантоване доње или горње границе на n_i
- Baker SUS алгоритам:
    - *n* еквидистантни граничник постављен на точку – једно окретање
    - Гарантује да је $floor(E(n_i)) \leq n_i \leq ceil(E(n_i))$

Фитнес-сразмерна селекција

- Проблем
  - Једна високо квалитетна јединка може брзо да преузме читаву популацију ако су остале јединке значајно лошије: рана конвергенција
  - Када су фитнеси слични (пред крај), селекциони притисак је лош
    - Селекциони притисак дефинише колико су фаворизована добра решења
    - Када су фитнеси релативно слични (блиски), смањује се и фаворизација
- Скалирање може да помогне
  - Скалирање према најгорем: $f'(i) = f(i) - \beta^t$
    - Где је β најгори фитнес у последњих *n* генерација

Ранг-базирана селекција

- Покушај да се превазиђу проблеми фитнес-сразмерне селекције
- Вредност фитнеса нема апсолутни већ релативни значај овде
- Најбоља јединка има највиши ранг μ, а најгора ранг 1
- Трошак примене сортирања је обично занемарљив

Турнирска селекција

- Претходне методе се ослањају на опште популационе статистике
  - Ово може бити уско грло, нпр. на паралелним машинама
  - Ослањају се на присуство екстерних фитнес функција
    које можда не постоје увек: нпр. еволуција ботова за игрице
    (овде не знамо који је фитнес, али можемо да утврдимо ко боље игра)
- Општа шема:
  - Одабери *k* чланова на случајан начин, а потом одабери најбољег од њих
  - Наставити процес за одабир још јединки

Турнирска селекција (2)

- Вероватноћа одабира јединке *i* зависи од:
  - Ранга *i*
  - Вредности *k*
    - веће *k* значи и већи селекциони притисак
  - Да ли се такмичару бирају са враћањем
    - Одабир без враћања појачава селекциони притисак
- За $k = 2$, време потребно да најбоља јединка преузме популацију

  је иста као код линеарног рангирања за $s = 2 \bullet p$

Селекција преживелих

- Методе сличне онима које се користе за одабир родитеља за укрштање
  - У генерацијском моделу тривијално, бришу се најстарији, тј. сви родитељи
  - У општем случају се могу бирати/брисати било које јединке из скупа родитеља и деце
- Две групе приступа:
  - Селекција заснована на старости
    - Као код SGA
    - SSGA може да имплементира брисање случајне (не препоручује се) или брисање најстарије
  - Фитнес-сразмерна селекција
    - Примена неке од раније поменутих метода: рулетска, турнирска, ...
- Специјални случај:
  - Елитизам
    - Често коришћен код оба популациона модела (GGA, SSGA)
    - Увек се задржава копија најбољег решења до сада

Теорема о схемама

- Теоријска основа иза генетских алгоритама и генетског програмирања (John Holland, седамдесете године)
- Неједнакост која објашњава еволутивну динамику
- ***Теорема (неформално): кратке схеме са натпросечним фитнесом постају експоненцијално учесталије током генерација***
- Схема је шаблон који идентификује подскуп ниски које су сличне на појединачним позицијама
- *Пример: за бинарне ниске дужине 6, пример схеме је 1*10*1 где оваква схема описује све ниске дужине 6 са фиксираним битовима на 4 описане позиције*

Теорема о схемама (2)

- Ред схеме *o(H)* је дефинисан као број фиксираних позиција
- *∂(X)* је удаљеност између прве и последње фиксиране позиције
- Фитнес схеме је просечни фитнес свих ниски које припадају схеми
- ***Теорема:***

$$\mathrm{E}(m(H,t+1)) \geq \frac{m(H,t)f(H)}{a_t}[1-p].$$

  - где је *m(H,t)* број ниски које припадају схеми *H* у генерацији *t, f(H)* просечни фитнес схеме *H,* док је *a(t)* просечни фитнес у генерацији *t*
  - *p* је вероватноћа да ће укрштање или мутација „разбити“ схему:

$$p = \frac{\delta(H)}{l-1}p_c + o(H)p_m$$

  - *l* је дужина генотипа док су p_c и p_m вероватноће укрштања и мутације