Extracting Navigation Hierarchies from Networks with Genetic Algorithms

Stefan John[1], Michael Granitzer[1] and Denis Helic[2]
[1]*Chair of Media Computer Science, University of Passau, Passau, Germany*
[2]*Knowledge Technologies Institute, Graz University of Technology, Graz, Austria*

Keywords: Decentralized Search, Genetic Algorithm, Optimization, Heuristic, Network Hierarchy.

Abstract: Information networks are nowadays an important source of knowledge, indispensable for our daily tasks. Because of their size, however, efficient navigation can be a challenge. Following the idea to use network hierarchies as guidance in human as well as algorithmic search processes, this work focuses on the creation of optimized navigation hierarchies. Based on an established model of human navigation, decentralized search, we defined two quality criteria for network hierarchies and propose a genetic algorithm applying them. We conducted experiments on an information as well as a social network and analyzed the optimization effectivity of our approach. Furthermore, we investigated the structure of the resulting navigation hierarchies. We found our algorithm to be well-suited for the task of hierarchy optimization and found distinct structural properties influencing the quality of navigational hierarchies.

1 INTRODUCTION

Although the Internet has become the primary platform of information retrieval and social interaction only over the last years, the analysis of human navigation in networks in general and the Internet in particular has been part of scientific research for several decades already. One of the most famous publications initiating the research on social networks is Milgram's small-world experiment (Milgram, 1967), which led to two assumptions. First, social networks are structured in a way that arbitrarily chosen people are connected by short chains of relationships, a property referred to as the "small world phenomenon". And second, humans are able to find these short paths among the nearly endless number of possible routes.

Incited by these results, a lot of effort was put into analyzing human navigation in networks. Describing the navigational concept applied in Milgram's experiment, Kleinberg formed the notion of *decentralized search* and investigated the human ability to find short paths to arbitrary people in social networks (Kleinberg, 2000a; Kleinberg, 2000b; Kleinberg, 2002). Along with analyzing "the structure of networks in which this phenomenon emerges" (Kleinberg, 2002), Kleinberg showed that using additional hierarchical knowledge, decentralized search can efficiently be used to navigate in such networks, suggesting the existence of human *hierarchical background knowledge*.

Based on this insight, Adamic and Adar implemented a decentralized search algorithm using background knowledge for its greedy navigation (Adamic and Adar, 2005). Feeding their implementation with a naturally existent hierarchy (the hierarchical structure of an organization), they were able to confirm Kleinberg's theoretical findings. Additionally, they found their algorithm to be less efficient using background knowledge which could not be represented as a hierarchy, further pointing out the *importance of hierarchical structures* for both, algorithmic and human navigation processes. Since then, various studies have shown humans to be efficient navigators in social as well as information networks. Unfortunately, our limitation of knowledge sometimes hinders us from taking shortcuts, slowing down search processes.

Problem. Previous research has shown that background knowledge and in particular a hierarchical background knowledge is of primary importance in human navigation in networks. Traditionally, in information systems category hierarchies are often used to support user navigation. Although the goal of these hierarchies is to support navigation they are typically optimized for *semantics*. Previous work has shown that semantically optimal categories often posses desirable navigational properties—at least in the domain

John, S., Granitzer, M. and Helic, D.
Extracting Navigation Hierarchies from Networks with Genetic Algorithms.
In *Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016) - Volume 2*, pages 63-74
ISBN: 978-989-758-186-1

of social tagging systems (Strohmaier et al., 2012). However, the question if and how these results can be generalized to other information networks still remains an open one in our community.

Objective. Therefore, in this work, we take another approach for obtaining navigationally sound hierarchies. We aim for *automatic extraction* of hierarchies while optimizing their *navigational properties*. The final goal of our work is to enhance navigational support for users by embedding such navigationally optimal hierarchies in a typical user interface of an information system.

Approach. To that end, we develop an approach for evaluating navigational properties of hierarchies. We base our approach on optimization of shortest paths between arbitrary pairs of nodes in extracted hierarchies. To tackle the complexity of the optimization problem we resort to a genetic algorithm for which we develop a novel crossover rule. Finally, we illustrate our approach by performing experiments on real datasets.

Contributions. With our work we make the following three contributions:

1. A methodology for assessing the quality of navigational network hierarchies.
2. An algorithmic approach for the creation of optimized navigation hierarchies.
3. Novel insights about the structure of navigationally good hierarchies.

An implementation of the algorithmic approach is publicly available at GitHub [1].

2 RELATED WORK

In recent years, several studies have investigated the relationship of decentralized search and human navigation in networks. For our work, especially those dealing with information networks were of interest.

2.1 Decentralized Search

With regard to navigation in networks, this work focuses on the concept of decentralized search first formalized by Kleinberg (Kleinberg, 2000b). Accordingly, a search starting at node s of a network has to be performed without global knowledge of the network. In order to reach a target node t, only local information may be used. This comprises knowledge of the direct neighbors of the currently visited node as well as some intuition about their distance to the target. Limited to local decisions, decentralized search is an iterative process. In each step, the search progresses by greedily selecting the neighbor closest to the target.

In the hierarchical network model introduced by Kleinberg (Kleinberg, 2002), a hierarchy of the studied network is used to measure distances. The length of a shortest path connecting two nodes in that hierarchy determines their distance. Therefore, each greedy step of a search depends on the nodes “proposed” by the hierarchy. Assuming the existence of a hierarchically structured background knowledge, this model can be applied to human search processes.

2.2 Model of Human Navigation

Gamifying the navigational concept of decentralized search, West and Leskovec studied human navigation in an online game[2] based on Wikipedia (West and Leskovec, 2012). Their work confirms that the efficiency of human navigation is not limited to social networks, but that we are also able to find short paths in information networks. Furthermore, it shows that people tend to visit network nodes of high degree, called *hubs*, in early stages of a search before closing in on the target.

Going one step further, Trattner et al. tried to model the observed human behavior using a decentralized search algorithm (Trattner et al., 2012). Based on the ideas of Adamic and Adar (Adamic and Adar, 2005), their approach relies on *hierarchical background knowledge* to determine the distance of nodes. To that end, they used two different types of hierarchies. Their results show that, paired with their algorithm, hierarchies built directly from the link structure of the underlying network are better suited to simulate human navigation than hierarchies built from external knowledge (like an existing semantic categorization). However, using such hierarchies, their algorithm slightly outperforms the human counterpart.

Helic et al. continued with this study by extending the greedy mechanism of pure decentralized search with stochastic decision processes (Helic et al., 2013). They invented a new selection mechanism for decentralized search, called decaying ε-greedy action selection. Applying this selection mechanism instead of simple greedy navigation, *hierarchical decentralized search* proves to be well suited for modeling human navigation. Picking up on the results of West and Leskovec (West and Leskovec, 2012), they also investigated the development of human search paths and conclude that human navigation can roughly be

[1] https://github.com/Leative/STOA (Feb. 2016)

[2] http://www.wikispeedia.net (Oct. 2015)

divided into two stages. Starting with an orientation phase, referred to as *exploration*, people try to reach familiar nodes. In this stage, while trying to reach hubs, humans are more likely to perform random decisions on where to go next. In later steps, however, as people become more confident in their background knowledge, they switch to *exploitation*, efficiently following their intuitions more often.

The above insights suggest that a the efficiency of human navigation may be increased by, for example, shortening the exploration phase. Intuitively, this can be achieved by offering additional hierarchical information extending the human background knowledge. In current information networks, a semantic categorization often serves as an additional layer of information. However, category hierarchies do not consider navigational properties of the underlying network. Therefore, this work focuses on the extraction of *navigationally optimized* hierarchies to further enhance the support of human navigation.

3 METHODOLOGY

In the following, a network is represented by an *unweighted* graph $G(V,E)$ with V being the set of nodes and E the set of edges of the network.

3.1 Hierarchy Creation

To our knowledge, the methods available to create hierarchies for information networks can roughly be split into two groups. Those following a probabilistic approach (Clauset et al., 2008) and those trying to uncover hierarchical structures inherently existing in the underlying network (Heymann and Garcia-Molina, 2006; Muchnik et al., 2007). The latter, usually rely on some notion of structural or semantic generality in order to relate the hierarchies' nodes to each other. Although these methods have shown to be viable for structural analysis or the generation of semantic overviews, our work follows a different path.

Rather than retaining semantic or structural relations, our approach seeks to optimize the quality of the produced hierarchies with respect to definable quality criteria. Relying on genetic algorithms, we perform a *global search* on the set of possible hierarchies of a network. This approach allows for an easy adaption to different use cases. By defining custom quality criteria, the created hierarchies can be optimized for the task at hand.

Focusing on the enhancement of human navigation, the quality criteria proposed in this work are tailored towards the navigational optimality of hierarchies. While most likely loosing structural information of the original network, this optimization facilitates the application to human search scenarios as well as algorithmic search processes based on hierarchical decentralized search.

3.2 Quality Criteria

In the context of decentralized search, efficiency can be defined considering the length of the search path used to succeed in a search. At the same time, an agents success rate determines her effectivity. Ideally, hierarchies created by our approach should increase both, the efficiency as well as the effectivity of a navigating agent.

In order to gain maximum *effectivity*, our created hierarchies have to meet a simple constraint. To facilitate success regardless of the search scenario, each pair of network nodes needs to be connected by at least a single path in the hierarchy. For this reason, we decided to restrict ourselves to *spanning trees*, for which this property naturally holds. Besides being optimal regarding search effectivity, this restriction also vastly reduces the search space for our optimization algorithm. As a drawback, however, to create spanning trees on directed networks we had to treat them as undirected.

As per above definition, improving the *efficiency* of a search requires an agent, either man or machine, to find a short path to the target. Intuitively, proposing a node lying on a shortest path to the target in each step of the search would yield unbeatable results. In general, however, many edges of the underlying network have to be sacrificed to create a meaningful hierarchy. Thereby, a lot of shortest paths become unavailable. To counter this problem, our work proposes two quality criteria for navigation hierarchies aimed at preserving as many of the shortest paths of a network as possible.

3.2.1 Global Stretch

For our first criterion, we resort to a measure used in a similar context by Helic et al. (Helic et al., 2013) and refer to it as *global stretch*. Given a spanning tree T of a network $N(V_N,E_N)$, for two nodes s and t we compare their distance $d_T(s,t)$ in the tree with their distance $d_N(s,t)$ in the network. In each case, the distance is measured by the length of a shortest path between s and t. We call this measure $\tau(s,t)$ the *local stretch* (eq. 1). By averaging over all combinations of network nodes, where n is the number of nodes in the network, we get the global stretch (eq. 2).

$$\tau(s,t) = \frac{d_T(s,t)}{d_N(s,t)}, s \neq t \tag{1}$$

$$\tau(T) = \frac{1}{n(n-1)} \sum_{\substack{s,t \in V_N \\ s \neq t}} \frac{d_T(s,t)}{d_N(s,t)} \tag{2}$$

Although it does not make a general statement about the quality of a solution, since we do not know the possible optimum, global stretch enables us to compare hierarchies.

3.2.2 Local Tree Fitness

Local tree fitness, is a bit more complex. Instead of comparing complete paths, we consider the neighborhood and the possible navigation steps proposed by the hierarchy at each node. $N(s)$ being the set of neighbors of a node s in our network, we first have to define two subsets of neighbors. Those which are lying on a shortest *network* path from s to a target node t (eq. 3) and those lying on a shortest path in the evaluated *tree* (eq. 4).

$$M_N = \underset{n \in N(s)}{\operatorname{argmin}}\, d_N(n,t) \tag{3}$$

$$M_T = \underset{n \in N(s)}{\operatorname{argmin}}\, d_T(n,t) \tag{4}$$

Considering a human navigation being stuck at node s, with these sets we can calculate the probability $f_L(s,t)$ (eq. 5) that the spanning tree proposes a node as the next navigation step which is lying on a shortest path to target t in the original network. As we did for stretch, we average over all source-target pairs to gather the overall local tree fitness $F_L(T)$, defined in eq. 6.

$$f_L(s,t) = \frac{|M_T \cap M_N|}{|M_T|} \tag{5}$$

$$F_L(T) = \frac{1}{n(n-1)} \sum_{\substack{s,t \in V_N \\ s \neq t}} f_L(s,t) \tag{6}$$

4 OPTIMIZATION ALGORITHM

Both of the presented quality criteria consider shortest paths, in general requiring computationally intensive *all pairs shortest path* calculations to be conducted on the studied network as well as on evaluated hierarchies. However, restricting the hierarchies to be spanning trees—thereby treating all networks as undirected—simple *breadth first search* can be used for this purpose. Although this restriction also decreases the search space of our global search, the upper bound for the number of possible spanning trees of a network with n nodes is still n^{n-2}, according to Caley's formula. Therefore, optimizing the quality of navigation hierarchies still constitutes a hard optimization problem.

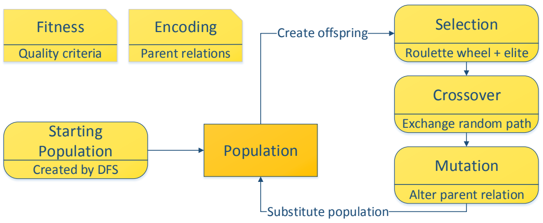


Figure 1: Overview of the spanning tree optimization algorithm (STOA).

Because of their versatility and modularity we decided to tackle this problem by applying *genetic algorithms*, a heuristic optimization approach. Genetic algorithms mimic natural evolution processes. In particular, they follow Darwin's theory of natural selection to optimize a *population of solutions*. Generation by generation, good individuals, hierarchies in our case, are chosen to "breed" offspring, while bad individuals are condemned to extinction.

Genetic algorithms are highly extensible and offer several ways to allow this natural selection to happen. However, our *spanning tree optimization algorithm (STOA)* mainly relies on the core components of simple genetic algorithms as described by Goldberg (Goldberg, 1989). Figure 1 shows an overview of our algorithm. The optimization process starts by providing a *starting population* of solutions (hierarchies). In each iteration (generation) of the algorithm, the *fitness* (quality) of all individuals of the current population is evaluated. A *selection* mechanism creates an *intermediate population* from which random *matings* are drawn. During the following *reproduction* step, the mated individuals are subject to two *genetic operators*. *Crossover*, where parts of the genetic information of the mates are interchanged and *mutation*, randomly altering single "genes" of solutions. In the following, the components of STOA are explained in more detail.

4.1 Starting Population

The creation of a starting population is the first task which has to be accomplished in order to run a genetic algorithm. As proved by Lovász (Lovász, 1993), random spanning trees could be generated with equal probability by conducting random walks on the underlying network. However, in large networks, random walks can take a considerably long time to visit every node. For this reason, we chose a less random, yet simple and fast approach. After selecting a random node as root, a *depth first search* is conducted, naturally generating a spanning tree. This approach is biased towards linear spanning trees, i.e. trees which are deep but not broad. Although leaving room for improvement, we accepted this approach to be sufficient for our study. Ultimately, the experiments showed that our recombination and mutation mechanisms can overcome the linearity in the first few generations.

4.2 Fitness

In genetic algorithms a fitness function is used to determine the fitness of individuals. Since our goal of optimization was to produce hierarchies of high quality with regard to the defined quality criteria, these quality criteria were also used as fitness functions for the algorithm.

4.3 Encoding of Spanning Trees

In our use case, the hierarchies we want to optimize constitute the solutions (individuals) the genetic algorithm is working with. Similar to nature, solutions in a genetic algorithm need to be defined by a set of “genes“ on which changes can be performed. This is usually done by choosing a proper encoding rather than working directly on the parameters relevant to the fitness function. We based the encoding on the work of Carvalho et al. (Carvalho et al., 2001). Considering a spanning tree to be rooted and directed, we encode it by the set of parent relations of its nodes. Each parent relation $P(a,b)$, representing an edge from a to b in the tree, can be considered a gene, constituting the smallest alterable unit of a tree.

4.4 Roulette Wheel Selection with Elitism

Being responsible for the selection of parents for the reproduction, the selection mechanism is crucial for every genetic algorithm. To this end, we rely on a commonly used strategy, known as *roulette wheel selection*. Enforcing survival of the fittest, individuals are chosen for an intermediate population with a probability proportional to their fitness value. However, while being an intriguingly simple concept, this purely probabilistic approach does not *guarantee* for the best individuals to be selected. For this reason, we extended the simple roulette wheel selection by the concept of *elitism*.

With elitism, the best k individuals of a generation (where k has to be specified as a parameter) are directly copied to the offspring, guaranteeing their genetic information to be available in the next iteration. Additionally, these individuals are still eligible for selection.

4.5 Random Path Crossover

Working with spanning trees as individuals imposes two requirements on viable recombination algorithms:

i) Spanning tree properties have to be preserved.

ii) Valuable partial structures of good trees should not be destroyed.

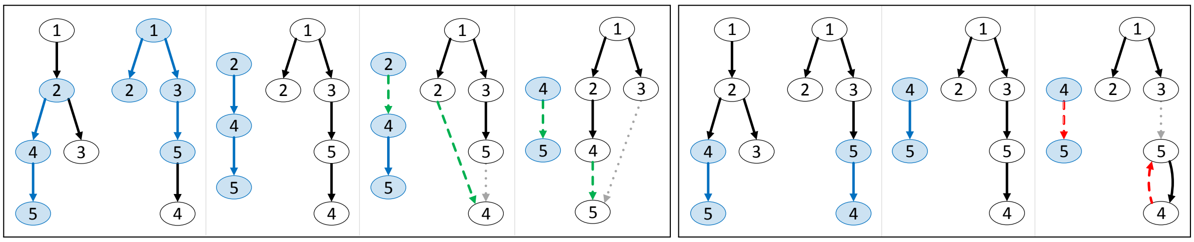
Built upon their proposed encoding, Carvalho et al. also presented a recombination algorithm, which satisfies both of these requirements (Carvalho et al., 2001). In the scenario of planning electrical distribution networks, which can also be formulated as a spanning tree problem, their approach outperformed the combination of binary encoding and one-point crossover (a standard approach for genetic algorithms) as well as an approach of randomly generating new spanning trees from the set of edges of the parent trees. It also has shown to be effective in moving towards better solutions in both theoretical and real world problems. Based on their work, we implemented a crossover operator which we refer to as *random path crossover*.

4.5.1 Mating

In random path crossover, pairs of trees are drawn at random from the intermediate population created by the selection mechanism. In case of an odd number of individuals in the population, the remainder of this mating passes the crossover process unchanged. Some approaches to genetic algorithms make use of a *crossover probability*, i.e. there is a fixed chance that a couple stays unaffected by crossover. As will be shown in the next section, our process of interchanging genetic information is not guaranteed to succeed. Inherent to the interchange algorithm, for a pair of trees a slight chance already exists to not being changed by random path crossover. Therefore, we decided to always apply crossover to all of the chosen pairs, as a side effect reducing the effort needed for parameterizing STOA.

4.5.2 Recombination

To recombine genetic information between the mated individuals, random path crossover interchanges ran-



(a) Valid crossover

(b) Invalid crossover

Figure 2: Concept of crossover. In the first section of each subfigure, a path between two nodes (blue filled nodes) is chosen for both trees. Afterwards, crossover tries to inject the path of the first tree into the second one (gray dotted lines indicate the original parent relation which is about to be substituted). While in subfigure (a) the injection succeeds for both path elements (injected parent relations are depicted by green dashed lines), subfigure (b) shows an invalid try. Injecting the red dashed parent relation would create a circle.

domly chosen paths of the trees. Originally, our encoding represents a spanning tree as a rooted tree and therefore imposes a direction on its edges. To choose paths, we have to abstract from this representation for a moment and consider the parent relations to be undirected. A path can then be defined by the parent relations connecting two nodes.

The algorithm randomly selects two nodes and identifies the paths connecting them in both of the parent trees. To exchange genetic information, the parent relations defining the path of the first tree are injected into the second tree (figure 2a) and vice versa. To preserve the spanning tree properties, and thereby fulfill requirement (i), each injection is tested and rejected if substituting the parent relation would result in a cycle (figure 2b). The order at which the crossover tries to substitute the single relations of a path is important, here. To avoid the rejection of valid paths, the parent relations have to be substituted in ascending order of their distance to the root, i.e. edges closer to the root of a path's tree have to be chosen first.

If a path is rejected during a recombination attempt, the process is canceled and a new pair of random nodes is chosen. This procedure is repeated until either a couple of interchangeable paths is found or a threshold of attempts is reached. In the latter case, both trees are passed unchanged to the next step of the genetic algorithm. Depending on the threshold, this is usually a rarely occurring event.

By exchanging paths, random path crossover only considers coherent structural units for the recombination of trees. In doing so, in accordance with the second requirement (ii), structures responsible for a high fitness are likely to either remain unchanged or to be transferred to the partner tree.

4.5.3 Random Root Switching

Performing the above recombination, a path can never be injected if it contains a parent relation where the root node of the mated tree is the child. This can be shown considering the following scenario. Let r be the root of tree T_2. Furthermore, let $P(a,r)$ be the parent relation found in the path chosen for tree T_1. During recombination, a will be chosen to substitute the original parent of r in T_2. Since r is the root, it is an ancestor of all other nodes in T_2. Therefore, selecting any of the other nodes as parent of r results in a cycle, which in turn is followed by the rejection of the path.

If the root nodes chosen for the initial population are never changed during an evolution, this behavior biases the exploration of the search space. Certain solution variants will be harder to generate than others, due to the limited ways of combining mated trees. Hence, before applying recombination, a new common root node is chosen randomly for the mated trees.

4.6 Random Edge Mutation

To prevent an early stagnation of the search, we implemented a mutation operator suitable for spanning trees. The genes of a tree, i.e. its parent relations, are independently mutated with a fixed mutation probability, which can be set as parameter for the genetic algorithm. Excluding the root of the tree, mutation of a parent relation is done by randomly selecting a new parent for the respective child node from the set of its neighbors in the original network. Depending on the actual structure of the spanning tree, not all of the neighbors are valid choices. Mutation attempts which would introduce cycles are ignored and, as long as other neighbors are available, another parent is tried. A relation remains unchanged if all neighbors were tested without success. Similar to crossover, the order in which parent relations are mutated influences the outcome. To prevent any unwanted bias, the list of parent relations is shuffled before applying mutation.

5 EXPERIMENTAL SETUP

To analyze the effectivity of our approach and gain insights on the structural properties of navigational hierarchies, we conducted experiments with two medium sized test networks. First, a condensed version of Wikipedia, which was already part of the studies of West et al. (West et al., 2009; West and Leskovec, 2012). Second, a sample of the Facebook network, collected by McAuley and Leskovec in the course of an analysis of social circles (McAuley and Leskovec, 2012). Both networks are offered by Stanford University as part of the *Stanford Network Analysis Project*[3]. In the following we will refer to them just as *Wikipedia* and *Facebook* network, respectively.

5.1 Parametrization Experiments

In order to gather the best possible hierarchies, we conducted a series of *parametrization experiments* to determine the settings needed to optimize the performance of our genetic algorithm. Both of the test networks are known to be scale-free and to reveal a similar structure. Hence, we based the parametrization solely on the Wikipedia network. However, due to the runtime of STOA being quadratically dependent on the size of the input network, we used a network sample to facilitate a meaningful number of optimization runs within an acceptable period of time.

5.1.1 Sampling

To guarantee the representativeness of the sample, we focused on retaining two measures of the original network which we assumed to be highly influential for the outcome of the fitness functions: its degree distribution and its density.

Changing the *distribution of node degrees* influences the availability of shortest paths in the network. Biasing the distribution towards lower degrees for example, weakens nodes of relatively high degree (known as hubs), which are important connectors and might lie on many shortest paths. This bias is likely to lead to a higher average length of shortest paths which in turn are the basis of the fitness evaluation.

The *density* for an undirected network N with n nodes and m edges, in accordance with Wasserman (Wasserman, 1994), is defined by eq. 7.

$$density(N) = \frac{2m}{n(n-1)} \tag{7}$$

It reflects how many of the possible edges are actually present in the network. In relative terms, a lower density means less available edges. This in turn can also negatively affect the average length of shortest paths. An extreme example to demonstrate this effect is that of a spanning tree of a complete network, i.e. a network where all pairs of nodes are connected by an edge. Containing only $n-1$ edges, almost all shortest paths of the original network (having a length of one) are lost and substituted by longer paths.

Furthermore, working with spanning trees, connectedness of the underlying network is a hard constraint. Accordingly, the generated sample had to be connected to be useable with STOA. Considering this restriction and our measures, we implemented and compared three sampling concepts. A bare *random walk* as discussed by Leskovec and Faloutsos (Leskovec and Faloutsos, 2006). An *induced random walk*, inspired by the *TIES* algorithm proposed by Ahmed et al. (Ahmed et al., 2011), adding additional edges to the sample in an induction step. And a *random deletion* method similar to the one described by Krishnamurthy et al. (Krishnamurthy et al., 2005).

For each algorithm we considered five different sample sizes and conducted ten sampling runs for each combination. Without going into detail here, we found the random deletion implementation to be most effective in retaining the original degree distribution. For sample sizes of at least 500 nodes (about 9% of the network size), it also showed reasonably good results with regard to the density. Hence, for our parametrization we used the best sample generated by this method.

5.1.2 Parametrization

We first had to decide on the two fundamental parameters of genetic algorithms: the size of the population and the number of generations each optimization run should evolve. For an optimal parallel processing, we let 24 individuals form our population. Conducting several test runs with different parameter settings, we found 50 generations to be enough for the genetic algorithm to converge, i.e. new offspring solutions being only marginally better than those of the last generation. To provide meaningful and comparable results, we performed 40 independent optimization runs with the same starting population for all of the parametrization experiments.

The parametrization experiments comprised the evaluation of two parameters: the size of the elite being copied directly to the offspring and the probability by which mutation should be applied to the evolved individuals. Assuming that the prevention of degradation introduced by elitism might highly affect the usefulness of mutation, we chose to investigate and determine the optimal size for the elite first. Further-

[3] http://snap.stanford.edu (Oct. 2015)

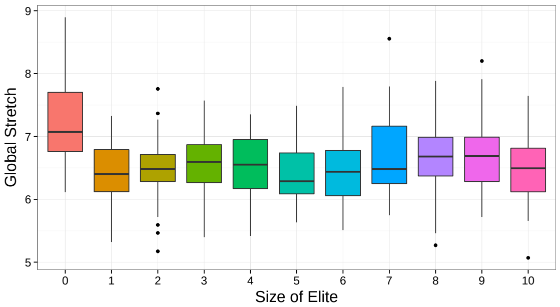


Figure 3: Global stretch results for different elite sizes. Lower values being better.

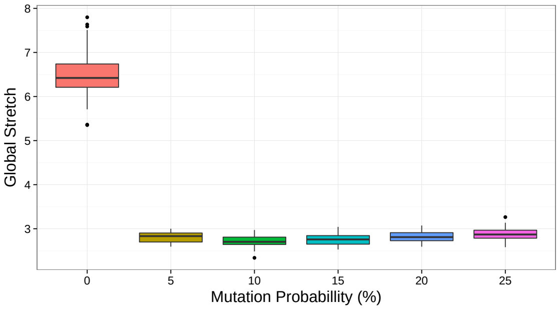


Figure 4: Global stretch results for different mutation probabilities. Lower values being better.

more, after settling both parameters, we investigated how changes to the population size and generation count affect the outcome in this setting.

The results of the optimization runs are presented for all tested parameter values as boxplots (Field et al., 2014) summarizing the fitness values of the *best* individuals of each run (e.g. figure 3). The horizontal line dividing each box denotes the median of the sample. The lower end of the box shows the value of the first quartile (25% of the results lie below that value), likewise the upper end of the box represents the third quartile (25% of the results lie above that value). Consequently, the box itself covers the 50% in the middle. The whiskers indicate the total range of values, with outliers exceeding a certain threshold plotted as separate dots.

Size of Elite. For the elite size we tried 0–10 individuals. This equals approximately 4% steps up to a value of 40% of the population. We did not find recommendations on how to use elitism in literature, but we assumed any values greater than 40% of the population size to be counterproductive. Figure 3 depicts the results for the first fitness function, the *global stretch*. Obviously, completely forgoing elitism delivers the worst optimization results. This finding confirms the negative impact of potentially replacing the best individual when evolving a new generation. According to the median, preserving the best 5 individuals for the next generation slightly outperforms other elite sizes. For higher values a small degradation can be observed acknowledging our decision to limit the test range. The situation for the local tree fitness was similar. However, the best results were achieved for a slightly larger elite with 6 individuals.

Mutation Probability. Applying the determined elite sizes, we started with a mutation probability of 0%, stepwise increasing it by 5%. As low probabilities (smaller than 20%) have shown to perform well (Haupt, 2000), we limited our experiments for that parameter to a maximum value of 25%. The results, shown in figure 4, are similar to those gained for elitism. Without mutation, the expected loss in the variety of genetic information is reflected by the creation of less optimal hierarchies. As soon as mutation is applied, the used probability plays only a secondary role. For both fitness functions, a probability of 10% produces the best individuals with regard to their median fitness values.

Population Size vs. Generations. Although with the above parameters the genetic algorithm starts to converge, for the experiments on the original networks we wanted to go beyond that point. To facilitate the algorithm to produce even better optimized hierarchies, we had two options. Either to increase the size of the population, allowing the algorithm to choose from a larger gene pool and to create more offspring in each generation, or to increase the number of generations, spending more time on refining already good solutions. To decide between those options, we performed another experiment. First, we doubled the size of our population using the parameters determined before. However, to check whether or not the size of the elite needs to be kept at a certain percentage, we included another test set where the elite size was doubled as well. Finally, we tested the algorithm's performance for 100 generations instead of 50 without changing the other parameters. Again, for both fitness functions we observed similar results (those for global stretch shown in figure 5). While all changes further increased the fitness of the best individuals, doubling the number of generations had the most notable effect. Compared to the values gathered in the mutation probability experiments the global stretch could be improved by approximately 10%. Aside from that, the size of the elite reveals to be a parameter which should be changed proportional to the population size.

5.2 Final Setup

Based on the findings of the parametrization experiments, we used mostly the same settings for both fitness functions when generating hierarchies for the

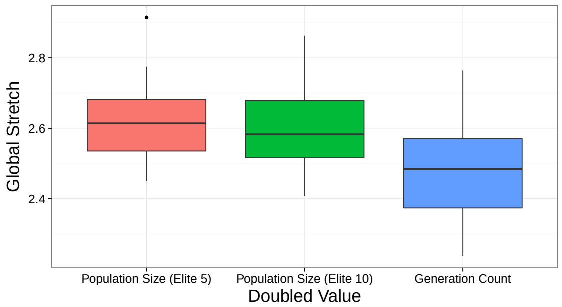


Figure 5: Global stretch results for different improvements of the optimization process. Lower values being better.

Table 1: Fitness values (global stretch) and standard deviations of the best individuals of the starting populations and the last generations.

	Wikipedia		Facebook	
	Starting pop.	Last gen.	Starting pop.	Last gen.
Best	497.47	8.54	192.06	6.16
Median	500.18	10.68	198.65	7.04
SD	1.21	1.05	11.68	0.48

original networks. For a population of 24 individuals we relied on a mutation probability of 10% and used 100 generations as a hard termination condition. The size of the elite was set to 5 individuals for the global stretch and 6 individuals for the local tree fitness, respectively. With respect to the time needed for evolving one generation on the original networks, we additionally used a stagnation soft limit, terminating the optimization when no improvement could be observed for 20 generations.

6 RESULTS

Applying global stretch as the fitness function, we performed 50 experimental runs on both original networks. The evaluation of the local tree fitness, however, showed to be about 20 times slower than evaluating a hierarchy's global stretch. For this reason, we had to rigorously limit the extent of the experiments using this quality criterion. In fact, we only conducted 5 runs on the Wikipedia network. Consequently, the following analysis of the quality and structure of optimized hierarchies, is based on the much larger data set collected for the global stretch. To compensate for this shortcoming, we analyzed the progression of the evolution processes for both fitness functions and compared the generated hierarchies.

6.1 Optimality of Hierarchies

In the course of the parametrization, increasing the number of generations showed to be an effective mean for extending the optimization process. However, in the final experiments, 74% of the evolutionary processes ended before the limit of 100 generations was reached. In average, the best hierarchy was already found after 51 generations. Without further experiments, we can only speculate whether or not an increase of the population size might yield better results for these networks, though.

Despite this discrepancy to the parametrization experiments, the optimization results are promising (table 1). Comparing the best hierarchy of all starting populations to the overall best hierarchy obtained by our experiments, a decrease in the global stretch of over 96% can be observed for both networks. Although we do not know the possible optimum values, this outcome at least indicates a good optimization performance. Since the Facebook network is about 10% smaller than the Wikipedia network, the better quality of the Facebook hierarchies is not a surprise. Interestingly, however, the relative improvement is nearly the same for both networks. This suggests that our algorithm performs well independently of the size and structure of the network. The best Wikipedia hierarchy has a fitness of 8.54, meaning its shortest paths are in average 8 times longer than those of the original network. For the smaller Facebook network, at least a value of 6.16 was reached. Taking into account that the hierarchies contain less than 5% of the network edges, these results can still be considered good. Additionally, the standard deviations for the best hierarchies of the last generation are pleasantly low. Although we have seen some outliers, our algorithm shows to deliver hierarchies of rather constant quality.

6.2 Hierarchy Structure

To get an idea about how valuable navigation hierarchies look like, we examined several measures revealing information about the structure of a network. We considered two centrality measures, which can be used as indicators for the importance of nodes. For a node v of a graph $G = (V,E)$ containing $n > 1$ nodes, we define the *degree centrality* to be the fraction of nodes v is directly connected to (eq. 8).

$$c_D(v) = \frac{deg(v)}{n-1} \tag{8}$$

Like the node degree itself, this measure reveals network hubs (high degree nodes), which have been

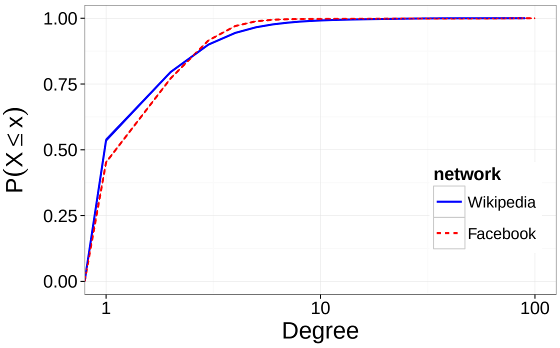


Figure 6: Average degree distribution functions of the best hierarchies. Confidence intervals are too small to be shown.

shown to be important for human navigation in information networks. In the following, such nodes will be called *degree hubs*. Furthermore, we investigated *betweenness centrality*, a measure indicating the importance of a node for the shortest paths of a network (Brandes, 2008). More precisely, in a directed network for a node v the betweenness centrality sums up the fractions of all shortest paths in the graph that pass through v (eq. 9).

$$c_B(v) = \sum_{\substack{s,t \in V_N \\ s \neq v \neq t}} \frac{sp(s,t|v)}{sp(s,t)} \tag{9}$$

While $sp(s,t)$ is the number of shortest paths between the nodes s and t, $sp(s,t|v)$ denotes the number of those paths passing through v. In an undirected network the score is typically halved to account for all pairs of nodes being considered twice. We additionally averaged that value by the total number of node combinations relevant for node v to get a more expressive and comparable score (eq. 10). We refer to this measure by the term *average betweenness centrality*. Analogous to degree hubs, in the following we refer to nodes with a high average betweenness centrality as *navigational hubs*.

$$avg(c_B(v)) = \frac{2}{(n-1)(n-2)} \cdot c_B(v) \tag{10}$$

Along with the centrality measures, we analyzed the hierarchies' degree distribution. Additionally, we compared their *diameter*, i.e. the length of the longest shortest path, to that of the networks. Naturally, we also considered the hierarchies' fitness values.

6.2.1 Good Hierarchies

First, we analyzed the best hierarchies of each run of our algorithm. We noticed that there is nearly no difference in the degree distribution between the hierarchies of one network. Furthermore, averaged over all of the hierarchies of one network, the distributions of both networks show to be similar (figure 6). While the major fraction of nodes are leaves, i.e. nodes only connected by a single edge, the optimized hierarchies also contain a small number of degree hubs. Since these have shown to be important first steps in human search paths (West and Leskovec, 2012), we further investigated them. Therefor, for a centrality measure c, a hierarchy H and a node m having the highest centrality value in H, we defined hubs of H as shown in equation 11.

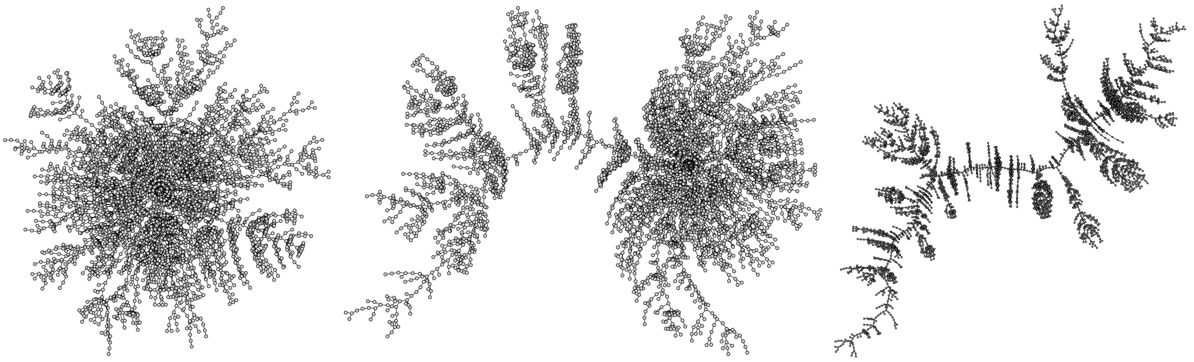
$$v \text{ is a hub of } H \iff c(v) \geq 0.75 \cdot c(m) \tag{11}$$

Although this approach might be problematic in networks without distinctive hubs, for our use case this showed to be a viable definition. Considering both centrality measures, we observed an interesting difference between the hierarchies of both networks. In all of the Wikipedia hierarchies there exists exactly 1 degree hub. At the same time, the number of navigational hubs varies between 1 and 10, with a median of 2 nodes. The Facebook hierarchies show a contrary structure. While 75% of them have exactly 1 navigational hub, in average 4 hubs of high degree can be found. We found, that in general the degree centrality of navigational hubs strongly correlates with their average betweenness centrality ($r > 0.61$). For degree hubs, however, this correlation is far less distinct ($r < 0.36$).

Analyzing the compactness of the hierarchies, in terms of their diameter, for both networks we found nearly identical, but high median values (≈ 75). Compared to the diameters of the underlying networks (Wikipedia: 5, Facebook: 8), this poses a drastic increase. To estimate how the diameter is reflected in the structure of the hierarchies, we analyzed the distribution of nodes around the most central navigational hub. We found a notably amount of nodes being situated far away of the central nodes. With this in mind, the high diameter is unlikely to be caused by single linear branches leading away from the core of the hierarchy. Instead, an overall widespread structure can be assumed.

6.2.2 The Good, the Bad and the Ugly

Calling to mind table 1, the linear structure of the starting population is obviously a degrading factor for the quality of hierarchies. Apart from this extreme, knowledge about the subtle differences distinguishing the best solutions from the second best might be of use. Either for manually improving hierarchies or estimating their optimality. To this end, we also examined the worst hierarchies of the last generation of our experimental runs.



(a) The good

(b) The bad

(c) The ugly

Figure 7: Three Wikipedia hierarchies of different quality (layouts are based on a spring model). Subfigure (a) shows the best hierarchy created, with a global stretch of 8.54. In subfigure (b) the global stretch is 11.97. A subtree splitting off from the navigational core can be observed. The last hierarchy's global stretch is 22.35. A linear structure is clearly noticeable.

Starting off by comparing the distribution of node degrees, no differences between good and bad hierarchies can be detected. A major difference, however, can be observed for the number of navigational hubs. While the *navigational core* of good hierarchies (figure 7a) is composed of very few dominant navigational hubs as well as some nodes of lower betweenness centrality, the core of bad hierarchies is distributed amongst a much larger set of nodes, skewed towards lower centrality. As discussed in the last section, this distribution and the lack of a specific central node are likely to introduce a separation of parts of the hierarchy (figures 7b and 7c). Consequently, for bad hierarchies a much higher spread, reflected by large average diameters, can be recognized. These observations suggest that the distribution of navigational hubs can be used as an indicator for the quality of navigational hierarchies. Condensed navigational cores of only few nodes with very high average betweenness centrality are beneficial. On the other hand, numerous nodes of mediocre centrality introduce linearity, which has been shown to be fatal.

6.3 Comparing the Fitness Functions

As the basis of comparison of our fitness functions, we chose one of the experimental runs in which the local tree fitness had been applied. In particular, we considered the run which created the best of all hierarchies with regard to local tree fitness. Throughout the evolutionary process of this run 3588 hierarchies were created. To get a comparable data set, we additionally calculated the global stretch for all of them. Analyzing this data, we first found the values of both fitness functions to highly and significantly correlate ($r = -0.52$, $p < 2.2 \cdot 10^{-16}$). Interpreting the correlation, care has to be taken. Both fitness functions indicate improvement in opposite directions. Therefore, the negative correlation of the fitness values corresponds to a positive correlation of the quality of the generated hierarchies. This result confirms that our quality criteria are likely to be interchangeable and, on the long run, should lead to the same hierarchies.

7 CONCLUSION

With the size and importance of information networks growing constantly, the necessity for retaining efficient navigation arises. Inspired by human example, our work focuses on network hierarchies used as means for efficient navigation. We offer a method for the creation of optimized navigation hierarchies based on genetic algorithms. Although ultimate optimality is unlikely to be reached by a heuristic approach, the hierarchies created for two sample networks are promising. Good hierarchies share structural properties. They possess a large number of leaf nodes, accompanied by a few degree hubs. Additionally, their navigational core consists of a few, extraordinary navigational hubs. From there, a rather homogenous network of nodes emerges, leading to hierarchies more widespread than their underlying networks.

By integration in user interfaces or back-ends these optimized navigation hierarchies might be valuable for guiding human navigation. Especially in early stages of a search, they can offer structural information about the network, which can be used to overcome human knowledge gaps in foreign domains. Focusing on the preservation of shortest paths, they might also be of interest for improving the performance of decentralized search algorithms.

Apart from applying optimized navigational hierarchies in practice there are other interesting aspects which should be considered in future work. Due to the modularity of genetic algorithms, STOA is changeable in various ways. As a side-effect of an early bug in the implementation of random edge mutation, for example, we conducted a few experiments with an unintentionally and randomly decaying mutation probability. Surprisingly, the optimization performed slightly better in these cases suggesting the purposeful application of a decaying mutation probability. Additionally, we did not rigorously analyze the performance of our optimization approach. Applying our algorithm to networks with known optima in future analysis, might clarify how much room for improvement is left.

REFERENCES

Adamic, L. and Adar, E. (2005). How to search a social network. *Social Networks*, 27(3):187–203.

Ahmed, N., Neville, J., and Kompella, R. R. (2011). Network sampling via edge-based node selection with graph induction. Technical report, Purdue University.

Brandes, U. (2008). On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145.

Carvalho, P. M. S., Ferreira, L. A. F. M., and Barruncho, L. M. F. (2001). On spanning-tree recombination in evolutionary large-scale network problems-application to electrical distribution planning. *Evolutionary Computation, IEEE Transactions on*, 5(6):623–630.

Clauset, A., Moore, C., and Newman, M. E. (2008). Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101.

Field, A., Miles, J., and Field, Z. (2014). *Discovering statistics using R*. SAGE Publications.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional.

Haupt, R. L. (2000). Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors. In *Antennas and Propagation Society International Symposium, 2000. IEEE*, volume 2, pages 1034–1037. IEEE.

Helic, D., Strohmaier, M., Granitzer, M., and Scherer, R. (2013). Models of human navigation in information networks based on decentralized search. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, pages 89–98. ACM.

Heymann, P. and Garcia-Molina, H. (2006). Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical report, Stanford InfoLab.

Kleinberg, J. (2000a). Navigation in a small world. *Nature*, 406(6798):845–845.

Kleinberg, J. (2000b). The small-world phenomenon: An algorithmic perspective. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 163–170. ACM.

Kleinberg, J. (2002). Small-world phenomena and the dynamics of information. *Advances in neural information processing systems*, 1:431–438.

Krishnamurthy, V., Faloutsos, M., Chrobak, M., Lao, L., Cui, J.-H., and Percus, A. G. (2005). Reducing large internet topologies for faster simulations. In *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, pages 328–341. Springer.

Leskovec, J. and Faloutsos, C. (2006). Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM.

Lovász, L. (1993). Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty*, 2(1):1–46.

McAuley, J. J. and Leskovec, J. (2012). Learning to discover social circles in ego networks. In *NIPS*, volume 272, pages 548–556.

Milgram, S. (1967). The small world problem. *Psychology today*, 2(1):60–67.

Muchnik, L., Itzhack, R., Solomon, S., and Louzoun, Y. (2007). Self-emergence of knowledge trees: Extraction of the wikipedia hierarchies. *Physical Review E*, 76(1):016106.

Strohmaier, M., Helic, D., Benz, D., Körner, C., and Kern, R. (2012). Evaluation of folksonomy induction algorithms. *ACM Trans. Intell. Syst. Technol.*, 3(4):74:1–74:22.

Trattner, C., Singer, P., Helic, D., and Strohmaier, M. (2012). Exploring the differences and similarities between hierarchical decentralized search and human navigation in information networks. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, page 14. ACM.

Wasserman, S. (1994). *Social network analysis: Methods and applications*, volume 8. Cambridge university press.

West, R. and Leskovec, J. (2012). Human wayfinding in information networks. In *Proceedings of the 21st international conference on World Wide Web*, pages 619–628. ACM.

West, R., Pineau, J., and Precup, D. (2009). Wikispeedia: An online game for inferring semantic distances between concepts. In *IJCAI*, pages 1598–1603.