SEARCH AND CREATIVE SUMMARIZATION USING GENETIC ALGORITHMS

Linda Andersson, *c99las@cs.umu.se*
Department of Computing Science
Umeå University
26th February 2004

Abstract

To address the needs for information filtering and support for creativity, a genetic algorithm is applied to, in a creative way, summarize the documents returned from a search. The application's intentions are to facilitate the user's creativity by the way the results are represented. Genetic algorithms is a search method based on the Darwinian ideas of evolution, natural selection and natural genetics. Experiments are made to optimize the genetic algorithm's performance.

Contents

List of Figures

List of Tables

1 Introduction

The amount of information facing businesses and organizations is huge and globally distributed to the extent that it is impossible for human beings to overview. It is difficult to discern new trends and combinations of areas in the body of information. The information is mostly found in databases, presented in different languages and formats, as well as in different systems of classification. Information retrieval and analysis can be improved qualitatively if information can be retrieved and compared from all types of sources. Business analyses are performed in order for an actor to be prepared for events in a turbulent business environment. Behind the concept of business intelligence hides a range of different methods and approaches. Basically, business intelligence is about selecting, collecting, structuring, and visualizing information about important conditions in the world. Most of these conditions lie beyond the control of the organization. Surveillance and analysis of strategic areas may contribute to increased readiness and facilitate decision making.

Most of the business intelligence systems today are just doing information retrieval. These systems are gathering information from several different sources and presenting it to the user at a single place. The data received are not processed or filtered in any way. Companies behind such system often brag about the amount of sources they have. Some systems are based on advanced linguistic search technologies, guaranteeing the user to get relevant information based on the search queries. Real business intelligence is about synthesizing useful knowledge from collections of data. It is about estimating current trends, integrating and summarizing disparate information, validating models of understanding, and predicting missing information or future trends. This process of data analysis is also called data mining or knowledge discovery. Typical analysis tools might use:

Probability theory	e.g. classification, clustering and Bayesian networks
Statistical methods	e.g. regression
Operations research	e.g. queuing and scheduling
Artificial intelligence	e.g. neural networks and fuzzy logic

This report brings up an approach where a genetic algorithm has been chosen for the analysis of the data. Genetic algorithms are adaptive heuristic search methods based on the evolutionary ideas of natural selection and natural genetics. They are especially suited for large complex search spaces, making internet and other sources used in business intelligence a perfect search domain. The genetic algorithm module in this system is used as a tool for creativity, presenting the analyzed data in a way that helps the user to interpret information in new and maybe more helpful ways. In this new information or knowledge society, original ideas and original results are becoming the only results that count. The only way to make real contribution, to add value, is to be creative [Janlert, 2002].

1.1 Background

This master's thesis is the final part of the Master's Programme in Computing Science and Engineering (Minoring in Cognitive Science). The thesis is performed in the project SIIBA (System for Intelligent Information Surveillance and Analysis) at the Interactive Institute, Umeå.

1.2 The Project - SIIBA

The project's purpose is to develop a system for intelligent business analysis, an expert system which should be able to detect trends, present research and development, internal and external processes, and other factors that can affect business development. The developed system for intelligent business analysis should continuously search, filter, gather, and analyze existing information. The intelligence is made up of several agents consisting of different intelligent techniques, which together gather and analyze the data available on the internet and on several databases. One module consist of a genetic algorithm which filter and analyzes the data and representing it in a different way, allowing the user to reinterpret the content.

The work is conducted in collaboration between research institutes, experts on business intelligence, content providers, information providers and biotech enterprises in Umeå. In co-operation with the company Nobicon, a portal for business intelligence and information processing is developed. The portal will gradually be endowed with news material and information based on search profiles specific to each participating company. Analyses and sources pertinent to specific companies will also be added. A number of diverse biotech companies in Umeå are forming a reference group. The common denominator is their need for information. The companies are test pilots within the project and are testing and evaluating new solutions and models, with existing systems of business intelligence as references.

1.2.1 Interactive Institute

The Interactive Institute is a multidisciplinary, innovation-oriented research institute working in the area of digital media. The Institute operates in the border zone between art, technology, science and enterprise. The goal for the research of the Institute is to develop innovative ideas and prototypes resulting in new knowledge and findings with a positive impact on society.

1.2.2 System Description

The system is dynamical and therefore uses a modular design, where each module is a sophisticated system that can be executed independently of the other modules and new modules can easily be added without need of redesign. The system's intelligence consists of the combination of the different modules.

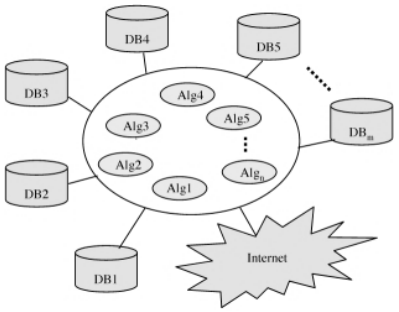

Figure 1: A basic System Architecture.

The existing system consists of four different modules; a Genetic Algorithm, a Latent Semantic Analysis-module, a statistical module with time series analysis, and a distributed meta-search engine.

The **Genetic Algorithm** (GA) currently works towards the pubMed database and generates new documents from the hits the search query results in, and from an alphabet that scores certain words. For example, a source that contains ".edu" can get a higher score than one that contains '.com', depending on the user's area of interest. The alphabet will grow and develop continuously. The API's are written in Python, the GA in C, and the generated database where documents and sentences are stored in MySQL-databases.

Latent Semantic Analysis (LSA) is a method to do a semantic model over a number of documents. The LSA-module can be trained by giving it a significant number of documents, and by mathematical manipulation get the relations between the documents or the consisting words. Then you can give it new documents to see its relation to the other text, which relative position it will get in the semantic model, and which documents that are most similar to the new document. LSA has been used within a many different areas: for advanced search in databases, thesaurus, pattern matching, and modeling of the brain's language processes.

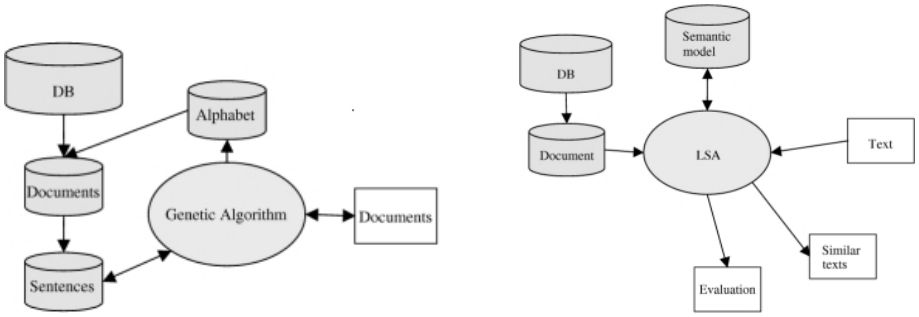

Figure 2: The GA Module and the LSA Module.

The **Distributed Meta-Search** engine renders possible emergence of more sophisticated patterns through individual search patterns. Different users have different search criteria sent to the web and their best hits are saved to a distributed database. Over time links of interest to several of the users will be collected. In this way it will be a gradual refinement of the immense amount of information available on the web.

The **Statistical** Module is a so called wrapper to the existing system, R. R is a language and an environment for statistical calculations and graphic techniques. The statistical module is responsible for statistical data analysis and contains several different models, e.g. time series. It works with a local database which gets data from Yahoo Finance. The time series is especially interesting because of the need to discover and understand trends.

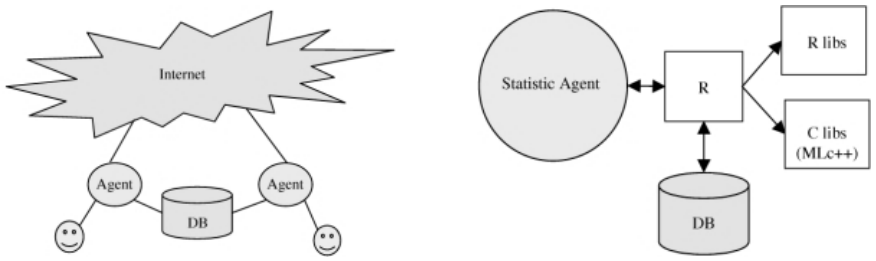

Figure 3: The Distributed Meta-Search engine and the Statistical Module.

1.3 Thesis Outline

The report starts with a literature review of genetic algorithms, and continues to examine how text generation earlier has been done and how the evolutionary principles has been used for creative aspects before. The following section deals with the design and implementation of the Genetic Algorithm-module. The experimentation section, begin with a discussion about what would be a successful result. After that the design, performance, and results of the experiments are presented. The report finishes with a discussion about the properties of the application and future works.

Part I
Literature Review

2 Genetic Algorithms

Genetic algorithms are adaptive heuristic search methods based on the evolutionary ideas of natural selection and natural genetics. The basic concept is to simulate processes in natural systems essential for evolution.

Genetic algorithms use a direct analogy of natural behavior, working with populations of individuals, each individual representing a string or a possible solution to a given problem. The individuals are assigned a fitness value representing their ability to perform a task. In nature this is equivalent to value how effective an organism is at competing for resources. The highly fit individuals have higher probability of being selected for reproduction, and the least fit individuals are thus less likely to get selected, and will therefore have relatively fewer offspring than the successful ones.

A new population is produced by selecting the best individuals from the current generation and mating them to produce a new set of individuals, with a hopefully higher fitness value. In this manner the average fitness value will increase over generations, and if the algorithm is well designed, the population will converge into an optimal individual. Genetic algorithms are not guaranteed to find the global optimal solution to a problem, but it can, under the right circumstances find acceptably good solutions within an acceptably short period of time [Beasley, 1993].

The algorithm performs the following steps to evolve the population:

1. Randomly generate an initial population.
2. Calculate the fitness for each individual.
3. Calculate the selection probability for each individual so that it is proportional to its fitness.
4. Select individuals for reproduction.
5. Insert the offsprings into the next generation.
6. Repeat from step 2 until a satisfactory solution is obtained.

The central theme of research on genetic algorithms has been robustness, the balance between efficiency and quality necessary for survival in many different environments.

These algorithms are computationally simple yet powerful in their search for improvement in complex spaces. Furthermore, they are not fundamentally limited by restrictive assumptions about the search space, assumptions concerning continuity, existence of derivates, unimodality, and other matters, properties that distinguish them from the majority of search algorithms [Goldberg, 1989].

2.1 The History of Genetic Algorithms

Genetic algorithms originated from the studies of cellular automata, conducted by John Holland and his colleagues at the University of Michigan. Holland's book, published in 1975, is generally acknowledged as the beginning of the research of genetic algorithms. Until the early 1980s, the research was mainly theoretical, with few real applications. This period is marked by extensive work in the domain of function optimization by, among others, De Jong and Hollstien. Hollstien's work provides a careful and detailed analysis of the effect of different selection and mating strategies on the performance of a genetic algorithm. De Jong's work attempted to capture the features of the adaptive mechanisms in the family of genetic algorithms that constitute a robust search procedure.

From the early 1980s the community of genetic algorithms has experienced an abundance of applications which spread across a large range of disciplines. Each and every additional application gave a new perspective to the theory. Furthermore, in the process of improving performance as much as possible via tuning and specializing the genetic algorithm operators, new and important findings regarding the generality, robustness and applicability of genetic algorithms became available.

Following the last couple of years of development of genetic algorithms in the sciences, engineering and the business world, these algorithms in various shapes have now been successfully applied to optimization problems, scheduling, data fitting and clustering, trend spotting and path finding.

2.2 Terminology - The Connection to Biology

In the context of genetic algorithms biological terms are used in the spirit of analogy with real biology. All living organism consist of cells, and each cell contains the same set of one or more chromosomes. These chromosomes can be conceptually divided into genes, each of which encodes a certain protein.

During sexual reproduction, recombination occurs. Genes from each parent are combined to form a new single chromosome. Offspring are subject to mutation, in which single nucleotides are changed from parent to offspring, the changes are often results from "copying errors". The fitness of an organism is typically defined as the probability that the organism will live to reproduce or as a function of the number of offspring the organism has.

In genetic algorithms, the term chromosome typically refers to a candidate solution to a problem, often encoded as a bit string. The genes are either single bits or short blocks of adjacent bits that encode a particular element of the candidate solution. The reproduction methods are commonly called genetic operators. The crossover typically consists of exchanging genetic material between two single-chromosome parents. Mutation consists of replacing a randomly chosen gene with a randomly chosen new gene [Mitchell, 1999].

2.3 Encoding

As for any search and learning method, the way in which candidate solutions are encoded is a central, if not the central, factor in the success of a genetic algorithm. Most applications of genetic algorithms use fixed-length, fixed-order bit strings to encode candidate solutions. The way to decide the format of the chromosome is called encoding. There are four commonly used encoding methods: binary encoding, permutation encoding, direct value encoding and tree encoding.

Binary encoding is the most common and simplest one. In binary encoding every chromosome is a string of bits, 0 or 1.

Permutation encoding can be used in ordering problems, such as the traveling salesman problem or task ordering problem. In permutation encoding, every chromosome is a string of numbers, which represents number in a sequence.

Direct value encoding can be used in problems where some complicated values such as real numbers are used. Use of binary encoding for this type of problems would be very difficult. In value encoding, every chromosome is a string of some values.

Tree encoding is used mainly for evolving programs or expressions, for genetic programming. In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language.

2.4 Fitness Function

Genetic algorithms most often requires a fitness function that assigns a score, fitness, to each chromosome in the current population. The fitness of a chromosome depends on how well the chromosome solves the problem at hand [Mitchell, 1999]. The fitness function is considered to be the most crucial part of this method. For many problems there are often many different factors to consider, which must be weighted against each other and sometimes they are contradictory. Ideally the fitness function should be smooth and regular, so that chromosomes with fairly equal abilities have fairly equal fitnesses. It is also preferable that the fitness function does not have too many local maxima or a very isolated global maximum [Beasley, 1993].

2.5 Selection Methods

The selection method's function is to pick individuals from the current generation in order to create new individuals for the next generation. The individuals are most often picked based on their fitness and the selection is usually stochastic, meaning that every individual has a probability of being selected. In principle the most fit individuals are more likely to be selected than those less fit. The purpose of selection is to emphasize the fitter individual in the population, resulting in an increase of the fitness in the next generation. The idea is to simulate natural selection [Mitchell, 1999].

Selection has to be balanced between exploration and exploitation. Exploration investigates new and unknown areas in search space, and exploitation makes use of knowledge found at points previously visited, to refine the search to find the global maximum. Too strong selection can result in a premature convergence, where suboptimal highly fit individuals may rapidly come to dominate the population, causing it to converge on a local maximum, and reducing

the ability to further change and progress. Too weak selection can lead to slow evolution. After many generations, the populations will have largely converged, but it still has not precisely located the global maximum. The best and the average individuals may be very similar, and consequently the selection would not have great effect in raising the evolution towards a maximum [Beasley, 1993].

There are different ways to change the composition of individuals. During strict generational selection, at each generation the new population consists entirely of offspring formed by parents from the previous generation, though some of these individuals can be identical to their parents. In steady-state selection only a few individuals are replaced in each generation. Usually the least fit individuals are replaced [Mitchell, 1999]. This may be a better model of what really happens in nature. In short-lived species, like insects, parent lay eggs, and die before their offspring hatch. But in longer-lived species, including mammals, offspring and parent are alive concurrently. This allows parents to nurture and teach their offspring, but also gives rise to competition between them [Beasley, 1993]. Different selection methods are described below.

Roulette Wheel selection is a fitness-proportionate selection, in which the expected number of times a particular individual is selected for reproduction, is that individual's fitness divided by the average fitness of the population. Each individual is assigned a slice of a circular "roulette wheel", the size of the slice is proportional to the individual's fitness. The wheel is spun as many times as there are individuals in the population. On each spin, the individual under the wheel's marker is selected to mate [Mitchell, 1999].

Sigma Scaling is an example of a scaling method, which keeps the selection pressure relatively constant over the course of the run rather than depending on the fitness variances in the population. Under sigma scaling, an individual's expected value is a function of its fitness, the population mean, and the population standard deviation. Initially, the standard deviation will be quite large and there will be relatively small differences in transformed fitness even if there are large absolute differences. As evolution proceeds the standard deviation reduces as the population gets fitter in average. Now small differences are important as the population converges, the fitter individuals will stand out more, allowing evolution to continue. Broad exploration of the solution space in the initial stages is replaced by more careful refinement towards the end [Mitchell, 1999].

Boltzmann selection adapts the selection pressure according to the needs at different periods in a run. Selection pressure starts out low, allowing less fit individuals to reproduce at about the same rate as fitter individuals. The selection pressure gradually increases in order to emphasize highly fit individuals, assuming that the early diversity with slow selection has allowed the population to find the right part of the search space. Boltzmann selection controls the rate of selection with a continuously varying temperature parameter. As temperature decreases, difference in expected value between high and low fitnesses increases [Mitchell, 1999].

Elitism is a selection method that forces the genetic algorithm to retain some number of the best individuals at each generation, and replacing the others with offspring as before. Thus avoiding the very fit individuals to be lost if not selected to reproduce or destroyed by crossover or mutation. Many researchers have found that elitism significantly improves the performance [Mitchell, 1999].

Rank selection is a method to prevent too quick convergence. The individ-

uals in the population are ranked according to the fitness, and a new fitness score is calculated from the rank of an individual. The conversion from rank to a new fitness score can be done linearly or exponentially [Mitchell, 1999].

Tournament selection is similar to rank selection, but computationally more efficient. Two individuals are chosen at random from the population. A number between 0 and 1 is randomly generated. If the randomized number is higher than a certain parameter, the fitter of the two individuals is selected, otherwise the less fit individual is selected. Both individuals are then returned to the population and can be selected again [Mitchell, 1999].

2.6 Genetic Operators

During the reproductive phase, individuals are selected from the current population and exposed to the genetic operators, producing offspring which will comprise the next generation. There are two basic operators; crossover and mutation.

Crossover and mutation are important in different stages of the search. Crossover speeds up the construction of good individuals. This is good in the beginning with a population that has a large diversity. When a population converge, mutation becomes more effective than crossover. When a population is near convergence, the individuals have almost identical chromosomes. Using crossover on those individuals creates chromosomes that probably already exist in the population [Beasley, 1993].

Cloning occurs when individuals are selected for mating but are exposed to neither mutation nor crossover, instead they are simply passed to the next generation.

2.6.1 Crossover

Crossover creates new individuals for next generation by recombining the chromosomes. There is a variety of different crossover operators. Single-point crossover is the simplest form, and recombines chromosomes by cutting the chromosomes at a single randomly chosen crossover point, and the parts after the crossover point at each parent's chromosome are exchanged to form two new offspring. The two offspring each inherit some genes from each parent. Multi-point crossover is like single-point except using several cutting points. A uniform crossover swaps, with a certain probability, each gene with the corresponding gene on another chromosome. Crossover can also be applied to more than two chromosomes at the same time [Goldberg, 1989].

Which of the crossover methods to chose depends often on the encoding, the fitness function, and other details specific for the particular problem [Mitchell, 1999].

2.6.2 Mutation

Mutation randomly alters a gene with a small probability. In a string of binary digits it can invert one or more of the digits. For a real valued chromosome a mutation could add or subtract some random value to one or more of the parameters. The implementation of the mutation method depends much on the encoding.

There is a challenge to find a mutation rate that will allow the population of solutions to converge to the best solution in the shortest amount of time. Too low

mutation rate can result in evolution getting stuck in a local maximum, unable to escape its current state. Too high mutation rate will tear apart any genetic adoptions of the parent to produce an almost totally random offspring. In order to evolve adaptation a middle course must be found where both extremes are avoided [Maley, 1996].

2.7 Applications

The basic theme of Genetic Algorithms have been used in a large number of scientific and engineering problems and models, for example in a wide variety of *optimization* tasks, including numerical optimization and such combinatorial optimization problems as circuit layout and job-shop scheduling. *Computer programs* have also been automatically evolved for specific tasks, and to design other computational structures such as cellular automata and sorting networks. Many *machine learning* applications have used genetic algorithms, including classification and prediction tasks, such as the prediction of weather or protein structure. It has also been used to evolve aspects of particular machine learning systems, such as weights for neural networks, rules for learning classifier systems or symbolic production systems, and sensors for robots. It has also been used to model processes of innovation, the development of bidding strategies, and the emergence of *economic* markets, and to model various aspects of natural *immune systems*, including somatic mutation during an individual's lifetime and the discovery of multi-gene families during evolutionary time.

Genetic algorithms are especially popular for model *ecological* phenomena such as biological arms races, host-parasite coevolution, symbiosis, and resource flow, and to study questions in *populations genetics*, such as "Under what conditions will a gene for recombination be evolutionary viable?". Evolutionary aspects on *social systems*, such as the evolution of social behavior in insect colonies, and, more generally, the evolution of cooperation and communication in multi-agent systems are also often studied using genetic algorithms [Mitchell, 1999].

Genetic algorithms has also more recently been used to generate images and sound. The basic idea behind *evolutionary* art is that the artist is able to control the development of a piece of work through manipulation of the selection. In all evolutionary art, one or more parent pictures or virtual sculptures are mutated and/or cross bred to produce a number of "children".

2.8 Genetic Algorithms Applied on Internet Search

Genetic algorithms has been applied on many of internet search's different functionalities. To get a better ideas of how it is done some fundamental parts of internet search is mentioned below.

2.8.1 Fundamentals in Internet Search

Search engines can save time by preprocessing the content of the web. That is, when a user issues a query, it is not sent to millions of web sites. Instead, the matching takes place against preprocessed data stored in one site. The preprocessing is carried out with the aid of a software program called a crawler. The crawler is sent out periodically by the database maintainers to collect web pages. A specialized computer program parses the retrieved pages to extract

words. These words are then stored along with the links to the corresponding pages in an index file. Users' queries are matched against this index file, not against other web sites.

The URLs or links produced as a result of searches are usually numerous. But due to ambiguities of language, the resulting links would generally not be equally relevant to a user's query. To provide quicker access to the most relevant records, and to place them at or near the top, the search algorithm applies various ranking strategies. A common ranking method known as term-frequency-inverse document-frequency (TFIDF) considers the distribution of words and their frequencies and generates numerical weights for words signifying their importance in individual documents. It produces word weights whereby words that are highly frequent, such as 'or,' 'to' or 'with', and appear in many documents have substantially less weight than words that are semantically more relevant and appear in relatively few documents.

In addition to term weighting, web pages can be weighted using other strategies. For example, link analysis considers the nature of each page in terms of its association with other pages, namely decide if it is an authority (having large number of other pages that point to it) or a hub (having large number of pages it points to). The highly successful Google search engine uses link-analysis to improve the ranking of its search results.

In order to identify the most relevant links quickly, certain search engines compare query terms to contextual information such as recent queries the user submitted. This technique is sometimes referred to as query catching, and involves collecting the words from recent queries and using these words to disambiguate, refine or expand the current query [Dharis, 2003].

2.8.2 Internet Search Applications using Genetic Algorithms

Genetic Algorithms can be used for many different purposes in information retrieval, a few examples are mentioned below.

Pathak et al. [Pathak, 2000] look at the possibilities of applying Genetic Algorithms to adapt the matching functions, used to match document description with query description. It is hoped that such an adaptation of the matching functions will lead to a better retrieval performance than that obtained by using a single matching function. An overall matching function is treated as a weighted combination of scores produced by individual matching functions. This overall score is used to rank and retrieve documents. Weights associated with individual functions are searched using Genetic Algorithms. Gordon [Gordon, 1988] uses Genetic Algorithms on a record, usually stored in a bibliographic database to describe a document, it consists of a set of subject terms or keywords. In this research, documents receive multiple descriptions in an attempt to resolve problems arising from different inquirers seeking the same document in dissimilar ways. Competing document descriptions are associated with a document and altered over time by a genetic algorithm according to the queries used and relevance judgments made during retrieval.

Billhardt et al. [Billhardt, 2002] uses Genetic Algorithms for automatically obtaining a suboptimal linear combination of retrieval experts for a given document collection. By combining the result of multiple information retrieval models or different query representations, the advantages of different models are brought together. The system automatically determines both, the retrieval

experts to be combined and the parameters of the combination function, e.g. the weights given to each expert in a linear combination.

One approach to information retrieval is to follow links from a number of documents submitted by the user to find the most similar ones, performing a genetic search on the internet. The application starts from a set of input documents, and by following their links it finds documents that are most similar to them. Milutinović et al. [Milutinović, 1998] use a concept called "genetic search with mutation" where both links from input documents and other links generated by the genetic algorithm are examined. The link's URL contains several fields that have different contents. So the URLs are represented as an array or list of these fields. Those lists or arrays are later exposed to modified variants of the genetic operators to make use of the mutation exploiting spatial or temporal locality. Spatial locality exploitation examines documents in the neighborhood of the best ranked documents so far, i.e. on the same server or local network. Temporal locality concerns maintaining information about previous search results and performing mutation by picking URLs from that set.

The research presented by Oren [Oren, 2001], attempts to find algorithms that will return documents more relevant to a user's needs, by performing a search, using genetic programming, through a portion of the space of possible techniques of weighting documents. Genetic programming is used to create new evaluation functions. These functions will then be rated based on how well they classify documents over queries taken from a training set. Functions that accurately rate documents as relevant or irrelevant will be allowed to propagate to the next generation.

3 Generating Text

The GA-module does in some sense summarize and regenerate text. To get more insight in this research area some earlier research and application is mentioned below.

The research on text generation is not a new issue. As early as 1948 did Claude Shannon [Shannon, 1948] use Markow chains to randomly generate text. In general, sentences and words have a statistical structure. For example a certain letter may occur more frequently than another, in average literature, and likewise for words. When randomly generating text, those statistical structures can be used. Every letter is chosen with a probability relative to the frequency of this letter. This method will result in a sequence of the most frequent letters, though without meaning. A more complicated structure is obtained if successive symbols are not chosen independently but their probabilities depend on preceding letters. In the simplest case of this type, a choice depends only on the preceding letter and not on ones before that. The statistical structure can then be described by the frequencies with which the various letters follow the first one. In the third-order approximation, each letter is chosen with probabilities which depend on the preceding two letters, and so on.

To give a visual idea of how this series of processes approaches a language, typical sequences in the approximations to English, constructed by Shannon, are given below. In all cases a 27-symbol alphabet is assumed, the 26 letters and a space.

Zero-order approximation (symbols independent and equiprobable).

> *"xfoml rxkhrjffjuj zlpwcfwkcyj ffjeyvkcqsghyd qpaamkbzaacibzlhjqd."*

First-order approximation (symbols independent but with frequencies of English text).

> *"ocro hli rgwr nmielwis eu ll nbnesebya th eei alhenhttpa oobttva nah brl."*

Second-order approximation (symbols dependent on the preceding letter).

> *"on ie antsoutinys are t inctore st be s deamy achin d ilonasive tucoowe at teasonare fuso tizin andy tobe seace ctisbe."*

Third-order approximation (symbols dependent on the two preceding letters.)

> *"in no ist lat whey cratict froure birs grocid pondenome of demonstures of the reptagin is regoactiona of cre."*

It is also possible to use word units instead of letters. Here is a **first-order word** approximation, were words are chosen independently, but with their appropriate frequencies.

> *"representing and speedily is an good apt or come can different natural here he the a in came the toof to expert gray come to furnishes the line message had be these."*

Second-order word approximation.

"the head and in frontal attack on an english writer that the character of this point is therefore another method for the letters that the time of who ever told the problem for an unexpected."

The resemblance to ordinary English text increases at each step. Sequences of four or more words can easily be placed in sentences without unusual or strained constructions.

Eduard Hovy [Hovy, 1993] considers that no language that communicate solely on the sentence level is adequate, and the same concerns programs. Moving the communication up a level has shown to be problematic, because when just stringing together sentences only a minority of the possible combination make semantic sense. Among the sentences which do, the meaning presumably differ widely. When the discourse is not properly structured, several interpretation mistakes can be made, and there are unfortunately no grammars of paragraph structure, and no general linguistic theories. The key insight for solving this issues is said to be the notion of text coherence. In other words, segments properly reflect communicative intention, and interrelationships among segments are properly expressed.

In planning text structure of multisentence paragraphs by computers, both a sound theory of text organization and an algorithm that can make efficient use of it is required. Rhetorical Structure Theory (RST) is an influential theory for text structure, it identifies the coherence relations that exist between individual segments of a text, and builds up a hierarchical structure of these relations. When captured the internal organization and rhetorical dependencies between sentences in the text, valuable knowledge about discourse can be obtained. Since it contains the derivation of each part of the paragraph, one knows the role each sentence plays with respect to the whole. Still no existing theory, RST included, describes enough to support all the needs of text planning. But the following general assertion about the structure can be formulated: A **discourse** is a structural collection of sentences, which by their semantic relatedness are grouped into segments. Each discourse segment has an associated **purpose**, the communicative goal of the speaker. A discourse is only communicative successful if it is mutually **coherent**.

The attempts Hovy did with RST structure planning established that it is possible to dynamically construct paragraph-length discourses. Additionally, it opens up a set of issues that has to be addressed before robust discourse planning and generation could become reality, from tense and aspect selection, to focus and theme development.

Pablo Gervás [Gervás, 2001] approaches the text generation problem from a more creative angle, by generating poems from prose. When a poem in one language is translated into a supposed equivalent poem in a different language, the poem may lose its traits and poetic form. The parameters that change most are rhythm, rhyme, metric distribution over lines, and word order. To recreate the effect of the original poem, this new version in the new language must be modeled into the corresponding poetic form in that language. Gervás discuss the issue of whether one should favour creative approaches that come up with new material or a faithful approach aiming for a final product as close as possible to the original.

To model the human creative process of poem composition, the issue of producing a specific aesthetically pleasing form for that message is needed to

look into. It concerns processes like selection of a strophic form, distribution of the message across the chosen strophic form, shaping the lines to an adequate length and metric, and finding appropriate rhymes.

One established technique that can simulate the existing skills of translating a prose into a poem without coding the knowledge in terms of rules is case based reasoning. ASPERA is a case based reasoning application that generates poetry versions of texts provided by the user, and applies a knowledge-based pre-processor to select the most appropriate metric structure for the user's wishes. The cases on its case base consist of a sentence of prose associated with a corresponding poem fragment. Each case includes information about how the words in the prose text are related to the words in the poetry solution. The system applies a generation process guided by metrical rules to adapt its best matching cases to the sentences in the text provided by the user. How strict the metric rules are imposed to the poem and whether the system is allowed to resort to additional vocabulary outside that presented are parameters that influence the amount of creativity the system is allowed to have.

4 Evolutionary creativity

A common definition of **creative** is "novel and useful". Creativity is defined as the tendency to generate or recognize ideas, alternatives, or possibilities that may be useful in solving problems, communicating with others, and entertaining ourselves and others. Creativity is a multi-dimensional phenomenon; it means different things to different people. At a micro-level, creativity may be seen in terms of four main elements: surprise, delight, effectiveness, and efficiency. Independently of which definition one chooses to use to describe creativity, the following applications will fit.

Among those are Douglas Lenat's [Lenat, 1982] classic programs, AM and EURISKO, two of the most creative programs in the AI literature. From a number of primitive axioms of set theory, AM was asked to find interesting conjectures, and found many important concepts and conjectures in number theory. Among other things, it rediscovered the concept of numbers with exactly two divisors, that is, prime numbers, as being an interesting set of numbers, quite correctly so.

Lenat's EURISKO program moved a level up to explore a space of heuristics so that a program such as AM would have the ability to move through its own well-defined space and make better conjectures than it was able to before. EURISKO could, learn heuristics for playing games. Looking at Extrema, for example, it was able to find loopholes in the rules of a tournament game that allowed a game-playing program to beat all human opponents. EURISKO has discovered concepts, conjectures, and heuristics in several domain, such as: design of naval fleets, elementary set theory and number theory, LISP programming, biological evolution, games in general, the design of three-dimensional semi-conductors, the discovery of heuristics which help the system discover heuristics, and the discovery of appropriate new types of slots in each domain.

Richard Dawkins [Dawkins, 1985] was one of the first to use evolution as an source to graphic development. He wrote with the help of the Darwinian theory, a computer program, called "Biomorph", to simulate the evolutionary process. The method used has principles similar to genetic algorithms, with cumulative selection, mutation and his own aesthetic taste as a fitness function. The program evolves graphical computer-generated trees and makes tiny changes to their genes by mutation to show how minor alterations in genetic pattern can produce "creatures", illustrated in figure 4, even beyond Dawkins' imagination. The "Biomorph" program, illustrates the design power of Darwinian evolution, that brings us to evolution as being the container of creative design of living things.

Figure 4: Examples of characters developed by "Biomorph".

The Generative Art and Design is a continuation in the genetic art domain, and it represents a design concept as code. This generative code is like DNA in nature. It uses techniques from Artificial Life to generate a multiplicity of possible artwork, artificial events, architectures and virtual environments. In the generative approach the real artwork is not only a product, like images or 3D-models. The generative artwork is an Idea-Product, a concept. It represents an artificial species able to generate an endless sequence of individual events, each one different, unique and unrepeatable, but each one belonging to the same identifiable Idea. Basilica is an Architectural Generative tool, designed by Soddu [Soddu, 2000], that is made for architectural design activity. It is based on his interpretation of the structure, 'DNA', of the Italian Renaissance. Basilica is able to generate an endless sequence of different architectural 3D models, all unpredictable, but nevertheless recognizable as part of the same idea, of the same architectural concept, illustrated in figure 5. The difference between one and another virtual model is like the difference between one and another individual belonging to the same species. A difference shaped by the random components of the evolution.

Argenia is a generative software designed to generate industrial object. It is a generative project able to interface with manufacturing machines to produce series of all different and unpredictable objects. The objects are recognizable because they belong to the same design concept, but the products may be unique and unrepeatable like handmade objects. One of Soddu's last projects of generative art is the Woman Portrait Generator, illustrated in figure 5, able to generate endless sequence of women's portrait in the spirit of Picasso. All these systems are based on two software engines that work together. The first tool represents a Generative Code, making an interpretation of the evolutionary system, and conceiving it as a generative DNA. The second tool emulates the evolution using Artificial Life procedures and manages the increasing complexity.

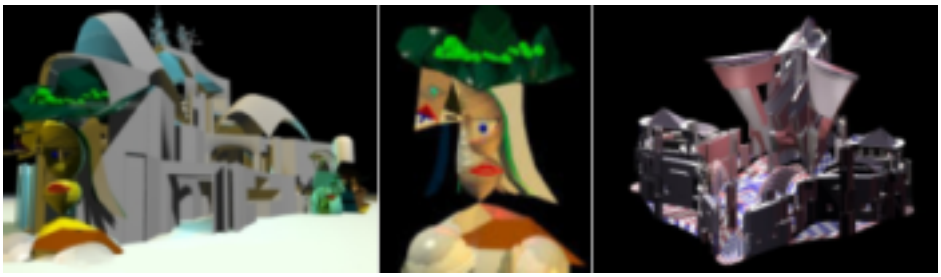

Figure 5: Some of the models generated by Basilica and Argenia.

The ideas of genetic art has also been applied to music. Douglas Hofstadter [Hofstadter, 1979] opened the eyes for many biologists and artists when he discussed the similarities between genes and music. The analogy is explicit in the following quote:

> "Imagine the mRNA to be like a long piece of magnetic recording tape, and the ribosome to be like a tape recorder. As the tape passes through the playing head of the recorder, it is 'read' and converted into music, or other sounds... When a 'tape' of mRNA passes through the 'playing head' of a ribosome, the 'notes' produced are amino acids and the pieces of music they make up are proteins."

Hofstadter also discusses how meaning is constructed in protein and in music:

> "Music is not a mere linear sequence of notes. Our minds perceive pieces of music on a level far higher than that. We chunk notes into phrases, phrases into melodies, melodies into movements, and movements into full pieces. Similarly proteins only make sense when they act as chunked units. Although a primary structure carries all the information for the tertiary structure to be created, it still "feels" like less, for its potential is only realized when the tertiary structure is actually physically created."

As Hofstadter first suggested, music is a natural medium for expressing the complex patterns of proteins and their encoding DNAs. Both consist of a linear sequence of elements whose real meaning lies in their combinations.

The artist Peter Gena [Gena, 1999] and medical geneticist Charles Strom used a sophisticated algorithm for converting the DNA sequences into music. Pitch is determined by a combination of the base composition of the codons[1] and the dissociation constant of the amino acid encoded. Tone intensity is determined by the number of hydrogen bonds between base pairs, and duration of the tone by a combination of the dissociation constants and atomic weights of the amino acids. The amino acids encoded by each codon were also separated into eight chemical categories, with different instrumental timbres assigned to each. Thus nearly all musical elements of their pieces are determined directly by the codon sequence.

It can be a little hard to give an example of this music in the report, but I would personally describe it as some different type of mellow classical background music, harmonious in some sense, and I recommend you to visit:
http://www.whozoo.org/mac/Music/samples.htm.

[1] A specific sequence of three consecutive nucleotides that is part of the genetic code and that specifies a particular amino acid in a protein or starts or stops protein synthesis.

Part II
Implementation and Design

5 Implementation and Design

The genetic algorithm (GA) tries to generate documents with the characteristics that the user wants. It is a system that generates documents from segments of other, already published documents, located somewhere on the web, these documents do not have to be related to each other, written by different researches, whose individual results may not even be of interest to the user, but when put together provide an interesting document. So if attention is directed towards all the publications of all the researchers who authored the documents that were used to generate this new document, maybe a pattern will be noticed that the user would want to pursue and find out more about. The results can be viewed as an index or a table of contents into other documents, the most interesting sentences from the most interesting documents.

5.1 Methods

When implementing a program there are several design choices that has to be done. Which language to use, how to arrange the cooperative parts of the system, etc. All these choices are described in detail in the following sections.

5.1.1 Choice of Algorithm

The goal is to find hidden information. Hidden in the sense that we do not see it, not that it is not there. The reason we do not see it is because of our own mind. Our prejudice gets in the way of our ability to observe what actually is in the data. Genetic algorithms can help us get beyond that and divided and mix the data in ways that allow us to see patterns that we would not have seen before because of our own mental predispositions. Additionally, when looking at a document, we can be misled by the author's formulation or our preconceived notion into thinking a certain way, or maybe the author is trying to trick us into accepting a specific opinion. The genetic algorithm divides the data into pieces and rearranges it, in ways which allow us to re-interpret the document without the author's original intentions. This is the primary reason genetic algorithms are useful, especially in the 'computer intelligence' domain.

One of the other strengths with genetic algorithms for this purpose is that genetic algorithms is a robust search method requiring little information to search effectively in large and complex search spaces. In particular a genetic search progress through a population of points, through parallel search, in contrast to the single point of focus of most search algorithms, making it much faster than traditional methods of search and optimization.

5.1.2 Choice of Programming Language

The programming language chosen for this application is C. Despite that genetic algorithms commonly have been implemented in LISP. One of the main reasons for the popularity of LISP in this area is because it allows dynamic code creation,

compilation and execution. LISP is different than procedural languages. It requires a different approach of programming style and planning. After studying the language fundamentals and functionality it became apparent that all the functionality of LISP is present in other common high level languages. The author had no detailed previous knowledge of LISP and the time it would take to manage it could have reduced the project scope.

The performance of programs often get more affected by choice of algorithm, and quality of compiler than by choice of language. The choice of language is an important part of the design though, since it plays an important part of the design and architecture of the system, but it will not necessary affect the performance. In this application a lot of data will be processed, and there are several time consuming functions, therefore a fast programming language is to prefer. But the main reason for choosing C as a program language is because its ability to structure the data in the desired way. C also has an interface against Python, the language in which the scripts that works against the databases are written.

5.2 The Architecture of the Genetic Algorithm-Module

The GA-application consists of four basic elements. First of all the genetic algorithm (GA) and a Graphical User Interface(GUI) which enables the user to add search queries and adjust the settings. The module is dependent of a script to maintain it with sentences. This script searches the pubMed database with the given query and processes the documents received. When the GA has finished processing the text, the text and belonging data get stored into a database.

The user adds a query and an alphabet. The script searches the database with the query. The found documents are saved to the database. The documents are evaluated with LSA and the given alphabet. The documents are then segmented into sentences, which gets values according to the occurrences of the words from the alphabet, and the document it is derived from. The sentences get a unique id number, and together with the calculated value and a reference to the original document, it forms a gene. The genes are the GA's smallest building blocks. Several genes form a chromosome, which gets processed by the GA according to the algorithms described later. As these chromosomes get combined and recombined and passed through generations, their relevancy increases.

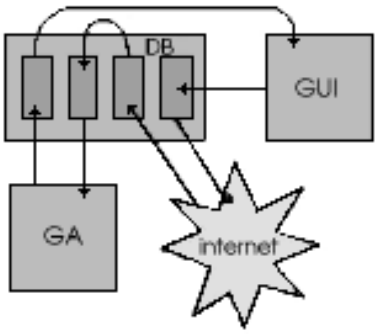

Figure 6: The architecture of the GA-module.

5.3 The alphabet

The earlier mentioned alphabet is a collection of terms, added by the user, to refine the search. It can be factors like authors, preferred or unpreferred, journals, publishers, and publish year. It can also be words that often occur in the same content as the search query terms, to assure that the search is done within the wanted field.

5.4 Evaluate Documents and Sentences

The function that evaluates the sentences is based on several factors. First of all, the document that it originates from gets evaluated using Latent Semantic Analysis (LSA), described in the section 5.4.1. The value resulted from this method is the LSA Value. The sentences also get evaluated on the basis of the number of query terms occurred in the sentence. The resulting value is called Query Value in the final equation. The Content Value is calculated on the basis of the number of words from the alphabet that occur in the sentence. The Source Value indicates if the source of the document is reliable. If it is a prominent author or publisher, etc. All of these constituents are weighted and put into a common equation:

$$TotalValue = 0.3 * LSAValue + 0.3 * QueryValue + 0.3 * ContentValue + 0.1 * SourceValue \quad (1)$$

The weighting in this equation is based on beliefs of the factors significant to for the user.

5.4.1 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is an approach that takes advantage of implicit higher-order structure in the association of terms with documents in order to improve the detection of relevant documents on the basis of terms found in queries. The primary assumption of LSA is that there is some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice with respect to retrieval, and statistical techniques can be used to estimate this latent structure.

In order to analyze a document, LSA first generates a matrix of occurrences of each word in each comparable document. LSA then uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis. The SVD scaling decomposes the word-by-document matrix into a set of, typically 100 to 300, orthogonal factors from which the original matrix can be approximated by linear combination [Foltz, 1996].

Instead of representing documents and terms directly as vectors of independent words, LSA represents them as continuous values on each of the orthogonal indexing dimensions derived from the SVD analysis, constructing a semantic space wherein terms and documents that are closely associated are placed near one another. Singular-value decomposition allows the arrangement of the space to reflect the major associative patterns in the data, and ignore the smaller, less important influences. As a result, terms that did not actually appear in a document may still end up close to the document, if that is consistent with the major patterns of association in the data [Deerwester, 1990].

5.5 Continuous Genetic Algorithm

The GA will continuously receive new genes as information on the web is added or changed. Therefore also the GA is continuous and will develop through time. Considering premature convergence, there is no guarantee that the algorithm will not get stuck on a local maximum, although it is constantly fed new genes. The solution is to reset the GA, and initiate a brand new population every time the derivate of the population's fitness has reached its zero-point. This is implemented by continuously saving the best individual at the moment, and if no individual reaches higher fitness than that individual in a certain number of generations, the GA is reset. If the fitness of the best individual of the population exceeds a given threshold, that individual is saved to the database.

Which stop criterion and threshold to use depends on the problem's complexity and the requirements on the solution. The advantages of having a high limit for the stop criteria is that the chances of just have reached a local maximum decreases, but on the other hand, many unnecessary generation may elapse. The threshold is set according to a specific need and is there for preventing too low fit individuals from being saved in the database.

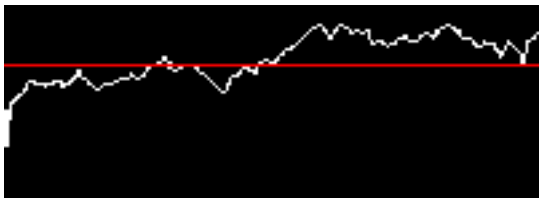

Figure 7: The fitness curve. The horizontal line represents the threshold.

5.6 The Outcome

The output from the GA is a collection of sentences, which together receive the highest fitness value. These sentences still retains their origin in form of a link to the source document. Thus enables the user to click the sentences to reach the original document, if it seems interesting. Examples of sentences produces by the GA are illustrated i figure 8.

5.7 Genetic Algorithm Specific Design

The critical components of a genetic algorithm include the representation of individuals, the fitness function, the crossover and mutation operators, and the selection mechanism. To design a genetic algorithm for a particular problem, these individual components each have to be addressed. The GA's properties varies according to the choices made.

5.7.1 Encoding - Data Structures

Most genetic algorithms use fixed-length, fixed-ordered bit strings to encode a candidate solution. This GA uses another kind of encoding. It allows variable chromosome length, the genes do not have any specific ordering in the chromosome, and it is no binary representation.

0.550656	Here we show that in a pine-inhabiting mirid, Phytocoris difficilis Knight, hexyl butyrate and (E)-2-hexenyl butyrate are abundantly produced only in males; we demonstrate that these metathoracic gland compounds elicit strong antennal responses in conspecific males, and that these butyrates totally interrupt attraction of males to the female-produced sex pheromone. In plant bugs (Miridae), females produce sex pheromones in the metathoracic scent gland, which in most other true bugs (Heteroptera) is responsible for chemical defense. Our results suggest that in at least some plant bugs the male metathoracic scent gland esters have a natural communicative function as anti-sex pheromones, probably to interrupt further mating attempts by other males.
0.545705	Abdomen and head emit only trace amounts of this pheromone. In plant bugs (Miridae), females produce sex pheromones in the metathoracic scent gland, which in most other true bugs (Heteroptera) is responsible for chemical defense. Our results suggest that in at least some plant bugs the male metathoracic scent gland esters have a natural communicative function as anti-sex pheromones, probably to interrupt further mating attempts by other males. Here we show that in a pine-inhabiting mirid, Phytocoris difficilis Knight, hexyl butyrate and (E)-2-hexenyl butyrate are abundantly produced only in males; we demonstrate that these metathoracic gland compounds elicit strong antennal responses in conspecific males, and that these butyrates totally interrupt attraction of males to the female-produced sex pheromone.
0.544055	Mature gregarious male desert locusts, Schistocerca gregaria, emit the courtship inhibition pheromone phenylacetonitrile. In plant bugs (Miridae), females produce sex pheromones in the metathoracic scent gland, which in most other true bugs (Heteroptera) is responsible for chemical defense. Here we show that in a pine-inhabiting mirid, Phytocoris difficilis Knight, hexyl butyrate and (E)-2-hexenyl butyrate are abundantly produced only in males; we demonstrate that these metathoracic gland compounds elicit strong antennal responses in conspecific males, and that these butyrates totally interrupt attraction of males to the female-produced sex pheromone.

Figure 8: An example of the top three sentences, produced by the GA.

Genetic algorithms usually process populations of strings. So the primary data structure for this GA is therefore a string population, where each individual contains the chromosome, the fitness value, and some auxiliary information. Each chromosome in turn is an array of genes, a gene is the smallest element in the GA and consists of an ID referring to a specific sentence and a value according to the sentence significance. But because the order is irrelevant in the chromosome, there would be more correct to see each chromosome as a set of genes.

5.7.2 Fitness Function

The individuals' fitnesses are dependent of the genes' values. The genes' values are calculated on the basis of the occurrences of certain keywords. The value is normalized to a float between zero and one, by dividing the number of words occurring in the sentence by the total number of words in the alphabet. The highest possible value for a gene is thus 1.0.

The chromosomes' fitness values are calculated by adding all its genes' values and normalizing them so that chromosomes with a length about five genes long will be favoured.

The individual's fitness does not depend on the specific combination of genes, and the function does not do anything to consider how the different sentences match together.

5.7.3 Selection Method

The selection method chosen for this solution is tournament selection. Two individuals are chosen at random from the population. A number between 0 and 1 is randomly generated. If the randomized number is higher than a certain parameter, the fitter of the two individuals is selected, otherwise the less fit individual is selected. Both individuals are then returned to the population and can be selected again.

5.7.4 Crossover

The crossover method for this solution is a modified 1-point crossover and is not quite as common since the chromosome length is variable, and the genes are sentences instead of binary strings. No chromosome should contain the same

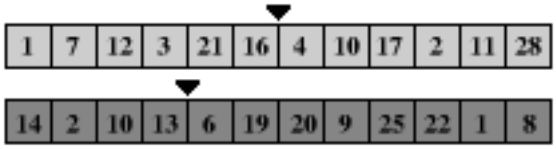

Figure 9: Two parent chromosomes with the cross point marked.

Figure 10: The chromosomes length can change through crossover.

Figure 11: If two identical genes exist, one is deleted.

sentence more than once. So if a sentence occurs in both of the parents and happens to end up in the same chromosome, one of the identical genes will be deleted and the chromosome length will consequently decrease. It does not matter which of the identical genes that gets deleted, because of the fact that the order of the genes inside a chromosome is irrelevant. The crossing point for each parent will be randomized independent from the other. The procedure is described in figure 9, 10 and 11.

5.7.5 Mutation

The mutation is divided into two parts. One which replaces a gene with another randomly picked from the gene pool, and one that changes the chromosome length, by adding or removing genes. The mutation can vary both a specific gene and the chromosome length with a certain probability.

Part III
Experimentations

6 What is Success - And How is It Evaluated?

What would be a successful result? With the complexity of this task it is hard to decide. It is obvious that we strive for the optimal fitness value. But is that really what the users are looking for? For the grade of the fitness to be relevant to the user, the evaluation of the sentence needs to be accurate and correspond to the user's needs. So would the best way to evaluate the GA's performance be to let a user make a solid appraisal of the result? Should it be up to the user to judge success or failure? What results could you get from just letting the user read some of the highest scoring documents?

But then a second question is raised: What do we really want from this application? If it is a tool for summarization, then it would be an idea to let the users test it and compare the generated sentences to a manually made summary of the same text. But first of all, the GA-generated text is not from just one document. To do a comparable summarization you will have to go through all the sentences the GA has access to and manually make a summary. This work would be both strenuous and time consuming. To limit the process to a few documents would take away some of the meaning of the GA: to search for an optimum in large complex spaces.

Should you just let the user specify a query and an alphabet, read some of the highest scoring documents, and judge success or failure? If any of the highest scoring documents appears to be very relevant to the concepts the user had in mind, then that document is a connection for some documents the user should read. If none of the highest scoring individuals seems relevant to the user, something about the keywords, the fitness function, the weighting, or the source is wrong.

But if you see the GA as a tool to recombine old information to help you see new relations and see beyond biases, see it as a tool for creativity, it becomes much harder to evaluate. In principle it would be enough with just one successful hit. If you most of the time get nonsense, but eventually get one hit that comes up with something revolutionary which can help you solve all your problems, would it then be classified as a successful system or not? Or maybe this 'nonsense' in fact is too complex for the user to understand. Should the system's 'intelligence' then be limited to the user's ability and knowledge?

Whatever the purpose really is, it surely is not just to summarize text, since in that area there are many systems that would outperform this solution. The GA should in some way help the user find relationships or information that our biased eyes earlier could not see without help. I prefer to see it as a tool for creativity. But how do you evaluate creativity?

7 Experiment Design

The experiment should preferably consist of two parts. The first part evaluates the common GA utilities. This will be evaluated with a simple statistics program that measures the fitness values, the number of generations elapsed until a maximum over the threshold is reached, the time it takes to reach a certain maximum and how many genes that have been in circulation during a run.

The aim of the other part of the experiment would be to evaluate how the GA fits for this kind of tasks. How well the evaluation of the documents and sentences work, and how the weighting should be adjusted to reach an optimal result. It would be up to a user to judge success or failure. But first of all, an experiment like this for this application would be too extensive for the time period available. Due to the fact that the GA so far just has pubMed as its only source, the experiment would not give much, since pubMed is rather specific, well-structured database, not the kind of search space genetic algorithms does best in. But as soon as the application is adapted to internet and several databases, a user evaluation will take place.

This evaluation should involve a population of researchers, and be performed during a longer period, giving the users a chance to learn how to obtain the real value from the system.

7.1 Data Sets

The data used by the GA are sentences derived from 100 documents published on pubMed, resulting in approximately 800 sentences. This is actually a very small search space for a genetic algorithm, which may result in inaccurate test results. But the restriction is necessary due to a limitation from pubMed. The sentences used are results from a search done in pubMed given a specific search query. The data the GA actually are working with are arrays containing the sentences' id numbers and their assigned fitness values. PubMed is a service of the National Library of Medicine, that includes over 14 million citations for biomedical articles back to the 1950's. These citations are from MEDLINE and additional life science journals. PubMed includes links to many sites providing full text articles and other related resources [pubMed].

To use pubMed as the only source of data is just an introduction phase. As the project proceeds the number of sources increases, and the internet will of course be one of them.

8 Evaluate Parameter Settings

The purpose of the experiments is to serve as a basis for the adjustment of the parameters, to strive for an optimal fitness value. The parameters typically interacts with each other in an non-linear manner making it hard to optimizing them one at a time. There is a great deal of research in parameter settings, but there is no conclusive results on what is best [Mitchell, 1999]. The parameter settings are usually problem dependent.

The reason that the fitness value is so low in the results below is that it is limited by the genes' values. A chromosome cannot possibly get a fitness value higher than the genes' values. In these tests the highest scored gene has a value just over 0.55, so fitness values close to 0.5 are in fact close to optimal. In all cases when testing one parameter, all the other parameters are set to their optimal value, which may lead to confusing results sometimes. But this course of action makes it easier to see the effect the varying values of that specific parameter makes.

The variables measured to evaluate the different parameters' effect on the performance is the fitness value, the time consumed, the number of generations elapsed, and the proportion of genes that have been in circulation during a run. The values shown in the tables is the mean value of 30 runs.

8.1 Populations Size

The population size is known to affect convergence. A very small population can result in premature convergence and a too large population may give slow convergence. Too small population sizes may also constrain the GA to a local maximum, while larger population sizes may provide enough diversity and robustness to quickly eliminate these individuals from the population and thus only increase the time taken to reach a solution. The use of various genetic operators influences population size. As a general observation, increase in complexity of the algorithm leads to a need for larger population size.

The aim with this experiment is to empirically test how various population sizes affects the performance of this GA. The population size is varied from 5 to 100 individuals. For every population size 30 runs are made by the GA.

Population size (number of)	Fitness value	Time (s)	Generations (number of)	Circulation (%)
5	0.501	1.9	221	93
10	0.536	2.7	261	93
20	0.538	4.9	222	93
30	0.546	10.1	351	93
40	0.551	15.8	334	93
50	0.552	20.1	254	93
60	0.553	30.6	396	94
70	0.556	46.9	614	94
80	0.555	55.6	600	95
90	0.555	48.4	354	95
100	0.557	64.7	461	95

Table 1: The performance of the GA using various population size.

As can be seen in table 1 the population size affects the GA's performance. The fitness rises with the increasing population size, until a maximum is reached. The GA performs best using population sizes from 40 individuals and larger. Concerning the time consumption, it can be seen that it also increases with the population size. Since the performance of the GA does not improve much with population sizes larger than 40 individuals, it is possible to prioritize speed over performance without greater effects on quality.

8.2 Variation in Crossover/Mutation Probability

According to many sources, it is possible to adjust the balance between exploration and exploitation by adjusting the rates for crossover and mutation.

Relating the role of mutation to the explorative, or exploitive part of the search, two different views are common. On one hand, mutation is often seen as an explorative operator, because it introduces new material, in an unbiased manner. On the other hand, it can also be seen as an exploitative operator, because it is conservative. That is, it causes only small changes, thus most of the old information, and genetic material, is preserved [Eiben, 1998].

Similar to mutation, crossover can be seen as both explorative and exploitative. Crossover is an explorative operator, because it recombines the material of the parent into new configurations. Meanwhile, it can be perceived as exploitative, because the material used is old [Eiben, 1998].

The aim with this experiment is to empirically test the effect of various crossover and mutation rates. How does the balance between the exploration and exploitation get affected by the variations? In the experiments of the crossover rate, the optimal mutation rate is used, and vice versa of the experiments on the mutation rates. The crossover rate was varied from 0.0-1.0, and the mutation rate from 0.0-0.1 with concentration on very small rate values. As can be seen in table 2 the crossover rate does not have great effects on

Crossover rate	Fitness value	Time (s)	Generations (number of)	Circulation (%)
0.0	0.516	85.2	744	91
0.1	0.514	53.5	485	92
0.2	0.517	52.8	541	96
0.3	0.520	41.3	280	94
0.4	0.514	30.9	210	93
0.5	0.518	26.2	171	92
0.6	0.513	27.0	264	96
0.7	0.516	23.7	170	95
0.8	0.510	21.1	148	94
0.9	0.506	19.0	114	92
1.0	0.511	20.7	188	93

Table 2: The performance of the GA using various crossover rates.

the GA's performance, though the time consumed and the numbers of generations decrease with the increase of the crossover rate. What can be seen from these results is that a slow convergence is the result of too low crossover rate. Because of the small search space and the relatively large population size, any

sharp changes in the percent of genes in circulation will not be noticed. The performance of the GA has a maximum when the crossover rate is at 0.3, showing that with rates higher than 0.3 the crossover rather destroys than improves the good chromosomes.

Mutation is a rather costly function, as seen in table 3, great decreases in time consumption are made when mutation is off. At the same time the number of genes in circulation makes dramatically changes early on when just very low rates of mutation is used. The fitness curve has a peak when the mutation rate is about 0.01, after that it decreases along with the increase of time consumption. Surprisingly the number of generations does not increase with the mutation rate. Due to the fact that the optimal grade of the probability rate was used when

Mutation rate	Fitness value	Time (s)	Generations (number of)	Circulation (%)
0.00000	0.497	4.5	50	40
0.00001	0.510	21.3	312	91
0.0001	0.517	24.7	374	94
0.001	0.512	23.0	348	92
0.01	0.520	38.4	342	98
0.05	0.513	67.9	203	100
0.1	0.501	98.8	216	100

Table 3: The performance of the GA using various mutation rates.

testing the effects of the other operator's varyings, the result can be a little hard to interpret. To clarify the results another table is added, where one operator's possibility rate is set to zero and the other is set to the optimal value.

Operator	Fitness value	Time (s)	Generations (number of)	Circulation (%)
Mutation	0.516	85.0	647	100
Crossover	0.497	4.5	50	40

Table 4: The performance of the GA, when only one of the genetic operators is in use.

The results in table 4 make clear conclusions about the balance between exploration and exploitation. It is mutation that stands for exploration, while crossover stands for the exploitation. It is the mutation that increases the number of genes in circulation, and crossover reduces the number of generations elapsed and the time consumption. Which gives that a combination of the two operators gives the best result, in both performance and efficiency.

Another common opinion about GAs is that the search space gets explored by the genetic operators, while exploitation is done by selection [Goldberg, 1989].

8.3 Selection Methods

Selection has to be balanced between exploration and exploitation. Too strong selection can result in a premature convergence, suboptimal highly fit individuals may rapidly come to dominate the population, causing it to converge on a

local maximum. Too weak selection can lead to slow evolution. After many generations, the populations will have largely converged, but still not have precisely located the global maximum.

As can be seen in table 5, the tournament selection results in higher circulation percent and more generations elapsed, in other words, higher exploration and slower convergence.

Method	Fitness value	Time (s)	Generations (number of)	Circulation (%)
Roulette Wheel	0.520	41.3	280	94
Tournament	0.551	59.3	596	100

Table 5: The performance of the GA using different selection methods.

The previous results shows that tournament selection is more suitable for this problem, offering a selection less strong than roulette wheel, also resulting in higher fit individuals.

8.4 Variation in Stop Condition

The stop criteria can be varied to affect the quality and the performance of the search. With low demands of the limits, the search is fast but it can be stopped too soon, before it in fact has found a maximum. With high demands the risk of reached a local maximum decreases, but the search may eventually get stuck at a maximum and waste time.

The value of the limit represents how many generations that proceed after a maxima is reached, to assure that the reached maxima is not just local maxima.

As can be seen in table 6, increasing the limit for the stop condition increases the values for all parameters, till a maximum is reached. In this experiment a limit at 500 generation reached the highest fitness, and is thus used in the application. What stop criterion to choose depends on the application's requirements. What is the optimal fitness value worth?

Limit (number of)	Fitness value	Time (s)	Generations (number of)	Circulation (%)
0	0.195	3.8	1	41
10	0.472	5.9	38	58
20	0.491	7.2	52	66
30	0.501	9.1	84	72
40	0.503	9.0	87	73
50	0.508	10.2	109	76
100	0.505	14.4	209	83
200	0.507	21.6	280	90
300	0.514	28.9	530	96
400	0.518	33.2	664	97
500	0.521	38.9	840	99
1000	0.519	61.2	1480	100

Table 6: The performance of the GA using various stop criteria.

Part IV
Finishing

9 Discussion

A tool for creativity could be just about anything. It can be a song, a painting, spending time with friends, blue skies, software, anything that helps to increase your creativity, and it could be used in almost any situation. This GA-module is supposed to be used when you are stuck and need new ideas, when you need to start thinking in new ways. What the GA does is collecting all information about the subject you are interested in, cutting it up in small pieces and rearranging it, in ways that hopefully can give you fresh ideas.

When an author wants to make a point the fact may be wrapped up in fancy words to convince the reader about his/her opinion. What would happen if the document gets split up into little pieces? With only the facts represented, unwrapped from all these fancy words, would the reader still get the same interpretation? Would the conclusions and arguments made still seem as convincing? These are parts of the goal this application wants to achieve. It is letting the user reach an independent conclusion, without getting influenced by the author's intention or interpretation. It does not have to be a conscious act from the author either, it can just be an interpretation of the given facts which is inadvertently delivered through the text. But the author's interpretation does not necessarily have to be the only possible one. This application does not just work with one document at the time, by just one author. What is represented in the end, is a collection of sentences taken from several different sources, and the final product may be something that helps the user to make a new interpretation about for example a phenomenon. When combining several authors' arguments in a certain matter you might come up with something none of the authors thought of.

An example to clarify this thought. There are several different scientist all working in the same domain, all trying to reach the same goal, and all of them have written papers about their discoveries and conclusions within this subject. Maybe all of them has got different facts and different conclusions from their work. By dividing all these papers into little pieces and selecting some of the most important pieces to form one single document, you may be able to see a relation none of the authors thought of. Maybe one of the researchers has made strong arguments for a potentional solution to convince the reader that these conclusions are the right ones. When applying these arguments on another researcher's arguments or conclusions, doors for other conclusions might open. Maybe exactly these arguments can help the user see something in another researcher's data, that makes the user reconsider a theory. It does not have to be a preexisting collective intention among the different researchers either, they may have contradicting opinions, or maybe even different domains. It is not certain that the pieces that gets connected in fact have something at all in common. But it might give the user what is needed anyway.

In the process of generating text for this purpose there are two properties that we need to consider: meaningfulness, and newsworthiness. The balance between those characteristics can be achieved by adjusting parameters like: the

size of the parts getting processed, the size of the generated text, the sources the data comes from, and the ordering within the generated text.

To achieve meaningful text, with a clear consistency, some text analysis is needed, both from a semantical and grammatical point of view. The order in which sentences are stringed together has to be considered. You might also want a relatively low semantic variation, which can be achieved by LSA-analysis and taking the information from rather few similar sources within the same area.

Text that appear meaningless may have high news value. By 'high news value' I mean information that consists of new unknown facts or interpretations. A way to achieve this might be to take data from many different sources, not necessarily belonging to the same domain and combining them in a way to achieve large semantic variation inside the text. In this case the demands on the meaningfulness or the understandability might have to be lowered. That means that the user will need to use more fantasy and creativity to get or rather produce a meaning from the context.

In the representation of the sentences, order is irrelevant. In which way will this affect the user's interpretation of the information? According to Hovy [Hovy, 1993] only a minority of the possible combination make semantic sense when just stringing sentences together, and among those which do, the meaning presumably differ widely. When an author is writing something, a book, an article, a letter, etc. the facts and sentences are mentioned in a certain order, to convey the author's intentions or to emphasize a certain opinion. In that case the reader will probably interpret the text in a way similar to the author's intentions. Which is often a good thing, but it does not always have to be a bad thing for the reader to be able to interpret it in another way. So, a way to achieve higher news value might be to switch the order between the sentences, making it easier for the the user to reach his own conclusions without the interference of the author.

How long the texts should be, is another question worth mentioning. In this application the average "document" is about five sentences long, which results in a rather short text. There are many ways to look at this issue. One way to see this, is that short documents do not give such a detailed text, letting the user interpret the document more freely, and long detailed documents might not leave anything for the imagination. But in the same time very long sentences can also include more angles to the question and might just increase the possibilities for creative interpretations, while short texts might just include a few similar sentences, not giving much for the imagination.

In this application the smallest pieces processed are sentences. There are pros and cons with this solution. It is computationally simpler to get meaningful text in the end, it needs less text analysis etc. The disadvantages of this solution are that it might be too much coherent text to make it free for interpretation, and giving any news value. Another drawback of using whole sentences is that the sentence is valued on the basis of the whole sentence's significance. This might result in that a sentence containing only one very valuable fact, gets disregarded if it besides that fact just contains nonsense. A sentence can be divided into different clauses and even down to the word level. There are many advantages in using clauses as the smallest element in the process, you can escape both problems mentioned above, though it will be more extensive text analysis needed. It may also remove some or all of the content in the original

sentence, resulting in nothing more than random clauses or words.

Another interesting thing in this ordering question is; how will the readers experience the text? Do they try to read it as an ordinary article or tale, trying to interpret it in the same way they would have done if the text was ordered as usual? Or will they learn to see patterns in it, instead of just reading the text line for line? That is where the true creativity becomes involved. Quoting Janlert [Janlert, 2002]:

> "Struggling with structure stimulates the creative process. It should be hard, not impossibly hard, but sufficiently hard to provoke a 'dialogue', an interactive process between creator and 'material' ".

In this context you should see the 'creator' as the reader who tries to construe the text. This is probably not a task you easily learn in a day, but maybe it will give you unexpected fortune when you do.

10 Future Work

One of the most important future improvements for the system is to connect all the different modules into a hybrid multi-agent system. In the future further modules will also be added to the system.

There are several improvements, and functionalities that can be added to the GA-module in the future. First of all a new fitness function should get developed, a function that calculates the chromosomes' fitness value according to the combination of sentences and not just sum the different elements' values, to reach a measure that is not linearly dependent of the parts. With that functionality added the performance of the GA can remarkably increase. But for that to be possible some further text analysis is needed. For example, the LSA can be used for several functionalities, like evaluating each chromosome. But the GA may also be an asset for the LSA, by showing it new areas to look into. Pronoun replacement is another simple text analysis process that could be done in the future, to receive more sentences that match the queries. In this application the smallest part the GA processes are sentences. In future versions, smaller parts like clauses, phrases and words might be of interest.

There should also be an extensive evaluation of the GA - module. First of all to see how it fits for the purpose of creative summarization, but also to be able to evaluate the different evaluation methods: how does it affect the GA's performance if the weighting of different alphabets is adjusted and what differences can be seen in the result if the LSA is switched on or off? Other aspects that need to get evaluated are some of the design choices for the GA, like the text size, and how the order of the sentences affect the rendering of the text. In future versions the order might not be irrelevant.

11 Acknowledgments

I am grateful to Lars-Erik Janlert at the Department of Computing Science, Umeå, who has been giving me ideas and inspiration and a lot of help with the report. I would also like to thank Annika Westergren, Sofia Persson, Torbjörn Johansson and all the others at the Interactive Institute - *Tools for Creativity* who have been very valuable through the process, and Glen Ropella for guiding and inspiration. I would also like to express my gratitude to Victoria Jansson and all my friends for all the love and support they have given me.

References

[Beasley, 1993] **Beasley, David & Bull, David R, & Martin, Ralph R.** (1993) *An overview of genetic algorithms: Part 1, fundamentals.* University Computing, 15(2):58–69. (http://citeseer.nj.nec.com/516528.html).

[Beasley, 1993] **Beasley, David & Bull, David R, & Martin, Ralph R.** (1993) *An overview of genetic algorithms: Part 2, research topics.* University Computing, 15(4):170–181. (http://citeseer.nj.nec.com/516528.html).

[Billhardt, 2002] **Billhardt, Holger & Borrajo, Daniel & Maojo, Victor.** (2002) *Using Genetic Algorithms to Find Suboptimal Retrieval Expert Combinations.* Proceedings of the 2002 ACM Symposium on Applied Computing, SAC 2002, Madrid, Spain, March 11-14, pp. 657-662, ACM.

[Dharis, 2003] **Dharia, A.** (2003) *How do Internet search engines work?* 2002-10-14 http://www.kubrussel.ac.be/onderwijs/etew/-tew/vakken/tweedekan/ecsector/kennistech/-search_engines_sciam.htm, 2003-12-29.

[Dawkins, 1985] **Dawkins, Richard** (1985) *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe Without Design.* W. W. Norton & Company.

[Deerwester, 1990] **Deerwester, Scott & Dumais, Susan T. & Furnas, George W. & Landauer, Thomas K. & Harshman, Richard** (1990) *Indexing by Latent Semantic Analysis.*

[Eiben, 1998] **Eiben, A.E & Schippers ,C. A.** (1998) *On evolutionary exploration and exploitation.* Fundamentae Informaticae, 35:35-50.

[Foltz, 1996] **Foltz, Peter W.** (1996) *Latent Semantic Analysis for Text-Based Research.* New Mexico State University.

[Gena, 1999] **Gena, Peter & Strom, Charles** (1999) *A Physiologial Approach to DNA Music.* (An update of their presentation "Musical Synthesis of DNA Sequences," presented at the Sixth Symposium on Electronic Arts: Emergent Senses. Montreal, Canada, 1995).

[Gervás, 2001] **Gervás, Pablo** (2001) *Generating Poetry from a Prose Text: Creativity versus Faithfulness.*

[Goldberg, 1989] **Goldberg, David** (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley.

[Gordon, 1988] **Gordon, Michael** (1988) *Probabilistic and Genetic Algorithms for document retrieval.* ACM 31:10, 1208-1218.

[Hofstadter, 1979] **Hofstadter, Douglas** (1979) *Gödel, Escher, Bach: an Eternal Golden Braid* Basic Books.

[Hovy, 1993] **Hovy, Eduard. H.** (1993) *Automated discourse generation using discourse structure relations.* Artificial Intelligence, 63. Special Issue on NLP.

[Janlert, 2002] **Janlert, Lars-Erik** (2002) *Tools for creativity - breakers and bracers.*

[Jones, 1995] **Jones, Gareth & Robertson, Alexander & Santimetvirul, M. Chawchat & Willett, Peter** (1995) *Non-hierarchic document clusteringusing a genetic algorithm.* Department of Information Studies, University of Sheffield.

[Khan, 2002] **Khan, Nazan** (2002) *Population Sizing in Genetic and Evolutionary Algorithms.*

[Lenat, 1982] **Lenat, Douglas B.** (1982). *The nature of heuristics.* Artificial Intelligence 19: 189-249.

[Maley, 1996] **Maley, Carlo** (1996) *Mutation Rates as Adaptions.*

[Milutinović, 1998] **Milutinović, Veljko & Kraus, Laslo & Mirković, Jelena & Nešić, Ljiljana & Tomča, Nela & Cvetićanin, Suzana & Cvetković, Dragana & Slijepčević, Saša & Obradović, Vladan & Mrkić, Mladen & Čakulev, Igor** (1998) *A Software Package for Experimenting in Genetic Search on Internet: Spatial versus Temporal Locality and Static versus Mobile Agents.*

[Mitchell, 1999] **Mitchell, Melanie** (1999) *An Introduction To Genetic Algorithms.* MIT Press.

[Oren, 2001] **Oren, Nir** (2001) *Improving the Effectiveness of Information Retrieval With Genetic Programming.*

[Pathak, 2000] **Pathak, Praveen & Gordon, Michael & Fan, Weiguo.** (2000) *Effective information retrieval using genetic algorithms based matching functions adaptation.* In Proceedings of the 33rd Hawaii International Conference on System Science.

[pubMed] **pubMed** http://www.ncbi.nlm.nih.gov/PubMed/.

[Shannon, 1948] **Shannon, E. Claude** (1948) *A Mathematical Theory of Communication, Bell System Technical Journal.*

[Soddu, 2000] **Soddu, Celestino** (2000) *New naturality: a Generative Approach to Art and Design.* Artificial Life 7 Workshop Proceeding. 145-148.

A Algorithms of the Genetic Algorithm

Below are algorithmic descriptions of the most distinguished processes in the module.

Initiate Population

1. Initiate data.
   1.1. Designate set specific data; probability for mutation, population size,etc.
2. Produce chromosomes at random.
   2.1. Randomize the chromosome's length.
   2.2. Choose genes to the chromosome.
      2.2.1. Make a call to the script for the database.
      2.2.2. Assure that the chosen gene does not already exist in the chromosome.
      2.2.3. If the chosen gene already exist, make another call.
   2.3. The genes values are summed to get the chromosome's fitness.
3. Generate the first population out of the random generated chromosomes.

Proceed the generations

1. Iterate the following steps for all generations.
2. Select individuals for reproduction.
3. Combine the individuals to reproduction.
   3.1. Crossover occurs with a certain probability for each chromosome.
      3.1.1. Randomize a cross site for both individuals.
      3.1.2. Divide the chromosome at the cross site.
      3.1.3. Exchanges the parts of the chromosomes between the individuals.
   3.2. Mutation occurs with a certain probability for each gene independent of the crossover.
      3.2.1. Exchange the given gene for a randomized picked.
4. Calculate the new individual's fitness.
5. Advance the generation.

Crossover

1. Randomize a cross point for each parent chromosome.
2. Go through all genes in both of the chromosomes until the cross point is reached and test for mutation.
3. Switch the endings of the chromosomes between the two parent chromosomes.

4. If any of the newcomer genes already exists in the chromosome, delete one of the identical genes.

Selection

1. Randomly select two individuals from the population.
2. Randomize a number between 0 and 1.
    2.1. If the randomized number is higher than a certain parameter, the fitter of the two individuals is selected.
    2.2. If the randomized number is lower than a certain parameter, the less fit individual is selected.
3. Both individuals are then returned to the population and can be selected again.

Mutation

1. Randomize a new gene.
2. Check if the new gene already exist in the chromosome.
    2.1. If it exist, randomize a new one until an unique is reached.
    2.2. If it doesn't exist an unique gene, replace the old one.
3. Flip a biased coin to decide whether the mutation should variate the length.
    3.1. If mutate length, randomize a number by which the chromosomes length shall be changed.
    3.2. If the length by which the chromosome is negative remove that many genes.
    3.3. If the length by which the chromosome is positive add that many unique genes.

Search

1. Find user and search profile according to arguments given.
2. Check for query string.
3. Get and parse primary IDs
4. Get the alphabet from the user.
5. Search the database.
6. Parse xml result record.
7. Save a limited amount of hits to file.
8. Store info about the search in the database.

Parse

1. Connect to GUI DB and find user and profile according to arguments.
2. Retrieve query string from GUI DB.
3. Search pubMed with query string and get xml result record.
4. Parse xml result record and extract number of hits and other search parameters.
5. Retrieve hits from pubMed in xml encoding.
6. Store info about search in siibaGA DB, table info.
7. Get profile settings for source evaluation from GUI DB.
8. Parse xml file with retrieved results and store in siibaGA DB, table document.
9. Score the documents.
10. Calculate LSA score by calling the LSA module.
11. Calculate source score from profile settings, i.e. author, affiliation, etc.
12. Save scores in DB siibaGA, table document.
13. Retrieve abstracts from siibaGA DB, table document.
14. Parse into separate sentences.
15. Score sentences.
    - 15.1. Calculate query value, i.e. query string matches.
    - 15.2. Calculate content value from profile settings, i.e. other important words matches.
16. Set fitness according to equation.
17. Store sentences and fitness values in siibaGA DB, table sentence.
18. Count number of sentences in siibaGA DB, table sentence.
19. Insert number of sentences in siibaGA DB, table info.