

The program Funcopt will be used for finding the maximum of the function

$$f(x_1, x_2) = \frac{e^{-x_1^2 - x_2^2} + \sqrt{5} \sin^2(x_2 x_1^2) + 2 \cos^2(2x_1 + 3x_2)}{1 + x_1^2 + x_2^2}.$$

in the interval  $x_1, x_2$  in  $[-3,3]$ .

The program Funcopt consists of the following parts:

- funcopt.m (the main program)
- initpop.m (initializes the population)
- decode\_chromosome.m (decodes the chromosomes)
- evaluate\_individual.m (evaluates the individuals)
- tournament\_select.m (carries out tournament selection)
- crossover.m (performs crossover)
- mutate.m (performs mutations)

## funcopt.m

```

npop = 50;
ngenes = 40;
pcross = 0.8;
pmut = 0.05;
ptour = 0.75;
range = 3.0;
maxgenerations = 200;

population = initpop(npop,ngenes);

for gen = 1:maxgenerations

    maxfitness = 0.0;
    for i = 1:npop
        x = decode_chromosome(population,i,range,ngenes);
        fitness(i) = evaluate_individual(x);
        if (fitness(i) > maxfitness)
            maxfitness = fitness(i);
            best_individual = i;
            xbest = x;
        end
    end

    temp_pop = population;

    temp_pop(1,:) = population(best_individual,:);
    temp_pop(2,:) = population(best_individual,:);

    ....

    .... (cont'd)

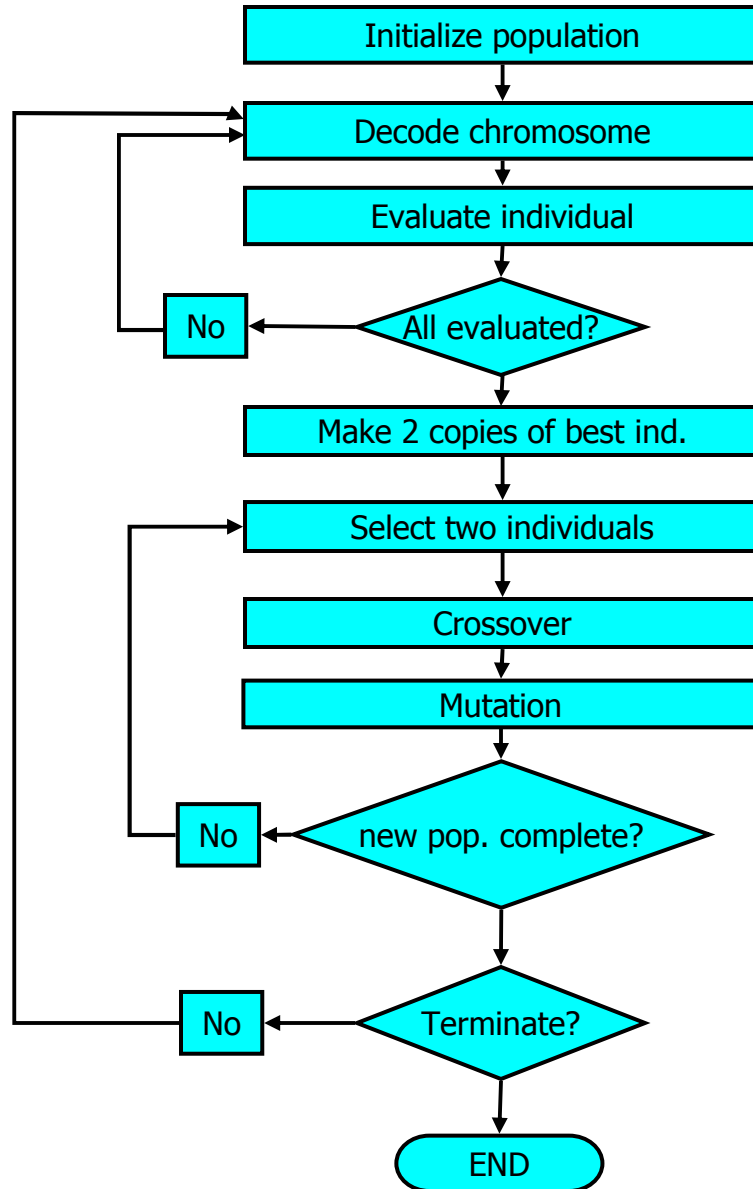
    for i = 3:2:npop
        i1 = tournament_select(fitness, npop, ptour);
        i2 = tournament_select(fitness, npop, ptour);
        r = rand;
        if (r < pcross)
            new_individuals = crossover(population, i1, i2, ngenes);
            temp_pop(i,:) = new_individuals(1,:);
            temp_pop(i+1,:) = new_individuals(2,:);
        else
            temp_pop(i,:) = population(i1,:);
            temp_pop(i+1,:) = population(i2,:);
        end
    end

    for i = 3:npop
        temp_individual = mutate(temp_pop(i,:), pmut, ngenes);
        temp_pop(i,:) = temp_individual;
    end

    population = temp_pop;
    maxfitness, xbest

end

```



```
population = initpop(npop,ngenes);
```

```
x = decode_chromosome(population,i,range,ngenes);
```

```
fitness(i) = evaluate_individual(x);
```

```
temp_pop(1,:) = population(best_individual,:);
```

```
temp_pop(2,:) = population(best_individual,:);
```

```
i1 = tournament_select(fitness,npop,ptour);
```

```
i2 = tournament_select(fitness,npop,ptour);
```

```
new_individuals = crossover(population,i1,i2,ngenes);
```

```
temp_pop(i,:) = mutate(temp_pop(i,:),pmut,ngenes);
```

**NOTE: unnecessary intermediate step in the lecture notes!**

## Initialization of the population

```
function population = initpop(npop,ngenes);  
  
for i = 1:npop  
    for j = 1:ngenes  
        s = rand;  
        if (s < 0.5)  
            population(i,j) = 0;  
        else  
            population(i,j) = 1;  
        end  
    end  
end  
end
```

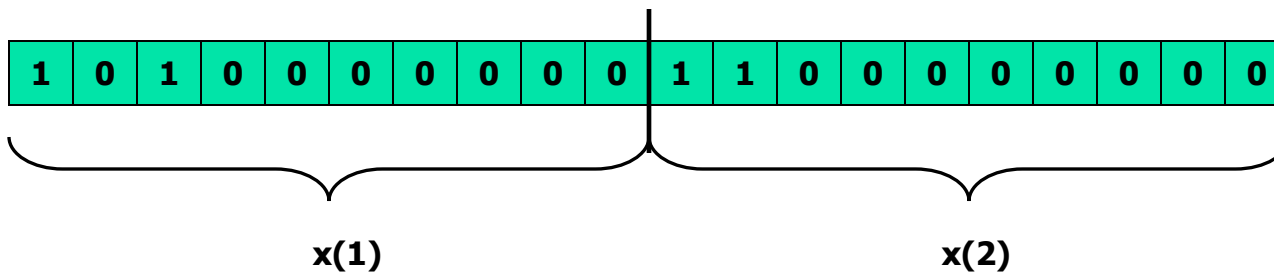
## Compact notation:

```
function population = initpop(npop,ngenes);  
  
population = fix(2.0*rand(npop,ngenes));
```

## Decoding chromosomes:

```
function x = decode_chromosome(population,i,range,ngenes);

nhalf = fix(ngenes/2) ;
x(1) = 0.0;
for j = 1:nhalf
    x(1) = x(1) + population(i,j)*2^(-j);
end
x(1) = -range + 2*range*x(1);
x(2) = 0.0;
for j = 1:nhalf
    x(2) = x(2) + population(i,j+nhalf)*2^(-j);
end
x(2) = -range + 2*range*x(2);
```



(In this case:  $x(1) = -range + 2*0.625*range = 0.75$ ,  $x(2) = -range + 2*0.75*range = 1.50$ )

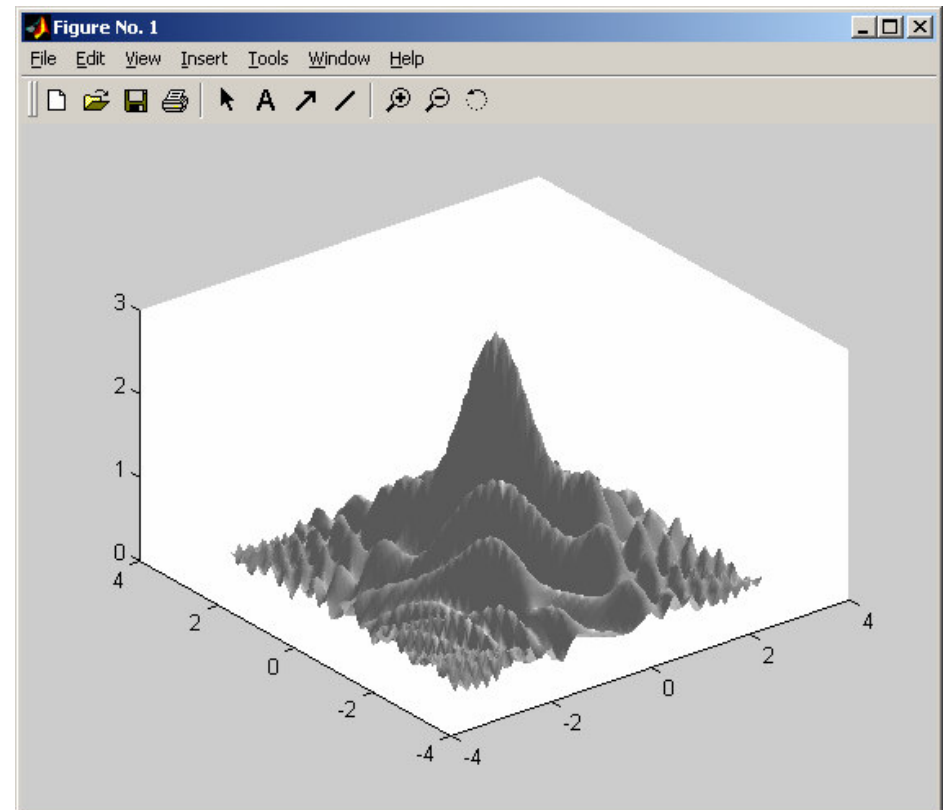
## Evaluating individuals

```
function f = evaluate_individual(x);
```

```
f = (exp(-x(1)^2 - x(2)^2)+ ...  
    sqrt(5)*(sin(x(2)*x(1)*x(1))^2) + ...  
    2*(cos(2*x(1) + 3*x(2))^2))/(1 + x(1)^2 + x(2)^2);
```

**Note:** ... means that an expression continues on the next line.

**f(1,1) = 0.62652...**



## Tournament selection:

```
function i = tournament_select(fitness, npop, ptour);

itmp1 = 1+fix(rand*npop);
itmp2 = 1+fix(rand*npop);

r = rand;

if (r < ptour)
    if (fitness(itmp1)>fitness(itmp2))
        i = itmp1;
    else
        i = itmp2;
    end
else
    if (fitness(itmp1)>fitness(itmp2))
        i = itmp2;
    else
        i = itmp1;
    end
end
```

**Note: in this simple example, the tournament size is set to 2.**



## Crossover

```
function new_individuals = crossover(population,i1,i2,ngenes);  
  
cp = 1 + fix(rand*(ngenes-1));  
  
for j = 1:ngenes  
    if (j < cp)  
        new_individuals(1,j) = population(i1,j);  
        new_individuals(2,j) = population(i2,j);  
    else  
        new_individuals(1,j) = population(i2,j);  
        new_individuals(2,j) = population(i1,j);  
    end  
end  
end
```

**Note: selection from population (left unchanged during the whole selection cycle). New individuals inserted in temp\_pop.**

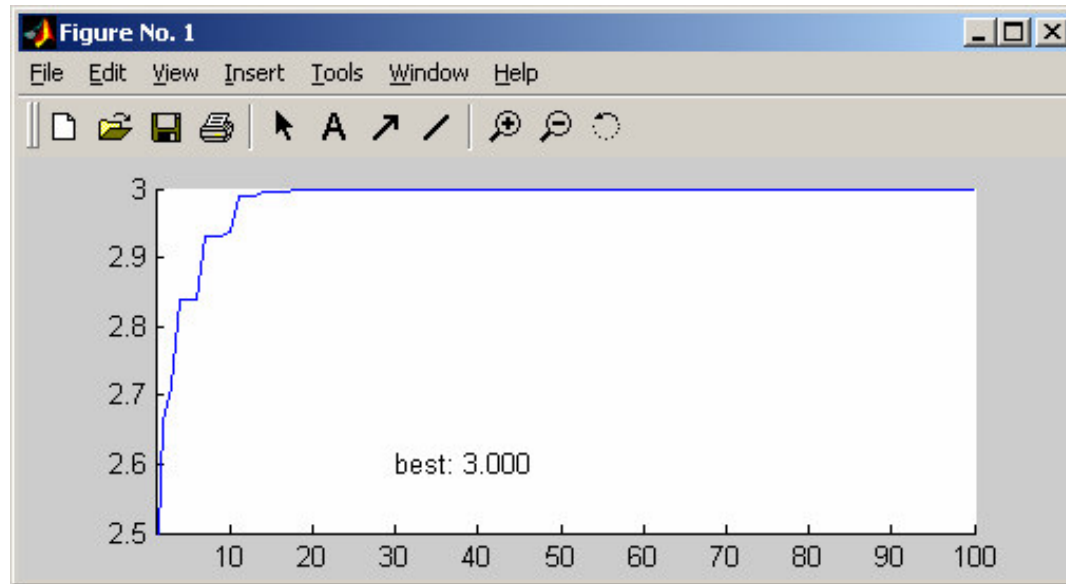
## Mutation

```
function mutated_individual = mutate(individual, pmut, ngenes);

mutated_individual = individual;
for i = 1:ngenes
    r = rand;
    if (r < pmut)
        mutated_individual(i) = fix(2.0*rand);
    end
end
```

Running the program

(1) Without graphics, (2) With population 2D graphics



**Note:** An introduction to Matlab programming (including *handle graphics*) can be found at

[http://www.mathworks.com/access/helpdesk/help/pdf\\_doc/matlab/getstart.pdf](http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/getstart.pdf)

(In particular, see chapters 3 and 4).

## Handle graphics in Matlab (see Handout 3, and e.g. the online help files for Matlab)

```
npop = 100;  
ngenes = 40;  
pcross = 0.8;  
pmut = 0.01;  
ptour = 0.75;  
range = 3.0;  
maxgenerations = 100;
```

```
hfig = figure;  
hold on  
set(hfig, 'Position', [50,50,500,200]);  
set(hfig, 'DoubleBuffer', 'on');  
axis([1 maxgenerations 2.5 3]);  
hbestplot = plot(1:maxgenerations, zeros(1,maxgenerations));  
htext = text(30,2.6, sprintf('best: %4.3f', 0.0));  
hold off  
drawnow;
```

```
population = initpop(npop,ngenes);  
neval = 0  
for gen = 1:maxgenerations  
maxfitness = 0.0;  
for i = 1:npop  
x = decode_chromosome(population,i,range,ngenes);  
fitness(i) = evaluate_individual(x);  
neval = neval +1;  
if (fitness(i) > maxfitness)  
maxfitness = fitness(i);  
best_individual = i;  
xbest = x;  
maxfitness, neval  
end  
end
```

```
plotvector = get(hbestplot, 'YData');  
plotvector(gen) = maxfitness;  
set(hbestplot, 'YData', plotvector);  
set(htext, 'String', sprintf('best: %4.3f', maxfitness));  
drawnow;
```

```
temp_pop = population;
```

```
temp_pop(1, :) = population(best_individual, :);  
temp_pop(2, :) = population(best_individual, :);
```

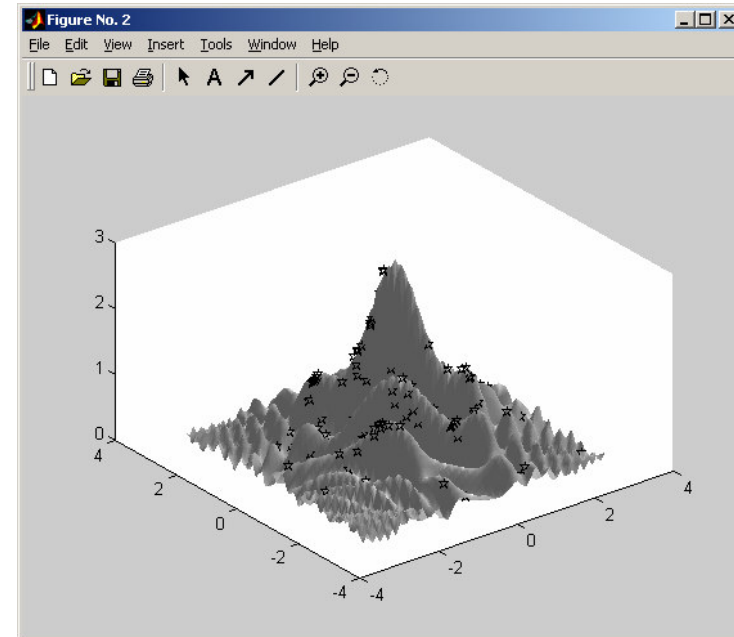
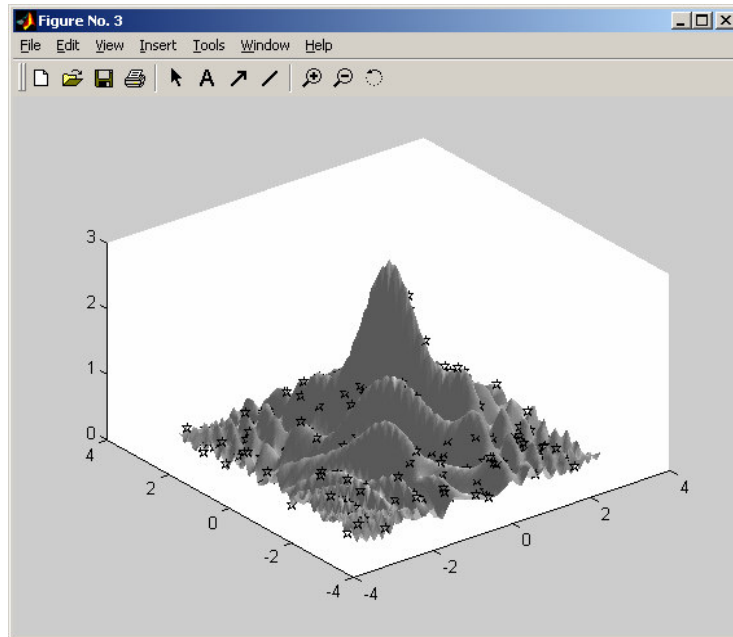
```
for i = 3:2:npop  
    i1 = tournament_select(fitness, npop, ptour);  
    i2 = tournament_select(fitness, npop, ptour);  
    r = rand;  
    if (r < pcross)  
        new_individuals = crossover(population, i1, i2, ngenes);  
        temp_pop(i, :) = new_individuals(1, :);  
        temp_pop(i+1, :) = new_individuals(2, :);  
    else  
        temp_pop(i, :) = population(i1, :);  
        temp_pop(i+1, :) = population(i2, :);  
    end  
end
```

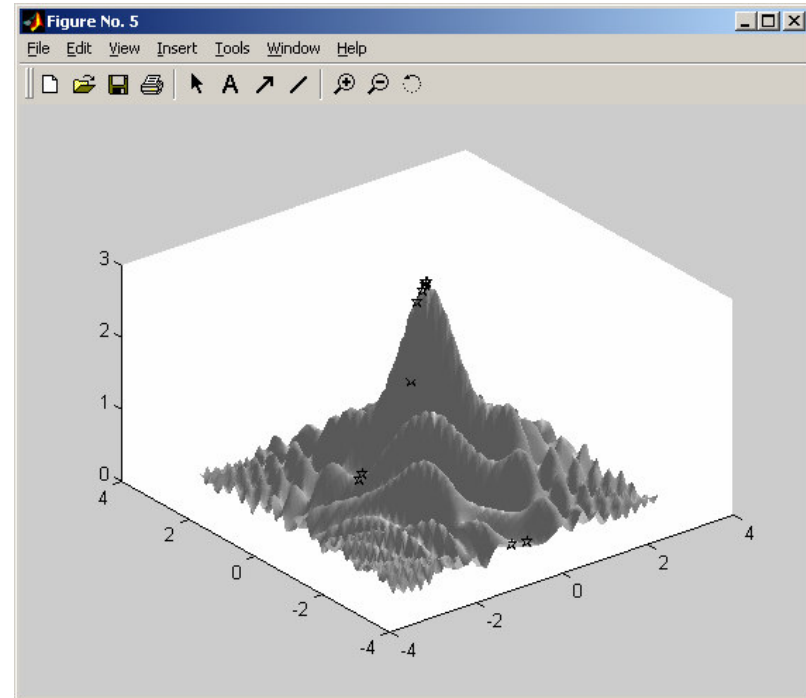
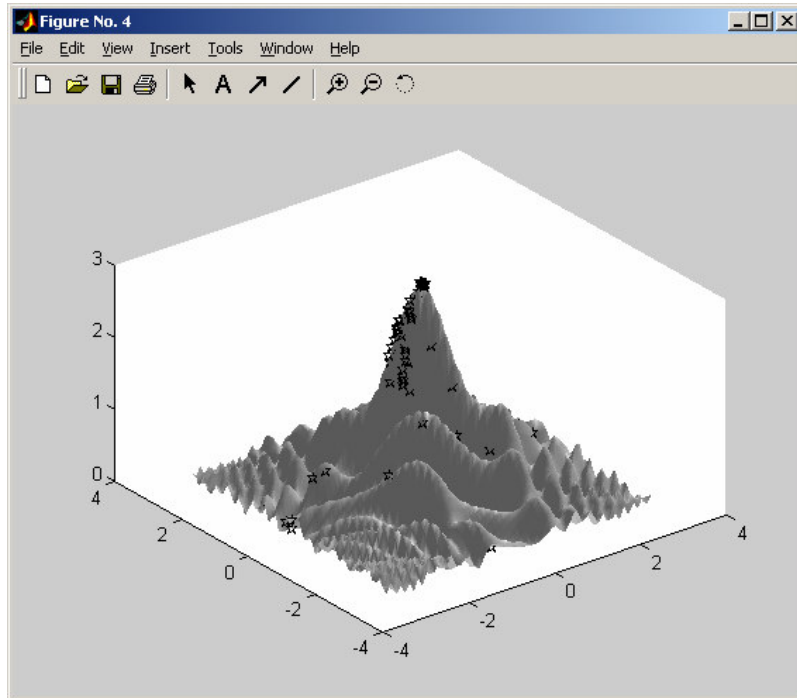
```
for i = 3:npop  
    temp_individual = mutate(temp_pop(i, :), pmut, ngenes);  
    temp_pop(i, :) = temp_individual;  
end
```

```
population = temp_pop;  
end
```

```
x = decode_chromosome(population, 1, range, ngenes)
```

## (3) With 3D graphics





3D graphics in Matlab (for more information, see e.g. the online Matlab help.)

```
npop = 200;  
ngenes = 20;  
pcross = 0.8;  
pmut = 0.01;  
ptour = 0.75;  
range = 3.0;  
maxgenerations = 100;
```

```
hfig = figure;  
hold on;  
set(hfig, 'DoubleBuffer', 'on');  
delta = 0.1;  
limit = fix(2*range/delta) + 1;  
[xg,yg] = meshgrid(-range:delta:range,-range:delta:range);  
zg = zeros(limit,limit);  
for j = 1:limit  
    for k = 1:limit  
        zg(j,k) = evaluate_individual([xg(j,k) yg(j,k)]);  
    end  
end  
  
surfl(xg,yg,zg)  
colormap gray;  
shading interp;  
view([-7 -9 10]);  
  
xind = zeros(npop,3);  
h_indplot = plot3(xind(:,1),xind(:,2),xind(:,3),'kp');  
  
hold off  
drawnow
```

```
population = initpop(npop,ngenes);
```



```
for gen = 1:maxgenerations
maxfitness = 0.0;

for i = 1:npop
x = decode_chromosome (population, i, range, ngenes);

xind(i,1) = x(1);
xind(i,2) = x(2);

fitness(i) = evaluate_individual(x);

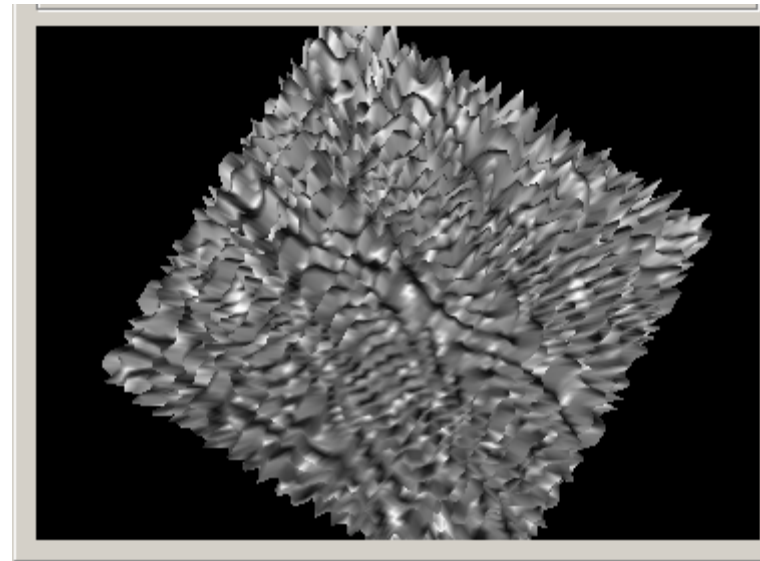
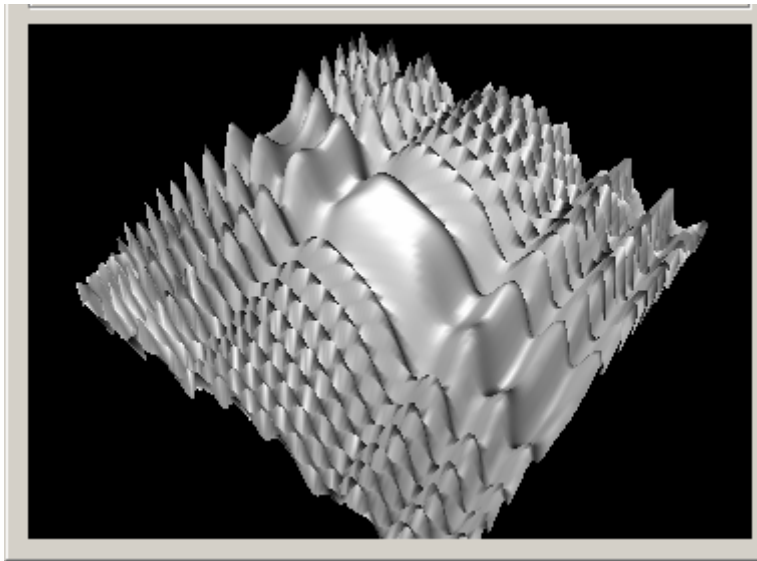
xind(i,3)=fitness(i);

if (fitness(i)>maxfitness)
maxfitness = fitness(i);
best_individual = i;
xbest = x;
end
end

set(h_indplot, 'XData', xind(:,1), 'YData', xind(:,2), 'ZData', xind(:,3));
drawnow;

ETC...
```

The function  $\Psi(x_1, \dots, x_{10})$ :



$$\psi_n(x_1, x_2, \dots, x_n) = \frac{1}{2} + \frac{1}{2n} \exp\left(-\alpha \sum_{i=1}^n x_i^2\right) \sum_{i=1}^n \cos\left(\beta \sqrt{i} x_i \sum_{j=1}^i x_j\right),$$

# Analysis of EA performance

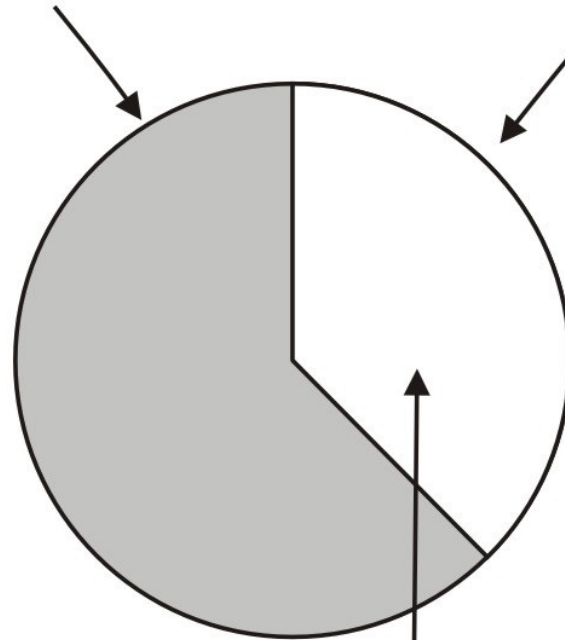
Population size	Avg. best fitness @ $10^5$ ind.	Avg. best fitness @ $10^6$ ind.
30	$0.95312 \pm 0.02832$	$0.97224 \pm 0.028194$
100	$0.97226 \pm 0.02142$	$0.99706 \pm 0.007562$
300	$0.99719 \pm 0.00718$	$1.00000 \pm 0.000000$
1000	$0.99997 \pm 0.00001$	$1.00000 \pm 0.000000$

Mutation rate	Avg. best fitness @ $10^5$ ind.	Avg. best fitness @ $10^6$ ind.
0.003	$0.92958 \pm 0.02677$	$0.93310 \pm 0.03389$
0.010	$0.95542 \pm 0.02018$	$0.97666 \pm 0.01693$
0.030	$1.00000 \pm 0.00000$	$1.00000 \pm 0.00000$
0.100	$0.89221 \pm 0.01961$	$0.95635 \pm 0.02226$

**Measure statistical significance e.g. through the average +/- standard deviation**

Individuals that  
do not contain S.

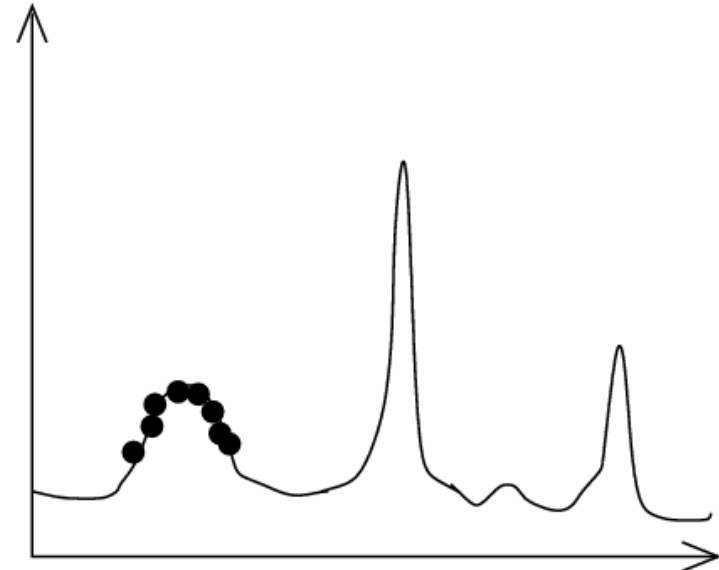
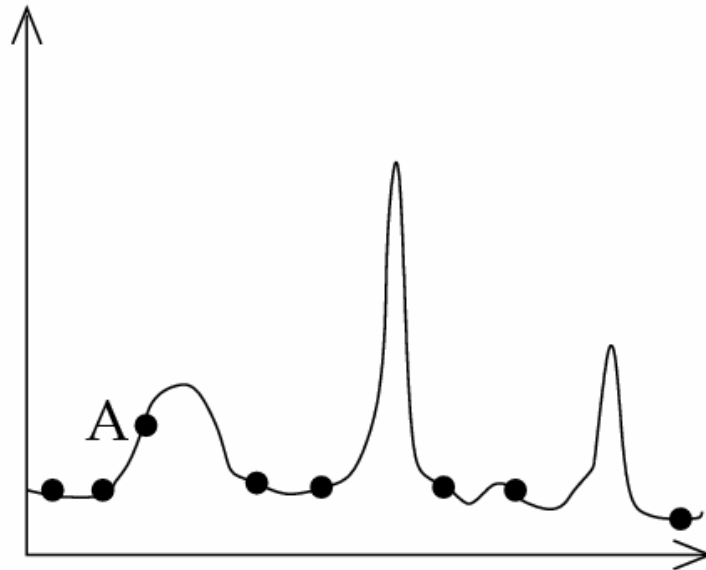
Individuals that  
contain S.



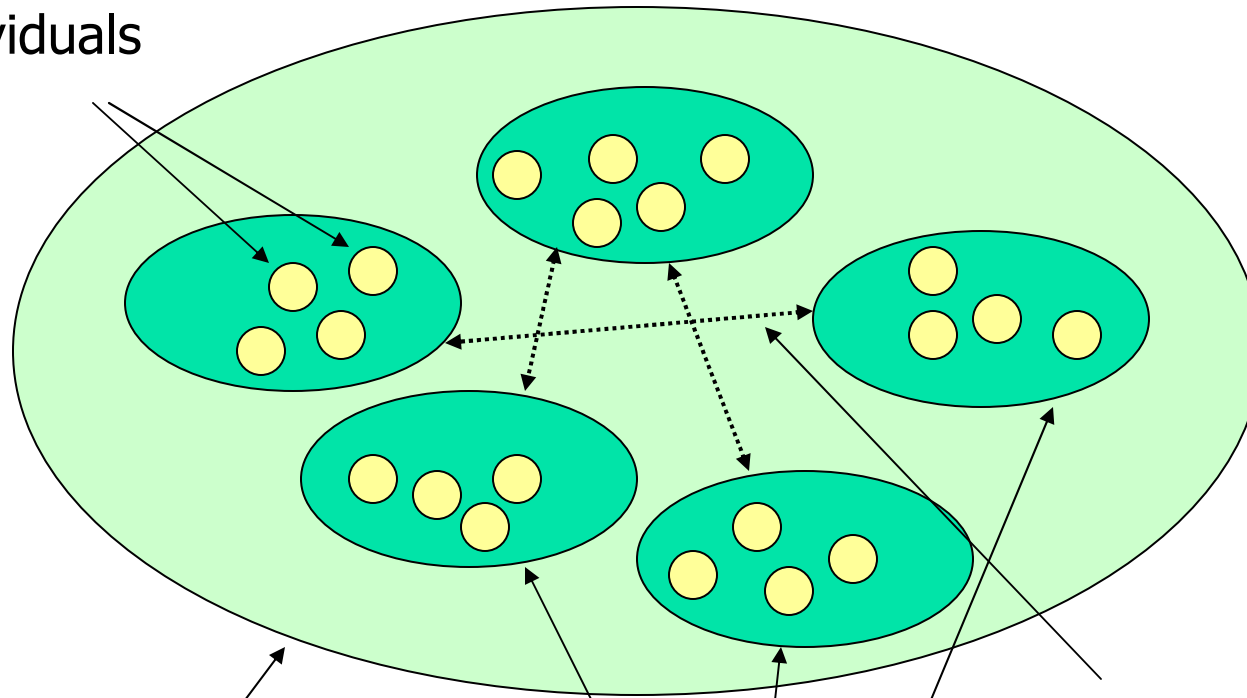
$$\Sigma f_i(S) = \bar{f}(S)\Gamma(S,g)$$

Sum of fitness values for individuals  
that contain S.

## Premature convergence:



Individuals



Population

Subpopulations

Tunneling  
(occurs with  
low probability)