Student: Eirik Baug
His group member: Martin Stigen

Project 2 - IT3708

1. As you heard in lecture, different MOEAs utilize different strategies to handle multiple objectives. You implemented MOEA have a certain strategy. In comparison to any one of the other two mentioned MOEAs, why do you think the strategy in your implemented MOEA is better? If not, why not?

For this project we chose to utilize the NSGA-II, non-dominated sorting genetic algorithm. This algorithm has the property of being fast and simple. On the other hand we have the slightly more complex SPEA, Strength Pareto Evolutionary Algorithm.

NSGA-II perfectly implements elitism in that it uses non-dominating sorting by sorting each individual after increasing domination rank. The domination rank of the individual is 1 + the number of solutions in the population the individual is dominated by. This is very easily calculated and the best individuals are always passed on to the next generation. In addition it ensures some form of diversity by implementing crowding distance, which is the distance of one individual to its neighbours with the same domination rank. We then want individuals with high crowding distance (large distance to its neighbours) as it increases the diversity of the population. We use crowding distance when when we have i individuals of rank x, but only j < i spaces left in the population.

SPEA, on the other hand, implements elitism by always keeping an archive of only non-dominated solutions. For every iterations the newly created individuals are compared with the the whole population (current + archive) and given that they are not dominated and there is more space in the archive, added to the archive. The archive should always contain only non dominated solutions and is also used for offspring creation. When the number of non-dominated solutions exceed the size of the archive an individual will need to be evicted. SPEA uses a clustering method to determine which of the non-dominated solutions to keep as it wants to keep the less crowded solutions on the pareto front.

I would argue that NSGA-II implements strong and fast elitism, while SPEA ensures greater diversity among the non-dominated solutions at the cost of slower convergence. They are both easy to calculate, and both of them would likely give good results.

I think NSGA-II is the stronger choice for most of the tasks in this project as the processing, memory and time required for each iteration is already pretty large, so a MOEA with faster convergence might be favourable, especially during the demo where time is very valuable. SPEA could, however, avoid converging to local optima as it ensures diversity among the elites while NSGA-II on the other hand only ensures diversity among the lowest ranked in the population. I believe SPEA would have been the stronger choice for some of the more difficult example images in this project as they are very hard to correctly segment with the given objectives, but could possibly be segmented better if we had greater diversity among the best individuals. I.e. NSGA-II reaches the goal faster but sometimes gets stuck, while SPEA could possibly reach better solutions at the cost of more time.

Student: Eirik Baug
His group member: Martin Stigen

2. Describe the Chromosome representation that you used in your implementation. Also, mention another representation that could be used for this problem and why this representation is also suitable. Defend your choice of the most suitable representation.

The chromosome representation used for the task is a 3-dimensional boolean array, boolean[rom][col][edge]. At index row,col you find an array of boolean edge values stating whether the pixel at row,col is connected to its neighbours or not. A true value indicates connection while false does not. The index in this array indicates the what edge it represents (0-up, 1-right, 2-down, 3-left).When we for instance use this to cross two pictures in a single-point crossover you would have to move along the crossover edge (where the two individuals has been put together) and make sure that the connecting boolean values are consistent as they need to be false, false (connection) or true,true (no connection). True, false is an inconsistent value as it states that one pixel is connected while the other state that they are not.

Another representation could be a string of connected edges. Every character in the string could be a character from 0 to 4, where 0 denotes the pixel pointing to itself and 1 to 4 denoting the pixel pointing to different surrounding pixels. Every segment would then be denoted by a single-linked cluster og edges. The number at index i would then denote the edge of pixel i in the image. This representation is simple, it is fast to convert to and small changes to the edges could have dramatic effect on the segments without sacrificing much processing power. By simply combining parts of two strings or just changing some edges in one string you could make a completely new segmentation of the image.

The problem with this representation is how difficult it is to recombine after a crossover, because when you change the string you no longer know what pixels constitutes the root nodes in the segments so you will likely have to do complex graph search finding the root node of every segment before recombining the phenotype. Our chosen representation, however, while it likely is not as fast in the crossover it is much simpler and very fast in the recombination phase as we have all the edges of every pixel and just have to make sure that they are consistent with each other which is why we ended up choosing this representation.

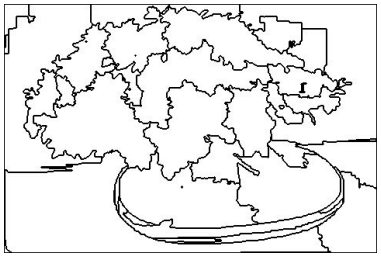
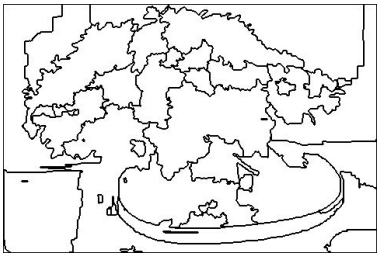
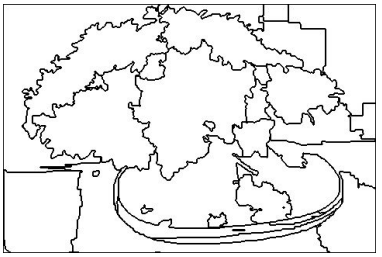
3. Using different set of weights in your multi-objective segmentation approach, explain the effects of weights on the weighted-sum method that you observed during your implementation. You need to justify your answer using proper numerical data. You can also use figures (if necessary).

The parameters used for weighted sum:
Minimum number of segments: 1, maximum number of segments: 50, mutation rate: 0.2, iterations: 20.

Weight parameters (Edge value | overall deviation)	**Segment numbers**	**Edgevalue and deviation of solution 1**	**Average PRI score**

Student: Eirik Baug
His group member: Martin Stigen

0.5 | 0.5	50, 50, 49, 50, 50	868211 | 4702042	75.52%
1.0 | 0.0	50, 50, 50, 50, 50	888325 | 5360664	74.29%
0.0 | 1.0	50, 50, 50, 50, 50	802746 | 4770668	75.23%

0.5 | 0.5	**1.0 | 0.0**	**0.0 | 1.0**
		

As the data show there isn't a whole lot of difference in the number of segments for the solutions as no matter what the algorithm will converge towards the highest number of segments possible. This is because, overall deviation and edge value both favor a large amount of segments. Overall deviation is calculated by the color difference within each segment. It is the RGB distance from the centroid (segment color) to every pixel in the segment. If it is 0.0 the whole segment has the same color and that is optimal. Therefore in an image with lots of different colors a large amount of segments with only the same colors is preferred as this would make the deviation score as low as possible.

Edge value, which is the contrast between the segment borders needs to be maximized and also increases the number of segments, because the algorithm favors a lot of border pixels with large contrasts. If you only have two large segments with one border between them the edge value will not be as large because there isn't a lot of border pixels for the edge value to sum up to. A large amount of segments with contrasting borders, on the other hand, will have a large amount of border pixels so the algorithm will naturally converge to large borders (many segments) with high contrast between them.

Other things to notice is when you completely optimize for one objective you still get decent values for the other objective. This is because the objectives are strongly connected. They converge towards the same amount of segments and while one makes sure the insides of a segments is as similar as possible the other makes sure the outside is as dissimilar as possible, meaning that they are both fairly easy to optimize together since one compliments the other. It's not a case of you can only have one or the other, since they don't work against each other as optimizing for one does not come at the cost of the other. Visually, optimizing for both objectives seemed to give the best result, but there wasn't a whole lot of difference. The same goes for the PRI value which again goes to show how well these objectives compliment each other. You can get near optimal segmentation by using only one of the objectives as they both prefer the same type of segments.