Genetic Algorithms and Evolutionary Computing Project

Jasper Hawinkel & Matthias Baeten

11 january 2016

Contents

1 Answer Question 2

Write the answer for question 2 in this file

Build the report by building the file `../build/build_report.tex`

Put the figures in `../figures/`

2 Path Representation

In this section an adjusted version of the existing genetic algorithm will be studied. The tours in the Traveling Salesman Problem will not be represented anymore by the adjacency representation. The path representation will be used instead. The ith element of the path representation denotes the ith city visited. This representation needs appropriate recombination and mutation operators. We chose to use the order crossover operator as recombination operator. For mutation we chose to use the inversion operator.

The order crossover operator recombines the genes of two parents to produce two children. To produce the first offspring a randomly chosen segment of the first parent is copied into the offspring. Secondly information about the relative order of the second parent is used to make the representation of the first offspring complete. The second offspring is created in an analogous manner, with the parent roles reversed. The working process of the order crossover operator is shown below:

1. Choose two crossover points at random, and copy the segment between them from the first parent into the first offspring.
2. Starting from the second crossover point in the second parent, and copy the remaining unused numbers into the first offspring in the order that they appear in the second parent.
3. Create the second offspring in an analogous manner, with the parent roles reversed.

The used mutation operator is the inversion mutation operator. This operator works by randomly selecting two positions in the chromosome and reversing the order in which the values appear between those positions. For more information about these operators we refer to [1]. The implementation of the order crossover operator is shown in the appendices. The path representation and the inversion mutation operator were already implemented in the template Matlab program for the TSP.

3 Answer Question 4

Write the answer for question 4 in this file

4 Answer Optional Question

Write the answer for the optional question in this file

References

[1] A. E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*, Second Edition.

Appendices

In the appendices you can find a small part of the code that is used to obtain the results discussed in this report. To make sure that the appendix is not extremely long, only the code is shown that is completely written by ourselves.

Matlab Code

Implementation Order Crossover

This function recombines two parents to create two children with the use of Order Crossover.

```
function Offspring = order_crossover(Parents)

% INPUT: -Parents: 2xn matrix containing the two parents
    representation
% OUTPUT: -Offspring: 2xn matrix containing the two offspring
    representation's

n = size(Parents,2);
rn1 = randi(n);
delta = randi(n-1); % size of copied piece
Offspring = zeros(2,n);

j = rn1;
Offspring_copy = zeros(2,delta);
idx = 1;

while delta >= 1
    Offspring(:, j) = Parents(:,j);
    Offspring_copy(:,idx) = Offspring(:, j);
    if j == n
        j = 1;
    else
        j = j+1;
    end
    delta = delta - 1;
    idx = idx+1;
end

offspring1_index = 1;
```

```
offspring2_index = 1;
while j ~= rn1
    % check for position j
    cand1 = Parents(2, offspring1_index);
    cand2 = Parents(1, offspring2_index);
    while (ismember(cand1,Offspring_copy(1,:)))
        offspring1_index=offspring1_index+1;
        cand1 = Parents(2, offspring1_index);
    end
    while (ismember(cand2,Offspring_copy(2,:)))
        offspring2_index=offspring2_index+1;
        cand2 = Parents(1, offspring2_index);
    end
    Offspring(1, j) = cand1;
    Offspring(2, j) = cand2;
    offspring1_index = offspring1_index +1;
    offspring2_index = offspring2_index +1;
    if j == n
        j = 1;
    else
        j = j+1;
    end
end

end
```

Implementation fitness function for path representation

This function compute the fitness values corresponding to the candidate solutions in path representation.

```

function ObjVal = tspfun_path(Phen, Dist)
%   Implementation of the TSP fitness function
%   Phen contains the phenocode of the matrix coded in
%   path representation.
%   Phen is a matrix of size NIND x NVAR with at every row
%   the path representation of a tour.
%   Dist is the matrix with precalculated distances
%   between each pair of cities.
%   ObjVal is a vector with the fitness values for
%   each candidate tour (=each row of Phen)

size_Dist = size(Dist);
ObjVal = Dist( sub2ind(size_Dist, Phen(:,end), Phen(:,1)) );
for t = 1:size(Phen,2)-1
```

```
16      ObjVal=ObjVal + Dist( sub2ind(size_Dist , Phen(:,t), Phen(:,t
            +1)) );
17  end
18
19  end
```