

# Диференцијална еволуција

- Storn, R. (1996) "On the usage of differential evolution for function optimization," *Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS)*. pp. 519–523.
- R. Storn, K. Price, (1997) "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization* 11: 341–359.

# Елементи диференцијалне еволуције

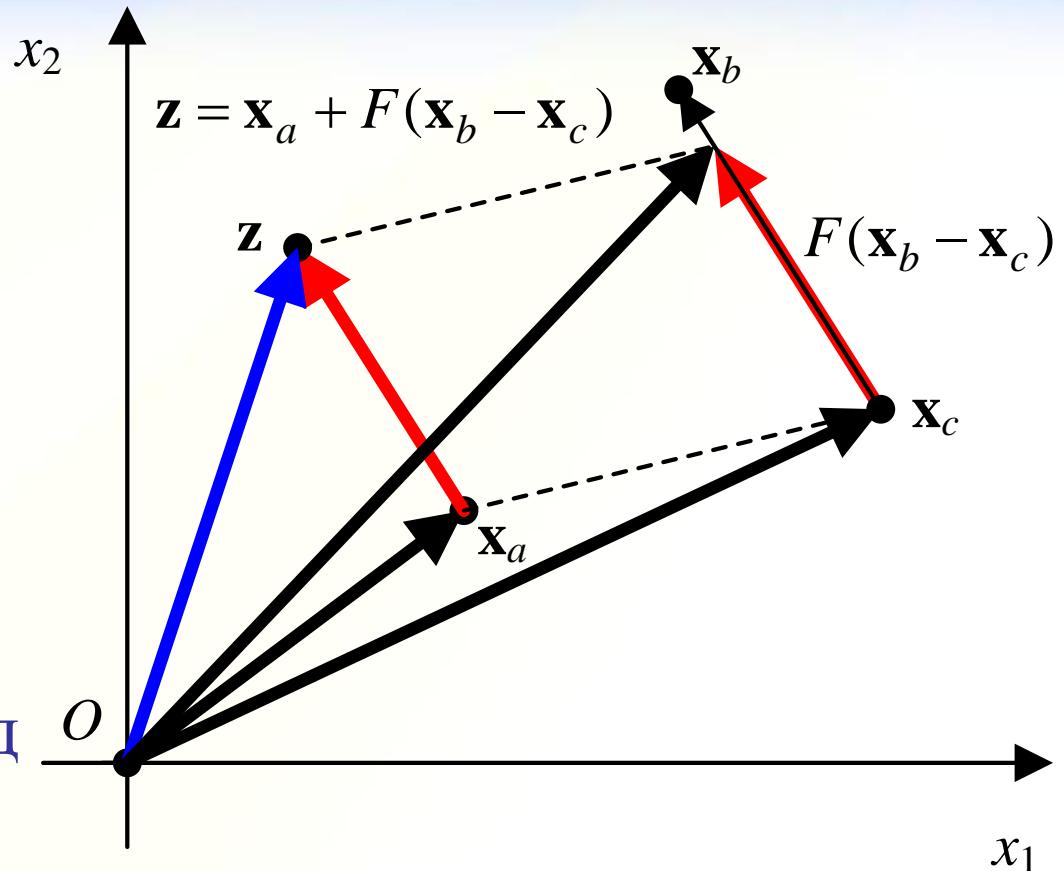
- У суштини је једна имплементација ГА/ЕА
  - ГА: селекција и укрштање (нема мутације!)
  - ЕА: селекција и варијација
- Алгоритам ради са скупом решења (популацијом)
- Почетни корак је иницијализација полазне популације
  - типично на случајан начин
  - израчунавање описне функције за полазну популацију
- У основној варијанти све оптимизационе променљиве су РЕАЛНИ бројеви (прилагођен за NLP проблеме)

# Селекција

- За свако решење  $\mathbf{x}$  из текуће популације изаберу се три различита вектора ( $\mathbf{x}_a$ ,  $\mathbf{x}_b$  и  $\mathbf{x}_c$ )
  - популација мора да има 4 или више елемента
- $\mathbf{x}_a$ ,  $\mathbf{x}_b$  и  $\mathbf{x}_c$  морају бити различити међусобно и различити од  $\mathbf{x}$
- Дефинишу се параметри
  - Диференцијални тежински фактор  $F$  из интервала од 0 до 2
  - Вероватноћа укрштања CR из интервала од 0 до 1

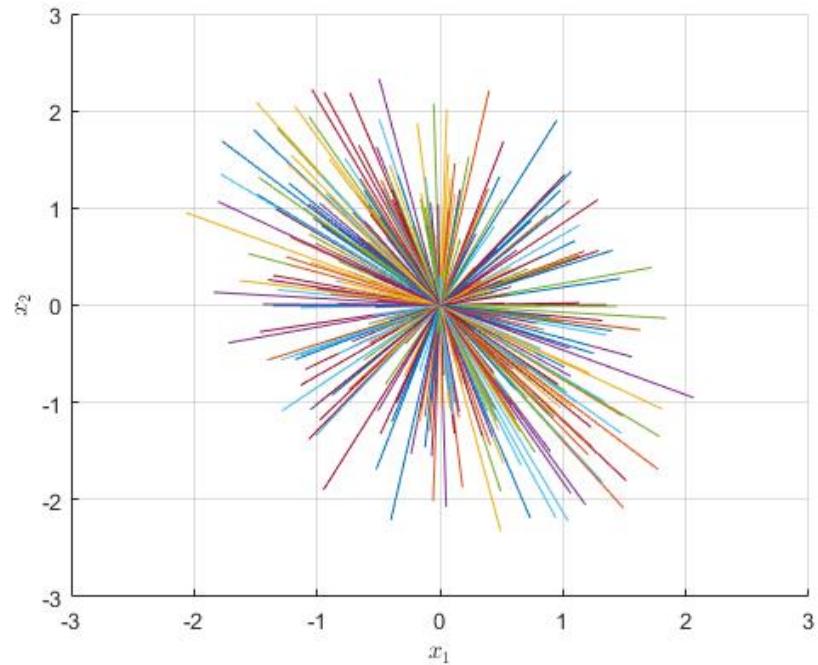
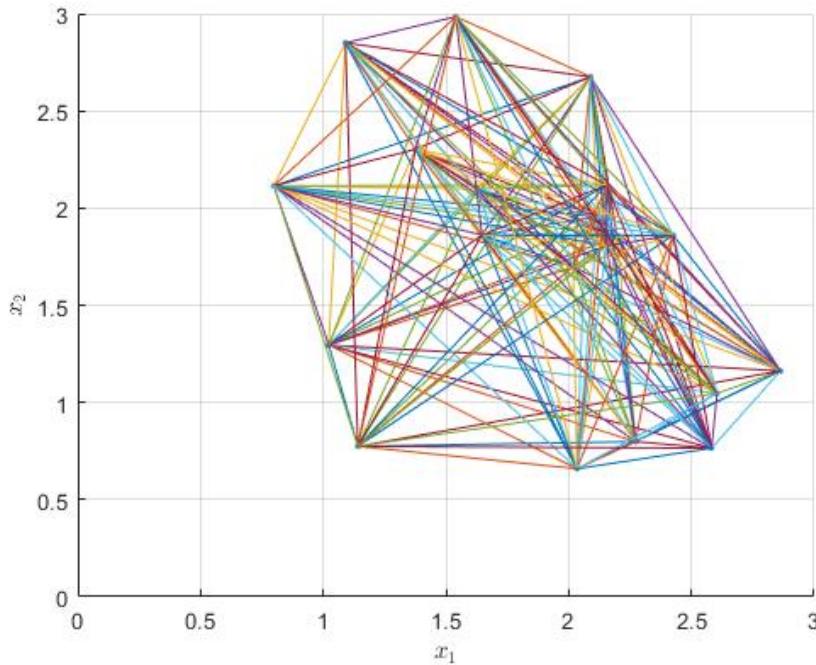
# Међурешење

- Полазећи од изабраних  $\mathbf{x}_a$ ,  $\mathbf{x}_b$  и  $\mathbf{x}_c$
- Формира се међурешење  $\mathbf{z} = \mathbf{x}_a + F^*(\mathbf{x}_b - \mathbf{x}_c)$
- Како је  $\mathbf{x}_b \neq \mathbf{x}_c$  нови вектор  $\mathbf{z}$  је сигурно различит од  $\mathbf{x}_a$ ,  $\mathbf{x}_b$  и  $\mathbf{x}_c$



# Расподела вектора разлика

- Слика лево: 20 тачака и сви вектори разлика
- Слика десно: сви вектори разлика у координатном почетку(симетрија)



# Укрштање (или оператор варијација)

- Изабере се случајан цео број  $R$  између 1 и  $D$ , где је  $D$  број димензија оптимизационог простора
  - Смисао случајног броја  $R$  је да ће променљива под редним бројем  $R$  сигурно бити замењена
  - Тиме се обезбеђује да резултат укрштања увек буде различит од полазног решења!
- За сваки елемент вектора решења (променљиву) изабере се случајан број  $r_i$  са униформном расподелом од 0 до 1

# Укрштање: услов и илустрација

- Нека је  $\mathbf{y} = (y_1, y_2, \dots, y_D)$  вектор новог решења
- Уколико је  
 $r_i < CR$  или  $i = R$   
 $y_i = z_i$
- У супротном  
 $y_i = x_i$
- Вектор бита приказан изнад слике је логичка вредност наведених услова (0-остаје  $x_i$ , 1-мења се као  $z_i$ )

$r_i < CR$  или  $i = R$ :

0 1 1 0 1 0 0 1

$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$

$\mathbf{z} = (z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8)$

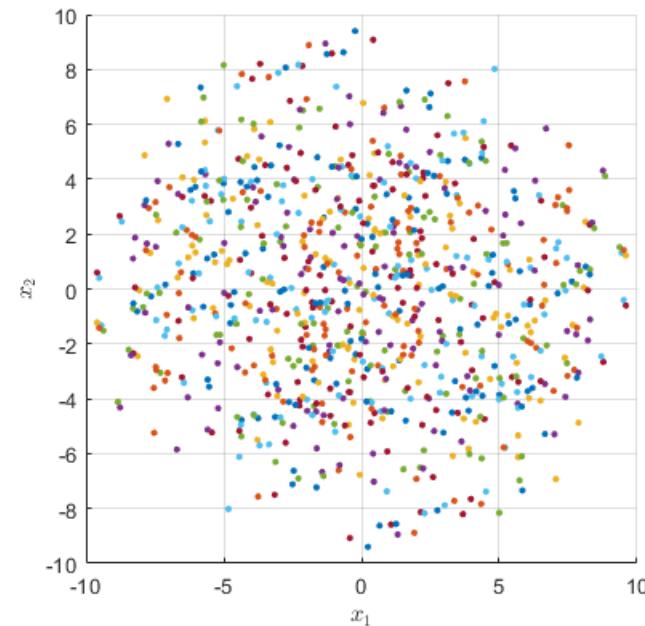
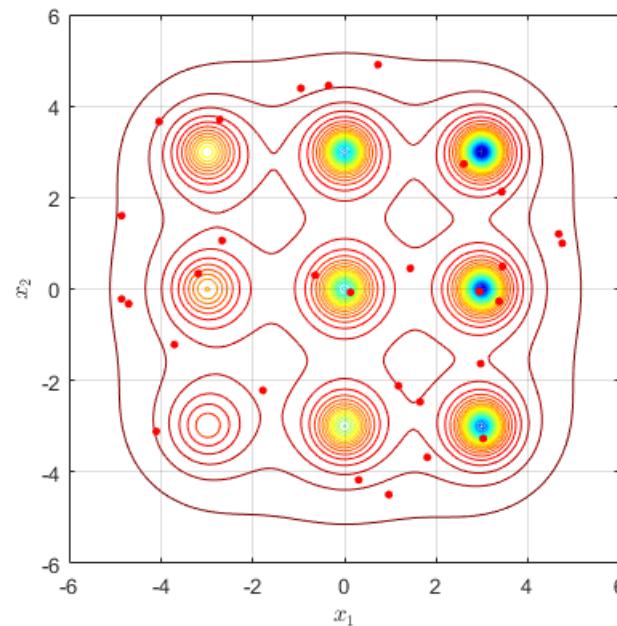
$\mathbf{y} = (x_1, z_2, z_3, x_4, z_5, x_6, x_7, z_8)$

# Услов за прихватање новог решења

- За свако решење  $\mathbf{x}_k$  из популације формира се ново решење  $\mathbf{y}_k, k = 1, 2, \dots, N_{\text{pop}}$
- У меморији се чувају две популације
  - текућа популација решења  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_{\text{pop}}}$  и
  - популација са кандидатима  $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_{N_{\text{pop}}}$
- Уколико је  $f(\mathbf{y}_k) < f(\mathbf{x}_k)$   
 $\mathbf{x}_k$  се замењује са  $\mathbf{y}_k$  у текућој популацији
- Уколико је  $f(\mathbf{y}_k) \geq f(\mathbf{x}_k)$   
 $\mathbf{x}_k$  остаје у текућој популацији

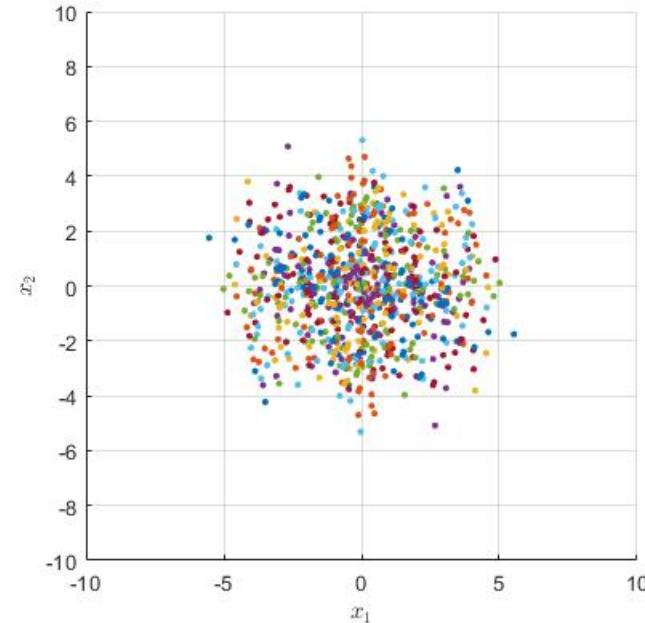
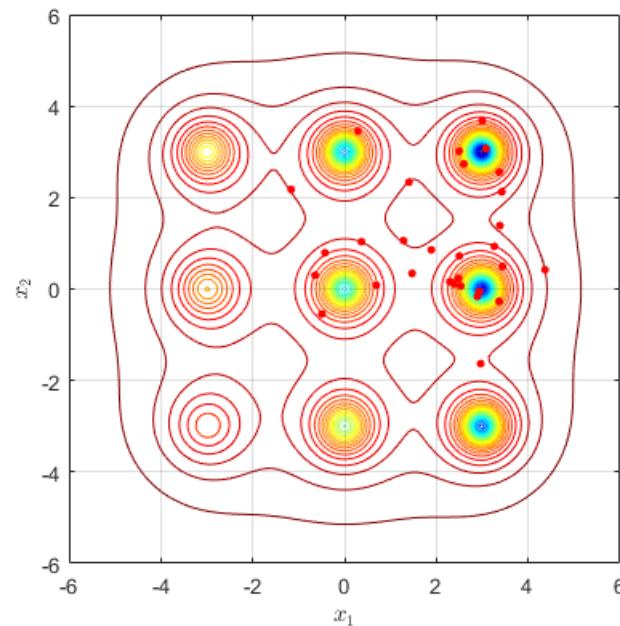
# Ток оптимизације: Шекелова функција са 9 различитих мин.

- Величина популације је 20
- Нулта генерација и сви вектори разлика транслирани у координатни почетак



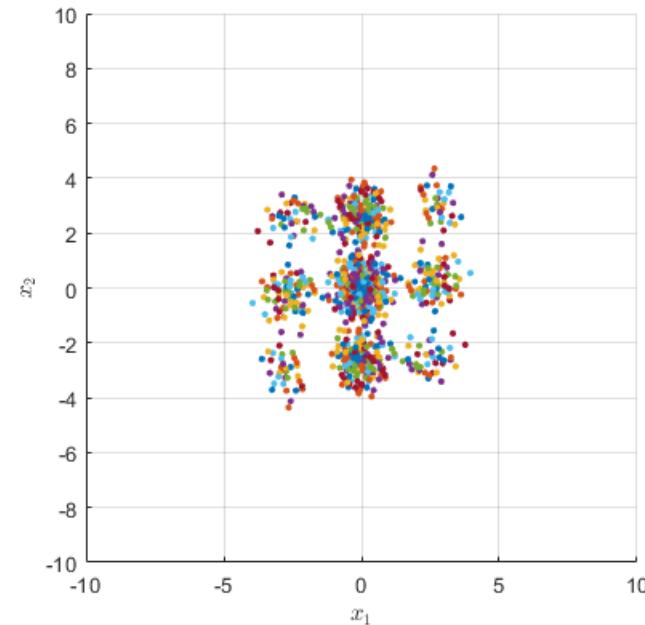
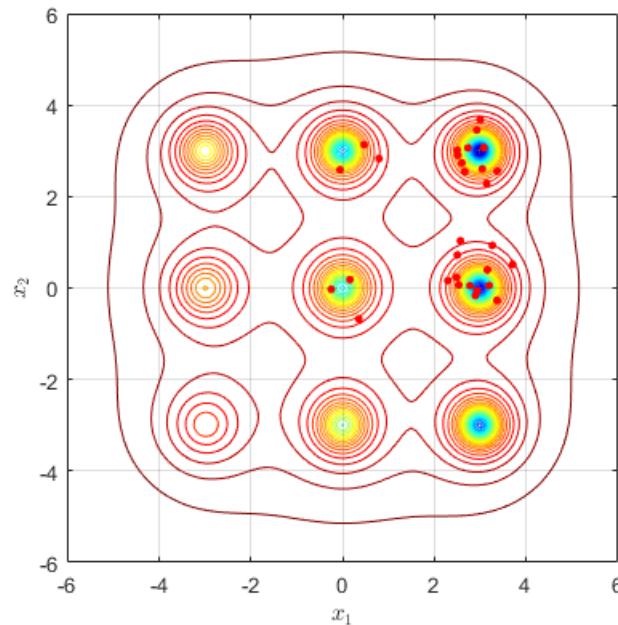
# Ток оптимизације: Шекелова функција са 9 различитих мин.

- Генерација #5 и сви вектори разлика транслирани у координатни почетак
- Вектори разлика се смањују



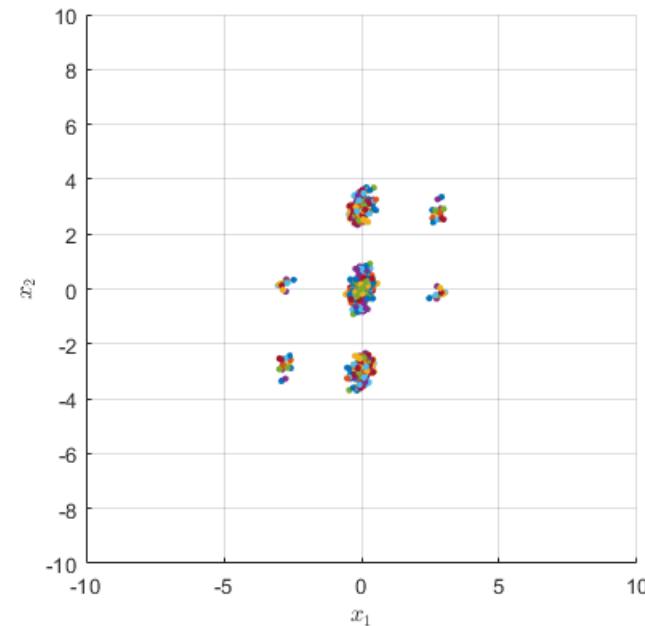
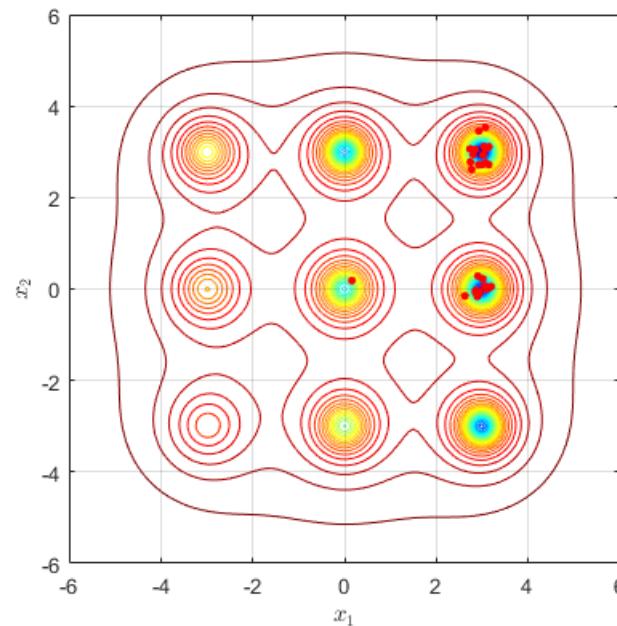
# Ток оптимизације: Шекелова функција са 9 различитих мин.

- Генерација #10 и сви вектори разлика транслирани у координатни почетак
- Груписани вектори разлика: претрага СВИХ минимума!



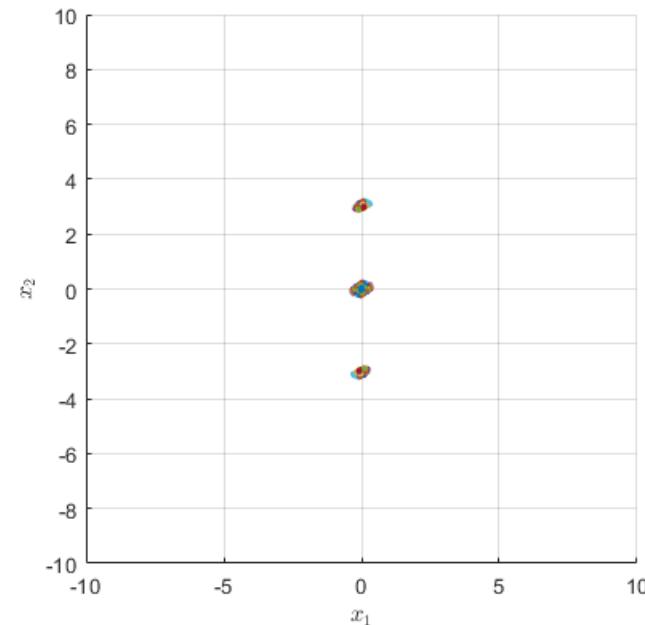
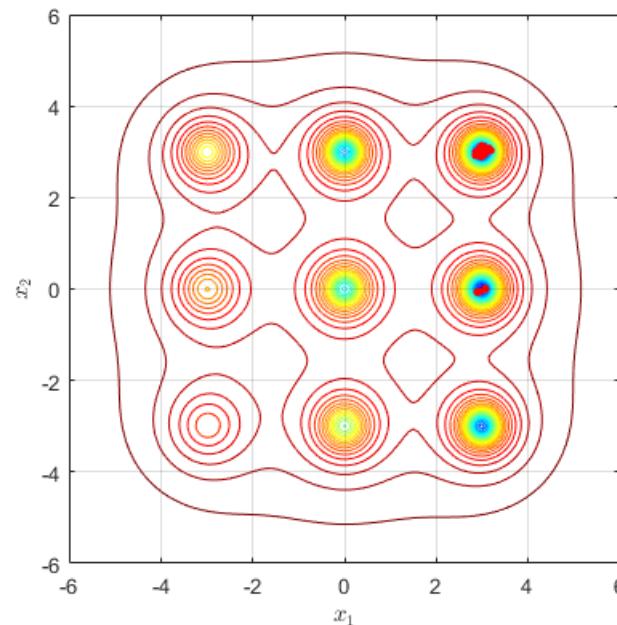
# Ток оптимизације: Шекелова функција са 9 различитих мин.

- Генерација #20 и сви вектори разлика транслирани у координатни почетак
- Смањује се могућност претраге, почиње конвергенција



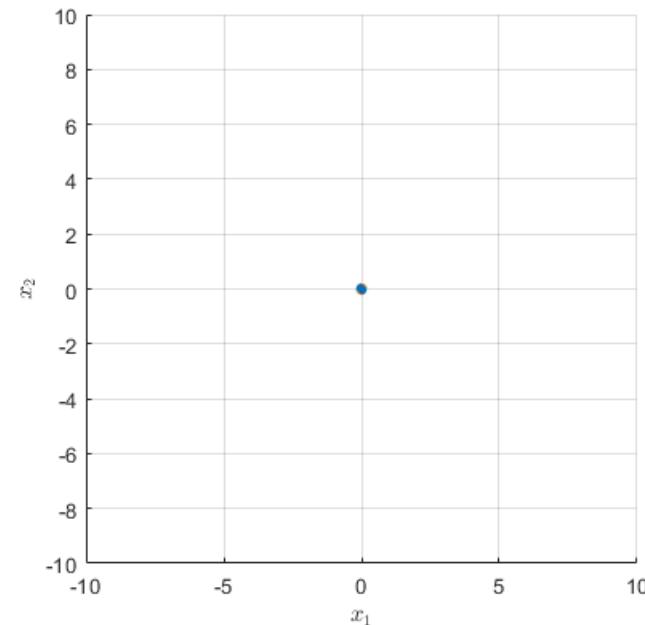
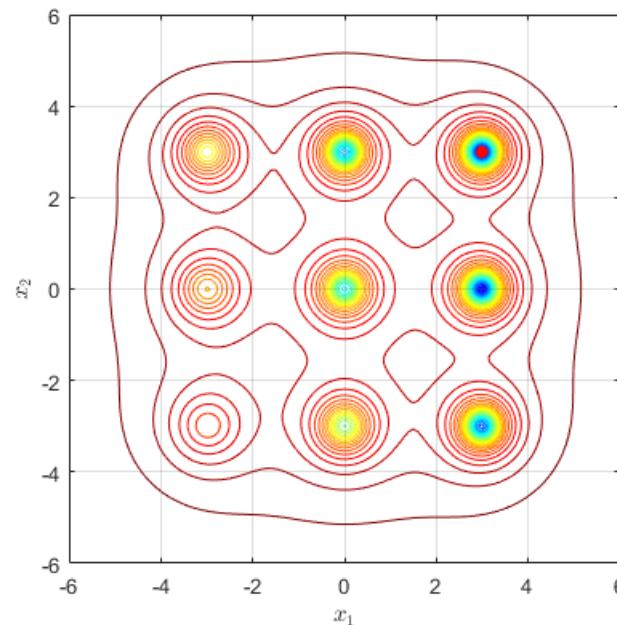
# Ток оптимизације: Шекелова функција са 9 различитих мин.

- Генерација #30 и сви вектори разлика транслирани у координатни почетак
- Претражују се само још 2 минимума, конвергенција



# Ток оптимизације: Шекелова функција са 9 различитих мин.

- Генерација #40 и сви вектори разлика транслирани у координатни почетак
- Претражују се само глобални минимум!



# Пример оптимизације са диференцијалном еволуцијом

# Означавање варијација алгоритма

- Класична диференцијална еволуција
- Стандардна ознака **DE/rand/1/bin**
  - Differential Evolution
  - random избор вектора  $\mathbf{x}_a$ ,  $\mathbf{x}_b$ ,  $\mathbf{x}_c$
  - 1 разлика
  - binomial расподела бројева који се добијају од вектора разлике
- $(\lambda+\mu)$ -EA са  $\lambda = \mu = N_{\text{pop}}$ 
  - оператор селекције:  $\mathbf{x}$ ,  $\mathbf{x}_a$ ,  $\mathbf{x}_b$ ,  $\mathbf{x}_c$
  - оператор варијације:  $\mathbf{x} \wedge \mathbf{z} = \mathbf{x}_a + F^*(\mathbf{x}_b - \mathbf{x}_c) \rightarrow \mathbf{y}$
- Постоје и другачије имплементације алгоритма диференцијалне еволуције

# Примена за SAT и TSP проблеме?

- SAT проблеми:
  - најједноставнија имплементација је тумачити бите као реалне бројеве
  - извршити израчунавње и
  - заокружити резултат за сваки бит на 0 или 1
- TSP проблеми:
  - потребно је дефинисати разлику два пермутована низа
  - добијени алгоритам више личи на ГА него на диферецијалну еволуцију
  - отворено је питање ефикасности за TSP проблеме

# Закључци о диференцијалној еволуцији

- Стандардан избор параметара
  - Популација 10 пута  $D$ , неки сматрају да је  $\sim 50$ овољно и за изузетно сложене проблеме
  - $F = 0,8$
  - $CR = 0,9$
- Диференцијална еволуција има знатно мање параметара од ГА
- Представља једну могућу имплементацију ЕА
- Ефикасност је врло слична ГА
- У пракси је изузетно добар алгоритам за решавање сложених NLP и SAT проблема са пуно променљивих
- Нема могућност проналажења глобалног оптимума после теоријски бесконачно много времена јер нема мутације

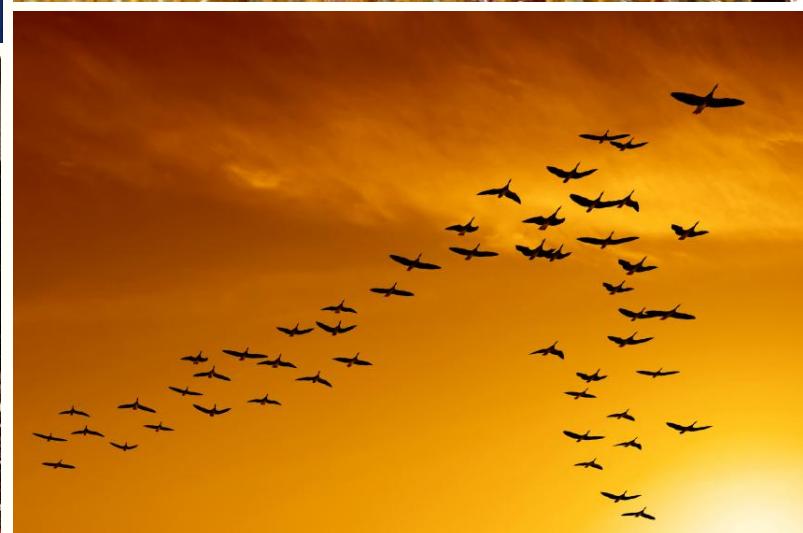
# Имплементације диференцијалне еволуције

- Mathematica:  
`NMinimize[f, vars, Method -> "DifferentialEvolution"]`
- Python:  
`scipy.optimize.differential_evolution(func, bounds, ...)`
- Постоје разни “незванични” кодови доступни на Интернету

# Particle Swarm Optimization (Оптимизација јатом): Референце

- J. Kennedy, R. Eberhart (1995), "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks IV*. pp. 1942–1948.
- J. Kennedy, R.C. Eberhart, (2001), *Swarm Intelligence*, Morgan Kaufmann, ISBN 1-55860-595-9

# PSO аналогије



# PSO и симулација одлучивања у групи

- Психологија: размишљање је процес који се нужно одиграва у групи!
- **Настајање норми и култура у оквиру једног друштва је оптимизациони процес!**
- Симулације друштвених процеса су истовремено и алгоритми за решавање оптимизационих проблема
- Мисли појединца могу се схватити као један елемент друштва (скупа појединаца)
- У комплексним ситуацијама, када је немогуће за појединца да рационално сагледа ситуацију која је сувише комплексна, људи прибегавају коришћењу веровања/убеђења/предрасуда која могу бити далеко од рационалног закључивања

# Дељење информација и имитација као оптимизациони алгоритам

- Дељење информација појединих јединки је корисно за опстанак и напредак групе тих јединки
  - Јединка има своје знање/информације
  - Знање се преноси кроз групу (кроз веровања, предрасуде, понашање, социјалне мреже итд. )
  - Култура настаје као резултат тог процеса (култура оптимизира спознају – резултати се преносе на јединке које су далеко у групи)
- Имитирање других јединки, које раде неку операцију боље, је један (могући) алгоритам за оптимизацију

# Оптимиизациони проблем: избор изборног предмета

- Пример једног избора као оптимизационог проблема
- **Одлука се доноси на основу два скупа информација**
  - Информације које **појединац сам прикупи** (градиво, наставник, начин полагања итд.)
  - Информације које **појединац добије од** других чланова групе (познаника)
- Без обзира који конкретан избор је у питању, увек постоје ова два скупа информација
- Утицај ова два скупа информација на коначну одлуку је различит за различите појединце и различите ситуације
- Симулација овог процеса је оптимизација јатом
- (Solomon Asch - Conformity Experiment)

# PSO терминологија

- Агент (елемент, честица) је једно могуће решење у оптимизационом простору
  - вектор координата решења,  $\mathbf{x} = (x_1, x_2, \dots, x_D)$
- Јато (група) је скуп агената помоћу којих се врши оптимизација (исто као популација код ГА)
- PSO је оптимизациони алгоритам који врши операције над скупом решења

# Оптимизација и кретање јата

- Агент води рачуна о најбољем решењу које је пронашао током оптимизације ( $p_{best}$ )
- Јато води рачуна о најбољем решењу које су пронашли сви агнети јата ( $g_{best}$ )
- Агенти мењају своју позицију у (оптимизационом) простору на основу ова два решења ( $p_{best}$  и  $g_{best}$ )

# Промена позиције

- Позиција агента  $\mathbf{x}_n$  у  $n$ -том кораку је
$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{v}_n \Delta t$$
- $\mathbf{x}_{n-1}$  је претходна позиција агента
- $\Delta t$  је прираштај времена
  - у оптимизацији је 1 ради једноставности
- $\mathbf{v}_n$  је брзина агента у  $n$ -том кораку
  - брзина је  $D$  димензиони вектор који одговара прираштају позиције
  - како се рачуна?

# Промена брзине

- Промена брзине се рачуна према формулама

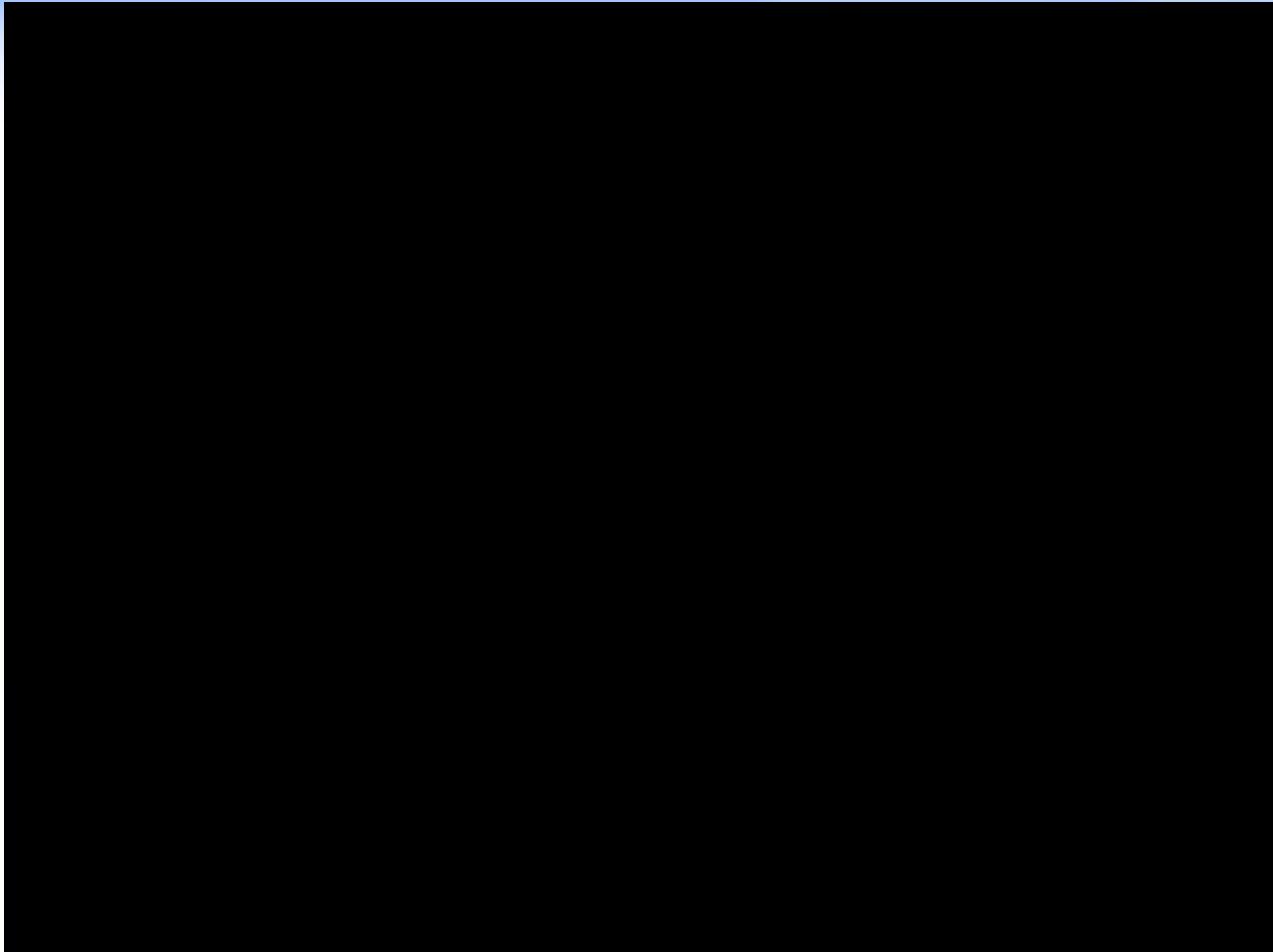
$$\mathbf{v}_n = w \cdot \mathbf{v}_{n-1} + c_1 \cdot \text{rand}() \cdot (\mathbf{p}_{\text{best}} - \mathbf{x}_{n-1}) + c_2 \cdot \text{rand}() \cdot (\mathbf{g}_{\text{best}} - \mathbf{x}_{n-1})$$

- $\mathbf{v}_{n-1}$  је брзина агента у  $n-1$  кораку
- $\text{rand}()$  је функција која генерише случајан број у интервалу  $[0,1]$
- $w$  је коефицијент инерције
- $c_1$  је когнитивни коефицијент
- $c_2$  је социјални коефицијент

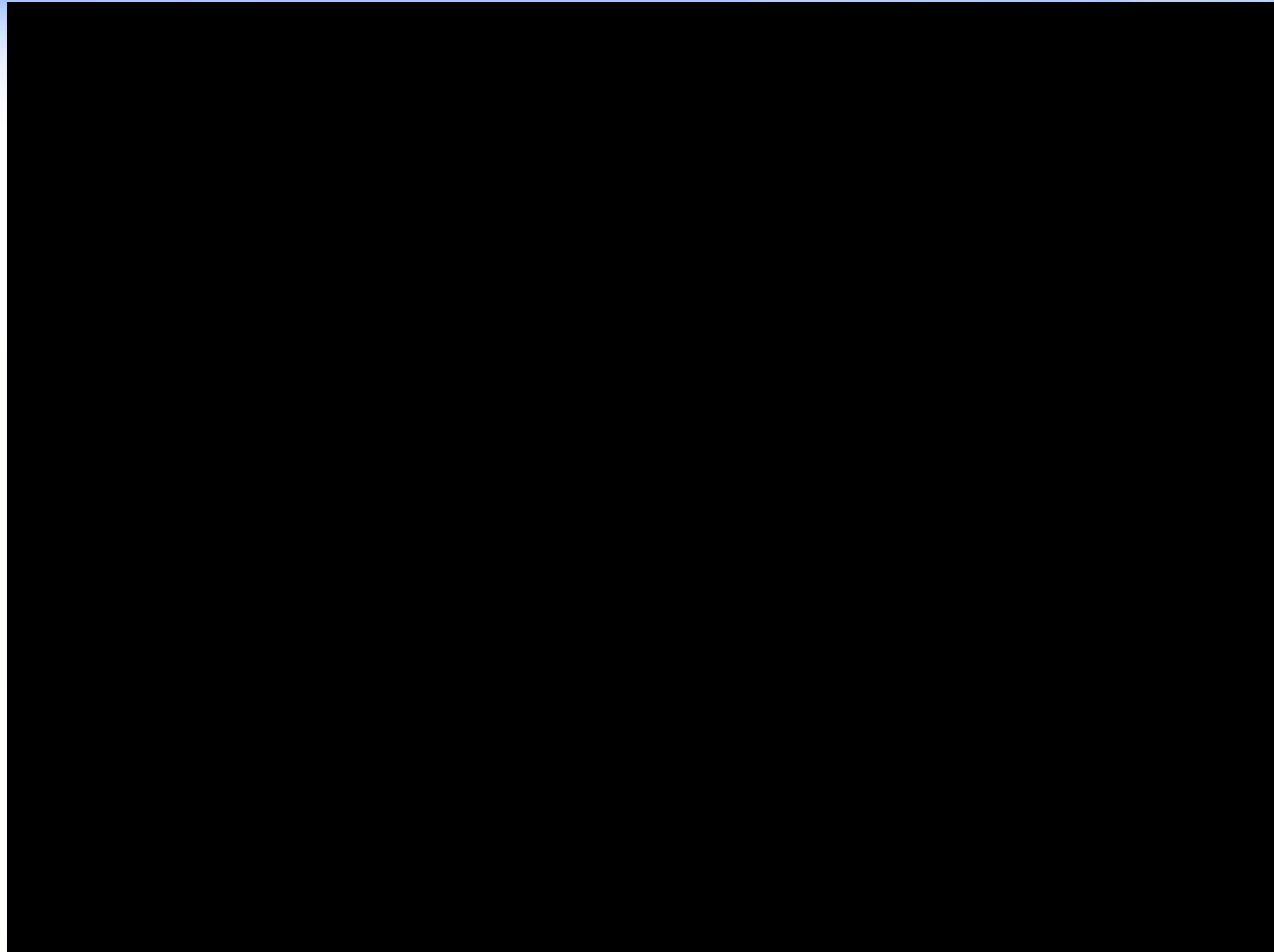
# Параметри PSO алгоритма и њихове вредности

- Број агената у јату
- Коефицијент инерције  $w = 0,729$
- Когнитивни коефицијент
- Социјални коефицијент  $(c_1, c_2) = (1,494, 1,494)$
- Максимална брзина  $v_{\max} = 0,2$   
[под условом да су променљиве  
у опсегу од  $-1$  до  $1$ ]  
тј.  $10\%$  опсега за сваку променљиву

# Илустрација: 15 агената

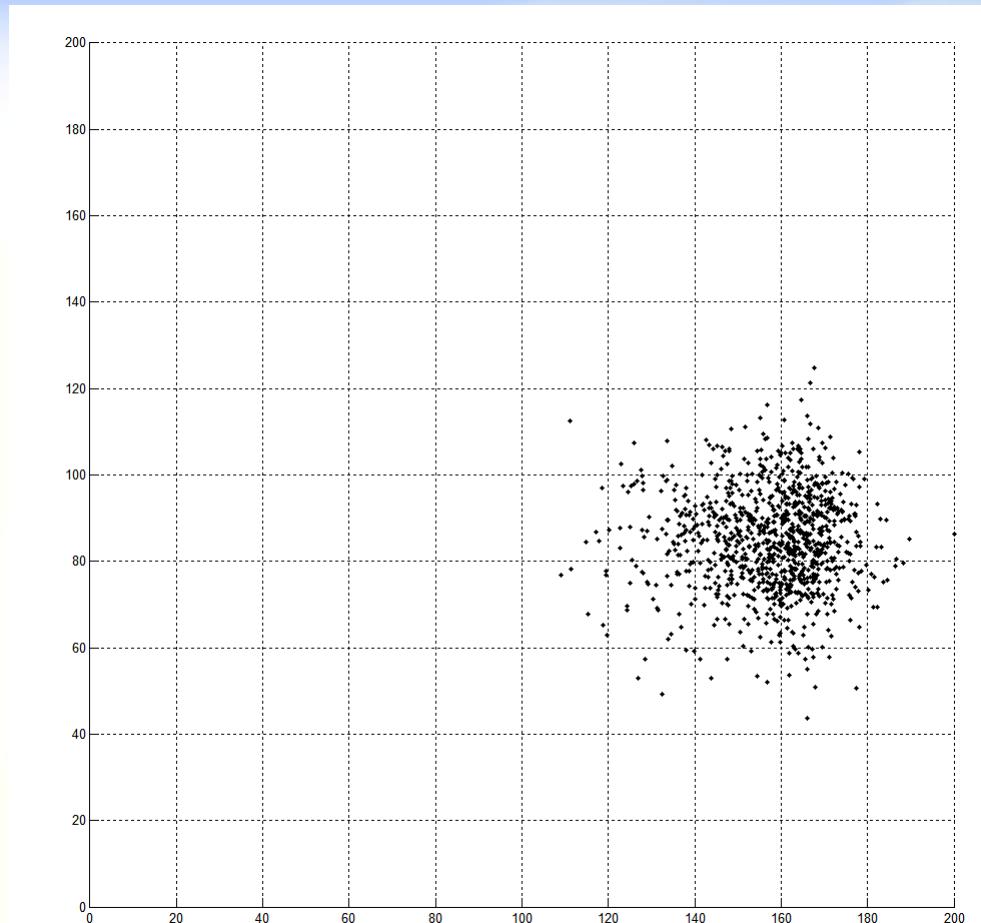


# Илустрација: 50 агената



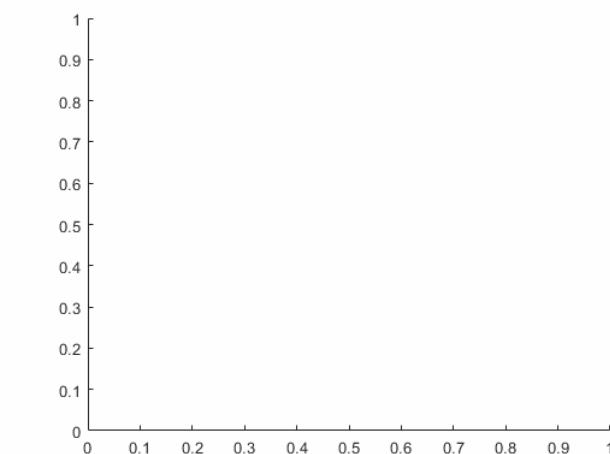
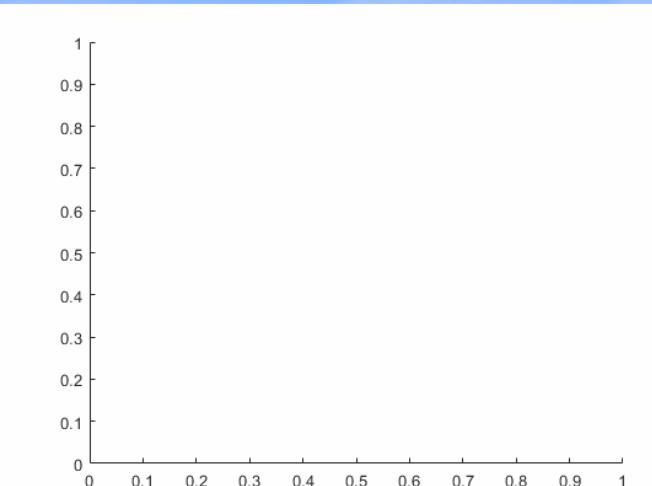
# Particle Swarm Optimization

## Илустрација



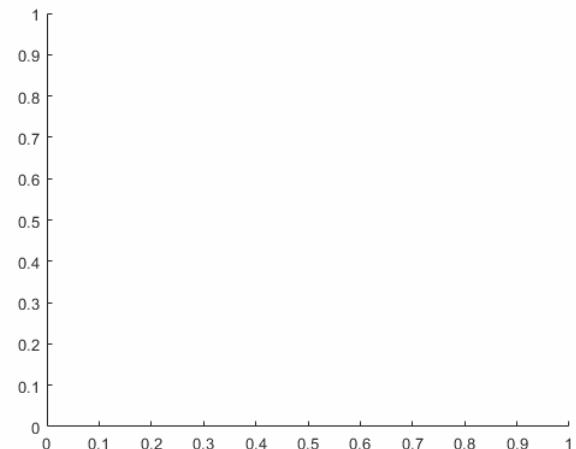
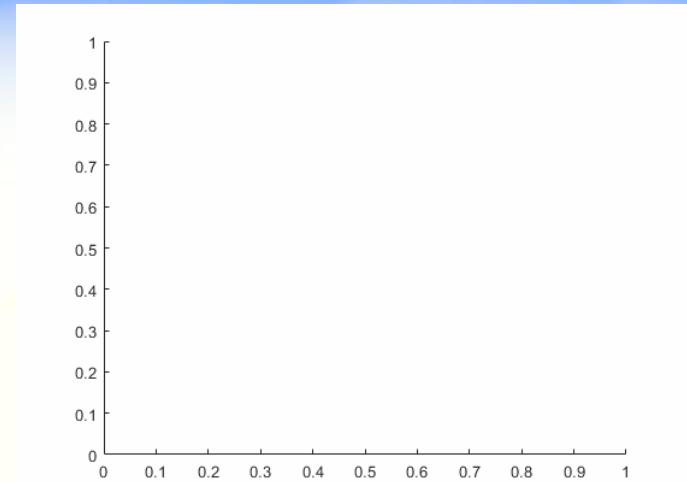
# PSO параметри и стабилност: без случајних променљивих

- $N=1000, v_{\max}=0,2,$   
 $c_1=c_2=1,5, w=0,8$   
све случајне  
променљиве су 1
- $N=1000, v_{\max}=0,2,$   
 $c_1=c_2=4,0, w=0,8$   
све случајне  
променљиве су 1



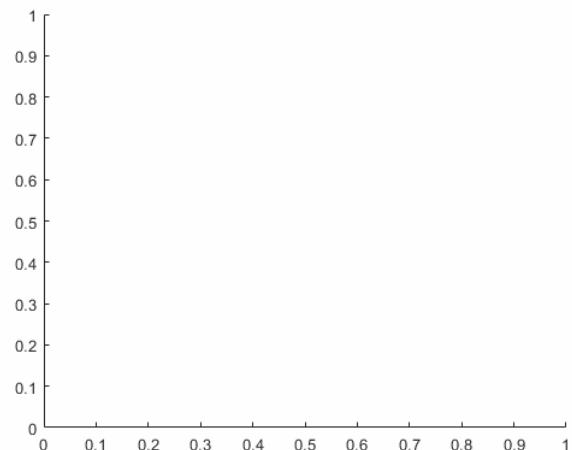
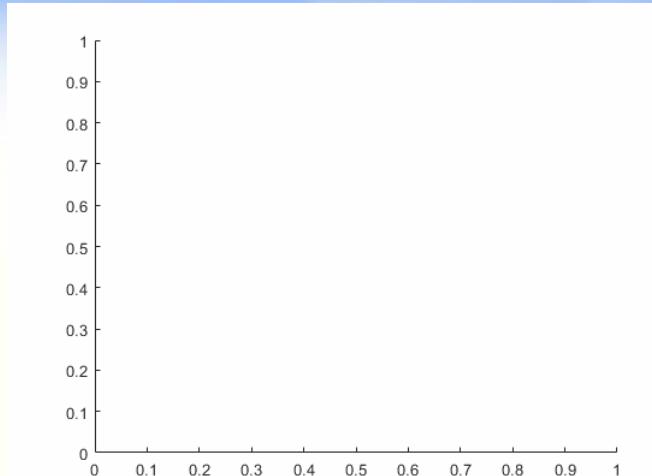
# PSO параметри и стабилност: са случајним променљивима

- $N=1000$ ,  $v_{\max}=0,2$ ,  
 $c_1=c_2=1,5$ ,  $w=0,8$   
све случајне  
променљиве су  $\text{rand}(0,1)$
- $N=1000$ ,  $v_{\max}=0,2$ ,  
 $c_1=c_2=4,0$ ,  $w=0,8$   
све случајне  
променљиве су  $\text{rand}(0,1)$



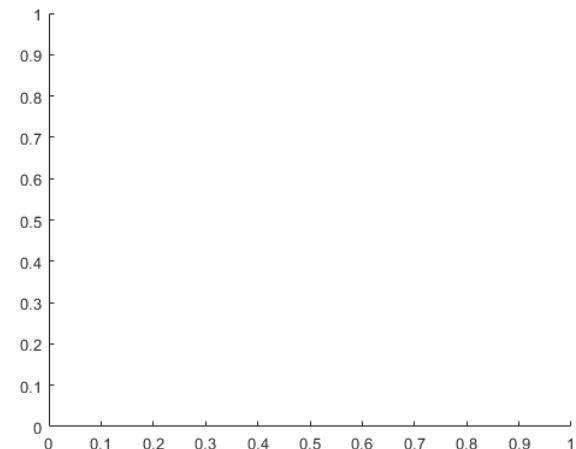
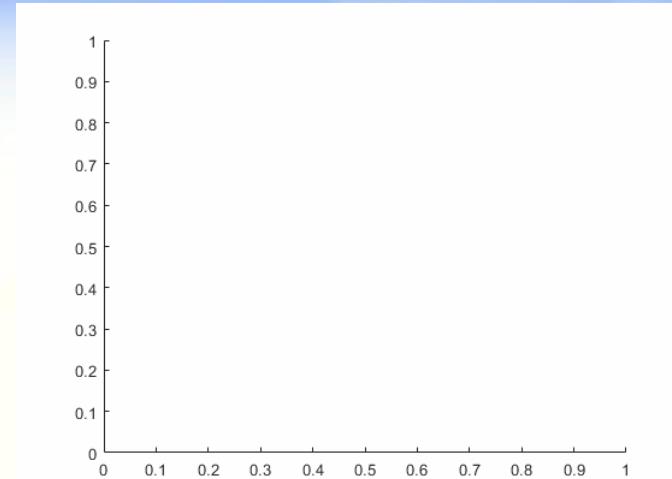
# PSO параметри и стабилност: са случајним пром. без $v_{\max}$

- $N=1000$ , без  $v_{\max}$   
 $c_1=c_2=1,5$ ,  $w=0,8$   
све случајне  
променљиве су  $\text{rand}(0,1)$
- $N=1000$ , без  $v_{\max}$   
 $c_1=c_2=4,0$ ,  $w=0,8$   
све случајне  
променљиве су  $\text{rand}(0,1)$



# PSO параметри и стабилност: са случајним пром. промена $w$

- $N=1000$ ,  $v_{\max}=0,2$ ,  
 $c_1=c_2=1,5$ ,  $w=0,2$   
све случајне  
променљиве су  $\text{rand}(0,1)$
- $N=1000$ ,  $v_{\max}=0,2$ ,  
 $c_1=c_2=1,5$ ,  $w=1,2$   
све случајне  
променљиве су  $\text{rand}(0,1)$

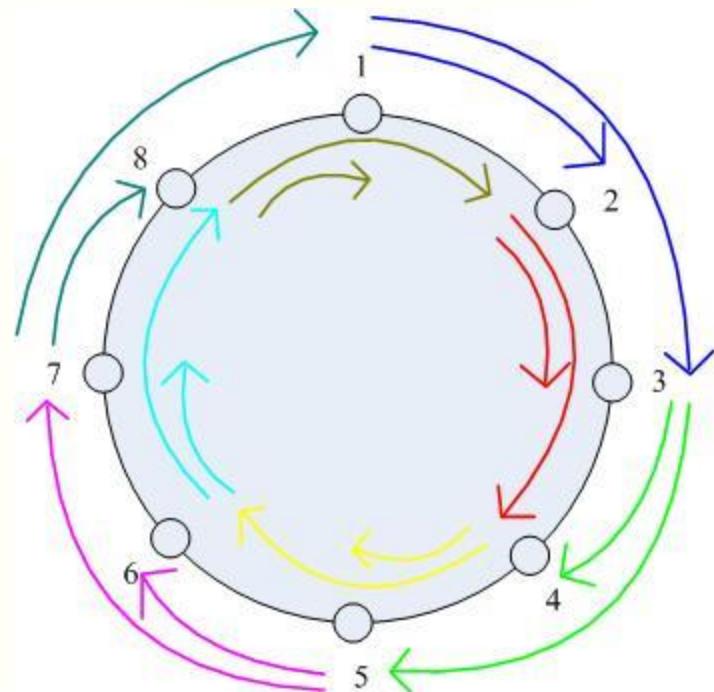


# PSO параметри и стабилност: закључци

- PSO није апсолутно стабилан алгоритам
- “Експлозија” јата дешава се уколико параметри нису добро подешени
- Посебно, коефицијенти  $c_1$  и  $c_2$  морају бити мањи од 4
  - Ова граница је теоретски показана
- Смањивање  $w$  води ка брзој конвергенцији повећавање  $w$  води ка расипању јата

# Тополошки PSO: варијације

- Уместо читавог јата агент има информације само о околини
- “Ring” топологија је најчешћа, са 2 суседа
- Може се генерализовати на “ $m$ -околину”



# PSO и EA

- Да ли се PSO може схватити као једна имплементација EA?
- $(\mu, \lambda)$ -EA,  $\mu = \lambda$  = укупан број агената
  - Оператор варијације је начин промене “позиције” агента
  - Нема селекције
  - $p_{best}$  и  $g_{best}$  су меморија!
- Формално EA нема меморију!
- Проширење EA уз дефиницију меморије?

# PSO закључци

- Једноставан оптимизациони алгоритам
- Стохастички оптимизациони алгоритам
  - случајно генерисане полазне позиције
  - `rand()` у сваком кораку алгоритма
- Може да се заустави у локалном оптимуму, ако сви агенти исконвергирају ка њему (нема даљег начина за излазак!)
- На граници између локалних и глобалних

# Имплементације PSO

- MATLAB:  
`particleswarm(fun, nvars)`
- Python: посебна библиотека  
`pyswarms`
- Разни кодови на Интернету...

# Оптимизација која опонаша колонију мрава

- Ant colony optimization (ACO)
- A. Colomni, M. Dorigo et V. Maniezzo, *Distributed Optimization by Ant Colonies*, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.
- M. Dorigo, *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italie, 1992.

# Основна идеја и терминологија

- Колонија мрава = популација
- Позиција мрава = једно могуће решење оптимизационог проблема
- Координате позиције одговарају оптимизационим променљивима
- Идеја: што више мрава прође неком путањом то је та путања боља

# Наредна решења

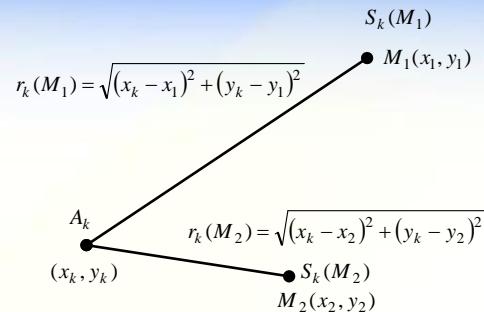
- Вероватноћа испитивања наредних решења (позиција) зависи од два параметра
  - Атрактивности решења
  - Нивоа пута (колико је мрава прошло туда)
- Ови параметри се мењају током оптимизације
- Релативно компликовано рачунање параметара

# Закључци о АСО

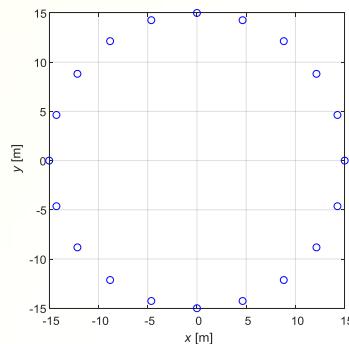
- Оптимизација са АСО алгоритмом је пре свега применљива за дискретне проблеме
- Алгоритам је прво примењен за проналажење пута у графовима
- Постоје варијанте алгоритма које се примењују на НЛР проблеме
- Комплексност алгоритма, чини се, није праћена значајним побољшањем ефикасности

# Задатак за вежбе

- Сигнал,  $S_k$ , који еmitује предајник  $k$  дат је изразом  $S_k = \frac{A_k}{r_k}$ , где је  $A_k$  константа предајника (реалан број), а  $r_k$  је растојање (у метрима) између тачке у којој се налази предајник и тачке у којој се мери сигнал (слика 1).
- Ради одређивања локације и константи два непозната извора сигнала, извршена су мерења у  $N = 20$  тачака.
- Мерне тачке су унiformно распоређене на кружници полупречника  $R_0 = 15\text{ m}$ , а координате мерних тачака су дате изразом  $(x_i, y_i) = \left( R_0 \cos \frac{2\pi i}{N}, R_0 \sin \frac{2\pi i}{N} \right)$ ,  $i = 0, 1, 2, \dots, N-1$  (слика 2).
- Извори сигнала се налазе у равни тог круга, у њему.
- Вредност сигнала у једној тачки простора једнака је збиру вредности сигнала које еmitују појединачни извори (важи принцип суперпозиције).



Слика 1. Предајник  $A_k$  који се налази у тачки  $(x_k, y_k)$  и две мерне тачке:  $M_1(x_1, y_1)$  и  $M_2(x_2, y_2)$ .



Слика 2. Распоред мерних тачака.

# Задатак за вежбе

- Вредности измереног сигнала у тим тачкама су, редом:

$$S = (2.424595205726587e-01, \quad 1.737226395065819e-01, \quad 1.315612759386036e-01, \\ 1.022985539042393e-01, \quad 7.905975891960761e-02, \quad 5.717509542148174e-02, \\ 3.155886625106896e-02, \quad -6.242228581847679e-03, \quad -6.565183775481365e-02, \\ -8.482380513926287e-02, \quad -1.828677714588237e-02, \quad 3.632382803076845e-02, \\ 7.654845872485493e-02, \quad 1.152250132891757e-01, \quad 1.631742367154961e-01, \\ 2.358469152696193e-01, \quad 3.650430801728451e-01, \quad 5.816044173713664e-01, \\ 5.827732223753571e-01, \quad 3.686942505423780e-01)$$

- Усвојени запис решења овог проблема је  $\mathbf{x} = (x_{P_1}, y_{P_1}, x_{P_2}, y_{P_2}, A_1, A_2)$ .
- Оптимизациона функција је

$$f_{\text{opt}}(\mathbf{x}) = \begin{cases} \sum_{i=0}^{N-1} \left( \frac{A_1}{\sqrt{(x_i - x_{P_1})^2 + (y_i - y_{P_1})^2}} + \frac{A_2}{\sqrt{(x_i - x_{P_2})^2 + (y_i - y_{P_2})^2}} - s_i \right)^2, & \sqrt{x_{P_1}^2 + y_{P_1}^2} < R_0 \text{ и } \sqrt{x_{P_2}^2 + y_{P_2}^2} < R_0. \\ 100, & \sqrt{x_{P_1}^2 + y_{P_1}^2} \geq R_0 \text{ или } \sqrt{x_{P_2}^2 + y_{P_2}^2} \geq R_0 \end{cases}$$

# Задатак за вежбе

- Потребно је пронаћи координате и константе непознатих извора
- Написати своју имплементацију алгоритма **диференцијалне еволуције**
- Уколико се користи готова функција за диференцијалну еволуцију (из Pythona или неког другог извора), задатак носи **2 поена**
- Пронаћи и у ASCII фајл записати решење задатог проблема за које је  $f_{\text{opt}} \leq 10^{-14}$