# Genetic algorithms with age structure and hybrid populations

Final report for the research project in Computer Science 416

Nils Knappmeier

April 10, 2003

**Abstract**

This paper deals with hybrid populations in genetic algorithms. Age structured genetic algorithms were introduced by N.Kubota and T. Fukuda in their paper [1]. This paper tries to combine the advantages of traditional genetic algorithms and age structured genetic algorithms by using a hybrid population that consists of individuals of both kind.

## Contents

**1 Introduction** **2**

**2 Genetic Algorithms** **2**
2.1 General concept . . . . . 2
2.2 Representing problem in GA . . . . . 2
2.3 Crossover . . . . . 2
2.4 The evolution cycle . . . . . 3
2.5 Parameters . . . . . 3

**3 GA with age structure** **4**
3.1 Problems with GA . . . . . 4
3.2 The idea of age structure . . . . . 4
3.3 Expected effects of age structure . . . . . 4
3.4 Experiments . . . . . 4
3.5 A note on partitioning . . . . . 4

**4 Hybrid Populations** **4**
4.1 The structure of hybrid populations . . . . . 5
4.2 Preventing extinction . . . . . 5
4.3 Expected effects . . . . . 5

**5 Experiments** **5**
5.1 General setup . . . . . 5
5.2 Interpretation of values . . . . . 6
5.3 Experiment: Partitioning scheme . . . . . 6
5.4 Experiment: Subpopulations . . . . . 6

**6 Conclusion** **7**

## 1 Introduction

Solving difficult problems is one of the major challenges of Computer Science. If we want to find the optimal solution for a problem, we often have to search the whole search tree of the problem. In some cases a suboptimal solution may be sufficient. There are ways to find such a solution without searching the whole tree of the problem.

Simple neighborhood search is one approach that points in this direction. But the solution is usually not be very good because the search might be stuck in a local maximum. There are modifications to such as simulated annealing that try to address this problem by allowing steps to worse solutions in some cases. Another approach to avoid local maxima, is to perform multiple neighborhood searches at the same time and allow only the $n$ best search states to survive. This can also be seen as a mutation, selection approach, so called evolutionary algorithms.

This approach already tries to simulate behavior that we find in nature. The next step in this direction is the paradigm of genetic algorithms. In this case, a potential solution is encoded in the DNA of an individual. Individuals may exchange parts of their DNA in a process called crossover.

This paper is structured as follows. Section 2 describes the functioning of genetic algorithms (GA). Section 3 presents the idea of age structured genetic algorithms (ASGA) and their differences to regular GA. Section 4 introduces the idea of hybrid populations. Section 5 applies the hybrid algorithm to the TSP problem and describes the experiments I performed. Section 6 contains the conclusion of the experiments.

## 2 Genetic Algorithms

### 2.1 General concept

Genetic algorithms are an extension of the evolutionary approach. The analogy to nature is the following: Evolution worked fine for millions of years. But the real boost came when animals started to exchange genetic material. Genetic algorithms are based on a evolutionary search with $n$ individuals. The generation cycle is enhanced by a phase called *crossover*, in which individuals exchange parts of their solutions to create new ones.

This is not be as easy as it sounds: The solution of a problem is usually restricted by a number of side constraints. Mixing two solutions often results in the violation of these constraints.

Furthermore, we would like to be able to provide a framework for genetic algorithms that can be used to solve many problems. The crossover method should be independent of the problem.

### 2.2 Representing problem in GA

Again, nature gives us some hints: The DNA of an animal only contains instructions on how to build the animal, not the animal itself. Following this, we let each individual only contain an implicit solution.

In GA, a problem is represented as a 3-tuple $\mathcal{P} = (l, s, f)$, where

- $l : I \rightarrow \mathbb{N}$, a function that maps the instance of a problem to the length of the gene.
- $s : I, \{0,1\}^n \rightarrow S$ a function that maps a DNA[1] problem instance to its solution.
- $f : I, \{0,1\}^n \rightarrow \mathbb{R}$ a function that maps a DNA and a problem instance to a value that describes how good the solution is. (fitness function)

In order to avoid invalid solutions, the fitness function has to include a penalty for this kind of solution.

### 2.3 Crossover

As said, crossover means exchange of genetic material. There are always two individuals involved in a single crossover. We will call these individuals the parents $p_1$ and $p_2$. Furthermore, a crossover results in the creation of two individuals. We will call them the children $c_1$ and $c_2$. Now, there are different ways to decide which bits of the parents should be parts of which child's gene. In this I will highlight three common approaches to do that.

**Roulette wheel selection** In this approach, one single random bit is exchanged in order to create the children's DNA.

Let $c_1[i], c_2[i], p_1[i]$ and $p_2[i]$ be the $i$-th bit in the DNA of $c_1, c_2, p_1, p_2$ respectively.

Let $r \in \{1..l\}$ be a random number, where $l$ is the length of the DNA.

$$\forall i \in \{1..l\} : c_1[i] = \begin{cases} p_1[i] & \text{if } i \neq r \\ p_2[i] & \text{if } i = r \end{cases}$$

Respectively, we do the same for $c_2$. The DNA of $c_1$ does not differ from $p_1$ but in the one bit that has been exchanged.

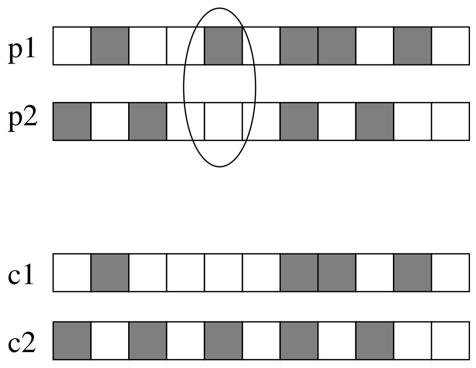


Figure 1: Roulette wheel selection

[1] I will use "DNA" in this report to refer to the genetic information of individuals. I'm aware that DNA is actually the shortcut for an acid.

**Cut selection** In this approach, a random number defines the position where the DNA is cut in both individuals. The right (or the left) part is exchanged

$$\forall i \in \{1..l\} : c_1[i] = \begin{cases} p_1[i] & \text{if } i \leq r \\ p_2[i] & \text{if } i > r \end{cases}$$

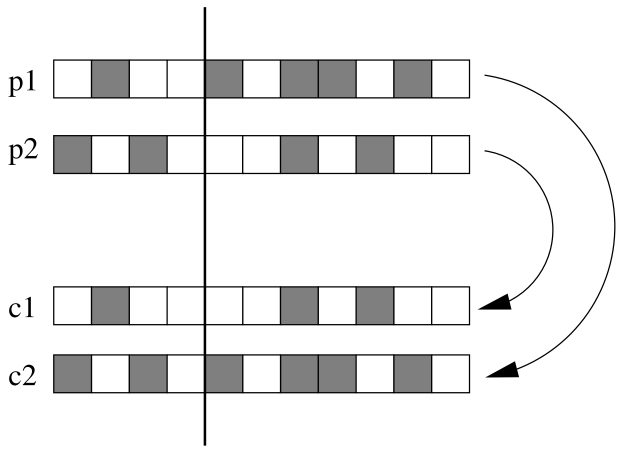


Figure 2: Cut selection

**Random selection** For each $i \in \{1..l\}$ we compute a random number $r_i \in [0, 1)$.

$$\forall i \in \{1..l\} : c_1[i] = \begin{cases} p_1[i] & \text{if } r_i < 0.5 \\ p_2[i] & \text{if } r_i \geq 0.5 \end{cases}$$

Respectively for $c_2$ with the <u>same</u> $r_i$.

This means: We decide for every bit in $c_1$ separately and randomly, whether it should be taken from $p_1$ or $p_2$

**Evaluation** Even though the crossover should work independently from the actual problem, the efficiency of different selection types might vary from problem to problem.

1. Knapsack problem: For the knapsack problem $\mathcal{P}_k = (l_k, s_k, f_k)$, we could define $s_k$ in a way, that every bit in the DNA represents an item in the set. 0 means that the item is not part of the solution, 1 means that it is. [1, page 158] It makes sense to use roulette wheel selection here. This is like trying to select/deselect a single item in every cycle. Cut selection might not work as well as the roulette wheel, because it selects/deselects many item at once.

2. TSP Problem: In section 5, I will use genetic algorithms to solve the TSP problem. In this case both cut selection and roulette wheel selection seem to be reasonable. I will justify this in section 5 since I do not want to get into to much detail right now.

## 2.4 The evolution cycle

One evolution cycle in genetic algorithms has the following structure:

1. Partitioning
2. Crossover
3. Mutation
4. Selection

**Partitioning** Since individuals die, when they perform the crossover, we do not want everyone to do that. In particular, we want to conserve the better individuals into the next generation and only let the worse individuals perform the crossover. The idea behind this is, not to loose good solutions.

Partitioning divides the population into two parts: The better individuals are put into a temporary data structure *tmp*. The others are marked to perform the crossover.

**Crossover** Each individual in the marked subpopulation is performing a crossover for a certain number of times. The parents are chosen randomly and are removed when they have reached their birth limit. The children go into *tmp*.

**Mutation** With a certain probability, a random bit in each individual in *tmp* is flipped. This still remains from the evolutionary algorithms.

**Selection** The $n$ best individuals from *tmp* are taken into the next generation

**The elitist scheme** A variation is the elitist scheme. This just means, that the best individual of the population is directly taken into the new generation, without mutation. It has been proved, that the elitist scheme converges to the optimal solution.

## 2.5 Parameters

In conclusion to this section, the goodness of the solution and the computation time of the algorithm depends on these parameters:

- $n \in \mathbb{N}$, the population size
- $cp \in [0, 1]$, the crossover probability. This is the percentage of the population that performs a crossover each generation.
- $br \in \mathbb{N}$, the birthrate. This is the number of crossovers, each individual is involved in each generation.
- $p_m \in [0, 1]$, the probability for each individual to mutate
- $\#g$: the number of generation cycles that are performed

# 3 GA with age structure

Genetic algorithms with age Structure are described in [1]. In this section I will give a short summary of the problems they should solve, the idea, the way the work.

## 3.1 Problems with GA

Genetic Algorithms are usually able to solve problems fairly well in a reasonable time. In general the principal holds: The better the solution, the longer the time to wait.

The population size $n$ and the number of generations $\#g$ mainly define what percentage of the solution space was actually covered by the search.

Local maxima are usually avoided because multiple searches are run. If, however, there is a large area in the search space that contains good solutions, surrounded by an area that contains worse solution, then there is still a chance for genetic algorithms to get stuck in the first area because all the individual that would be able to escape it, die. Of course, it is possible to escape, but it might take a long time until that happens.

If we were able to spread the individuals over a wider area in the search space, then situations like this would be less likely. We would eventually be able to find better solutions in a better time.

## 3.2 The idea of age structure

The idea of age structure contains two observation from nature concerning higher animals:

- Individuals usually do not die when they bear children.
- Individuals do not live forever. They die of old age.

Age structured genetic algorithms (ASGA) integrate this idea of age into the idea of Genetic Algorithms.

Every individual in the population gets an additional feature: its age. When a child is born, the age is set to zero. When the individual performs a crossover, it is still put into the temporary data structure along with its children. The age of every individual is increased in every generation cycle. Finally we define a probability $p_a$ for each age $a$ that describe how likely it is for an individual at age $a$ to survive. For one $a$, the lethal age, the survival probability is always zero.

The partitioning phase in the generation cycle is now modified to apply the age operator to every individual. First it checks, if this individual should perform and crossover. And then it checks independently from that, if the individual should die or survive.

## 3.3 Expected effects of age structure

The general cause of these changes is:

- There are more possibilities for bad individuals to survive, because they do not automatically die, when the are performing a crossover. If they are good enough to survive the selection process, they will survive.
- There are more possibilities for good individuals to die, because they do not survive indefinitely, but die at a certain age.

This reduces the evolutionary pressure that lies on the population. The expectation is that the genetic diversity is age structured populations is higher remains higher than in traditional populations.

## 3.4 Experiments

The paper [1] describes some tests using the knapsack problem. One of the main results is showed in figure 3. We are not interested in the AGA graph. The other graphs show, that it takes longer for the ASGA approach to gain fitness, but the final fitness is higher then the one in the GA approach.

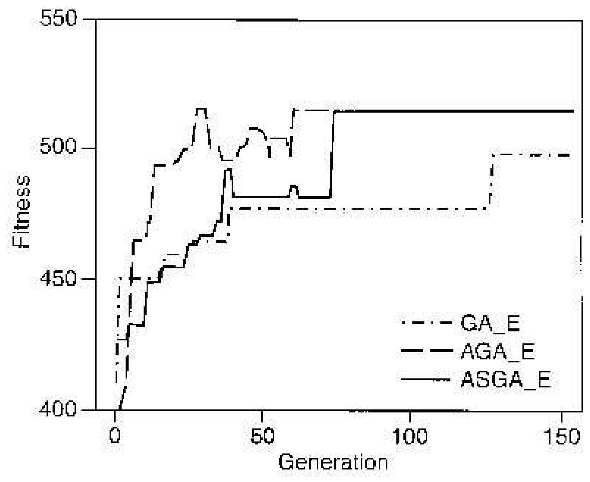


Figure 3: Fitness after $n$ generations in ASGA, AGA and GA [1]

## 3.5 A note on partitioning

It is not specifically described in [1] how the partitioning process works in detail. It makes sense, for genetic algorithms always to have to worst individuals perform a crossover and let the best individuals survive. The experiment in section 5.3 shows that. The alternative would be to pick the individuals randomly.

In ASGA however, individuals do not die on crossover. It might make sense to pick individuals randomly in this approach. One reason, why I was not able to completely reproduce the results from [1] is, that it lacks of a details description of the experiments.

# 4 Hybrid Populations

Obviously both approaches have there benefits and drawbacks:

- GA has a less overall performance concerning the goodness of the solution.
- ASGA has less evolutionary pressure, so it might take longer to get to a good solution.

The main goal is in all algorithmic problems is to get a good (optimal) solution fast. The idea of hybrid populations is to combine the approaches of GA and ASGA to get the benefits of both approaches.

### 4.1 The structure of hybrid populations

A hybrid population (HPGA) consists of two subpopulations:

- Individuals that behave like in the ASGA approach.
- Individuals that behave like in the GA approach.

It is possible for individuals of both subpopulations to mix their genetic material.

**Concretely** In HPGA, age is a feature of an individual. Although it is not included in the DNA, it is treated like a part of it:

Let $p_1, p_2$ be the parents and $c_1, c_2$ be the children involved in a crossover, $\alpha, r \in [0, 1)$ where $r$ is a random number and $\alpha$ is defined. Furthermore $c_1.asga = true$ if the individual $c_1$ has the *age* feature.

Then

$$c_1.asga = \begin{cases} p_1.asga & \text{if } r > \alpha \\ p_2.asga & \text{if } r \leq \alpha \end{cases}$$

Respectively for $c_2$ with the same $r$.

### 4.2 Preventing extinction

As said in section 3.4 it takes longer for ASGA population to get to a higher fitness than for GA populations. For hybrid populations, this means that in the beginning of the algorithm, GA individuals are be preferred by the selection process. In the extreme case, the ASGA subpopulation would shrink to such a small size that it cannot recover. It might even be extinguished. For this reason, I chose to modify the selection and the crossover process as well:

- The sizes of the subpopulations are fixed. Instead of a total population size, the parameters for the algorithm are a size for the ASGA subpopulation and a size for the GA subpopulation.
- The selection process ensures that the sizes of the subpopulations match these parameters in every generation.

This can mean that good solutions are discarded while worse solutions are chosen, if there are already too many good solutions taken from one subpopulation.

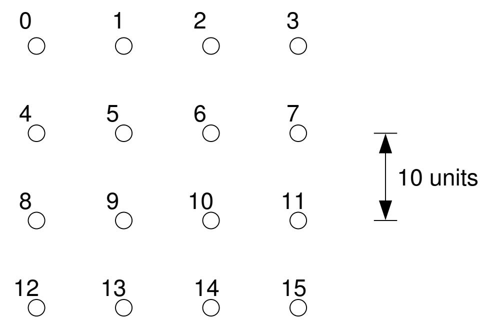


Figure 4: The TSP instance used for the simulations

### 4.3 Expected effects

Hybrid Populations can be viewed as two subpopulations with different purposes:

- The GA subpopulation applies more evolutionary pressure, thus converges fast.
- The ASGA subpopulation has less evolutionary pressure, thus converges slower, but finds ways out of local maxima faster that the GA subpopulation.

By performing crossovers between the populations, both subpopulations have the ability to communicate their results to each other.

The expectation is, the hybrid population finds better results than GA and finds them faster than ASGA.

## 5 Experiments

### 5.1 General setup

All the experiments performed use the same general setup. I use a specific instance of the euclidean TSP Problem (figure 4) and try to solve it using different population sizes. The different parameters are stated in the discussion of each particular simulation. In this section, I will discuss the model that I used to represent TSP. I define $\mathcal{P}_t = (l_t, s_t, f_t)$ as follows:

**Solution representation** An instance of the TSP consists of a set of (numbered) cities. A solution is a permutation of the order of these cities.

In order to represent the number of a city in binary, we need $x = \lceil \log_2 c \rceil$ bits, where $c$ is the number of cities. The DNA just represents the permutation of cities in binary.

Example: For 4 cities, the DNA String 00|10|11|01 would represent the permutation $0, 2, 3, 1$. Each city is represented in two bits.

The length of a gene is then $c \cdot \lceil \log_2 c \rceil$.

**Fitness function** The TSP tries to minimize the length of the path, but the genetic algorithm tries to maximize the fitness. Thus, I define the fitness to be

$$f_t = -\ \text{length\_of\_path}\ -\ \text{penalty}$$

The penalty counts the number of different cities that are contained in the solution. In a valid solution, every city appears exactly once. In order to ensure that valid solutions always have a better fitness than invalid ones, the penalty function evaluates to $x \cdot 2.0 \cdot d_{max}$, where $x$ is the number of cities that are not contained in the solution and $d_{max}$ is the maximal distance between two cities.

**Crossover scheme** The crossover scheme is roulette wheel selection, although cut selection would make sense as well. I used only one scheme for the experiments, because I wanted to test the effect of hybrid populations and not of different crossover scheme.

Roulette wheel selection and cut selection both make sense for this representation of the TSP problem, because the do not destroy existing sub-paths in the solution. Roulette wheel might change one city in the path. If the path was already correct in mentionable parts, it is likely, that it does not destroy these parts. The same is true for cut selection but not for random selection.

## 5.2 Interpretation of values

There are lots of values that can be derived from the simulations. I consider two values to be important:

- Best fitness: The individual with the best fitness represents the solution that would be presented if the algorithm stopped in this generation. Therefore, the best fitness of the population is the value that describes, how successful the population is.
- Standard deviation: The standard deviation of the fitness distribution can be seen as a measurement for the genetic diversity of the population. A high standard deviation means that the algorithm also allows weaker individuals to survive. The genetic diversity in the population is higher. A standard deviation that is very close to zero often means, that only one solution exists in the population, maybe with very few variations caused by mutation.

## 5.3 Experiment: Partitioning scheme

This experiment determines, whether it is more adequate to use random selection or selection of the worst in the partitioning phase (see section 3.5).

**Setup** I run 30 experiments in the every combination of GA/ASGA and random/worst selection. The values for population size, birthrate, crossover probability[2] and mutation probability is 50, 5, 0.7 and 0.05 respectively. The survival probability for ASGA individuals is 0.9, 0.9, 0.5, 0.0 for the first, second, third and fourth year.

[2]For worst-selection this means that the $30 \cdot 0.7$ worst individuals perform the crossover

Each simulation runs for 200 generations. Figure 5 shows the average best fitness of each generation and the average standard deviation.

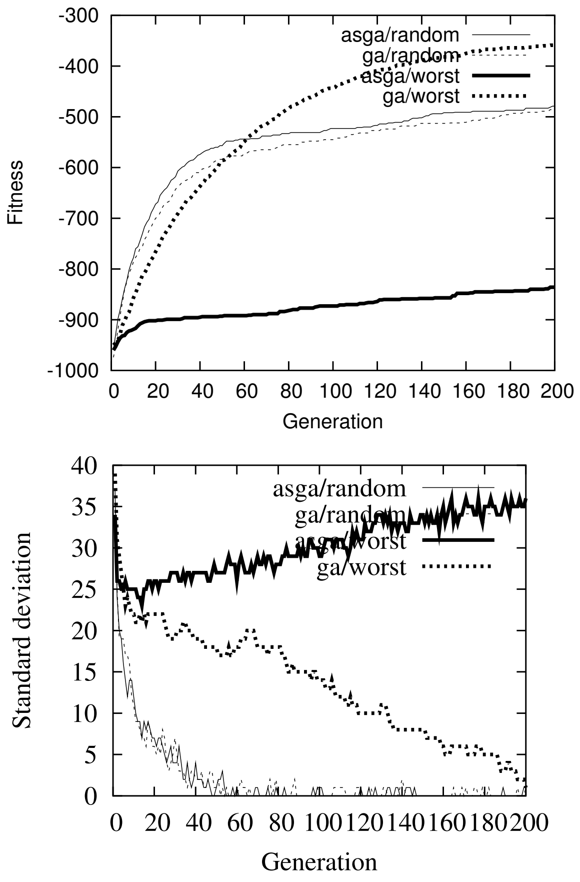


Figure 5: Best fitness and standard deviation with different partitioning strategies

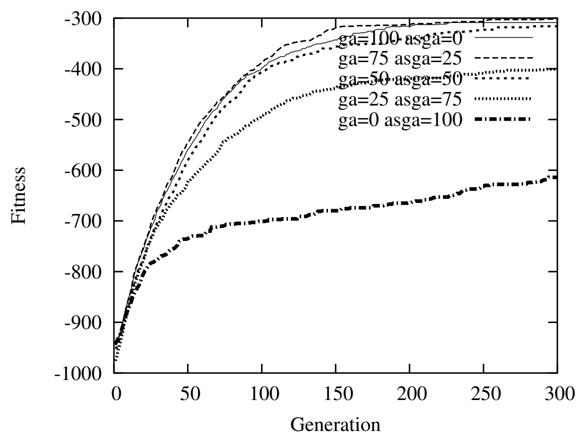
**Analysis** Considering GA, the worst-selection scheme seems to be the better choice, since it brings both better results and a higher diversity than the random selection scheme. For ASGA the result is quite unexpected. Random selection brings a better result, but the standard deviation converges to zero. Worst selection on the other hand maintains the diversity, but the best fitness is worse than in any other approach.

The bad performance of ASGA may have been caused by the small lethal age. However, different other tests have shown, that the standard deviation in both ASGA and GA converges to zero very quickly, when the random selection scheme is applied.

**Thus, for further experiments, I will not use random selection anymore**

## 5.4 Experiment: Subpopulations

This experiment tests hybrid populations with different sizes of the subpopulations.



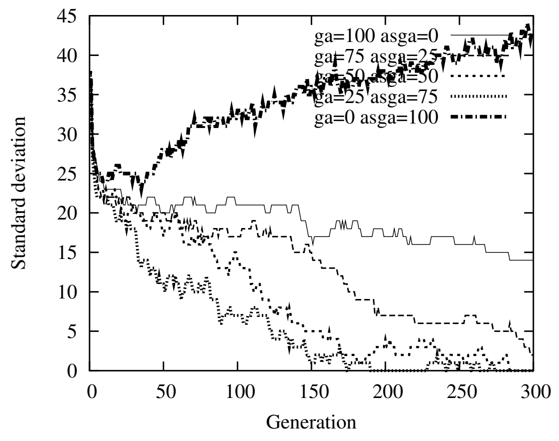


Figure 6: Fitness and standard deviation with different subpopulation sizes

**Setup** The population size, birthrate, crossover probability and mutation probability is 100, 5, 0.7, 0.05 respectively. The survival probability for ASGA individuals is 0.9 until and including the age of four, 0.5 at the age of five and 0.0 at the age of six. The paper [1] suggested a lethal age of 5 generations, which I used as basis.

I performed 30 simulations, each with subpopulations of 0, 25, 50, 70 and 100 ASGA individuals. I measured the average values of best fitness and standard deviation. The results are shown in figure 6.

**Analysis** Figure 6 shows, that the pure ASGA Population (ga=0 asga=100) has a bigger standard deviation than the other populations. The genetic diversity is higher. On the other hand, the value for the best fitness is far below the on of the other populations.

This result does not meet the expectations. In fact, it also contradicts the results of [1]. I will talk about this later in my conclusion.

All hybrid populations have a lower diversity than both pure populations. The fitness values show that there is no big difference between 0, 25 and 50 ASGA individuals.

The conclusion of this experiment is, that pure GA works better than both ASGA and Hybrid Populations.

## 6 Conclusion

I have performed more simulations with different configurations and some simulations with the knapsack problem. There was one configuration for the knapsack problem that really achieved a better result using ASGA. For the TSP and all configuration, the result was similar the one describe in section 5.4.

There might be different reasons for this result:

1. My program is buggy. I doubt it, since the program is not very big and I debugged it extensively. It is however very hard to identify this kind of bug because the program is highly depending on random numbers.

2. I did not exactly reproduce, the test conditions that N.Kubota and T.Fukuda used. This might be, because the configuration was not explained in the necessary detail (e.g. the size of the problem) in the paper. However, they performed multiple tests with different configurations as well and in every configuration, ASGA performed better that GA.

3. Since the paper does not talk about the size of the simulated problem, it is not possible for me to compare the length of the DNA for TSP and knapsack. Though, I also tried the TSP Problem with less cities (9) and the result was the same.

I can think of several reasons for my results, but they all don't seem to be very convincing.

One of the hybrid populations performed about as good as the pure population, but the genetic diversity was always much lower.

As the experiment in section 5.3 shows, the diversity also depends on selection scheme that is applied during the partitioning phase.

In contrast to my assumptions, the ASGA individuals performed rather badly. But the worst selection scheme is applied to both subpopulations. If the whole ASGA population has a lower fitness than the GA population, then some individuals with a better performance were chosen for the crossover. The number of individuals and their rating depends on the size of the subpopulations. In the cases "ga=25 asga=75" and "ga=50 asga=50", the worst GA individuals have a rather high fitness compared to whole population. This might be a reason for the rapid decrease of the standard deviation.

For further experiments, one might want to ignore the danger of extiction and just select the worst individuals of the whole population for crossover.

The complete source of my simulation suite is available on [2].

**A note on my references** I attended a lecture about optimization algorithms[3] held by Prof. Karsten Weihe at the University of Technology in Darmstadt. Genetic Algorithms were discussed in this lecture very briefly.

I read a lot about genetic algorithms in the WWW, but I did not read it specifically for this project. In fact, I read most of it a few years ago. That is why I didn't give any references to papers about genetic algorithms.

## References

[1] N.Kubota and T. Fukuda, Genetic Algorithms with Age Structure, Soft Computing 1 (1997), 155-161 © Springer Verlag 1997

[2] `http://www.knappi.org/asga`

[3] Lecture about Optimization Algorithms, Prof. Karsten Weihe (german web-page, but the linked material is in English)

`http://www.algo.informatik.tu-darmstadt.de/lehre/2002ss/optalgo/`