پیاده سازی مساله کوله پشتی با الگوریتم ژنتیک در Matlab

مراحل الگوریتم ژنتیک

* ۱-ایجاد جمعیت اولیه تصادفی و ارزیابی آنها
* ۲-انتخاب والدین و ترکیب آنها برای ایجاد جمعیت فرزندان (Cross over)
* ۳-انتخاب جمعیت برای اعمال جهش و ایجاد جمعیت جهش یافتگان
* ۴-ادغام جمعیت اصلی ، فرزندان و جهش یافتگان و ایجاد جمعیت اصلی جدید
* ۵-اگر شرایط خاتمه محقق نشده باشد از مرحله ۲ تکرار می کنیم.
* ۶-پایان

* مفاهیم اولیه در درک الگوریتم ژنتیک بسیار مهم هستند عبارتند از:

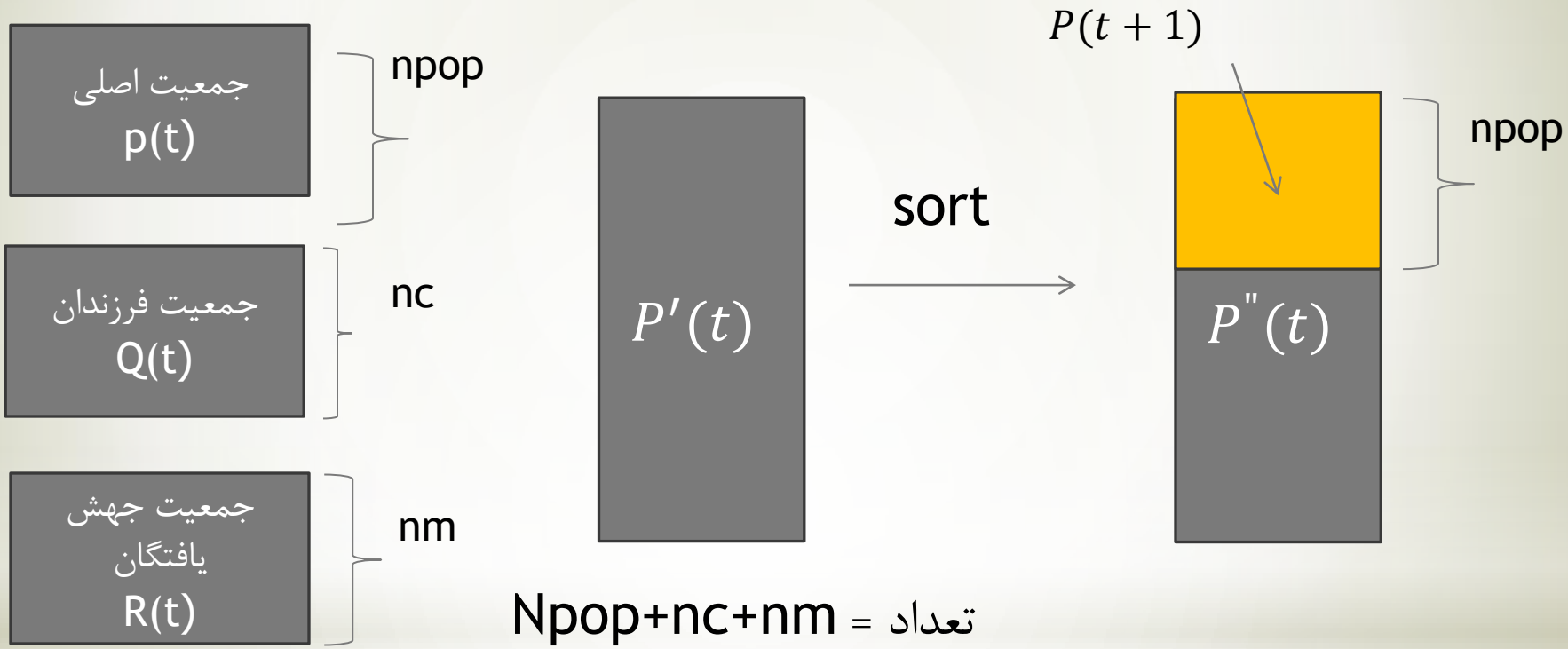
کروموزوم :اساس الگوریتم ژنتیک تبدیل هر مجموعه جواب به یک جواب کدینگ است که این کد را کروموزوم می گویند.

نسل:هر تکرار از الگوریتم را یک نسل می گویند.

جمعیت:مجموعه ای از کروموزومها را یک جمعیت می گویند.

ژن: عناصر تشکیل دهنده یک کروموزوم را که معمولا اعداد هستند ژن می گویند.

merge, sort , trancate*



$$nc = p_c \times npop$$
$$nm = p_m \times npop$$

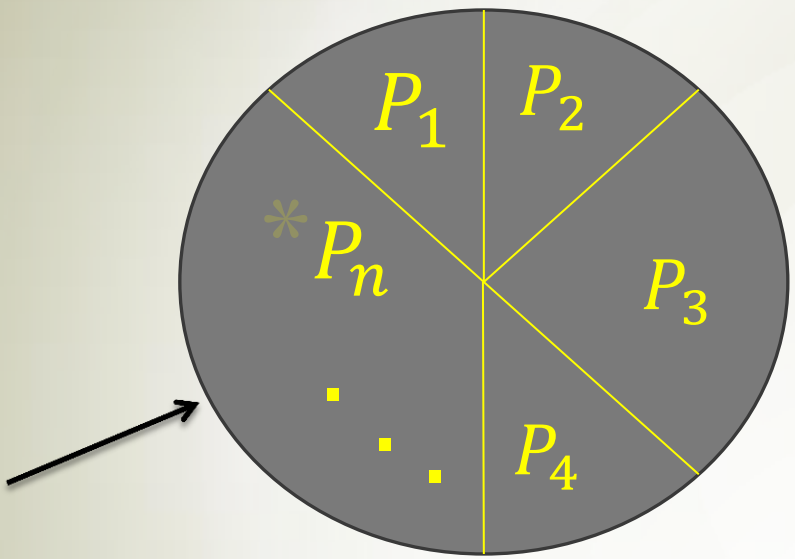
$$X = (x_1, x_2, \dots, x_{nvar}) \qquad x_i \in \{0,1\}$$

شکل کروموزوم (بردار جواب)

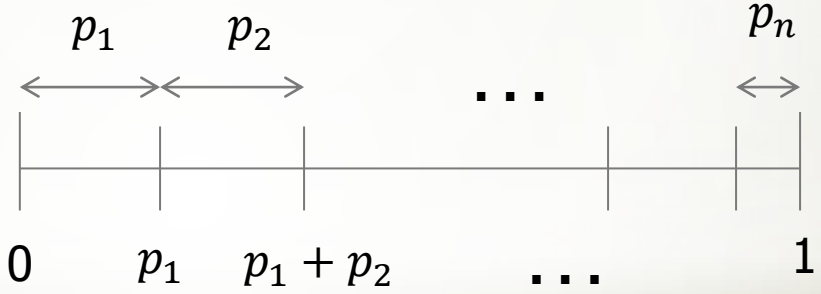
روشهای انتخاب والدین

* انتخاب تصادفی
* انتخاب بر اساس شایستگی یا رتبه (با یک توزیع احتمال گسسته نمونه برداری می کنیم)
(از چرخ رولت استفاده می کنیم)
* انتخاب رقابتی

چرخ رولت برای انتخاب والدین



$$p_1 + p_2 + p_3 + \cdots + p_n = 1$$



در بازه ۰ تا ۱ باز می کنیم

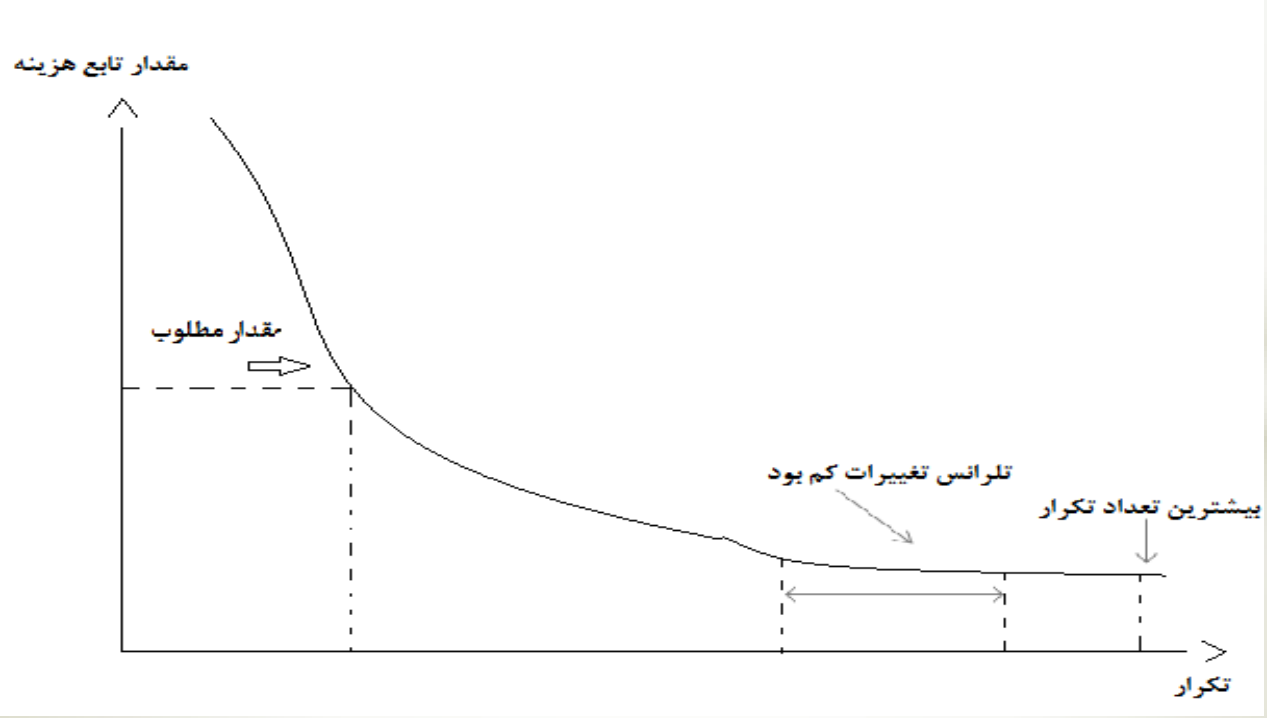
$$r \sim u(0,1)$$

$$c_i = \sum_{j=1}^{i} p_j$$

Find smallest i , where $r \leq c_i$

انواع شرایط خاتمه

* ۱- رسیدن به حد قابل قبولی از پاسخ
* ۲- سپری شدن زمان یا تکرار معین
* ۳- سپری شدن زمان یا تکرار معین بدون مشاهده بهبود خاصی در نتیجه



مراحل کدنویسی مساله

- تعریف مساله
- پارامترهای الگوریتم
- تنظیمات اولیه (Initialization)
- حلقه تکاملی
- پردازش نتایج

۱- تعریف مساله*

```
%% Problem Definition

global NFE;
NFE=0;

model=CreateModel();          % Create Knapsack Model

CostFunction=@(x) KnapsackCost(x,model);   % Cost Function

nVar=model.n;          % Number of Decision Variables

VarSize=[1 nVar];   % Size of Decision Variables Matrix
```

۲-پارامترهای الگوریتم

```
%% GA Parameters
MaxIt=200;          % Maximum Number of Iterations

nPop=50;           % Population Size

pc=0.8;                     % Crossover Percentage
nc=2*round(pc*nPop/2);  % Number of Offsprings (Parents)

pm=0.3;                     % Mutation Percentage
nm=round(pm*nPop);      % Number of Mutants

mu=0.02;               % Mutation Rate

ANSWER=questdlg('Select the Parent Selection Method:','GA','Random','RWS','TS','RWS');
UseRandomSelection=strcmpi(ANSWER,'Random');
UseRWS=strcmpi(ANSWER,'RWS');
UseTS=strcmpi(ANSWER,'TS');

if UseRWS
    beta=10;               % Selection Pressure
end

if UseTS
  TournamentSize=3;       % Tournament Size
end

pause(0.1);
```

تنظیمات اولیه (Initialization)*

```
%% Initialization

% Create Empty Structure
empty_individual.Position=[];
empty_individual.Cost=[];
empty_individual.Sol=[];

% Create Population Matrix (Array)
pop=repmat(empty_individual,nPop,1);

% Initialize Population
for i=1:nPop

    % Initialize Position
    pop(i).Position=CreateRandomSolution(model);
```

تنظیمات اولیه (Initialization)*

```
% Evaluation
    [pop(i).Cost pop(i).Sol]=CostFunction(pop(i).Position);

end
% Sort Population
Costs=[pop.Cost];
[Costs SortOrder]=sort(Costs);
pop=pop(SortOrder);

% Update Best Solution Ever Found
BestSol=pop(1);

% Update Worst Cost
WorstCost=max(Costs);

% Array to Hold Best Cost Values
BestCost=zeros(MaxIt,1);

% Array to Hold NFEs
nfe=zeros(MaxIt,1);
```

۴- حلقه تکاملی *

```
%% GA Main Loop
for it=1:MaxIt
    if UseRWS
        % Calculate Selection Probabilities
        P=exp(-beta*Costs/WorstCost);
        P=P/sum(P);
    end
    % Crossover
    popc=repmat(empty_individual,nc/2,2);
    for k=1:nc/2
        % Select Parents
        if UseRandomSelection
            i1=randi([1 nPop]);
            i2=randi([1 nPop]);
            p1=pop(i1);
            p2=pop(i2);
        end
        if UseRWS
            i1=RouletteWheelSelection(P);
            i2=RouletteWheelSelection(P);
            p1=pop(i1);
            p2=pop(i2);
        end
        if UseTS
            p1=TournamentSelection(pop,TournamentSelection);
            p2=TournamentSelection(pop,TournamentSelection);
        end
```

روش بولتزمان:

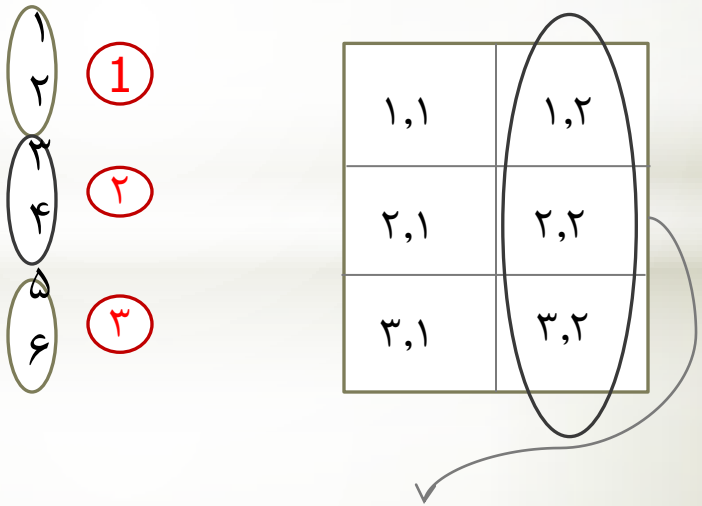
$$P_i \propto e^{-c_i}$$

$$P_i = \frac{e^{-c_i}}{\sum_j e^{-c_j}}$$

اعمال فشار انتخاب

$$P_i \propto e^{-\beta c_i}$$

$$P_i = \frac{e^{-\beta c_i}}{\sum_j e^{-\beta c_j}}$$



۴-حلقه تکاملی*

```
% Apply Crossover
    [popc(k,1).Position popc(k,2).Position]=BinaryCrossover(p1.Position,p2.Position);
    % Evaluate Offsprings
    [popc(k,1).Cost  popc(k,1).Sol]=CostFunction(popc(k,1).Position);
    [popc(k,2).Cost  popc(k,2).Sol]=CostFunction(popc(k,2).Position);
end
popc=popc(:);
% Mutation
popm=repmat(empty_individual,nm,1);
for k=1:nm
    % Select Parent Index
    i=randi([1 nPop]);
    % Select Parent
    p=pop(i);
    % Apply Mutation
   popm(k).Position=Mutate(p.Position,mu);
% Evaluate Mutant
 [popm(k).Cost  popm(k).Sol]=CostFunction(popm(k).Position);
end
```

۴-حلقه تکاملی*

```
% Merge Population
    pop=[pop popc popm];

% Sort Population
    Costs=[pop.Cost];
    [Costs SortOrder]=sort(Costs);
    pop=pop(SortOrder);

    % Truancate Extra Memebrs
    pop=pop(1:nPop);
    Costs=Costs(1:nPop);

    % Update Best Solution Ever Found
   BestSol=pop(1);

   % Update Worst Cost
   WorstCost=max(WorstCost,max(Costs));
```

۴-حلقه تکاملی*

```
% Update Best Cost Ever Found
    BestCost(it)=BestSol.Cost;

    % Update NFE
    nfe(it)=NFE;

    % Show Iteration Information
    if BestSol.Sol.IsFeasible
        Flag=' *';
    else
        Flag='';
    end

    disp(['Iteration ' num2str(it) ': NFE = ' num2str(nfe(it)) ', Best Cost = '
num2str(BestCost(it)) Flag]);

end
```

```
function model=CreateModel()

  v=[45  83  38  83  45  58  25  78  43  87 ...
     34  27  17  34  21  64  49  57  66  45 ...
     20  31  64  16  18  80  84  60  30  38 ...
     64  75  55  67  85  54  53  22  64  62 ...
     50  63  17  51  21  13  69  42  19  64];

 w=[ 403   672   876   833   462   421   155   359   117   825 ...
     752   541   693   357   432   449   672   736   368   739 ...
     859   698   468   827   572   493   883   217   355   396 ...
     788   746   543   154   452   147   217   748   316   202 ...
     624   453   163   234   296   143   461   406   831   154];

 n=numel(v);

 W=10000;

 model.n=n;
 model.v=v;
 model.w=w;
 model.W=W;

end
```

```
function [z sol]=KnapsackCost(x,model)
    global NFE;
    if isempty(NFE)
        NFE=0;
    end
    NFE=NFE+1;
    v=model.v;
    w=model.w;
    W=model.W;

    GainedValue=sum(v.*x);
    LostValue=sum(v.*(1-x));
    GainedWeight=sum(w.*x);
    LostWeight=sum(w.*(1-x));
    Violation=max(GainedWeight/W-1,0);
    %alpha=10000;
   %z=LostValue+alpha*Violation;
   beta=10;
    z=LostValue*(1+beta*Violation);

    sol.GainedValue=GainedValue;
    sol.LostValue=LostValue;
    sol.GainedWeight=GainedWeight;
    sol.LostWeight=LostWeight;
    sol.Violation=Violation;
   sol.z=z;
    sol.IsFeasible=(Violation==0);
end
```

```
function x=CreateRandomSolution(model)
    n=model.n;
     x=randi([0 1],1,n);

End
------------------------------
function i=RouletteWheelSelection(P)
     r=rand;
     C=cumsum(P);
     i=find(r<=C,1,'first');
end
------------------------------
function p=TournamentSelection(pop,m)
  n=numel(pop);
    A=randsample(n,m);
    spop=pop(A);
    costs=[spop.Cost];
 [~, i]=min(costs);     p=spop(i);          end
```

روشهای انتخاب والدین

```
function [y1 y2]=BinaryCrossover(x1,x2)

    M=randi([1 3]);

    switch M
        case 1
            % Single Point Crossover
            [y1 y2]=SinglePointCrossover(x1,x2);

        case 2
            % Double Point Crossover
            [y1 y2]=DoublePointCrossover(x1,x2);

        case 3
            % Uniform Crossover
          [y1 y2]=UniformCrossover(x1,x2);
    end
end
```

```
function [y1 y2]=SinglePointCrossover(x1,x2)
   nVar=numel(x1);
    c=randi([1 nVar-1]);
  y1=[x1(1:c) x2(c+1:end)];
  y2=[x2(1:c) x1(c+1:end)];
End
----------------------------
function [y1 y2]=DoublePointCrossover(x1,x2)
    nVar=numel(x1);
    c=randsample(nVar-1,2);
    c1=min(c);
    c2=max(c);
    y1=[x1(1:c1) x2(c1+1:c2) x1(c2+1:end)];
    y2=[x2(1:c1) x1(c1+1:c2) x2(c2+1:end)];
end
```

```
function [y1 y2]=UniformCrossover(x1,x2)

    alpha=randi([0 1],size(x1));
   y1=alpha.*x1+(1-alpha).*x2;
  y2=alpha.*x2+(1-alpha).*x1;
End
---------------------
function y=Mutate(x,mu)

    nVar=numel(x);

   nMu=ceil(mu*nVar);
   j=randsample(nVar,nMu);

   y=x;
   y(j)=1-x(j);
end
```