

# 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 развојити
- 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  $\mathbf{x} = (x_1, x_2, \dots, x_D)$
- 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

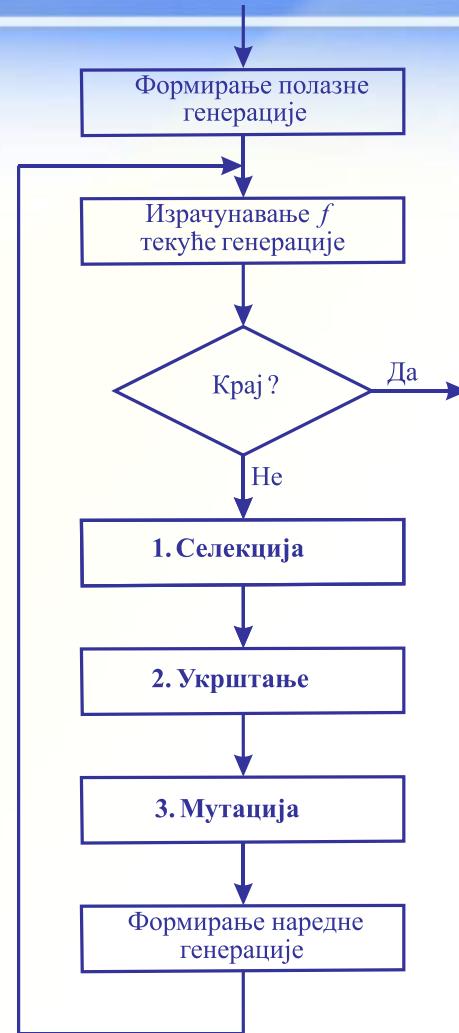
- ГА врши операције над **скупом решења** у оптимизационом простору
  - сви други претходно описани алгоритми оперишу само са једним (најбољим) решењем које “поправљају”
  - NM симплекс ради са скупом од тачно  $D+1$  решења али се једно (најбоље) решење стално прати
- Скуп индивидуа (решења) назива 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, прескацањем 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
- Тада 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.

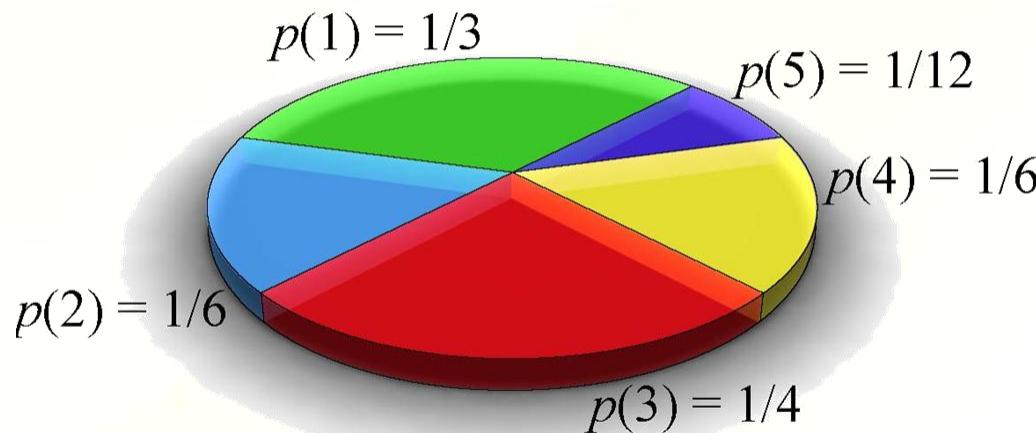
**Gandalf:** And I hope you never have to. But if you do, remember this: true courage lies in knowing not when to take a life, but when to spare one.

# 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  $p_i = \frac{f_i}{\sum_i f_i}$  избора  $p_i$  пропорционална способности индивидуе  $f_i$
- 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

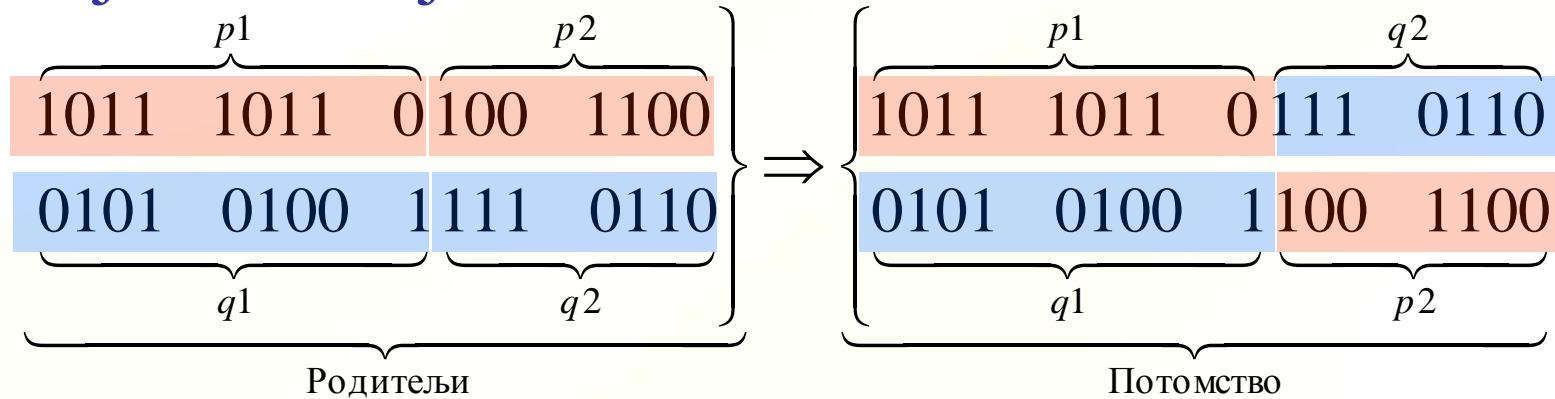
- Избором два или више решења из скупа најбољих формира се група решења за укрштање
- Њиховим укрштањем формирају се нова решења
- Укрштање се може реализовати
  - стохастички или
    - Вероватноћа укрштања је вероватноћа са којом се изабрана група индивидуа укршта и типично износи од 0,6 до 0,9
    - 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)  
 $x = (0,1,1,0,1,0,0,1)$
- 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

# Континално укрштање (NLP проблеми)

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

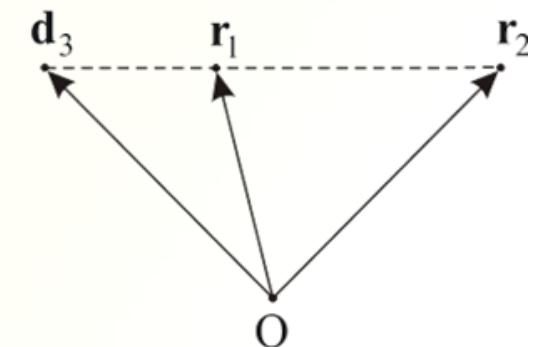
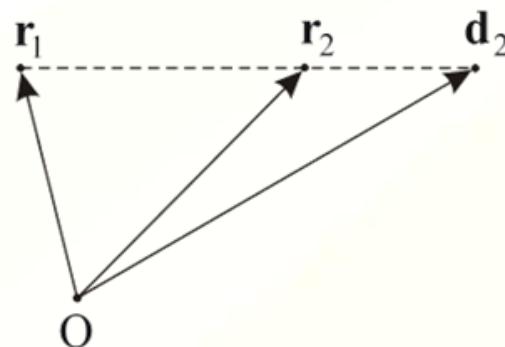
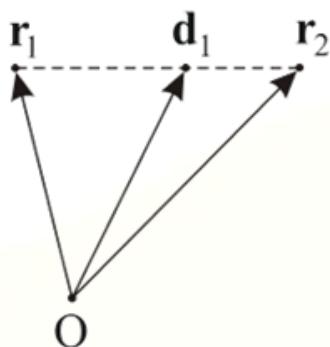
$$\mathbf{d}_1 = \mathbf{r}_2 + \alpha \cdot (\mathbf{r}_1 - \mathbf{r}_2) = \alpha \cdot \mathbf{r}_1 + (1 - \alpha) \cdot \mathbf{r}_2,$$

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

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

# Пример континуалног укрштања

- Пример континуалног укрштања  
(два родитеља  $r_1, r_2$ , три потомка  $d_1, d_2, d_3$ )



- 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

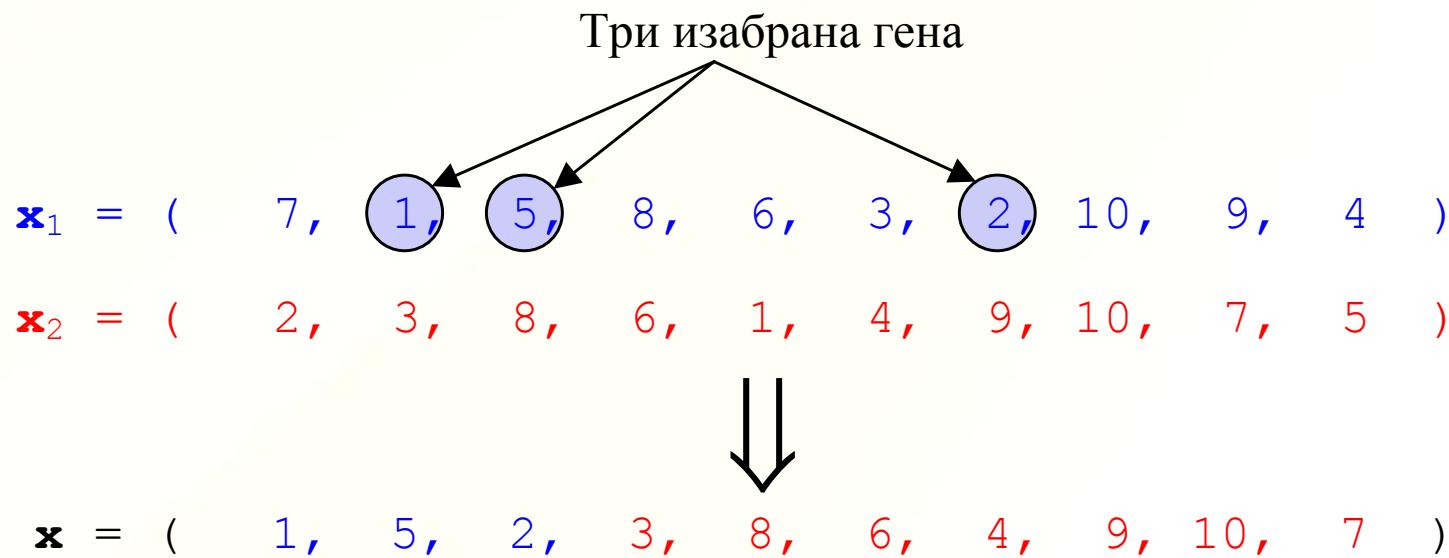
# Илустрација укрштања за TSP проблеме, два пресека

- 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

$$\begin{array}{c} \text{прва тачка} \\ \text{пресека} \\ \text{друга тачка} \\ \text{пресека} \\ \hline \mathbf{x}_1 = ( \quad 7, \quad 1, \quad | \quad 5, \quad 8, \quad 6, \quad 3, \quad | \quad 2, \quad 10, \quad 9, \quad 4 \quad ) \\ \mathbf{x}_2 = ( \quad 2, \quad 3, \quad 8, \quad 6, \quad 1, \quad 4, \quad 9, \quad 10, \quad 7, \quad 5 \quad ) \\ \downarrow \\ \mathbf{x} = ( \quad 5, \quad 8, \quad 6, \quad 3, \quad 2, \quad 1, \quad 4, \quad 9, \quad 10, \quad 7 \quad ) \end{array}$$

# 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 променљивих,  $k = 2$  (две замене)

$$\mathbf{x} = (7, 1, 5, 8, 6, 3, 2, 10, 9, 4)$$

$$\mathbf{x} = (7, 1, 5, 8, \textcolor{red}{6}, 3, 2, 10, 9, \textcolor{red}{4})$$

$$\mathbf{x} = (7, 1, 5, 8, \textcolor{red}{4}, 3, 2, 10, 9, \textcolor{red}{6})$$

} Замена  
места гена  
**првог** паре

$$\mathbf{x} = (\textcolor{blue}{7}, 1, 5, 8, \textcolor{red}{4}, 3, 2, \textcolor{blue}{10}, 9, \textcolor{red}{6})$$

$$\mathbf{x} = (\textcolor{blue}{10}, 1, 5, 8, \textcolor{red}{4}, 3, 2, \textcolor{blue}{7}, 9, \textcolor{red}{6})$$

} Замена  
места гена  
**другог** паре

# 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
- Границе 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  $\mathbf{x} = (1, 0, 0, 1, 0, 1, \textcolor{red}{1}, 0)$
- 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  $\mathbf{x} = (0.81, \textcolor{red}{-0.62}, 0.93, -0.35)$
- 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 дugo**, пре или касније ће се десити мутације које ће довести до проналажења **глобалног минимума**
- У пракси је готово увек недопустиво чекати толико дugo и алгоритам се прекида после одређеног броја генерација уколико критеријуми за завршетак нису пре тога достигнути
- Експериментално је утврђено да после  $\sim 50$  генерација алгоритам конвергира за популације до 200 000 индивидуа

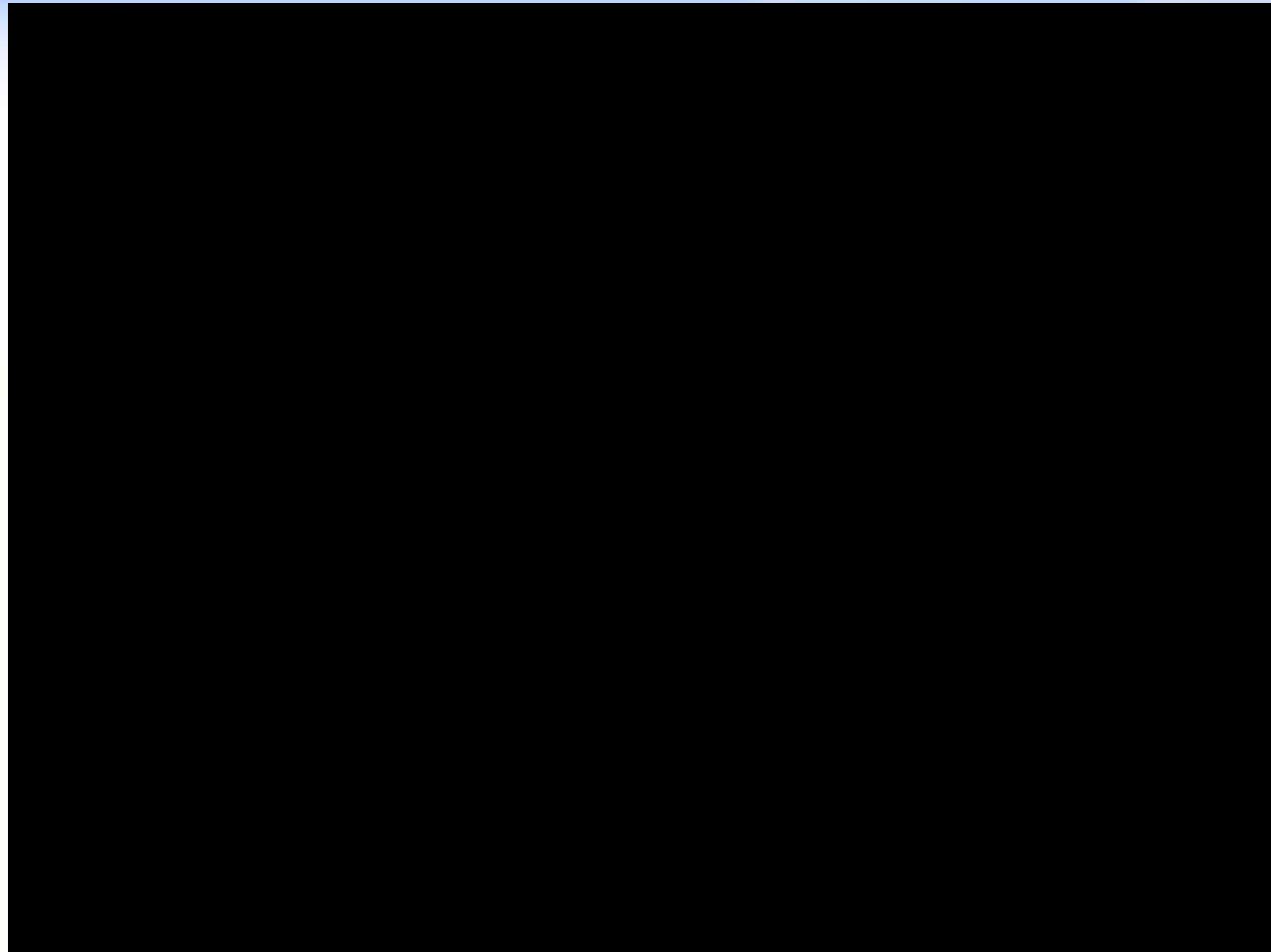
# Параметри ГА

- Величина популације ( $N_{\text{pop}}$ )
- Број генерација ( $N_{\text{gen}}$ )
  - Укупан број итерација (израчунавања опт. функције) је  $N_{\text{pop}} * N_{\text{gen}}$
- Запис променљивих
- 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 придржени оптимизационим променљивима  $\mathbf{x}_{\text{tot}} = (\mathbf{x}_{\text{opt}}, \mathbf{x}_{\text{GA}})$ 
  - 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

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

# Примери ЕА: претходни алгоритми

- $(\mu+\lambda)$ -ЕА уз  $N_{\text{pop}} = \mu + \lambda$  даје ГА са елитизмом
  - оператор варијације = оператор (ГА) укрштања + оператор (ГА) мутације
  - оператор селекције је исти за ЕА и ГА
- $(\mu, \lambda)$ -ЕА уз услов  $N_{\text{pop}} = \lambda$ ,  $\mu < \lambda$  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

- $(1, \lambda)$ -ЕА
  - на основу једног решења генерише се  $\lambda$  решења у којима се израчунава  $f$ , а затим се бира једно најбоље решење и понавља се процес
  - нових  $\lambda$  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

- Дат је скуп од  $D = 64$  фајла. Величине фајлова у [В] су редом  
 $\mathbf{s} = (173669, 275487, 1197613, 1549805, 502334, 217684, 1796841, 274708, 631252, 148665, 150254, 4784408, 344759, 440109, 4198037, 329673, 28602, 144173, 1461469, 187895, 369313, 959307, 1482335, 2772513, 1313997, 254845, 486167, 2667146, 264004, 297223, 94694, 1757457, 576203, 8577828, 498382, 8478177, 123575, 4062389, 3001419, 196884, 617991, 421056, 3017627, 131936, 1152730, 2676649, 656678, 4519834, 201919, 56080, 2142553, 326263, 8172117, 2304253, 4761871, 205387, 6148422, 414559, 2893305, 2158562, 465972, 304078, 1841018, 1915571);$
- Потребно је што боље искористити меморију величине  $64 \text{ MiB} = 2^{26} \text{ B}$ , за складиштење ових фајлова
- Усвојени запис решења овог проблема је  $\mathbf{x} = (x_1, x_2, \dots, x_D)$ ,  $x_k \in \{0,1\}$ ,  $k = 1, 2, \dots, D$  (0: фајл се не складишти, 1: фајл се складишти у датој меморији)
- Оптимизациона функција је:  $f(\mathbf{x}) = \begin{cases} F, & F \geq 0 \\ 2^{26}, & F < 0 \end{cases}$ , где је  $F = 2^{26} - \sum_{k=1}^D x_k s_k$
- Тражи се минимум (проблем SAT класе)

# 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)
- Израчунати и нацртати средње најбоље решење на основу претходних резултата
- Оба графика представити у log-log размери
- ASCII фајл уз решење би требало да садржи низ 64 бита који одговара најбољем пронађеном решењу и минималну вредност оптимизационе функције (решење које задовољава услов  $f(\mathbf{x}) \leq 32$  је довољно добро)