Генетички алгоритам: основне референце

- Goldberg, David (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley Professional
- Fraser, Alex S. (1957). "Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction," *Australian Journal of Biological Sciences* **10**: 484–491.

Еволуција и оптимизација

- У природи, еволуција се заснива на
  - размножавању
  - мутацији
  - такмичењу
  - селекцији
- Овај процес се примењује на све живе организме, генерацију за генерацијом
- Немогуће је за једну врсту (живог света) да буде потпуно прилагођена на околину, јер се околина стално мења
- **Еволуција је оптимизациони процес**
  - Ко је започео овај процес оптимизације?
  - Ко је крајњи корисник резултата те оптимизације (еволуције)?

Дарвинова теорија еволуције и ГА

- ГА је настао из покушаја симулација живог света (нумеричка биологија)
  - Game-of-life
- Теорија еволуције (Дарвин) је објединила биологију, али је корак даље укључивање интелигенције
- Данас се сматра да се интелигенција и еволуција не могу раздвојити
- Нивои учења
  - филогенетички (кроз наследство)
  - онтогенетички (кроз искуство током живота јединке)
  - социогенетички (кроз интеракцију у групи)
- Локални оптимизациони алгоритми се могу схватити као апстракција онтогенетичког учења: једно решење се поправља на основу неких правила
- ГА се може схватити као апстракција филогенетичког учења – кроз искуство претходних генерација које је сачувано у генима

Генетички алгоритам: симулација опстанка

- Основна идеја: имплементације алгоритама опстанка из природе
- Генетички алгоритам (ГА) је један од генералних приступа оптимизацији
- Заснива се на принципу природне селекције и опстанка **најприлагођенијих** јединки околини
- ГА спада у **стохастичке** алгоритме за **глобалну** оптимизацију
- Алгоритам се најчешће описује
  - строге математичке дефиниције захтевају апстрактне дефиниције оператора

Елементи ГА: индивидуе, гени и способности

- Једно могуће решење оптимизационог проблема (једна тачка оптимизационог простора) назива се **индивидуа** и састоји се од **гена**
- Вектор оптимизационих променљивих је индивидуа $\mathbf{x} = (x_1, x_2, \ldots x_D)$
- Свака индивидуа има онолико гена колико променљивих има оптимизациона функција (број гена једнак је броју димензија оптимизационог простора, D)
- **Једна оптимизациона променљива**, за изабрани запис оптимизационог проблема, представља **један ген**
- Ген је најмања јединица за рекомбинацију (претрагу простора)
- Вредност оптимизационе функције, $f(\mathbf{x})$, у једној тачки оптимизационог простора, назива се **способност** (енг: fitness) индивидуе
  - Један ген нема f
  - Група гена која дефинише индивидуу има f

Елементи ГА: популација и генерација

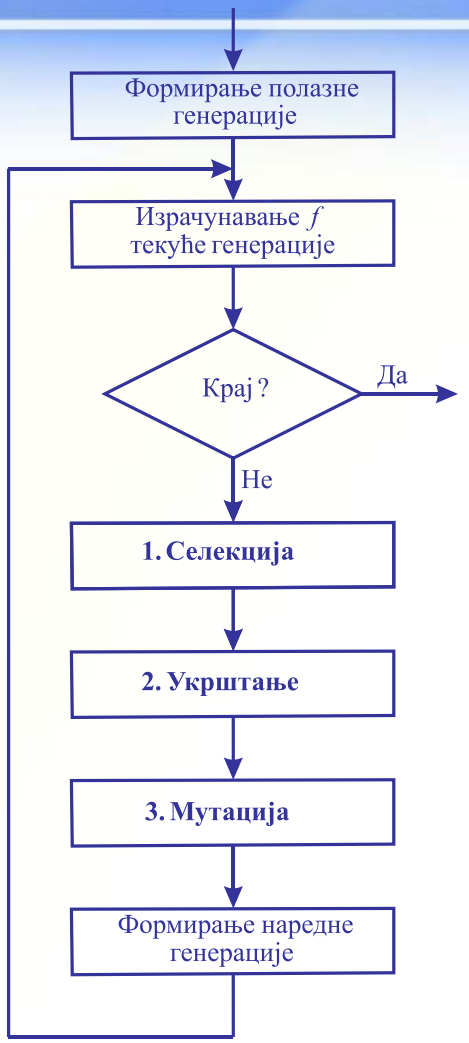
- ГА врши операције над **скупом решења** у оптимизационом простору
  - сви други претходно описани алгоритми оперишу само са једним (најбољим) решењем које "поправљају"
  - NM симплекс ради са скупом од тачно $D+1$ решења али се једно (најбоље) решење стално прати
- Скуп индивидуа (решења) назива се **популација**
- Број чланова популације је нужно већи од један
  - типично 100, 1000, или (много) више
- Током оптимизације, текућа популација замењује се наредном популацијом решења
- Популација у једном кораку алгоритма назива се генерација

Основна идеја ГА и формирање почетне генерације

- Идеја: од најбољих решења из претходне генерације формира се наредна генерација решења која (би требало да) су боља
- Формирањем сукцесивних генерација долази се до све бољих решења оптимизационог проблема
- ГА почиње формирањем почетне популације (полазног скупа решења) које се обично назива полазна или нулта генерација
  - најчешће је то скуп случајних вектора у опт. простору
  - може се формирати и на неки други начин (детерминистички, униформно, полазећи од процене, итд.)
  - ГА не залази у начин формирања нулте генерације

Блок дијаграм и формални кораци ГА

- Један корак ГА се састоји од 3 операције
  - селекција
  - укрштање
  - мутација
- Коришћењем ових операција од претходне генерације добија се наредна генерација
- Ово је најопштији случај алгоритма, прескакањем појединих оператора добијају се посебни случајеви



Бинарни и континуални ГА

- У зависности од представе гена, разликујемо две класе ГА:
  - бинарни ГА и
  - континуални ГА
- У литератури се под називом ГА најчешће подразумева бинарни ГА
- Од представе гена зависи реализација оператора укрштања и оператора мутације
- У литератури постоје опречна мишљења о томе које је представљање боље (зависи од конкретног проблема)

Представљање гена

- У природи су гени живог света представљени дискретним скупом елемената
  - број могућих комбинација људског генома је $4^{3\,000\,000\,000}$ (макс.)
- Из тог разлога су у првим применама ГА оптимизациони параметри представљани бинарно
- Бинарна представа је врло погодна за дискретне комбинаторно-оптимизационе проблеме (SAT, TSP)
- Реалне променљиве:
  потребно је извршити конверзију и дискретизацију
  - компликује имплементацију алгоритма
- Гени могу бити и реални бројеви, са тачношћу до машинске
- Сви оптимизациони проблеми (SAT, TSP, NLP) могу се решавати помоћу ГА

1. Оператор селекције

- У оквиру текуће генерације селектује се скуп родитељских решења
  - У природи су то они чланови популације који су оставили потомство
- Тај скуп служи за стварање наредне генерације, док остали чланови текуће генерације не учествују у формирању наредне генерације
- На овај начин се фаворизује простирање добрих решења и омогућава се њихово даље побољшавање
- Селекција се може реализовати
  - стохастички или
  - детерминистички

Стратегије селекције

- Најчешће коришћене стратегије селекције
  - Децимација
  - Пропорционална (рулет) селекција
  - Турнир селекција
- Постоји варијација за сваку од ових стратегија
- Постоје и многе друге стратегије које су предложене у литератури и коришћене у пракси
- Не постоји закључак која стратегија је (нај)боља

Стратегије селекције: Децимација

- Децимација популације
  - сортирање популације према оптимизационој функцији
  - затим се из тог скупа изабере подскуп првих k (најбољих) решења
- Предност овог приступа је **једноставност** имплементације
- Мана је сувише **брз губитак разноликости** генетског материјала унутар популације

Bilbo: I have... I have never used a sword in my life.
Gandalf: And I hope you never have to. But if you do, remember this: true courage lies in knowing not when to take a life, but when to spare one.

Елитизам и децимација

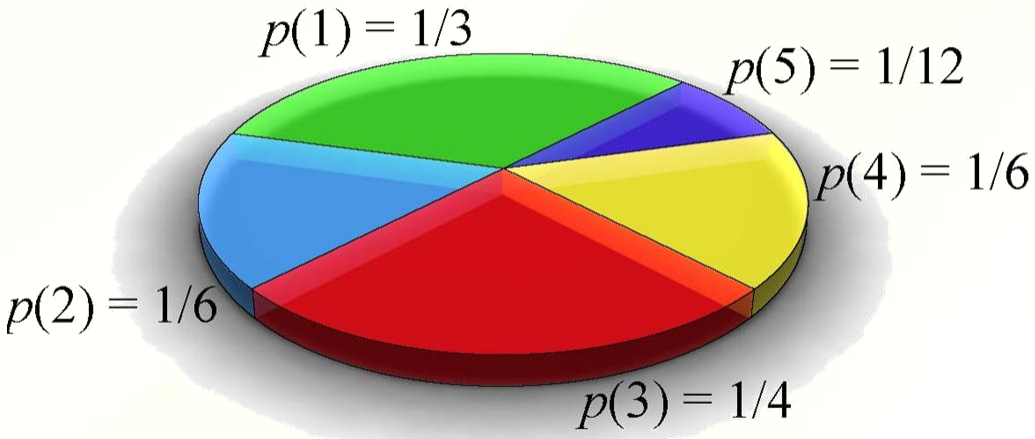
- При формирању следеће генерације могу се
  - задржати најбоља изабрана решења (елитистички приступ)
  - потпуно избацити решења из претходне генерације
- Иако се на први поглед може чинити да је добро оставити бар најбоље решење из претходне генерације, нумерички експерименти (статистички) показују **супротно**
- Елитистичка стратегија
  - доприноси бржој конвергенцији алгоритма
  - повећава вероватноћу конвергенције читаве генерације ка локалном минимуму из ког је алгоритму потребно много времена да изађе
- Елитистички приступ није погодан у општем случају, али се може користи као опција за убрзање конвергенције алгоритма

Стратегије селекције: Пропорционална (рулет) селекција

- Свакој индивидуи се додељује вероватноћа избора p_i пропорционална способности индивидуе f_i

$$p_i = \frac{f_i}{\sum_i f_i}$$

- Случајним избором селектује се једна индивидуа која учествује у стварању наредне популације



- Илустрација: популација са 5 индивидуа, случајном променљивом из скупа [0,1] бира се једна од индивидуа

Стратегије селекције: Пропорционална (рулет) селекција

- Вишеструким понављањем избора добија се група индивидуа за укрштање
- **Фаворизују се најбоља решења**, али постоји коначна вероватноћа да **понеко лоше** (али “срећно”) **решење учествује у укрштању**
- Повећава се разноликост генетског материјала током оптимизације, али се конкретна имплементација селекције усложњава
- Конвергенција ГА се успорава на овај начин

Стратегије селекције: Турнир селекција

- Из популације се на случајан начин изабере подскуп од N индивидуа
- Најчешће се користи бинарни турнир код кога је $N = 2$
- У изабраном подскупу пронађе се индивидуа са најбољом способношћу која је селектована за укрштање
- Изабрана индивидуа се избацује из основне популације како се не би догодило да поново буде изабрана
- Понављањем се добију све индивидуе за укрштање
- Смањује се вероватноћа брзог губитка генетске разноликости
- Има мању сложеност и рачунарске захтеве од пропорционалне селекције што га чини погоднијим за имплементацију

Оператор селекције: закључци

- Селекција се врши на основу оптимизационе функције (способности)
- Избором **стратегије селекције** се **прави компромис** између брзине конвергенције алгоритма и вероватноће проналажења локалног уместо глобалног минимума
- Избор стратегије селекције **не зависи од записа** оптимизационог проблема
- Све описане стратегије селекције могу се применити на било који оптимизациони проблем (SAT, TSP, NLP)
- Формално, било која операција која издваја подскуп решења из текуће генерације је **оператор селекције**
  - Селекција на случајан начин: ГА = случајно претраживање

2. Оператор укрштања

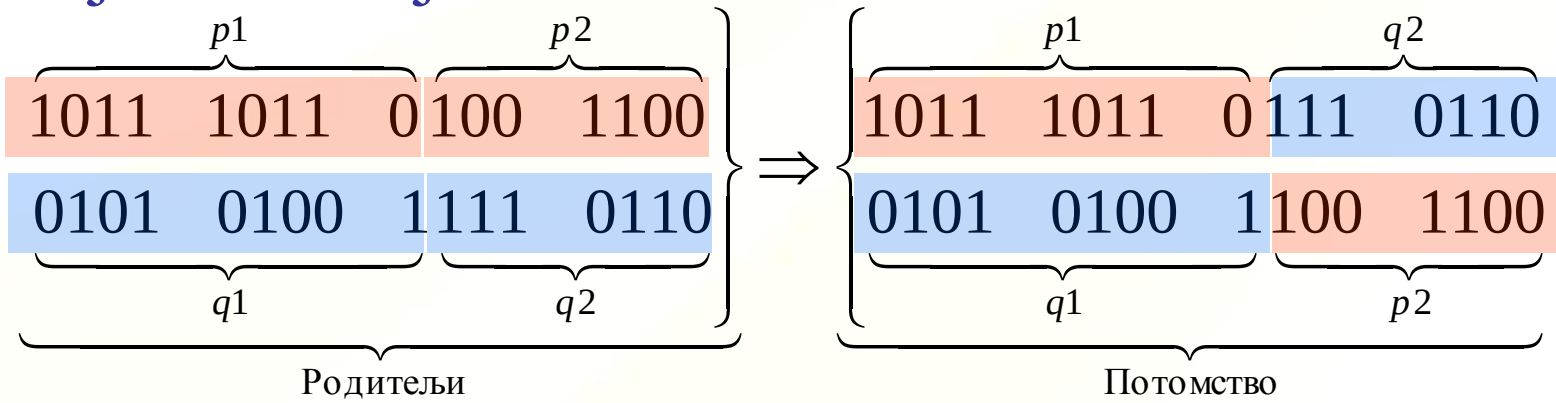
- Избором два или више решења из скупа најбољих формира се група решења за укрштање
- Њиховим укрштањем формирају се нова решења
- Укрштање се може реализовати
  - стохастички или
    - Вероватноћа укрштања је вероватноћа са којом се изабрана група индивидуа укршта и типично износи од 0,6 до 0,9
  - детерминистички
- Конкретна реализација оператора укрштања **зависи од записа проблема**

Бинарно укрштање (SAT проблеми)

- Гени су у овом случају бити $x_k = \{0,1\}$
- Индивидуе (решења) су секвенце бита (низ карактера поређаних један за другим)
  $\mathbf{x} = (0,1,1,0,1,0,0,1)$
- Нека постоје два селектована решења за укрштање
- На случајан начин се изабере тачка (бит) укрштања у којој се пресеку низови оба родитељска решења
- Надовезивањем добијених поднизова добијају се два нова решења “потомка”
- На први подниз првог родитеља надовеже се други подниз другог родитеља и обрнуто

Пример бинарног укрштања

- Укрштање два решења
  која се састоје од по 16 бита



- Описани поступак назива се укрштање са једном тачком пресека (one-point crossover)
- Могућа је модификација тако да се прекидање генетског низа врши у две, три или више тачака

Континуално укрштање (NLP проблеми)

- Континуална представа гена (реалне промењљиве, $x_k \in \mathbb{R}$)
- Укрштање се најчешће врши линеарном комбинацијом вектора који представљају родитеље ($\mathbf{r}_1$ и $\mathbf{r}_2$), параметар $0 < \alpha < 1$

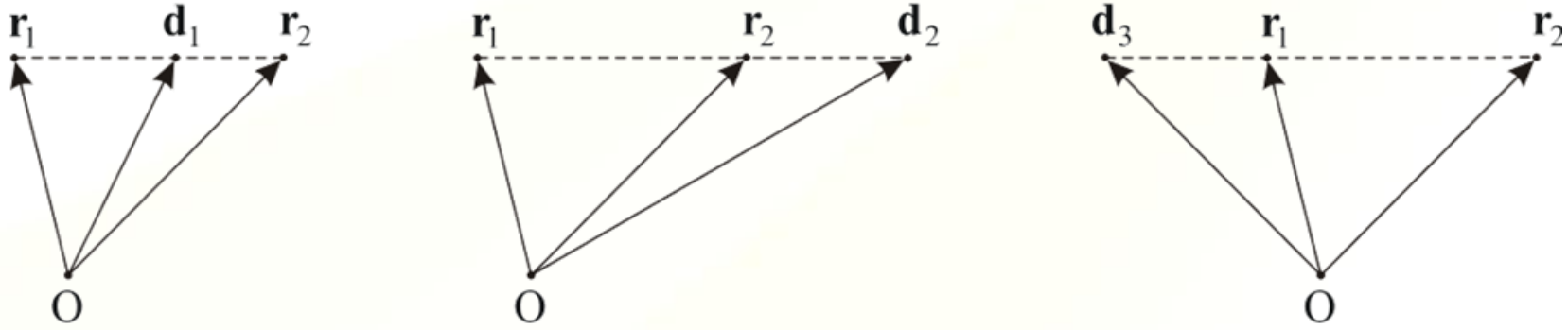
$$\mathbf{d}_1 = \mathbf{r}_2 + \alpha \cdot (\mathbf{r}_1 - \mathbf{r}_2) = \alpha \cdot \mathbf{r}_1 + (1-\alpha) \cdot \mathbf{r}_2,$$

$$\mathbf{d}_2 = \mathbf{r}_2 - \alpha \cdot (\mathbf{r}_1 - \mathbf{r}_2) = -\alpha \cdot \mathbf{r}_1 + (1+\alpha) \cdot \mathbf{r}_2,$$

$$\mathbf{d}_3 = \mathbf{r}_1 + \alpha \cdot (\mathbf{r}_1 - \mathbf{r}_2) = (1+\alpha) \cdot \mathbf{r}_1 - \alpha \cdot \mathbf{r}_2,$$

Пример континуалног укрштања

- Пример континуалног укрштања (два родитеља $\mathbf{r}_1$, $\mathbf{r}_2$, три потомка $\mathbf{d}_1$,$\mathbf{d}_2$,$\mathbf{d}_3$)



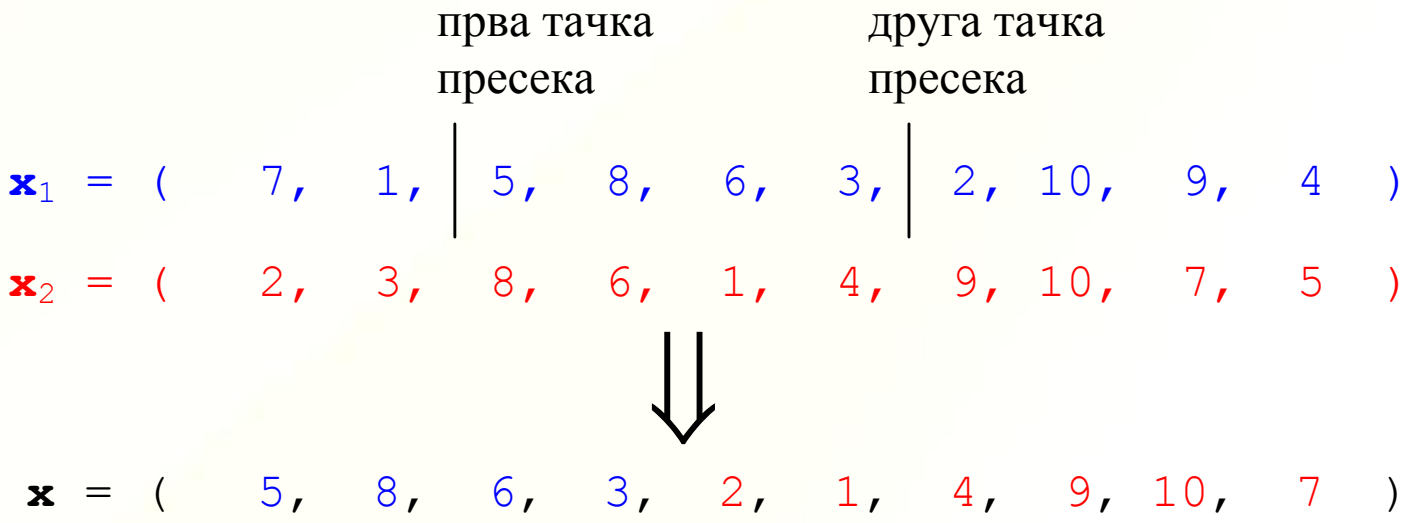
- У литератури су предложене различите реализације оператора укрштања за NLP
- Другачија укрштања нису драстично побољшала конвергенцију алгоритма на већем броју оптимизационих примера

Укрштање за TSP проблеме

- Запис TSP проблема је низ пермутованих елемената
- Два решења $\mathbf{x}_1$={1,3,2,4,5} $\mathbf{x}_2$={3,2,5,4,1}
- Потребна је креативност за укрштање две пермутације
- Најједноставније је користити једног родитеља и заменити места неколико гена
- Замена места гена *k* парова
- Може се увести вероватноћа са којом се замена места извршава

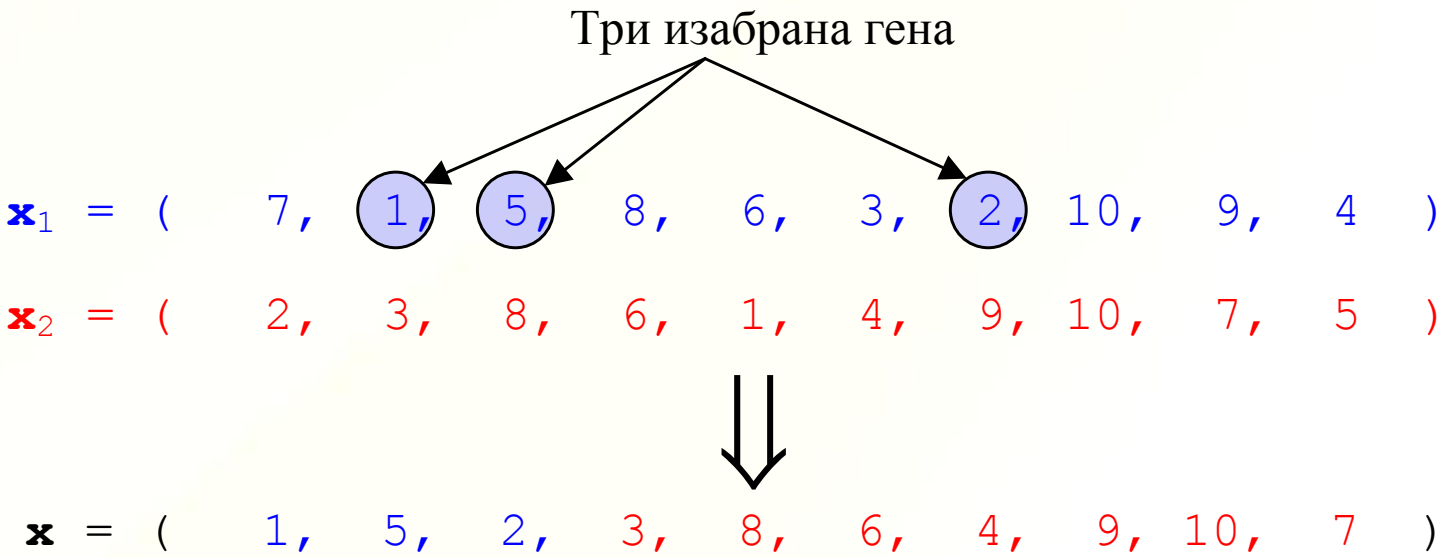
Илустрација укрштања за TSP проблеме, два пресека

- Запис са 10 променљивих
- Изаберу се две тачке пресека првог родитеља
- Гени између тачка пресека препишу се у ново решење, а остали гени се препишу из другог



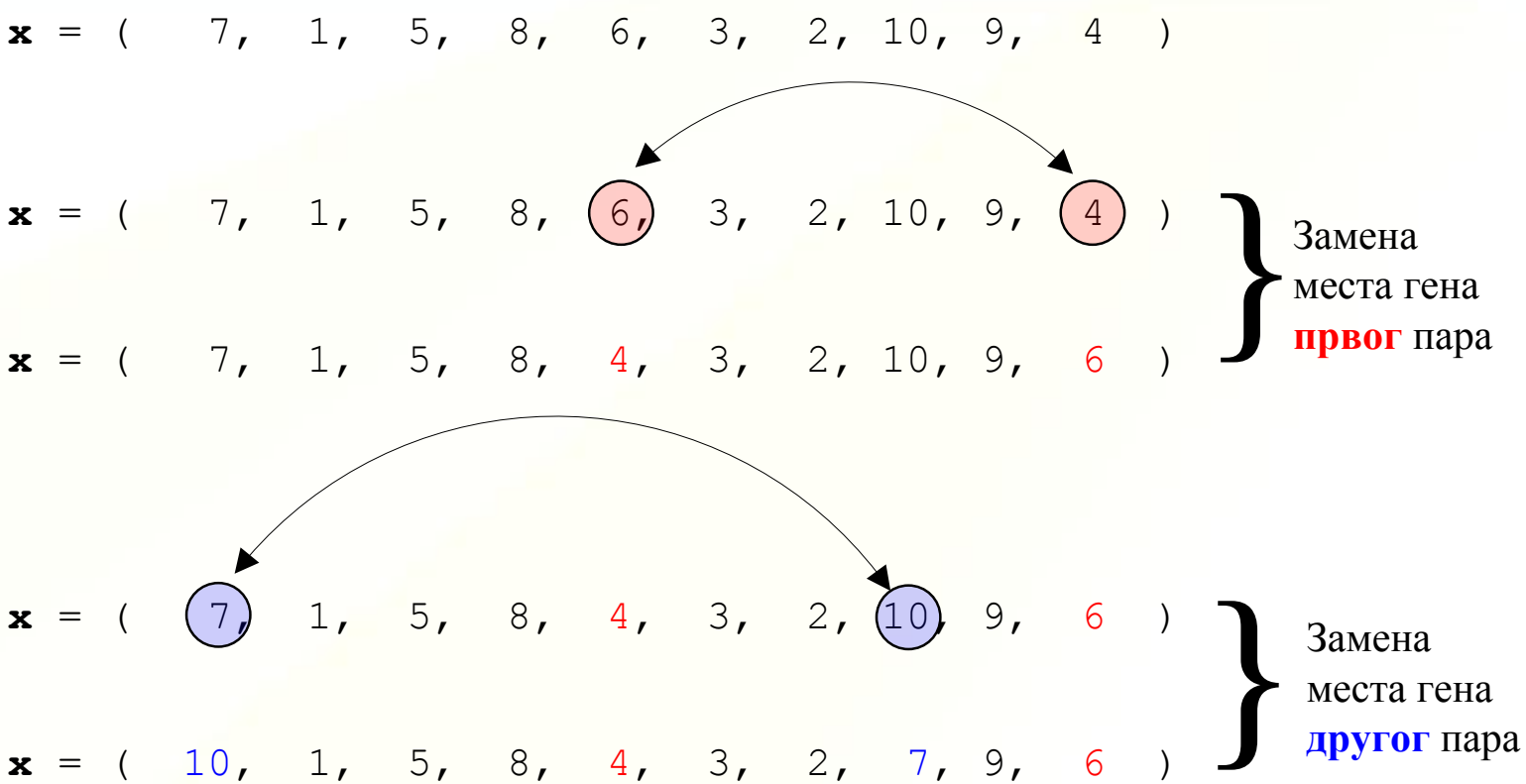
Илустрација укрштања за TSP проблеме, *k* гена

- Запис са $D = 10$ променљивих, избор $k = 3$ гена
- Изабре се k гена из првог родитеља и они се препишу у ново решење
- Осталих $D–k$ гена се препише из другог родитеља



Илустрација укрштања за TSP проблеме, *k* замена

- Запис са 10 промењљивих, $k = 2$ (две замене)



Оператор укрштања: закључци

- Укрштањем се формирају
  нова решења оптимизационог проблема (нове индивидуе)
  полазећи од селектованих решења
- Укрштање се понавља онолико пута колико је потребно да се формирају сва решења (индивидуе) за наредну генерацију
- Избор начина укрштања
  зависи од записа оптимизационог проблема
- За сваки тип записа (SAP, TSP, NLP) постоје посебне реализације укрштања
- Формално, било која операција која, на основу једног или више селектованих решења, формира нова решења је **оператор укрштања**

3. Мутација

- Реализује се искључиво стохастички
- Са унапред задатом вероватноћом изаберу се гени унутар новоформиране генерације који се на случајан начин промене
- Мутацијом се додаје нови генетски материјал у следећу генерацију (уноси се нова информација о оптимизационом простору)
- Мутација представља могућност да се истраже неистражени делови оптимизационог простора
- Вредност за вероватноћу мутације је обично мала, типично од 0,01 до 0,15
- Граничне вредности вероватноће мутације:
  →1: ГА = случајно претраживање
  →0: ГА = локално претраживање

Мутација: SAT, NLP, TSP

- SAT: своди се на промену бита у генетском низу са унапред задатом вероватноћом
  - оригинални низ $\mathbf{x}$ = (1,0,1,1,0,1,0,0)
  - мутирани низ $\mathbf{x}$ = (1,0,0,1,0,1,1,0)
- NLP: изводи се тако што се цео ген замени случајном вредношћу, у дозвољеном опсегу гена, са унапред задатом вероватноћом мутације
  - оригинални низ $\mathbf{x}$ = (0.81, 0.24, 0.93, -0.35)
  - мутирани низ $\mathbf{x}$ = (0.81, -0.62, 0.93, -0.35)
- TSP: изабере се подскуп гена којима се на случајан начин промене вредности (слично као укрштање са *k* замена!)
- Оператор мутације: случајна промена вредности гена једног решења

Критеријуми за завршетак

- Алгоритам се прекида онда када је решење пронађено или када се гени читаве генерације разликују за мање од неког унапред задатог прага
- Уколико се **оптимизација врши довољно дуго**, пре или касније ће се десити **мутације** које ће довести до проналажења **глобалног минимума**
- У пракси је готово увек недопустиво чекати толико дуго и алгоритам се прекида после одређеног броја генерација уколико критеријуми за завршетак нису пре тога достигнути
- Експериментално је утврђено да после ~50 генерација алгоритам конвергира за популације до 200 000 индивидуа

Параметри ГА

- Величина популације (N_{pop})
- Број генерација (N_{gen})
  - Укупан број итерација (израчунавања опт. функције) је $N_{pop} * N_{gen}$
- Запис промењљивих
- Начин реализације сваког оператора (селекција, укрштање и мутација)
- Избор вероватноћа код стохастичких оператора

Који су оптимални параметри ГА?

- Величина популације треба да буде већа од броја оптимизационих променљивих
- Што већа популација то је боље решење на крају, али већа популација значи оптимизацију са више итерација
- Број генерација 30-100 довољан је за релативно мале популације (до 200 000)
  - за веће популације и даље је ?
- Кодовање:
  бинарно или континуално у зависности од проблема
- Вероватноћа укрштања: 0,8
- Вероватноћа мутације: 0,1
- Број индивидуа које се укрштају ~популација / 5

Пример оптимизације са ГА

Имплементација ГА

- MATLAB: genetic algorithm (Global Optimization Toolbox)
- Mathematica, Python, C/C++
  - постоје само “незванични” кодови
- Алгоритам је сложен и имплементације имају пуно степени слободе
- Тешко је користити ГА имплементацију коју добро не познајете, због великог броја степени слободе

Варијације ГА

- ГА са више популација које коегзистирају заједно са разменом од неколико индивидуа у свакој генерацији (енг: multipopulation GA)
- Микро ГА: вишеструко понављање ГА са релативно малим популацијама
- Коришћење локалних оптимизационих алгоритама у спрези са ГА (вишестепене оптимизације)
- ГА код кога су параметри алгоритма придружени оптимизационим променљивима $\mathbf{x}_{tot} = (\mathbf{x}_{opt}, \mathbf{x}_{GA})$
  - параметри ГА су променљиве које се мењају током оптимизације
  - ГА сам подешава своје параметре (могуће осцилације и дивергенција!)

Закључци о ГА

- Група решења (популација) уместо једног решења
- Добре стране:
  - Проналази глобално решење
    (после довољно дуго времена појавиће се
    мутација која алгоритам пребацује у околину глобалног
    минимума)
  - Алгоритам се изузетно лако паралелизује
    (прилагођен архитектури модерних рачунара)
  - Што већи број оптимизационих варијабли то је алгоритам ефикаснији у поређењу са другим алгоритмима
- Лоше стране:
  - Алгоритам има доста параметара
  - Релативно комплексан за имплементацију
  - Споро конвергира ка решењу
  - Захтева пуно израчунавања оптимизационе функције

Генерализација ГА: Evolutionary Algorithm (EA)

- EA = Genetic algorithm, Genetic programming, evolution strategies, evolutionary programming…
- Марковљев ланац (енг: Markov chain)
  будуће стање (стохастичког) система
  зависи само од текућег (садашњег) стања, а
  не зависи од претходних стања
- Наредна генерација решења
  добија се само на основу претходне,
  стога ГА/ЕА представља Марковљев ланац

Генерални опис EA

- Генерално $\mathbf{x}[n+1] = s(v(\mathbf{x}[n]))$
- $\mathbf{x}[n]$ је генерација n,
  тј. скуп свих индивидуа у n-тој популацији
- $\mathbf{x}[0]$ је почетна популација (нулта генерација)
- $v(\,)$ је **стохастички** оператор варијације (промене)
- $s(\,)$ је оператор селекције
- Могуће је изоставити $s()$
  није могуће изоставити $v()$

Стандардни запис ЕА

- (μ, λ)-ЕА
  - μ је број селектованих решења (родитеља)
  - λ је број нових решења која се генеришу (деца)
    $\lambda \geq \mu$
  - из скупа од λ нових решења селекцијом се бира μ
    (претходна популација се не памти)
- $(\mu+\lambda)$-ЕА
  - λ решења се генерише на основу μ решења
  - сви се заједно пореде
  - из скупа од $\mu+\lambda$ решења селекцијом се бира μ
    (претходна популација се памти и пореди са наредном)

Примери ЕА: претходни алгоритми

- (μ+λ)-ЕА уз $N_{pop} = \mu + \lambda$ даје ГА са елитизмом
  - оператор варијације = оператор (ГА) укрштања + оператор (ГА) мутације
  - оператор селекције је исти за ЕА и ГА
- (μ, λ)-ЕА уз услов $N_{pop} = \lambda$, $\mu < \lambda$ је ГА без елитизма
  - оператор варијације = оператор (ГА) укрштања + оператор (ГА) мутације
  - оператор селекције је исти за ЕА и ГА
- (1,1)-ЕА је генерализовано симулирано каљење
  - оператор варијације је избор наредне тачке, као што је дефинисано за симулирано каљење
  - оператор селекције је начин прихватања (или одбацивања) наредне тачке, као што је дефинисано за симулирано каљење

Примери ЕА: могуће имплементације

- (1, λ)-ЕА
  - на основу једног решења генерише се λ решења у којима се израчунава *f*, а затим се бира једно најбоље решење и понавља се процес
  - нових λ решења се генерише на случајан начин са Гаусовом расподелом око основног решења (селектованог из прошле генерације)
- Једноставне имплементације ЕА су могуће
  - Избор оператора селекције и варијације је потпуно слободан
- Резултати показују да су једноставне ЕА имплементације сличних перформанси као и ГА (који је сложенији)

Закључци о ЕА

- ЕА је “швајцарски ножић” међу оптимизационим алгоритмима
- Може да се примени на СВЕ оптимизационе проблеме
- Избором оператора
  - Могу да се добију и локални и глобални алгоритми
  - Може да се мења брзина конвергенције алгоритма
- Избор параметара алгоритма
  - Величина популације
  - Број генерација
  - Оператори варијације и селекције
- Могућности имплементација су практично неограничене

Задатак за вежбе

- На процесору који се састоји од 4 језгра потребно је извршити 100 процеса. Времена трајања процеса, у μs, записана су у низу ***t***.

t=(227, 316, 797, 676, 391, 332, 598, 186, 672, 941, 248, 948, 667, 95, 441, 886, 697, 326, 733, 220, 81, 159, 340, 465, 266, 815, 193, 129, 91, 598, 854, 601, 931, 724, 860, 929, 546, 937, 494, 273, 451, 665, 330, 903, 257, 339, 258, 355, 5, 628, 282, 68, 616, 176, 304, 440, 150, 217, 474, 476, 255, 297, 279, 260, 482, 211, 495, 246, 838, 180, 862, 178, 750, 611, 209, 759, 249, 85, 618, 536, 634, 174, 248, 684, 80, 875, 428, 618, 313, 178, 9, 210, 870, 972, 441, 378, 275, 966, 58, 408)

- Потребно је распоредити процесе на језгра, тако да за извршавање свих процеса буде потребно најкраће време.

Задатак за вежбе (поступак решавања)

- Усвојени запис решења овог проблема је $\mathbf{x}=(x_1,x_2,...x_D)$, $x_k \in \{1,2,3,4\}$, $k=1,2,...D$ (1: процес се извршава на првом језгру, 2: процес се извршава на другом језгру, 3: процес се извршава на трећем језгру, 4: процес се извршава на четвртом језгру).
- Оптимизациона функција је: $f(\mathbf{x})=\max\{t_1,t_2,t_3,t_4\}$, где је t_{xk} укупно време извршавања процеса на језгру x_k.
- Тражи се минимум.

Имплементација

- Написати имплементацију **генетичког алгоритма**
- За величину популације узети 2000
- Максималан број итерација (израчунавања оптимизационе функције) је 100 000 за једно покретање, тј. 50 генерација
- Пустити 20 независних покретања и сачувати ток оптимизације (вредности оптимизационе функције у свакој итерацији)
- Израчунати и нацртати кумулативни минимум за свако покретање (20 кривих на једном графику) у semilogx размери
- Израчунати и нацртати средње најбоље решење на основу претходних резултата у semilogx размери
- Решење садржи
  - ASCII фајл: низ дужине 100 који одговара најбољем пронађеном решењу (записано искључиво у дефинисаном формату $\mathbf{x}=(x_1, x_2, \ldots x_D)$) и минималну вредност оптимизационе функције
  - графике у png или jpg формату
  - Код
- Рок за слање решења: **11. децембар 12.00 часова**