Thomas Langerak
604914

Assignment 9-3: Optimizing Link Architectures

Introduction

The goal of this assignment to first of all create an objective function to judge the quality of a link architecture based on information scent. Secondly we had to use this function to optimize a given website. Since links cannot all be presented at every page it was defined that the number of links per page, should be more than 2 and less than 6. During this assignment it was furthermore assumed that parents do not automatically link to children and vice versa. Practically, no attention was paid to the given structure, when optimizing the link architecture. Another big assumption that was made, was that page 1 (electronics) was the homepage of the website and the users always start here. This greatly reduced the computation needed for my objective function.

The given were a site structure (not used), the frequency how often a webpage was visited and the semantic relevance between to pages.

In the report I will go more in detail about the objective function that was defined, the optimization algorithm that was used and the corresponding results for $2<k<6$, with k being the amount of links per page.

Thomas Langerak
604914

Assignment 9-3: Optimizing Link Architectures

Functions

Objective Function

For the objective function the following assumption was made, which was quite critical in the design process; the hyperlinks on the page are 100% about the target, this results in the semantic pairwise relationship being equal to the information scent.

The first challenge was to define how the link architecture looks like. Throughout both the objective function and genetic algorithm three main notations are used. In the genetic algorithm the only notation used is a string (e.g. 'ahgefowelfj') in which each letters corresponds with a link *to* a specific page. It does not matter whether this letter is point from page 1 or page 2, it always will point to page x.

In the objective function this is converted to a double of [numberOfPages x k], with numbers resembling the pages (a=1, b=2, c=3,…,etc.), therefore each row has k elements of number pointing to a new page.

From this a new double is constructed for path finding (explained later). This new double has the size of [numberOfPages x numberOfPages] and contains either 0s or 1s. A 1 means page a links to page b, since links are unidirectional, a 0 means no link. This is for all combinations in both directions done.

The following steps are done from the moment a link architecture into the objective function:

1) calculate all possible simple paths, a simple path is defined as each path that does not visit a node (page) twice

- Throughout is assumed that the total scent of the path must by higher than 0.05, this is done by multiplying for each link that is taken along the path, the semantic relevance of the page where that link is pointing to, towards the target page.

-This could be improved by assuming that the overall scent should rise throughout the path, this is not implement however.

-This part is heavily based on the answer given on the question asked here: http://stackoverflow.com/questions/28824640/find-all-possible-paths-in-a-graph-using-matlab-brute-force-search

2) for each path calculate the probability the correct link on a page is pressed and sum for all paths for a target.

- The probability of a link being pressed is defined:

$$\Pr(l|t) = \frac{semantic\_relevance\_to\_target_l}{\sum_i semantic\_relevance\_to\_target_i}$$

$$i\ being\ all\ links\ on\ a\ given\ page.$$

- Probability of a path is a multiplication of all links need for that path.
- Probability for a target is the sum of all probabilities of a path

3) Calculate the quality of a link architecture.

- Weighted average of all targets (pages) in a architecture.
- *Something goes wrong here, since my average can be >1, this should not be possible I think.*

Thomas Langerak
604914

Assignment 9-3: Optimizing Link Architectures

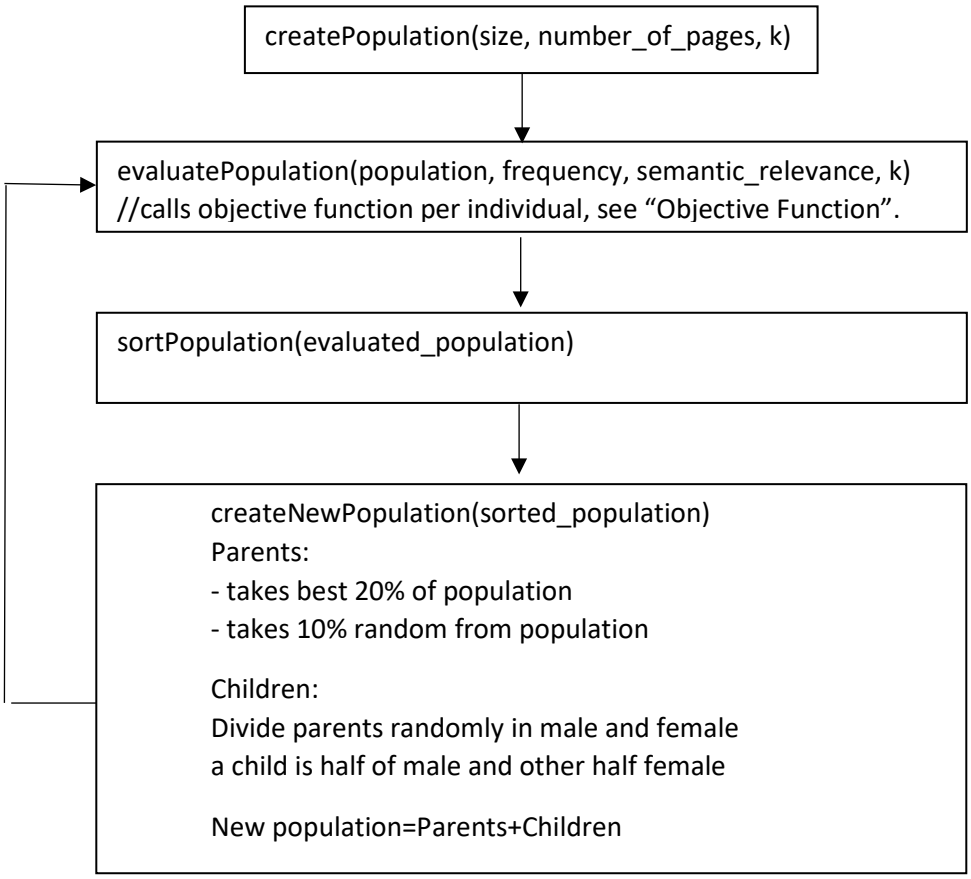
Optimization Algorithm

For the optimization part of this assignment I used a very simple version of a Genetic Algorithm (GA). I have chosen for this algorithm for two reasons. Below can be found a flowchart of this algorithm (simplified).

First of all, I have used it in a previous assignment. Therefore I knew its structure and how to implement it. This made using the GA fast and easy, yet still good. This was important, since the focus for this assignment mainly lay on the objective function, I did not want to waste time on this.

Secondly, when comparing a GA to the greedy search algorithm (the other optimization algorithm I have used before), the GA relatively a few times needs to call the objective function. This was one of the most important factors, since my objective function is far from optimal and takes quite some time to execute. When this function is called less often, my overall program comes quite a lot faster.

One of the things that I did not implement in the GA, although I should have for proper execution, is the mutation factor. Currently my GA does not mutate throughout the generations and only tries to find optimal combinations of parents. Making it heavily reliant on the initial link architectures. This could have major influences on the results gotten.



Thomas Langerak
604914

Assignment 9-3: Optimizing Link Architectures

Results

I have used rather arbitrary numbers for my optimization algorithm. I have not based them on anything, so there is definitely improvement to be gained here. I have evaluated my model for k=3,k=4 and k=5. Nonetheless it should work for all numbers of k, with 0<k<number of pages. The population size used is 100, the number of generations is 150. The best 20% o in addition to 5% random individuals of the population will be parents.

The numbers in the graph below correspond to the following pages:

1	Electronics
2	Accesoires
3	Household
4	Displays
5	Computing
6	Gifts
7	Jewelery
8	Refrigerator
9	Doshwasher
10	Television
11	Computer Monitors
12	Computer
13	Ipod
14	Diamond Rings
15	Gift Vouchers
16	Wrapping Papers
17	Emerald
18	Sapphire

Thomas Langerak
604914

Assignment 9-3: Optimizing Link Architectures

K=3

In the initial population the worst fitness was 1.3389. After optimizing the best fitness was 1.4904. This corresponds with the following link architecture.

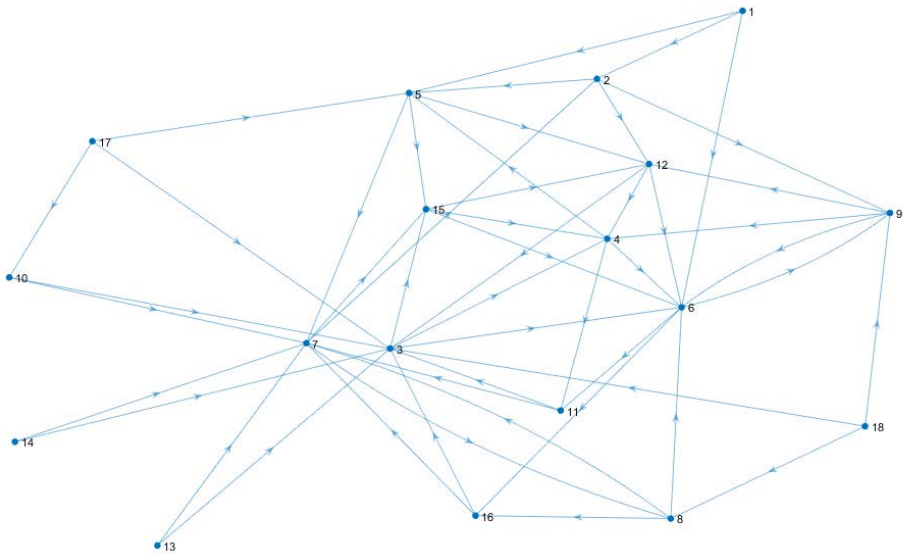


figure 1. Link Architecture k=3

K=4

The initial population had an architecture with the worst fitness of 1.2826. The final best architecture was 1.6257. This corresponds in the following link architecture.

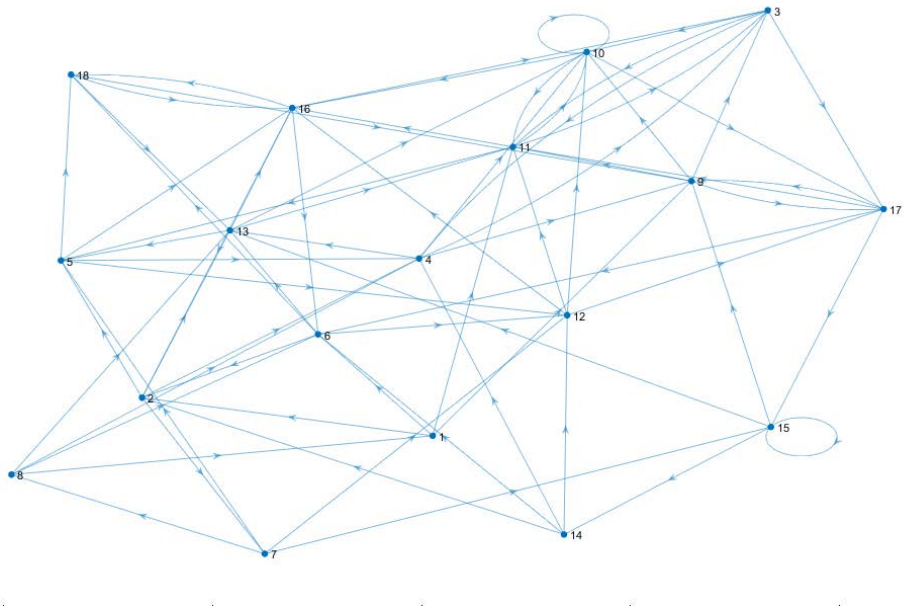


Figure 2. Link architecture k=4

Thomas Langerak
604914

Assignment 9-3: Optimizing Link Architectures

K=5

With k=5 the worst architecture in the original population had a result 1.0663. After 150 generations the most optimal result was 1.5057.

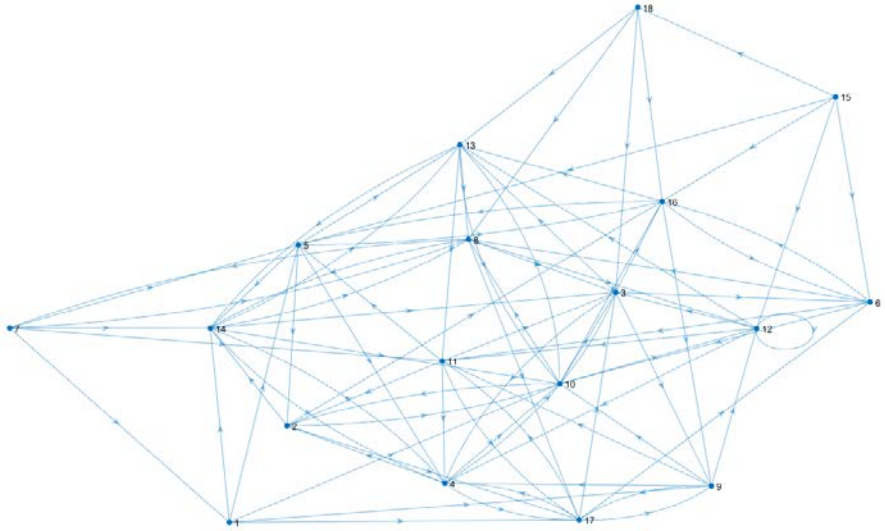


Figure 3. Results with k=5

Thomas Langerak
604914

Assignment 9-3: Optimizing Link Architectures

Conclussion & Discussion

Overall there is improvement to be seen of around 20%. Yet there are still a lot of improvements to be made.

In some of the results I have seen it can happen that a complete page is not pointed towards. This is definitely something that should not happen, therefore I recommend in the future to use the site structure and make sure that parent and child pages are always pointing to each other (this could be excluded from k), due to time constraints I did not implement that in this version.

As said early I am of the opinion that it would be logical that the scent should rise throughout the path a user is taking. Currently only the minimum total scent is implement. I think implementing also the former will give better and more realistic results. Furthermore, when implemented correctly it will greatly increase the speed of the objective function, since the number of paths per target will drastically drop.

Though I defined in the initial creation of the population I defined that a page should not be allowed to link to its self, this is not taken into account while creating the children in the GA. The result being that in the most optimal solution some pages point to themselves. This could solved probably in some different manners. Most probably adding some mutation in the GA would be enough, it could also be decided to actively excluded these children. It could be however that something went wrong when defining my optimal solution, yet I do not expect this.