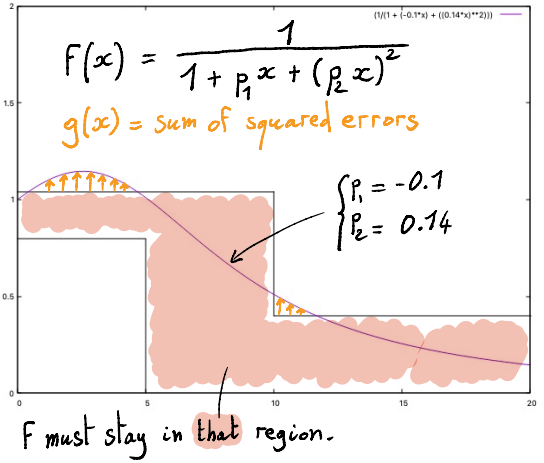
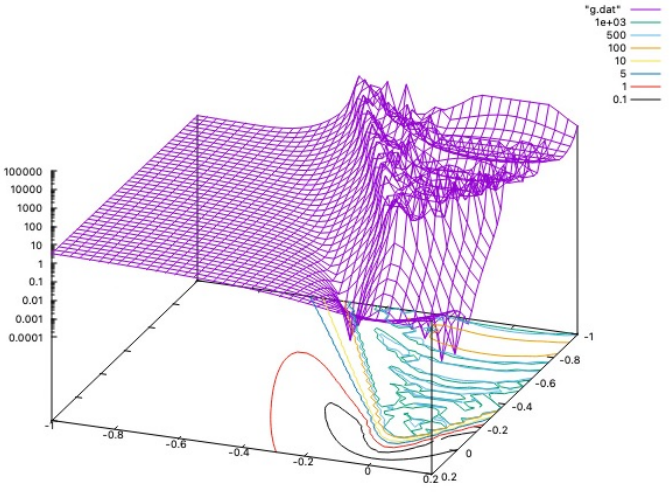
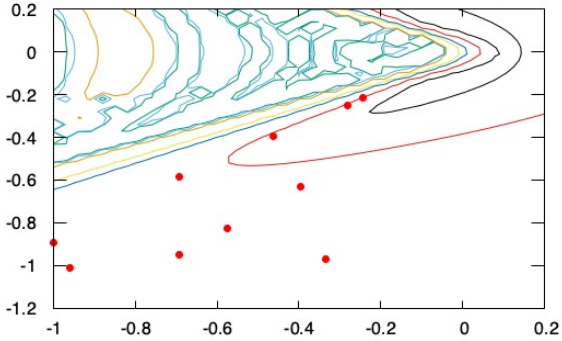
# Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces

Rainer STORN
Kenneth PRICE
1997






Non-linear objective function of, usually, many real parameters.




Generation 0

Evolution of a population of vectors by
- mutation -> diversity
- recombination -> knowledge sharing
- selection -> survival of the fittest

Price & Storn 2005 : Differential evolution, a practical approach to global optimization. pp.31-…

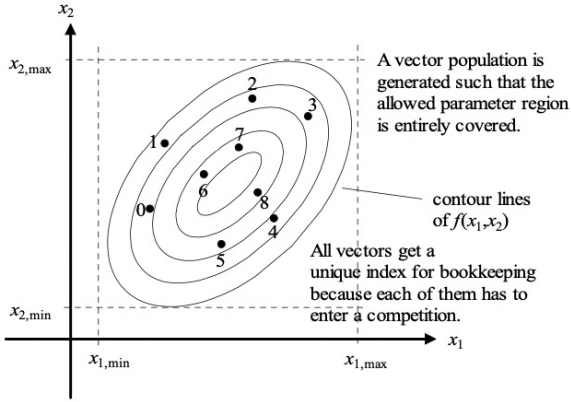



**Fig. 1.24.** Initializing the DE population

```
int i,j;
D = d;
NP = 5*D;
X = (float**)malloc(NP*sizeof(float*));
for (i=0;i<NP;i++) X[i] = (float*)malloc(D*sizeof(float));
U = (float**)malloc(NP*sizeof(float*));
for (i=0;i<NP;i++) U[i] = (float*)malloc(D*sizeof(float));
for (i=0;i<NP;i++)
{
  for (j=0;j<D;j++) X[i][j] = -1 + rand01()*(fabs(0.2-1));
}
```

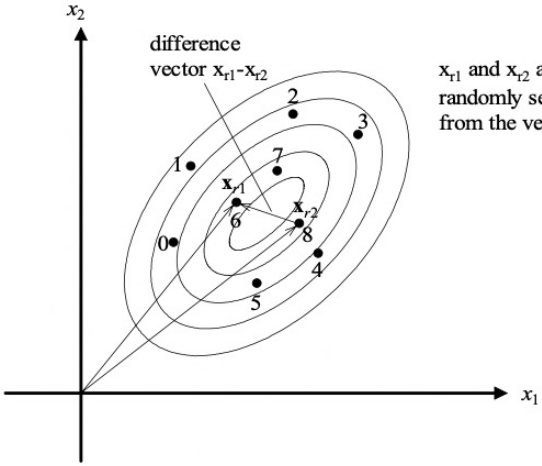



**Fig. 1.25.** Generating the perturbation: $\mathbf{x}_{r1} - \mathbf{x}_{r2}$

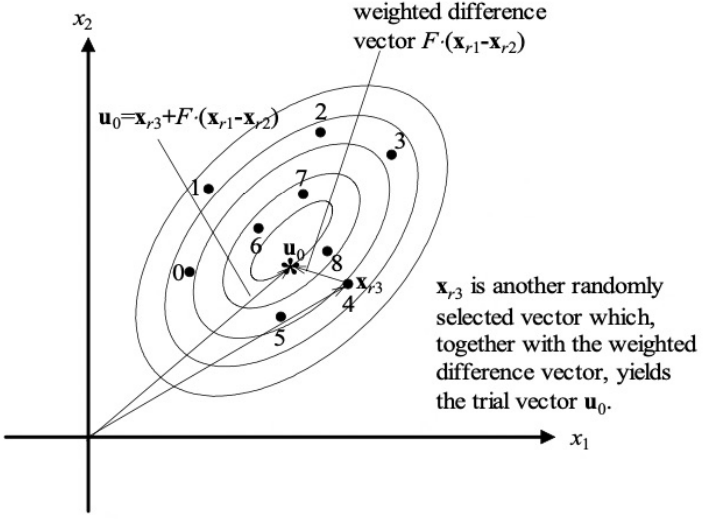



**Fig. 1.26.** Mutation

```
do R0=floor(rand01()*NP); while (R0==i);
do R1=floor(rand01()*NP); while (R1==R0 || R1==i);
do R2=floor(rand01()*NP); while (R2==R1 || R2==R0 || R2==i);
```

```
U[i][j] = X[R0][j] + F*(X[R1][j]-X[R2][j]);
```

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } \left(\text{rand}_j(0,1) \le Cr \text{ or } j = j_{\text{rand}}\right) \\ x_{j,i,g} & \text{otherwise.} \end{cases}$$

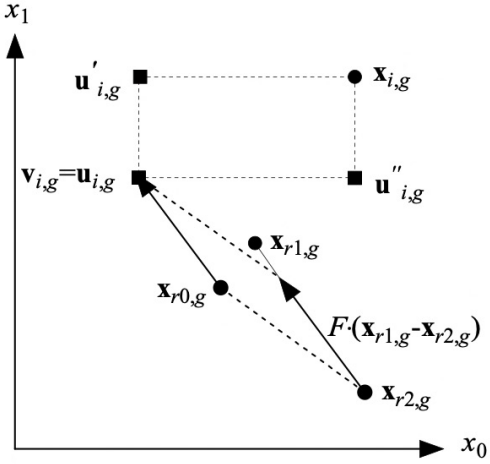



**Fig. 2.2.** The possible additional trial vectors $\mathbf{u}'_{i,g}$ , $\mathbf{u}''_{i,g}$ when $\mathbf{x}_{i,g}$ and $\mathbf{v}_{i,g}$ are uniformly crossed

## Recombination

```
jrand=floor(D*rand01());
for (j=0;j<D;j++) /* generate trial vector */
{
  if (rand01()<=CR || j==jrand)
    U[i][j] = X[R0][j] + F*(X[R1][j]-X[R2][j]);
  else
    U[i][j] = X[i][j];
}
```

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases}$$

## Selection

```
for (i=0;i<NP;i++) /* select next generation */
{
  if (g(U[i][0],U[i][1])<=g(X[i][0],X[i][1]))
    for (j=0;j<D;j++) X[i][j] = U[i][j];
}
```

```
int i,j,jrand;
for (i=0;i<NP;i++) /* R0!=R1!=R2!=i */
{
  do R0=floor(rand01()*NP); while (R0==i);
  do R1=floor(rand01()*NP); while (R1==R0 || R1==i);
  do R2=floor(rand01()*NP); while (R2==R1 || R2==R0 || R2==i);
  jrand=floor(D*rand01());
  for (j=0;j<D;j++) /* generate trial vector */
  {
    if (rand01()<=CR || j==jrand)
      U[i][j] = X[R0][j] + F*(X[R1][j]-X[R2][j]);
    else
      U[i][j] = X[i][j];
  }
}
for (i=0;i<NP;i++) /* select next generation */
{
  if (g(U[i][0],U[i][1])<=g(X[i][0],X[i][1]))
    for (j=0;j<D;j++) X[i][j] = U[i][j];
}
```