Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Population Size Adaptation in Evolutionary Algorithms

Adéla Šterberová

Supervisor: Ing. Petr Pošík, Ph.D.
Field of study: Cybernetics and Robotics
May 2019

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Ing. Petr Pošík, Ph.D. for his guidance, patience, motivation, and immense knowledge. Besides my advisor, I would like to thank my family for supporting me throughout writing this thesis.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 24. May 2019

Abstract

The population size is one of the most important parameters of evolutionary algorithms that affects their performance. There are several existing approaches for adapting the population size during the run of the algorithm (Parameter-less EA, Population pyramid, etc.) and for restarting the algorithm with a different population size (IPOP, BIPOP, etc.). The goal of this project is to compare these two methods for the genetic algorithm and the algorithm with the 'steady-state' model and for a chosen real-valued functions and the travelling salesman problem.

Keywords: evolutionary computation, evolutionary algorithm, parameter-less EA, IPOP, population size, genetic algorithms, steady-state, parameter

Supervisor: Ing. Petr Pošík, Ph.D.
FEE, Department of Cybernetics

Abstrakt

Jedním z nejdůležitějších parametrů evolučních algoritmů je velikost populace. Tento parametr ovlivňuje do velké míry jejich chování a výsledky. Avšak existuje několik metod jak se nastavení tohoto parametru vyhnout. Jednou je adaptace velikosti populace za běhu algoritmu (Parameter-less EA, Population pyramid, atd.) a druhou je restartování algoritmu s jinou velikostí populace (IPOP, BIPOP, atd.). Cílem této práce je porovnat tyto dvě metody pro genetický algoritmus a algoritmus s modelem nahrazování 'steady-state' na vybraných funkcích reálných čísel a na problému obchodního cestujícího.

Klíčová slova: evoluční algoritmus, parameter-less EA, IPOP, velikost populace, genetický algoritmus, parametr

Překlad názvu: Adaptace velikosti populace v evolučních algoritmech

Contents

Figures

Tables

Chapter 1

Introduction

Evolutionary computation (EC) represents one of the most powerful problem-solving tools. The EC comprises a variety of evolutionary algorithms (EAs). Their development might be seen as a reaction to the growing complexity and time requirements of mathematical problems being solved and corresponding demand for the automation of the calculations.

However, the general interest in the EC has increased rapidly in the second half of the twentieth century along with the increasing use of the computer technology. Another demand that the EAs meet is the robustness of their use, which means that these algorithms are capable of solving a great variety of computational problems ranging from a complex mathematical function to a design of a construction, all of these in an acceptable time. The robustness of the use is ensured mainly by two common features of the EAs. Firstly, the EAs are non-gradient methods and secondly, the EAs are stochastic. Owing to these features, the EAs (unlike any other optimization method) are able to find even the solution of the discontinuous functions. Therefore, the EAs have important role in so-called 'black-box' optimization. The 'black-box' optimization is a method employed when a user does not have enough information about the problem being solved, which is similar to the real-world problems.

Nevertheless, there are also some negative aspects of the EC, especially the requirement for a suitable parameters setting. The EAs have many parameters, some of them might affect the behavior of the algorithm and the result more than others. The necessity to set the parameters requires at least a basic user's knowledge of the EAs, which eliminates the robustness of their use. The size of the population is the parameter that affects the performance of the evolutionary algorithm most.

1.1 Related Work

The population size ranks among the most important performance-affecting parameters of EAs. For each problem solved by EC, there is a different suitable population size. The population size parameter affects the search for the optimum of a problem to a large extent. There are a lot of mistakes that can be made while estimating the optimal population size. Whereas underestimation of the population size might lead to never finding the optimal solution, overestimation of the size of the population might result in spending an unnecessary computational time. No methods have been found so far that would accurately calculate or determine the most suitable value of the population size.

However, several approaches exist that can find the optimal solution of the problem without knowing the ideal population size. The first approach adapts the population size during the run of the algorithm. The second approach restarts the algorithm every time with s different size of the population. In the context of this work, mainly two algorithms will be discussed - the parameter-less EA introduced by Harik and Lobo [HL99] focusing on the adaptation approach and the IPOP presented by Auer and Hansen [AH05] as the restart approach. Despite the fact that both of these algorithms are presented as parameter-less methods, they are not completely free of all parameters. They focus mainly on the population size parameter and how to avoid its setting.

1.2 Goals of Thesis

The aim of this thesis is to implement the two selected algorithms - the parameter-less EA and the IPOP - and to compare their behavior when dealing with various problems. An important part of the thesis is the comparison of these two algorithms and the EAs with a constant size of the population, which should clarify the advantages of the parameter-less EA and the IPOP algorithm. Another aim of the thesis is to determine whether these two algorithms are suitable to be used on selected class of the problems but also whether they provide equally good results on different models of the EAs. Another important goal of this work is to examine the 'parameterlessness' of these two algorithms, their attitude to the parameter setting and handling.

This thesis will be organized as follows. Opening chapter introduces the topic of evolutionary algorithms, their principles and variants. Next chapter describes the importance of the population size parameter along with the possible approaches how to avoid its setting and with the algorithms which applies these approaches.The last chapter presents the results of the

experiments on these two algorithms and the discussion of the results.

Chapter 2

Background

2.1 Evolutionary Algorithm

In order to understand the EAs, we first need to understand what the term 'evolution' means. Darwin [DC03] explained the evolution as a long-term process of changes that occurs over time and generations by which biological organisms acquire new heritable characteristics. The EAs are based on Darwin's Theory of Evolution and on his thought of natural selection. Furthermore, in nature all candidates of one population compete with each other and only the fittest candidates can survive and create a new generation.

Evolutionary computation is a subfield of artificial intelligence which deals with the algorithms inspired by the theory of evolution and applies them to the optimization problems, especially to the difficult ones which when being solved with the application of other methods would take a lot of time, as mentioned in chapter 1. The terminology used to describe the evolutionary computation borrows words from the evolution in nature; we use terms such as reproduction, genetic information, mutation, recombination, selection etc.

2.1.1 Principle of Evolutionary Algorithm

A candidate or an individual in the biological evolution is a living creature characterized by its genetic information. In terms of the EA it is similar. The candidate is a representation of a solution to a particular problem. In the real-valued optimization problems an individual is for example a vector of real numbers and the length of the vector depends on the dimension of a problem. The group of individuals that one can work with is called a population. It is a set of possible candidate solutions. The number of candidates in the population is called the population size. For the context of this work, it is

the most important parameter of an EA as mentioned in chapter 1.

An optimization problem is described by a function that determines the quality (fitness) of an individual. Correspondingly, it is called a fitness function and it evaluates all candidates. According to Eiben and Smith [ES03], the fitness function in terms of the biological evolution measures the ability of a living organism to survive and adapt to changes. In the context of mathematics, a fitness function determines the quality of a solution.

The fitness of individuals has a big influence on the selection of the next-generation parents. The fitter individuals have higher chance to be selected as parents because they are better adapted to the environment and thus their genes have a greater chance to be copied to offspring. However, the EAs are stochastic methods, which means that even the candidates with lower fitness have an opportunity to become parents and to survive. This also eliminates the situations where the EAs would end up in a local optimum.

Eiben and Smith [ES03] mention that when creating a new offspring there are two types of variation operators. Firstly, there is a unary variation operator which is called the mutation. When used as a tool for reproduction, it is the asexual reproduction. A new offspring is created by random changes of the parent and only one parent is needed to create a new individual. However, a mutation does not always have the role of the creator of a new candidate. It is an operator that different types of algorithms use differently. For example, evolution strategies use mutation as an asexual reproductive tool but genetic programming uses recombination for reproduction and it uses mutation only to change created individuals. These algorithms will be discussed later. Secondly, there is a binary operator - recombination. It is an operator that needs two parents. It creates a new offspring by randomly combining genes from both parents. Both of these methods are always stochastic.

Another step in the evolution is a replacement (sometimes called 'survival selection') which must be performed after the creation of the new individuals. This is usually the deterministic part of an EA and the process depends on a model that is used. This replacement is closely related to the selection. An example of the replacement strategy is when the number of children is bigger than the population size, only the same number of children as the size of the population can survive and then replace their parents and create a new generation. So the selection is made on the side of the offspring. On the contrary, when there is less children than parents the selection must be done from parents. The children replace the selected parents, that means even some older individuals may survive the selection.

Finally, the last part of an EA is a termination condition. There are many kinds of termination conditions, the most common are an evaluation termination, a time termination or a user termination. An evaluation termination

condition means that the total number of evaluations that the EA can perform is given to it by the user. When the number is reached the algorithm ends. A time condition is very similar, the algorithm ends after a specified period of the time. The last one means that the user decides when the algorithm should end, usually when he is satisfied with the found optimum of the function.

In algorithm 1 below, the principle of EAs explained in this chapter is described in a pseudo-code.

Algorithm 1: How evolutionary algorithm works in pseudo-code

```
initialization of random population;
evaluation of each individual in population;
while not termination condition do
    select parents for a new generation;
    make a new offspring through recombination and mutation;
    evaluate new candidates;
    select new candidates to replace their parents;
end
```

2.2 Variants of Evolutionary Algorithms

There are many types of the EAs. Algorithms differ from one another mainly in two aspects - the manner how it works with the populations and how it creates new offspring. Moreover, they also differ in using mutation and recombination. In the following sections the basic principles of the most important algorithms or methods are explained.

2.2.1 Evolutionary Programming

Fogel et al. [FOW66] presented evolutionary programming (EP) in 1966. In the past, this method was used to develop solutions in the form of finite state machines. De Jong [DJ06] clarified the idea behind EP very simply. He explains that the size of the population is N therefore there are N parents. The reproduction in this case is asexual. In other words, each parent makes one new offspring using mutation. Afterwards, the population is twice the size of the previous one. All candidates are evaluated by a fitness function and only N number of fittest individuals is allowed to survive. Those N fittest candidates become a new generation.

2.2.2 Evolution Strategies

Evolution strategies (ES) described by Schwefel [Sch75] are methods for real-valued functions where each individual is given by a vector (of a real numbers). The ES consists of many strategies. The strategy where the behavior of the ES might be easy to understand is $(1+\lambda)$-ES. This is also an asexual reproductive method and creation of a new offspring is done by mutation. Mutation in this case means normally distributed perturbation of some of the vector component. $1+\lambda$ means that one individual of the population creates λ new offspring. All these $1+\lambda$ candidates compete with each other. The one with the best fitness is chosen into new generation. Interestingly, when looking for the optimal solution, the parameters of this algorithm evolve during the run along with the solution.

2.2.3 Genetic Algorithm

Genetic algorithms (GAs) were introduced by Holland and Goldberg [GH88] and they are used mainly for the binary problems. De Jong [DJ06] proposed that mutation is performed as a 'bit flip' with fixed probability and recombination is done randomly by choosing a sequence of genes from one parent, the rest of the genes is taken from the other parent. Both sequences are juxtaposed to create a sequence of a new individual. This is how a new offspring is created. The population of the size N creates N new candidates. All of these N new candidates replace their parents and they make a new generation of parents. The fitness of parents does not play any role in the replacement. However, it affects the origin of the new individuals. The parents with better fitness provide more genes to the new offspring.

2.2.4 Differential Evolution

Differential evolution (DE) is a heuristic method proposed by Price and Storn [SP95] for real-valued functions. The initial population is created randomly. The creation of a new individual might be simply called as a mixture of a parameters. Price and Storn [SP95] reported that the DE generates a new offspring (a vector of numbers) by 'adding the weighted difference between two population vectors to a third vector'. This process might be called the mutation. During the process, many vectors are created. Afterwards, one vector is mixed with another - the target vector - and creates a trial vector. The selection process decides whether the trial or the target vector should survive and it chooses the one with a better fitness.

2.2.5 Genetic Programming

Genetic programming (GP) introduced by Koza [RK92] is a very similar method to the genetic algorithms except the fact that an individual in GP is represented by a computer program which should solve a problem. The population is a set of programs that change and improve randomly. A program may be represented in many different ways, e.g. as a parse tree, a linear sequence of instructions, indices to grammar production rules, etc.

Chapter 3

Population Size and its Adaptation Methods

3.1 Population Size

The importance of the population size has already been mentioned in chapter 1. It is the parameter whose settings is required by all of the algorithms mentioned above. Every problem solved by evolutionary computation has a different level of difficulty. The optimal size of a population depends on this difficulty and other parameters that are hardly possible to define for the problems of the real world. According to Lobo and Goldberg [LG04] there are only two types of errors which one can make when guessing the appropriate population size:

- Undersized population: when the guessed size of the population is too small, GA converges before it reaches the optimal solution which brings deterioration of quality. (Convergence means that GA population loses diversity, and thus the ability to search further.)
- Oversized population: when the size of the population is oversized, finding an optimal solution takes a lot of time and fitness evaluations, which means a delay in the computational time.

These two mistakes and the reason why population size is so important parameter might be seen in figure 3.1. The quality and the time penalty is not wanted. However, searching for this parameter requires to carry out a lot of experiments and it might be time-consuming for a user. Therefore, it is better to avoid searching for and setting of the optimal population size. Some approaches that do not require this setting are described in the next chapter.

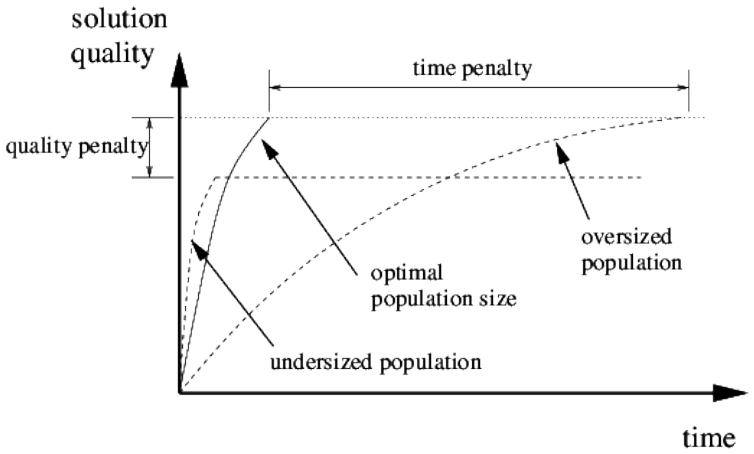

Figure 3.1: An example of a run of the EA with undersized population, oversized population and an optimal size of population. [LG04]

3.2 Adaptation and Restart

It has been mentioned that the population size is the crucial parameter in the EAs. However, there are some approaches that effectively avoid setting this parameter at the beginning of the algorithm. For the purpose of this work, two of them will be discussed. The first one is the parameter adaptation. It was described by Harik and Lobo [HL99] as a centralized control method where the parameter changes with central learning. In practice, this means that the main program maintains more populations (or even algorithms), gathers information and then decides for example which population will run. Another work which deals with the adaptive population sizing in more details is Smith and Smuda [SS93]. The second one is the restart approach, which is easier to describe. At any time when algorithm is heading to a dead end it restarts the EA with a different size of the population. This work focuses mainly on the parameter-less EA which belongs to the first approach and on the restart evolution strategy - the IPOP.

3.3 Parameter-less Evolutionary Algorithm

The parameter-less EA is an algorithm that adapts the most important parameter - the population size. It was introduced by Harik and Lobo in 1999 [HL99]. In order to make the algorithm run easier, their algorithm ignores the mutation. It also disposes of the selection rate and the crossover probability by setting these parameters to the constant values at the beginning. The reason for that is the same as for ignoring the mutation. However, the values

Algorithm 2: The parameter-less EA in pseudo-code.

```
initialize the population size;
initialize a random population;
evaluate each individual in population ;
while not termination condition do
    choose population to run by the counter;
    if not created then
        create population
    end
    run one generation of it;
    if a bigger population has better average fitness than a smaller then
        delete the smaller population;
        reset the counter;
    end
end
```

must be chosen wisely so that the combination of these parameters ensures the growth of the fitness.

Elimination of the population size is a little more complicated. Harik and Lobo [HL99] reported that 'the idea is to establish a race among populations of various sizes' which is the core part of the parameter-less EA. The algorithm maintains more populations and it is the smaller population that it favours. It gives them a greater opportunity to converge to the optimal solution. For that a counter of base 4 is used. The position of the counter that has changed determines which population should be running and make a new generation of this population. The lowest and the highest position of the counter means the smallest and the biggest already created population, respectively. Each newly-created population is twice the size of the previous one. And after a population is created, it is placed at the end of the list of the populations which keeps them arranged by the size in the growing order.

Altogether a smaller population has the opportunity to create four times more generations than the larger population. Moreover, each population is allowed to use twice as many function evaluations as the bigger population. The population is deleted from the list when a bigger population has better average fitness. Every time it happens the counter is reset and starts counting from zero again. The behavior of the parameter-less algorithm is described in algorithm 2.

3.4 Parameter-less Population Pyramid

The parameter-less population pyramid (P3) evolutionary algorithm was introduced by Goldman and Punch [WGP14]. It is a technique that also does not require the population size parameter or any other parameter to be set. Nevertheless, it is a very effective method that finds an optimal solution in a short time. Similarly to the parameter-less algorithm, P3 does not have only one population but it keeps more populations in the structure of a pyramid. That means each level of the pyramid is considered one population. A newly-created candidate belongs to a higher level than its fittest parent. However, a new candidate may only be included to the pyramid if he does not already exist in any other layer of the pyramid. The algorithm stops searching when it runs out of time (or possible evaluations given) or the solutions lose diversity.

3.5 Restart Evolution Strategy IPOP

Auger and Hansen [AH05] introduced the restart evolution strategy with an increasing population size (IPOP) in 2005. The algorithm firstly generates a random population. The initial size of the population is supposed to be rather small. Then there are two possible conditions for restarting the algorithm. The first one is the convergence of the population. Its criteria may vary for different problems being solved. However, the idea is the same, the convergence of the population is noted when most of the population is identical. Another possible case of the convergence is the situation when all individuals in the population are so similar that their fitness is almost the same.

The other condition is the stagnation of the algorithm. The stagnation is often difficult to recognize, what is means in general is a failure to develop or to progress. In terms of EAs, it means that the algorithm creates new generations but there is no new individual whose fitness would bring improvement in searching for the optimal solution. That means the algorithm heads to a dead end wasting a lot of fitness evaluations, which means a waste of time from the user's point of view. The difficulty with this condition was to determine whether the number of evaluations conducted that were spent is already a stagnation or the population still has the chance to get closer to the optimum. The exact determination of the stagnation condition fulfillment required a lot of experiments. Both conditions will be specified in chapter 4.5.

When one of these conditions occurs, the algorithm should restart itself. Afterwards, the IPOP launches a new EA with the previous population size multiplied by two. According to Auger and Hansen [AH05] the multiplication

Algorithm 3: The IPOP algorithm in pseudo-code.

```
initialize the population size;
initialize a random population;
evaluate each individual in the population;
while not termination condition do
    if population has converged then
        restart EA;
        double the population size;
    end
    if stagnation then
        restart EA;
        double the population size;
    end
    do next generation;
    evaluate the new generation;
    replace the old one;
end
```

of the population size might be by any value from 1.5 to 5, however they also noted that the factor of 2 might be appropriate because it is big enough and it allows to perform more restarts than a higher factor. The behaviour of the IPOP algorithm is described in algorithm 3.

3.6 BI-Population Restart Strategy

The BI-population restart strategy (also called BIPOP) was introduced by Hansen [Han09], whose work is based on the article on the IPOP [AH05] with some modifications done in the algorithm. Hansen [Han09] proposed a multistart scheme for the BIPOP algorithm consisting of two interconnected restart methods, both with the equal number of evaluations. One of the method of BIPOP increases the population size, the size is doubled with each restart. The conditions for the restart are the same as in the article on the IPOP [AH05]. This tactics should ensure a global search for the optimum of the problem. The second method uses varying small sizes of the population. Local scanning and searching for the optimum is robust and fast thanks to this tactics.

3.7 Are Parameter-less Approaches Really Parameter-less?

When speaking about parameter-less methods, it is important to explain what it means when a method has no parameters. In reality, more possible things might be meant by this term. The parameter-less method might be a method whose parameters are set to the constant values which generally give the best result. Another possible explanation of a parameter-less method might be the situation when a method with some parameters is inside another method which controls the parameters of the inner method and which as well has its own parameters, to which the optimization result is not much sensitive. One such example of the last-mentioned approach towards the meaning of the term 'parameter-less' is the parameter-less EA. There are almost no parameters that a user should set but the algorithm itself chooses and works with the population size which is an important parameter. The IPOP algorithm might be considered also a parameter-less method where the method with the parameters is inside another method. However, is seems that even the parameters that the exterior method uses are important for the result.

Chapter 4

Experimental Study

This thesis focuses mainly on the parameter-less EA and the IPOP algorithm from the reasons which are as follows. The method of the parameter-less EA, which represents the adaptation approach, is easily transferable from the original algorithm to the other population algorithms. The IPOP represents a restart evolution strategy with an increasing population size. The experimental study is exclusive of the P3 evolutionary algorithm because its method of handling the population is highly linked to a particular EA and cannot be easily transferred to another EA. Another algorithm which is not included in the experimental part of this work is the BIPOP. The reason is that Hansen et al. [HAR+10] reported that the results of the BIPOP do not differ significantly from the IPOP when tested on the problems selected and listed below in chapter 4.1.

4.1 Test Problems

Altogether, the algorithms - the parameter-less EA and the IPOP - were tested on five benchmark problems that can be divided into two categories: real-valued functions and travelling salesman problems (TSPs). All of these problems are the minimization problems which means that the faster and lower the graph develops the better. It is Rosenbrock, Griewank and Rastrigin functions that were chosen from the real-valued problems. All of them were performed on dimension 5 and 20. The other two benchmark problems chosen are the TSPs. The first one is a TSP where the candidates (the solutions) are generated randomly. The second one is enhanced by the nearest neighbor heuristic function that improves the generated solutions. The dimension of the TSP problem is determined by the number of cities that are given. The data set with specified cities and coordinates used in this thesis was found on [Rei95]. These are the file names that were used: 'eil51', 'berlin52', 'eil76', 'pr76', 'rat99'.

To compare all the problems mentioned above, the evolutionary algorithm using the generational model and the 'steady state' model were used. Both were taken from the python library called 'inspyred' available from [Gar15]. The generational model is used in the genetic algorithm which is also its name in the 'inspyred' library. Its behavior is described in the chapter 2.2.3. The 'steady state' model is used in the differential evolutionary algorithm (DEA) in the library.

4.1.1 'Steady State' Model

The 'steady state' model can be described fairly simply. Only two parents are selected and recombined in each iteration. They create two new offspring on whom the mutation is applied. These offspring replace the worst two candidates in the old population. The advantage of this model is the small number of evaluations for each generation. However, it takes a long time (a lot of fitness evaluations) to find an optimal solution.

4.2 Experiment Settings

In this section the setting of the GA and the DEA model (from the inspyred library) will be described. The GA model uses rank selection where the most important part is the ranking of the fitness of the candidate relative to other individuals. It uses a single-point crossover where one point in the genetic information of the parent is chosen randomly and from that point the genetic information from both parents are swapped. The replacement in the GA is generational as mentioned in the chapter 4.1.

The DEA model uses, by default, the tournament selection. The two candidates are chosen randomly and they compete with each other, then the fitter candidate is selected. It also uses the heuristic crossover, which combines the information from both parents in order to produce a better candidate, and the Gausian mutation, where a Gaussian distributed random value is added to the chosen gene. Finally, the replacement uses the 'steady-state' model which has been described in chapter 4.1.1.

4.3 Performance Measurement Methods

For the purpose of observing the behavior of the algorithms, an observation function was created. The function is called for every iteration of the algorithm and it was designed to record the best solution found so far. Therefore, if

there is an improvement, the fitness value of the candidate is written to the file along with the number of total evaluations already conducted. Also additional information is written to the file, such as the size of the population from which the candidate was found, the number of generations of this population and finally the solution itself.

In total, 15 experiments were carried out for each setting of the algorithms and they were saved into the separate files. In order to depict the development of the solution all these runs had to be averaged. At any moment the best result so far for the given number of evaluations has to be found in all of these 15 files and the arithmetic mean is calculated from these 15 values. The total number of evaluations given to the algorithm varies with the different dimensions of the problems. As far as the real-valued functions are considered, the total number of evaluations for each run was $10000 \cdot dimension$ of the problem. A smaller number of evaluations should be sufficient for the TSPs, therefore $2000 \cdot dimension$ was used.

4.4 Parameter Sensitivity Study for Algorithms

Despite the fact that both algorithms compared in this work are supposed to be without parameters, as explained in the description of the parameter-less EA and the IPOP algorithms (chapters 3.3 and 3.5), it has been also suggested that they have some settings that might influence the results. The correctness of these settings has to be tested. It is not only the influence of the correct settings on the parameter-less and the IPOP algorithm that has to be tested, but also the influence on the methods like the GA or the DEA which were used. All these following observations might help with the explanation and the description of the results.

4.4.1 Influence of Population Size on GA or DEA

The overall results of this work were influenced not only by the parameters of the two algorithms but also by the behavior of the two methods used for the purpose of testing - the GA and the DEA. The population sizes 20, 40, 80, 160, 320 and 640 were set to the EA, firstly with use of the GA (figure 4.1) and afterwards with use of the DEA (figure 4.2). The labels 'popsize_number' describe what size of the population was tested (the number is the size of the population).

The behavior of the GA was as expected. An EA with a small population size improved the solution faster than one with a larger population, but it stopped improving its solution very early in the algorithm. Therefore, it ended far from the optimal solution. On the contrary, an EA with a larger

size of the population was improving more slowly. Yet, it reached a better solution than one with a smaller population. However, the behavior of the DEA was not expected at all. For any problem that was tested, the graph showed almost the same result for the experiment with the small size of the population as for the one with much bigger size of the population. It means that for the DEA (from the inspyred library) the size of the population is not the important parameter. The possible explanation of this behavior is that this model (the DEA) works differently with the population than it was described. It only creates two new offspring in each generation and those two candidates replace the worst two individuals from the previous generation. It can seem that the DEA model uses only these two individuals as a population but what would one call a population in the EA this model uses only as an archive of the individuals. That might be the possible explanation of why the size of the population does not matter in the DEA model.

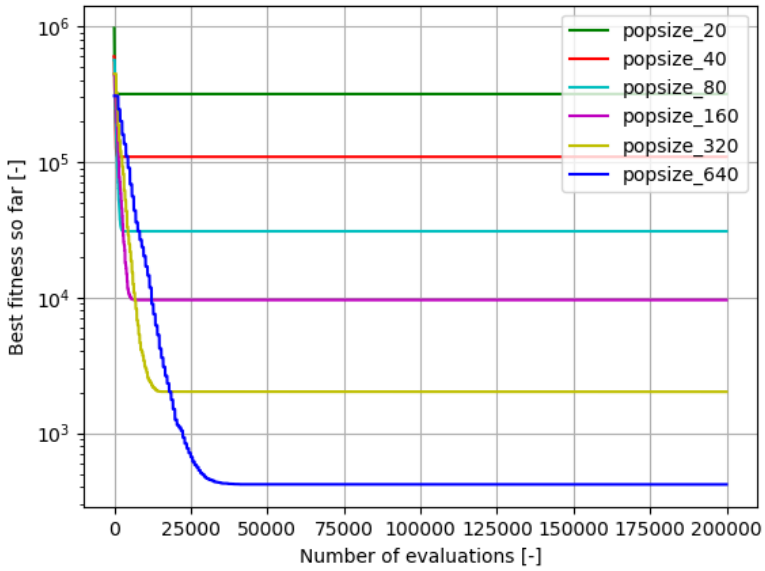

Figure 4.1: Comparison of the EAs with constant sizes of the population on the Rosenbrock function of dimension 20 with use of the GA. The Y axis is depicted in the logarithmic scale.

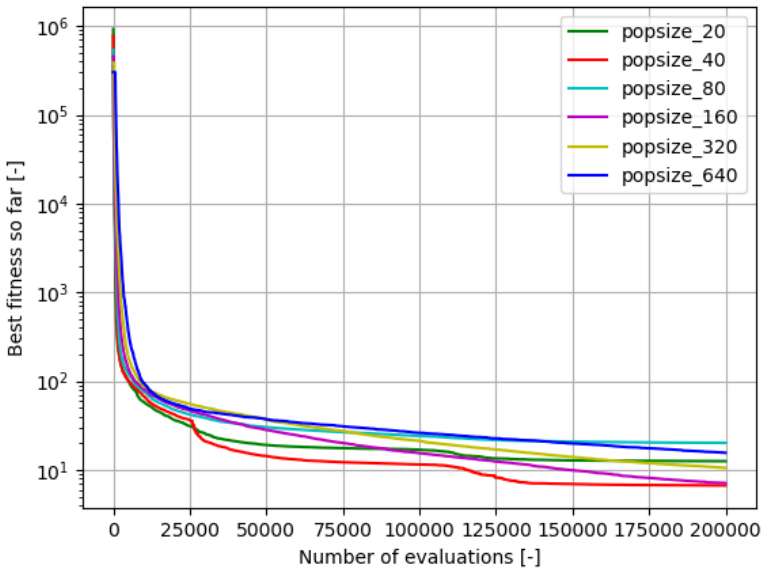

Figure 4.2: Comparison of the EAs with constant sizes of the population on the Rosenbrock function of dimension 20 with use of the DEA. The Y axis is depicted in the logarithmic scale.

4.4.2 Influence of Initial Population Size on Parameter-less and IPOP Algorithm

Both algorithms (the parameter-less GA and the IPOP) were designed so that the size of the population does not matter. The algorithms do not require the user to set the optimal population size to return good results. Nevertheless, an initial population size must be set somehow and a mistake might be made there.

The initial population sizes 2, 10, 20, 50, 100 and 200 were set at the beginning of each algorithm. The following figures (4.3-4.4) show the behavior of the parameter-less EA and the IPOP algorithm with these initial population sizes. The last number in the label names indicates which size was tested. The graphs show that the setting of the initial population size does not influence the behavior of any algorithm. The solution developed similarly in all tested cases.

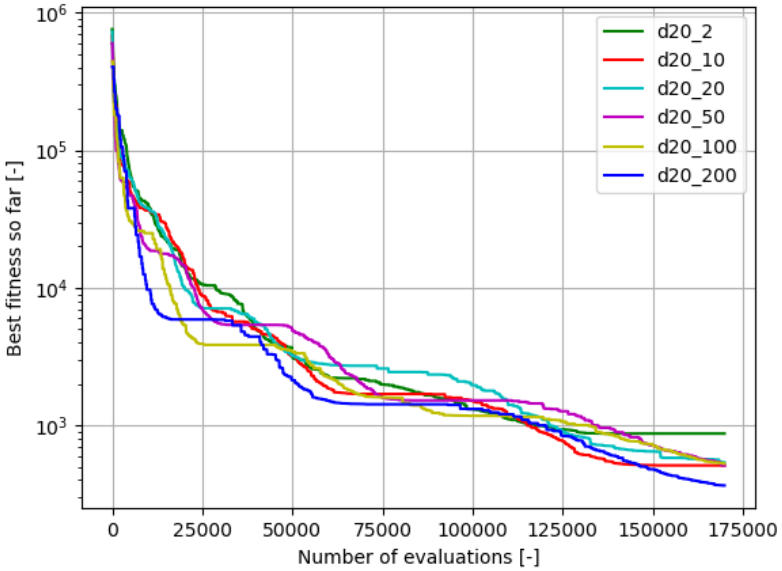

Figure 4.3: Comparison of the parameter-less EA with different settings of the size of the initial population on the Rosenbrock function of dimension 20 with use of the GA. The Y axis is depicted in the logarithmic scale.

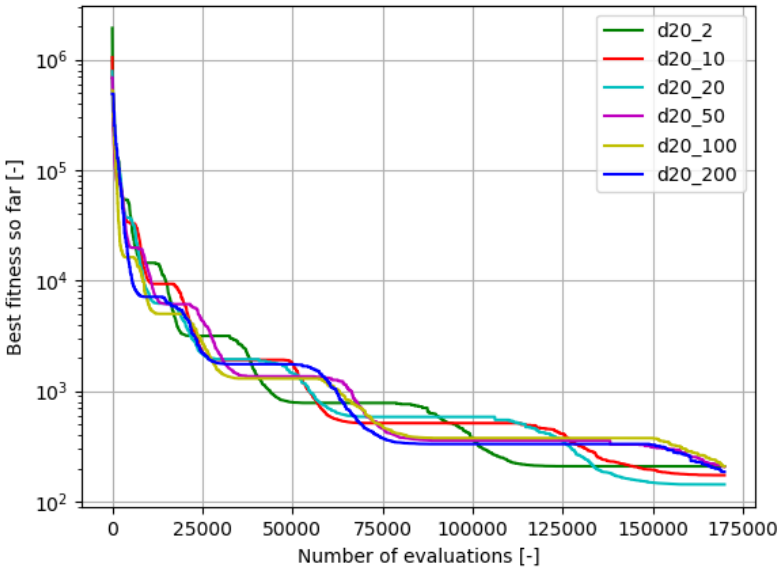

Figure 4.4: Comparison of the IPOP with different settings of the size of the initial population on the Rosenbrock function of dimension 20 with use of the GA. The Y axis is depicted in the logarithmic scale.

4.4.3 Influence of Restart Conditions on IPOP

As mentioned in the chapter 3.5, there are two conditions for the IPOP restart. The first one is the convergence and the other one is the stagnation. The convergence condition was not that difficult to determine. In case of the TSPs the condition was fulfilled when most of the solutions were the same. For real-valued functions it was enough when all the candidates were so similar that their fitness was almost identical. The exact setting of these conditions will be mentioned later in chapter 4.5. To find that the condition of stagnation was fulfilled needed more effort. The stagnation was described in chapter 2 as a long time without improvement but what a 'long time' means is not clearly defined by any source. The time was measured by the number of evaluations in these experiments. It is obvious from the tests that the number of evaluations needed to the recognition of the stagnation is different for each size of the population. If the population size is 20 and stagnation occurs after n evaluations, the stagnation for the population of size 1000 cannot be recognized also after n evaluations.

The following figures (4.5-4.7) show the behavior of the algorithms with five different stagnation conditions. In the labels the 'conv' ending indicates the condition of convergence and the 'stag' ending indicates the stagnation condition. The number connected to the 'stag' is the number of evaluations that must be wasted without improvement before the stagnation is recognized. It may also be a multiple of a population size (not only a concrete number) before the 'stag'. The tested conditions were: 200, 1000, $10 \cdot \textit{population size}$, $100 \cdot \textit{population size}$, $1000 \cdot \textit{populations size}$. As might be seen in the figures 4.5-4.7 the results were very close. However, the condition of $10 \cdot \textit{population size}$ evaluations had the best results. Therefore, this condition was used for all the experiments.

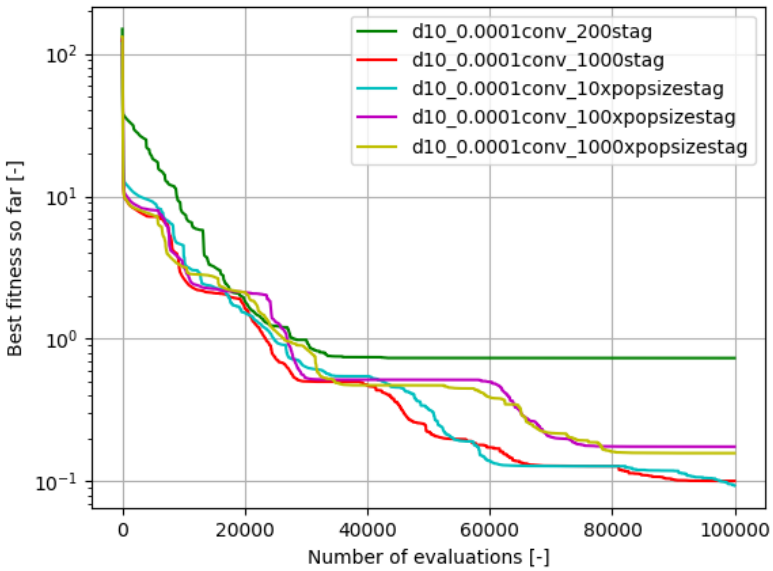

Figure 4.5: Comparison of different stagnation conditions on Griewank function of dimension 10 with the use of the DEA. The Y axis is depicted in the logarithmic scale.

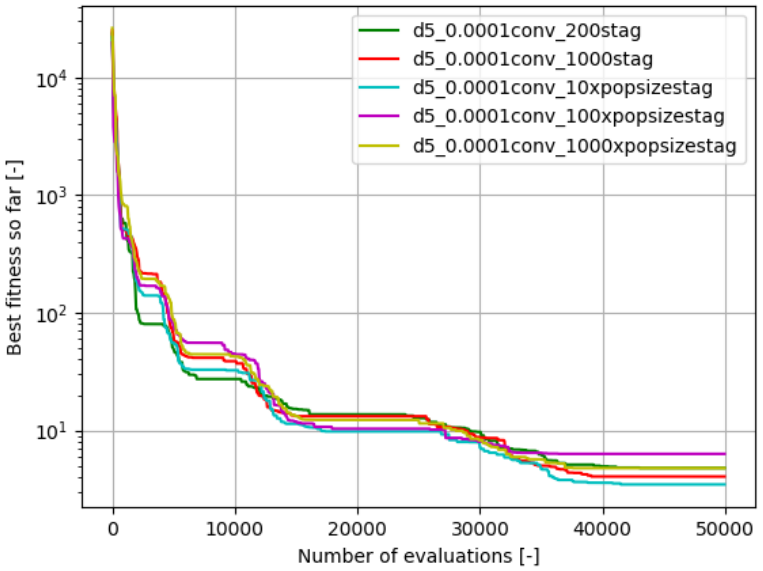

Figure 4.6: Comparison of different stagnation conditions on Rosenbrock function of dimension 5 with the use of the GA. The Y axis is depicted in the logarithmic scale.

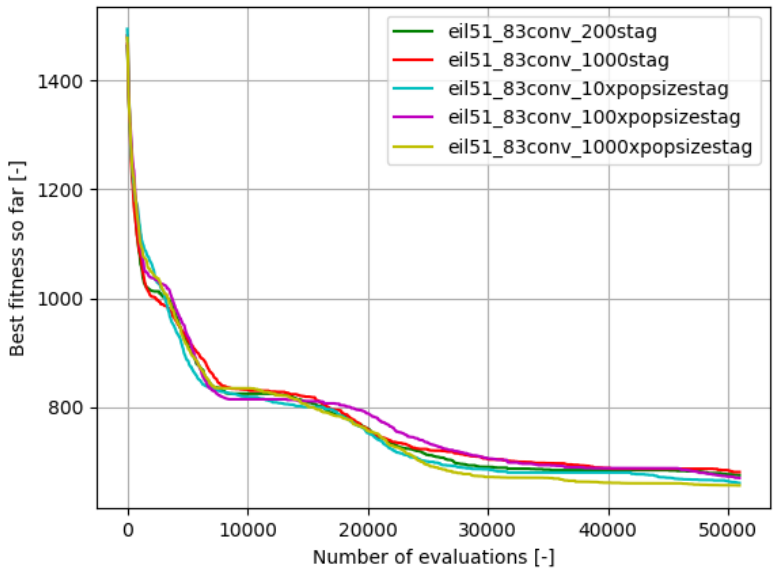

Figure 4.7: Comparison of different stagnation conditions on the TSP problem (eil51) of dimension 51 with the use of the GA.

4.4.4 Influence of Counter Base on Parameter-less EA

The counter determines which population should make the next generation as described in chapter 3.3. Harik and Lobo [HL99] reported that the base of this counter should be set to the number 4. The reason is that each population is allowed to create 4 generations before the next population, which is twice the size of the previous one, can create its one generation. As a result, when any population uses $2n$ evaluations, the next bigger population uses only n evaluations.

Bases 3,4,5,6 and 10 were gradually set up to the counter for the testing purposes. The following figures (4.8-4.9) show the behavior of the parameter-less EA with these bases. The last number in the label section indicates which size of the counter base was tested. Conclusion that can be made from the figures is that it is the base 4 which had the best result, which was also proposed in the article [HL99].

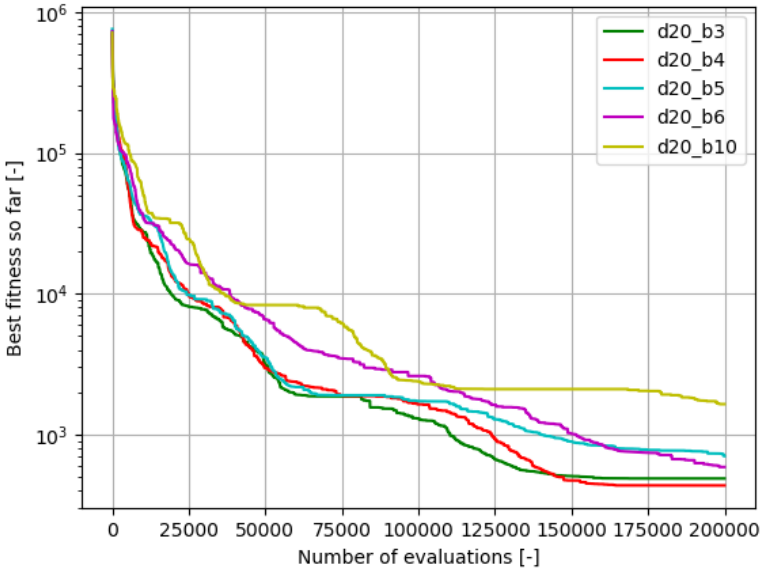

Figure 4.8: Comparison of the parameter-less EA with different settings of the base of the counter on the Rosenbrock function of dimension 20 with the use of the GA. The Y axis is depicted in the logarithmic scale.

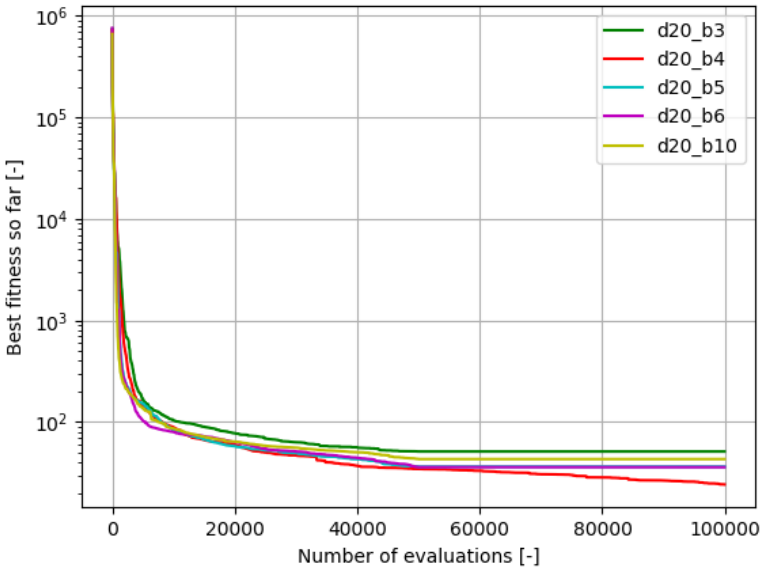

Figure 4.9: Comparison of parameter-less EA with different settings of the base of the counter on the Rosenbrock function of dimension 20 with the use of the DEA. The Y axis is depicted in the logarithmic scale.

4.5 Comparison of Population-size Adaptation Techniques

The graphs showing the behavior of the algorithms for different benchmark functions are presented in this section. All figures capture the process of developing of the optimal solution but also the average size of the population. The two algorithms - the parameter-less EA and the IPOP are compared with the EAs with a constant size of the population on chosen models (meaning GA and DEA). The Y axis is depicted in the logarithmic scale for the real-valued problems but in the standard scale for the TSP problems. To describe the label section, the label 'popsize_number' means the EA with a constant population size of the number. The abbreviation 'PL' in the labels stands for the parameter-less EA and 'IPOP' is simply the IPOP algorithm. For the parameter-less EA and the IPOP, the graphs also contain the average size of the population that generated the best solution so far (denoted with labels '_averagepopsize').

Both algorithms started with the population size set to 20. Counter with the base 4 was used for the parameter-less EA because it provided the best results as mentioned in the chapter 4.4.4. As far as the IPOP algorithm is concerned, there are 4 conditions that had to be set. The first one is the convergence condition for the TSPs which was set to the 83%. That is to say, the condition was fulfilled when 83% individuals of the population were identical. The second one, the convergence condition for the real-valued functions, was met when the difference between the candidates' fitness (in all coordinates) was less than a 10^{-4}. The third and the fourth one is the stagnation condition for the TSPs and for the real-valued functions, respectively. Setting of these conditions was very similar. Improvement in the fitness of the solution must have been 0 for the TSPs to notice the stagnation, whereas the real-valued functions noticed the stagnation when the improvement was less than 10^{-5}. However, these conditions had to be fulfilled repeatedly for $10 \cdot \textit{population size}$ number of evaluations before the stagnation was acknowledged, both for the TSPs and the real-valued functions.

4.5.1 Experiments on Real-valued Functions with GA

All figures in this section show results of the experiments conducted on the GA model, where the size of the population is an important parameter. Tested on the real-valued functions.

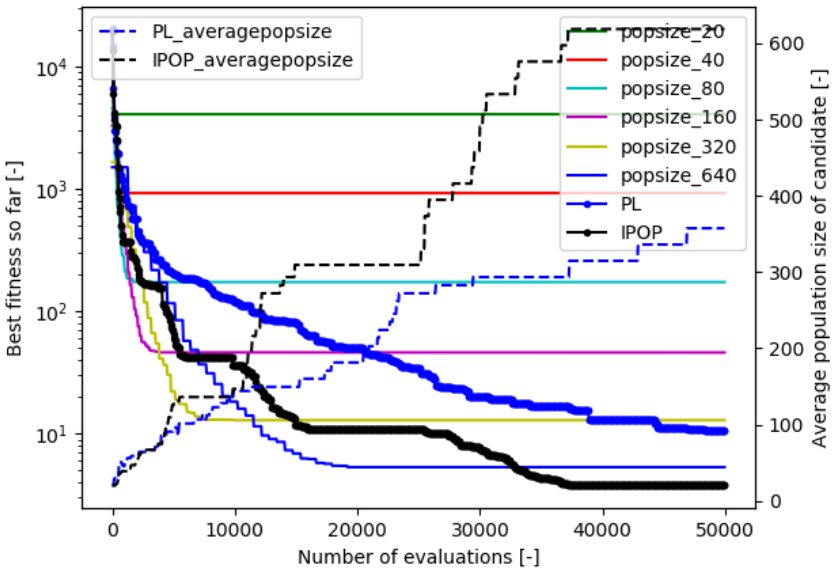

Figure 4.10: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Rosenbrock function of dimension 5. The Y axis is depicted in the logarithmic scale.

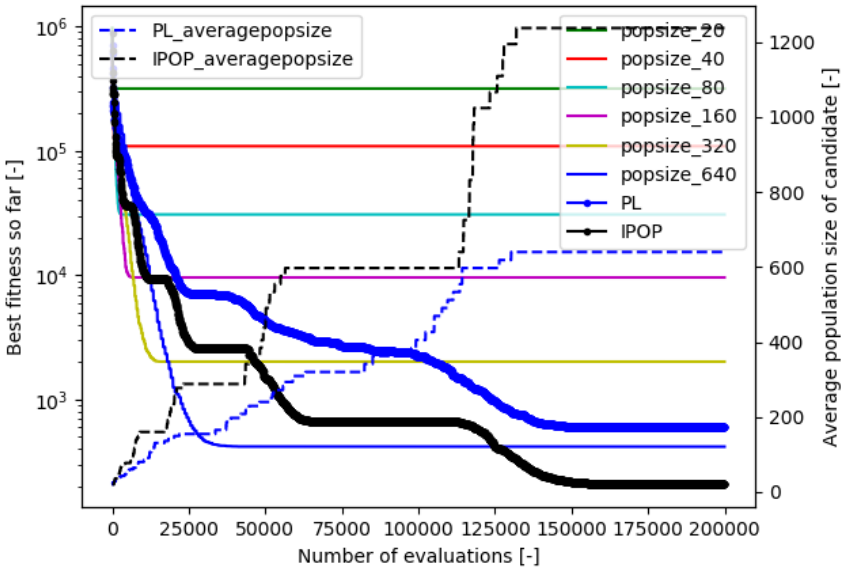

Figure 4.11: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Rosenbrock function of dimension 20. The Y axis is depicted in the logarithmic scale.

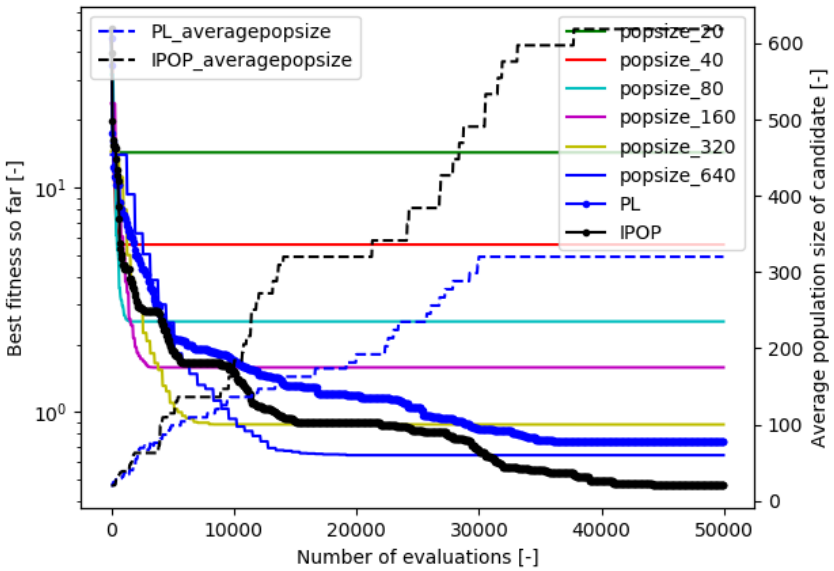

Figure 4.12: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Griewank function of dimension 5. The Y axis is depicted in the logarithmic scale.

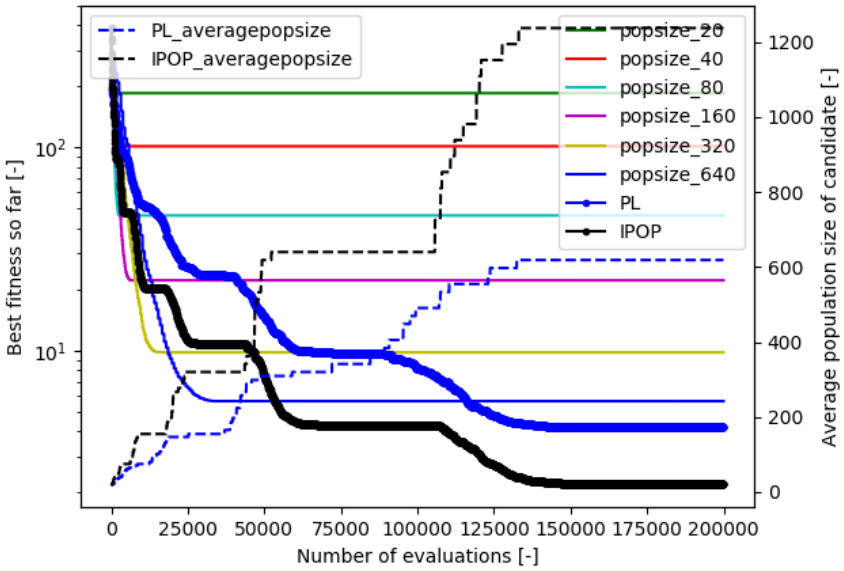

Figure 4.13: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Griewank function of dimension 20. The Y axis is depicted in the logarithmic scale.

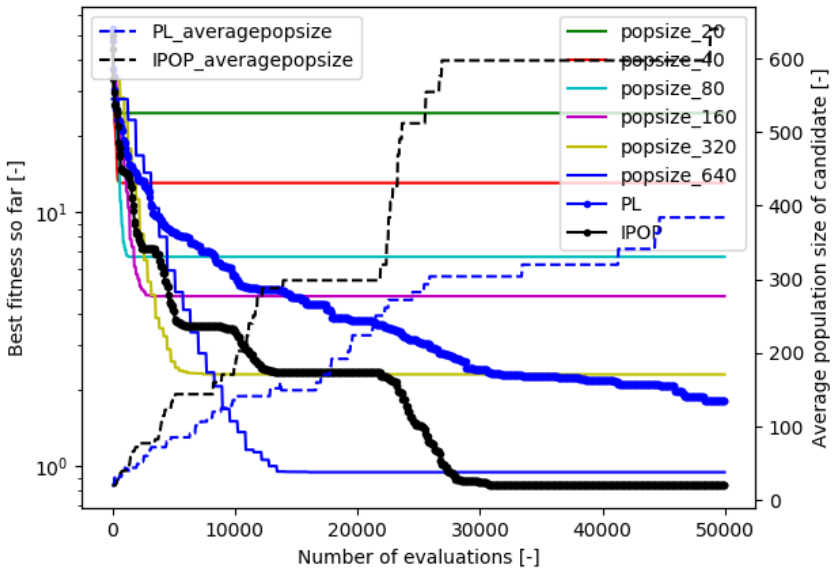

Figure 4.14: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Rastrigin function of dimension 5. The Y axis is depicted in the logarithmic scale.

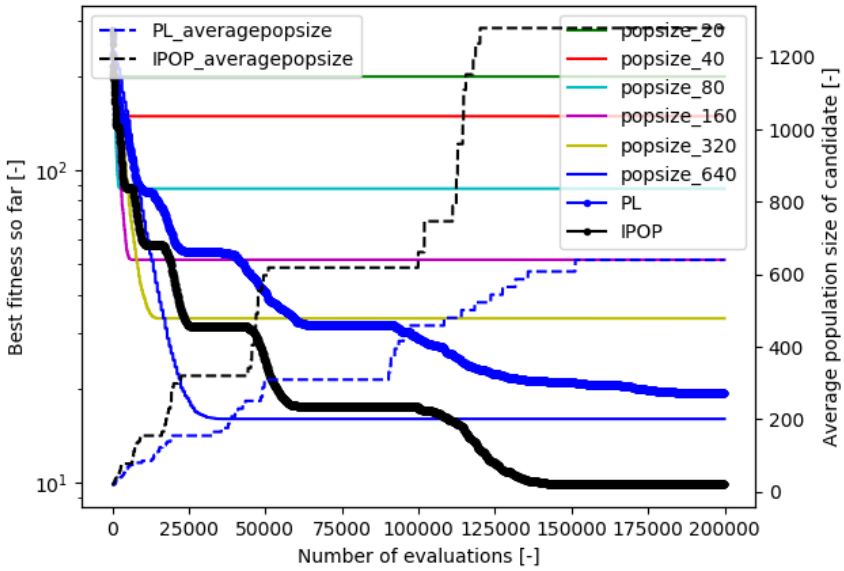

Figure 4.15: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Rastrigin function of dimension 20. The Y axis is depicted in the logarithmic scale.

The experiments (figures 4.10-4.15) testing the GA model for the parameter-

less EA and the IPOP algorithms behaved as expected as it was described in section 4.4.1. It has been mentioned that the population size is an important parameter for the GA model. Therefore, these two algorithms - the parameter-less EA and the IPOP - capable of adaptation of the population size provided better results than the EAs with the constant population sizes. The algorithms with the small population sizes were far from the optimal solution. The larger populations approached the optimal solution, but they also stopped improving after some time. On the contrary, the parameter-less EA and the IPOP algorithms were still improving the solution when the termination condition was met. In general, these two algorithms approached a better solution than the algorithms with a constant size of the population. In all these figures (4.10-4.15) there might be seen that the IPOP provided a better result than the parameter-less EA.

4.5.2 Experiments on Real-valued Functions with DEA

The behavior of the algorithms on the DEA model might be found in this section. Tested on the real-valued functions.

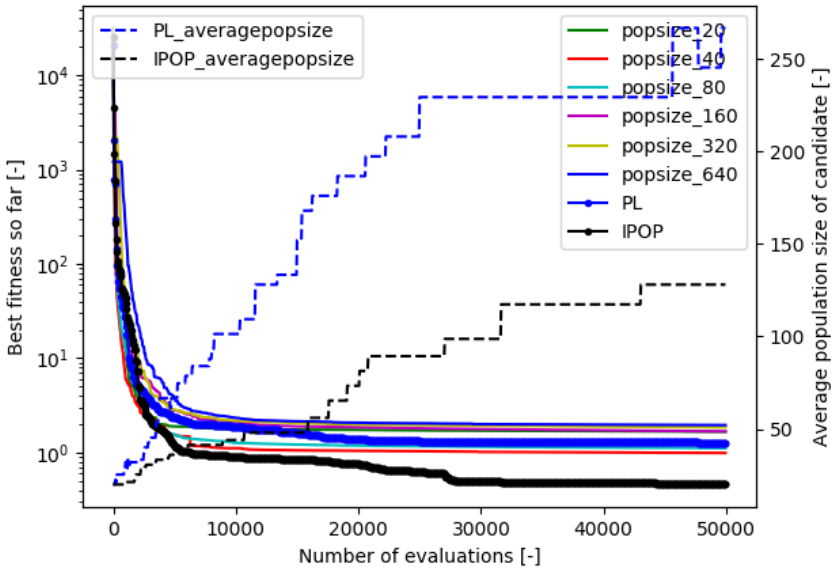

Figure 4.16: Comparison of the parameter-less AE and the IPOP algorithm on the background of the EAs with a constant population size on the Rosenbrock function of dimension 5. The Y axis is depicted in the logarithmic scale.

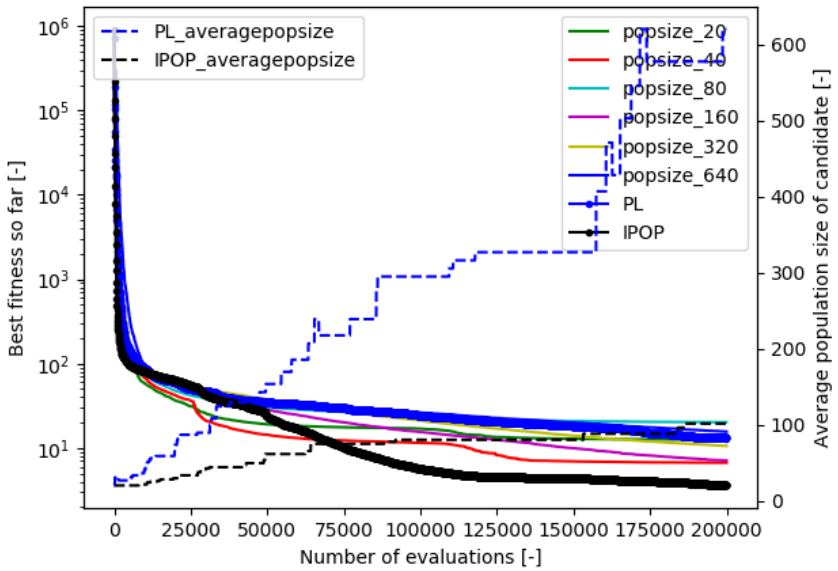

Figure 4.17: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Rosenbrock function of dimension 20. The Y axis is depicted in the logarithmic scale.

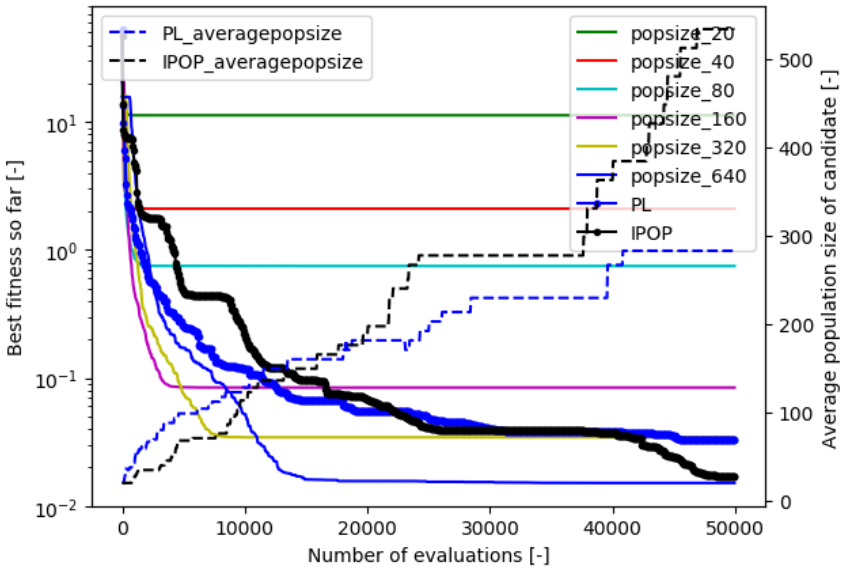

Figure 4.18: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Griewank function of dimension 5. The Y axis is depicted in the logarithmic scale.

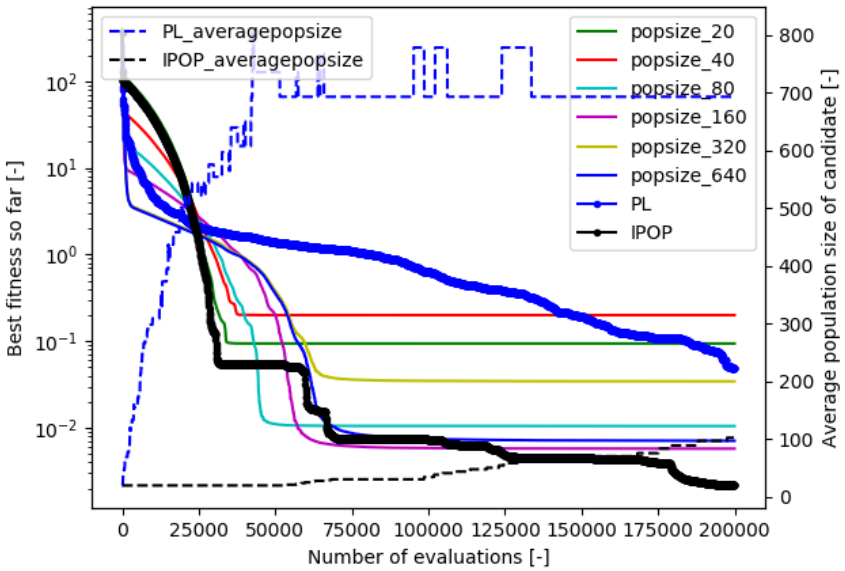

Figure 4.19: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Griewank function of dimension 20. The Y axis is depicted in the logarithmic scale.

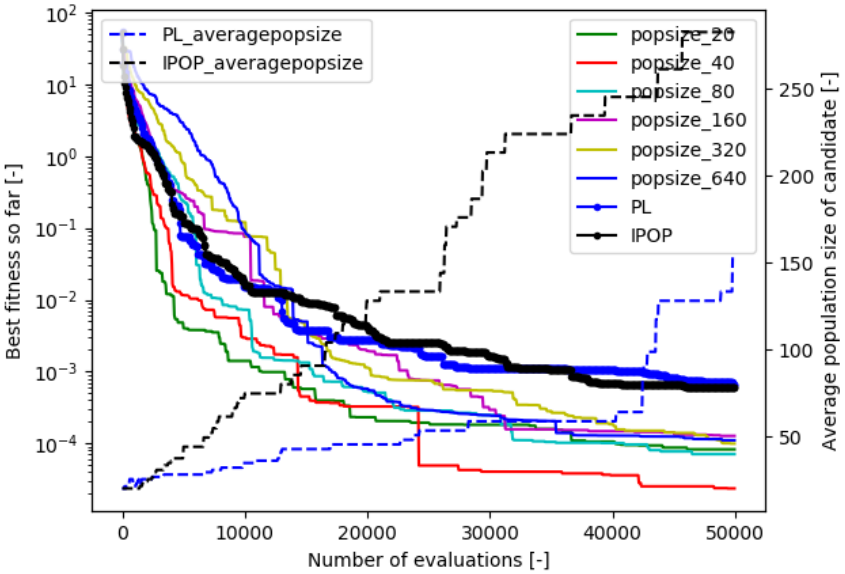

Figure 4.20: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Rastrigin function of dimension 5. The Y axis is depicted in the logarithmic scale.

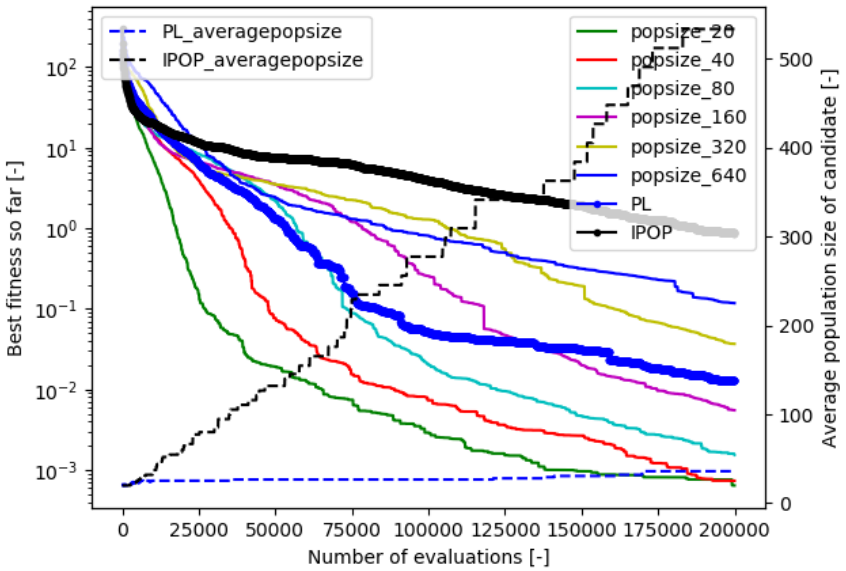

Figure 4.21: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the Rastrigin function of dimension 20. The Y axis is depicted in the logarithmic scale.

The parameter-less EA and the IPOP algorithm using the DEA model, where the size of the population is not important as mentioned in chapter 4.4.1, showed a similar behavior as the algorithm with various constant sizes of the population using the same model (figures 4.16-4.21). However, the parameter-less EA and the IPOP algorithm almost never find as good solution as the algorithm with a constant population size. It might be caused by the fact that both algorithms - the parameter-less EA and the IPOP - begin with the size of the population 20 and the development of the solution follows the DEA with a smaller population at the beginning. After several evaluations they find a better solution in a larger population and so on. However, in total the parameter-less EA and the IPOP algorithm lose more function evaluations on smaller populations, which do not find as good result as the bigger population. This development in the population size is what delays these two algorithm, as opposed to the DEA that is competent in finding the optimal solution itself with the population size chosen at the beginning. With the exception of the Rastrigin function of dimension 20 (figure 4.21), in all remaining experiments from this section (figures 4.16-4.20) the IPOP approached a better solution than the parameter-less EA.

4.5.3 Experiments on TSPs with GA

The figures in this section show results of the experiments on the TSPs with the use of the GA model.

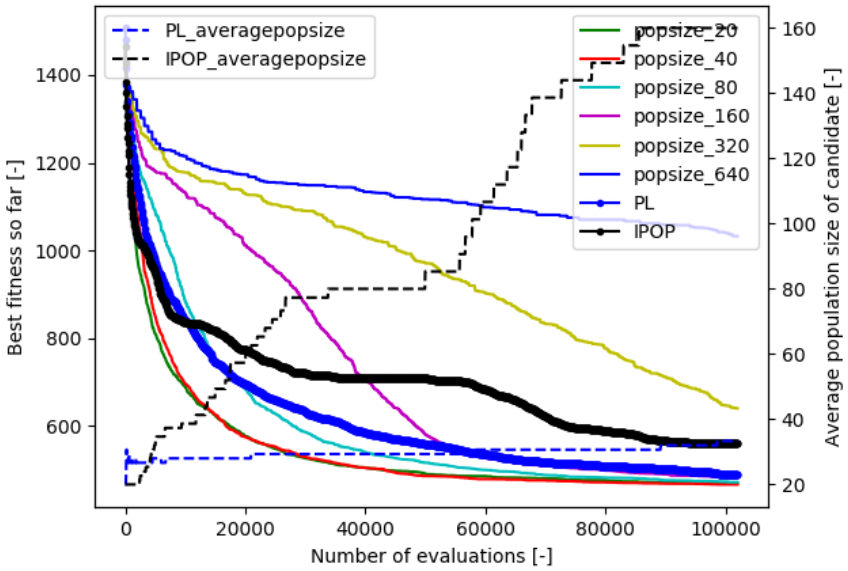

Figure 4.22: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 51 (filename 'eil51').

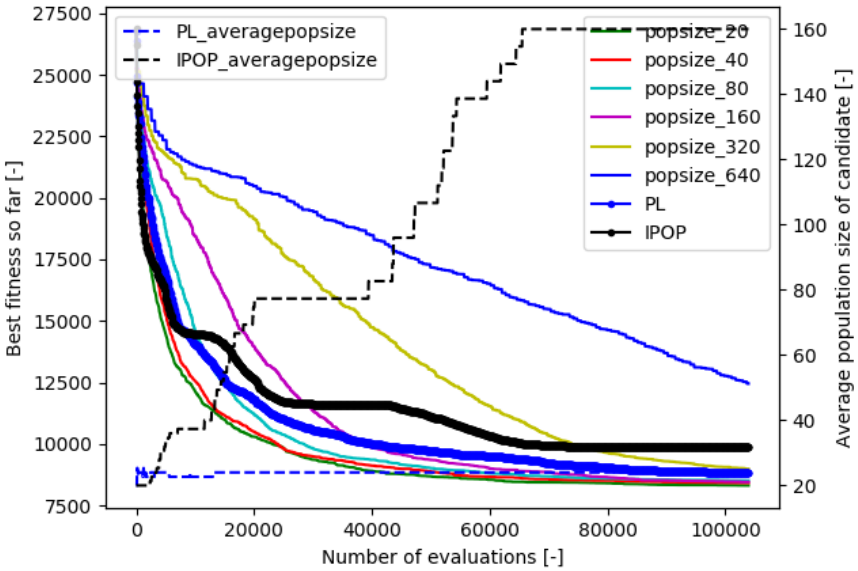

Figure 4.23: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 52 (filename 'berlin52').

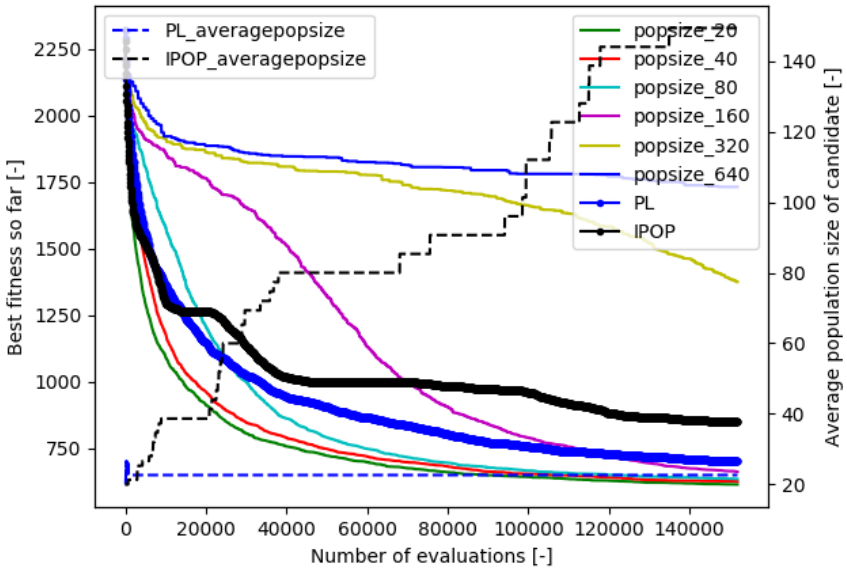

Figure 4.24: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 76 (filename 'eil76').

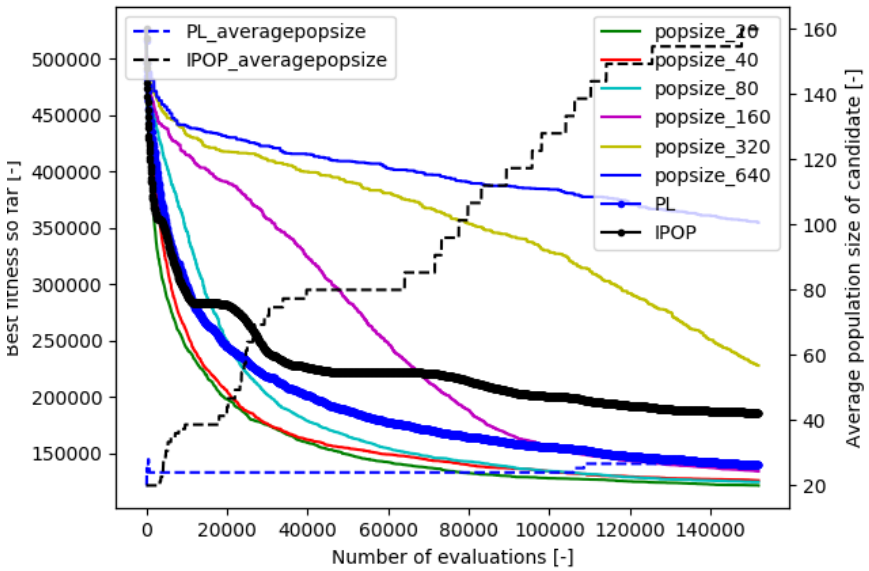

Figure 4.25: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 76 (filename 'pr76').

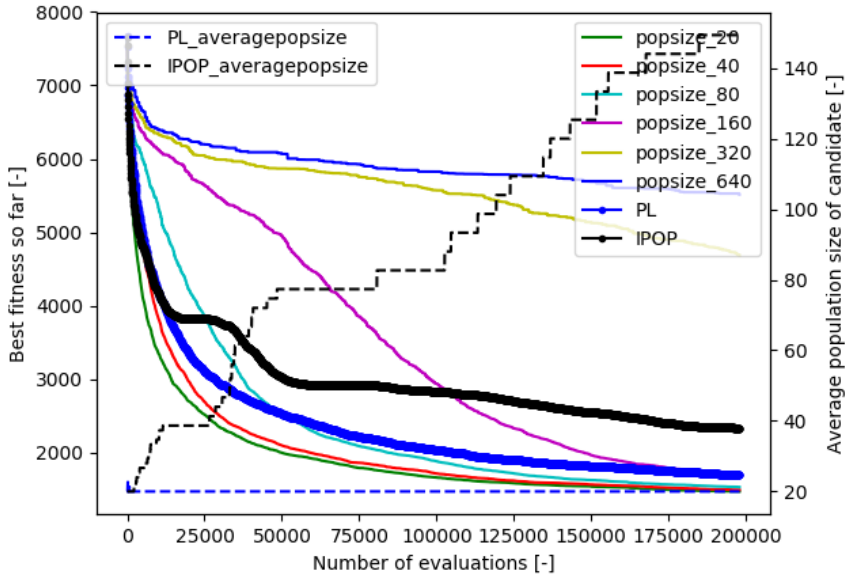

Figure 4.26: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 99 (filename 'rat99').

The experiments (figures 4.22-4.26) testing the TSPs on the GA model for

the parameter-less EA and the IPOP algorithms looked similarly as described in section 4.5.1 with the real-valued functions. These two algorithms - the parameter-less EA and the IPOP - provided better results than the EAs with the constant population sizes with the exception of the smaller population. The algorithms with the population sizes set to 20, 40 and 80 approached a better solution than the parameter-less EA and the IPOP. These sizes of the population were probably close to the ideal population size. After comparing the parameter-less EA and the IPOP it might be said that the parameter-less EA provided a better results than the IPOP in all these experiments (figures 4.22-4.26)

4.5.4 Experiments on TSPs with DEA

The behavior of the algorithms tested on the TSPs with the use of the DEA model can be found in this section.

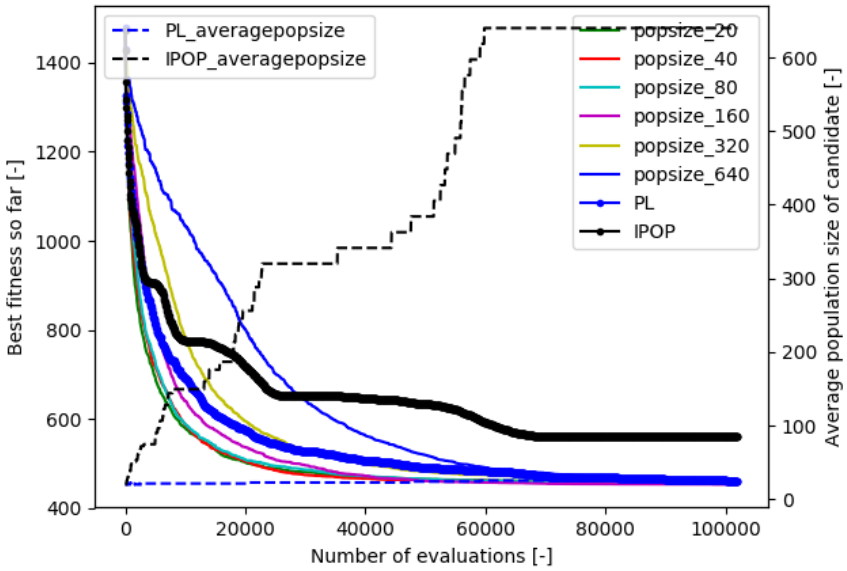

Figure 4.27: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 51 (filename 'eil51').

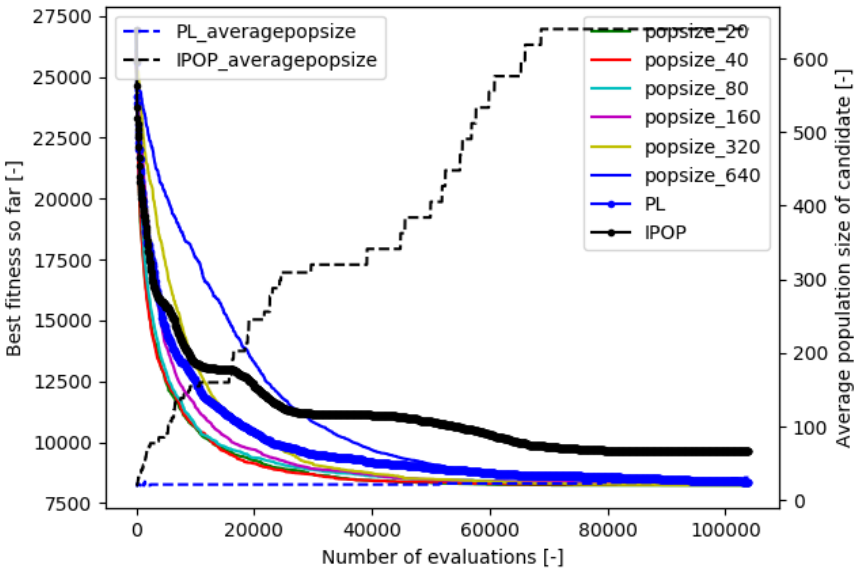

Figure 4.28: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 52 (filename 'berlin52').

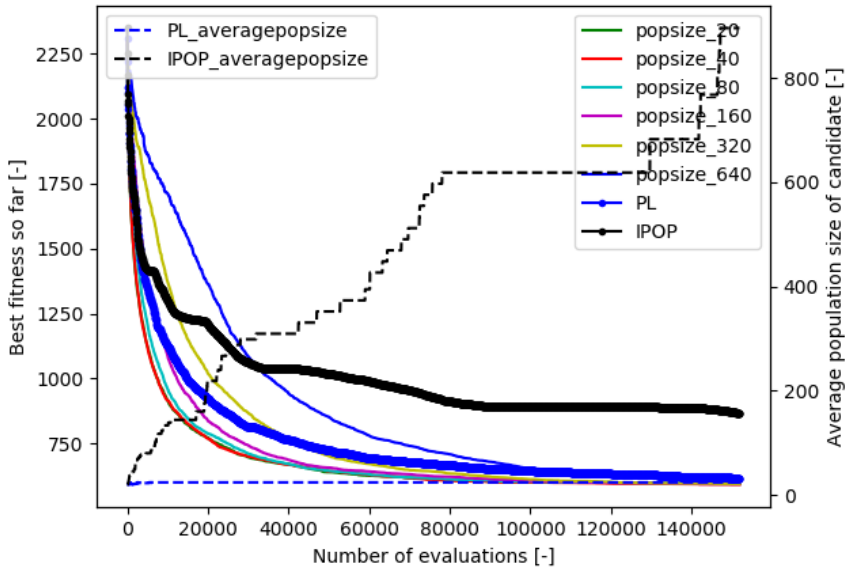

Figure 4.29: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 76 (filename 'eil76').

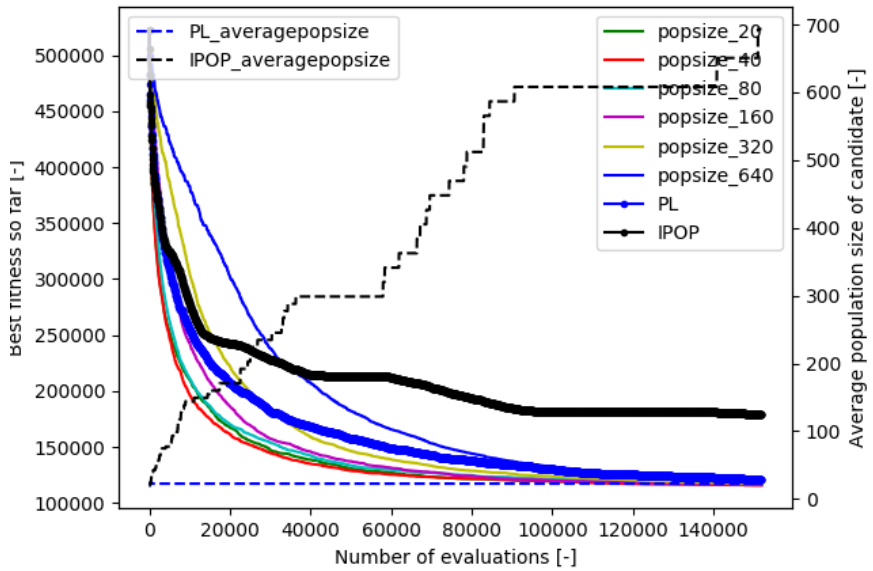

Figure 4.30: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 76 (filename 'pr76').

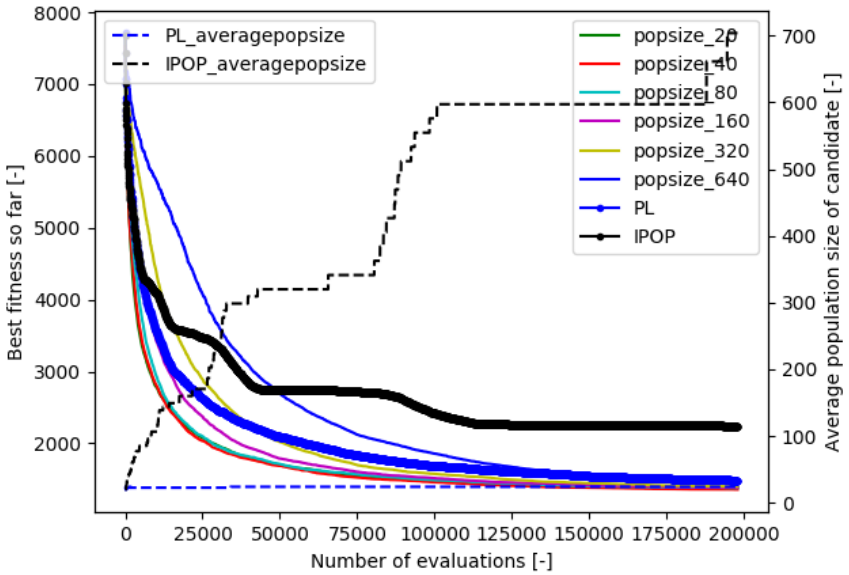

Figure 4.31: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP of dimension 99 (filename 'rat99').

The experiments in this section (figures 4.27-4.31) shows similar behavior as

the figures in section 4.5.2. The parameter-less EA and the IPOP algorithm using the DEA model, provided a worse results than the algorithms with a constant size of the population. The reason for that is the same as explained in section 4.5.2. However, in all these experiments (figures 4.27-4.31) the parameter-less EA approached a better solution than the IPOP.

4.5.5 Experiments on TSP Nearest Neighbor

The behavior of the algorithms tested on the TSPs with the nearest neighbor heuristics might be found here. In figures 4.32-4.34 the GA model is used and in figures 4.35-4.36 it is the DEA model.

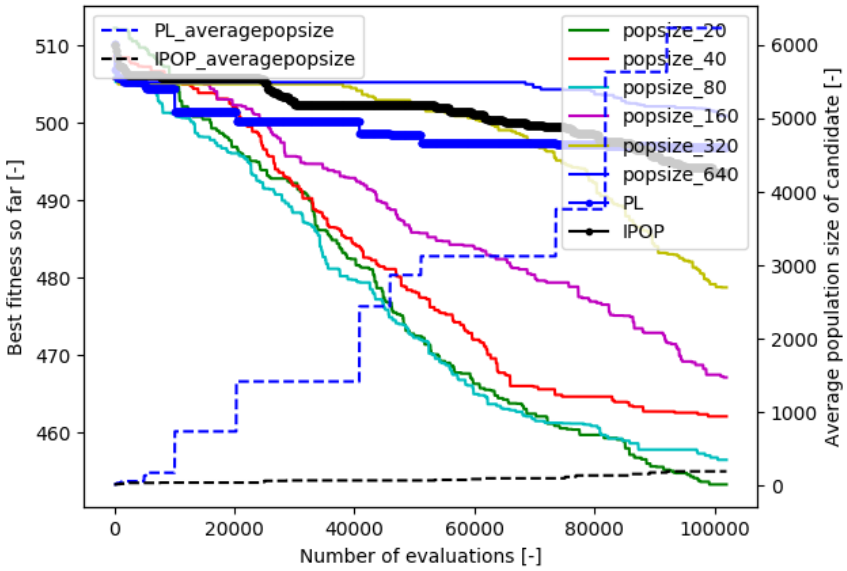

Figure 4.32: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP Nearest Neighbor of dimension 51 (filename 'eil51') with the use of the GA model.

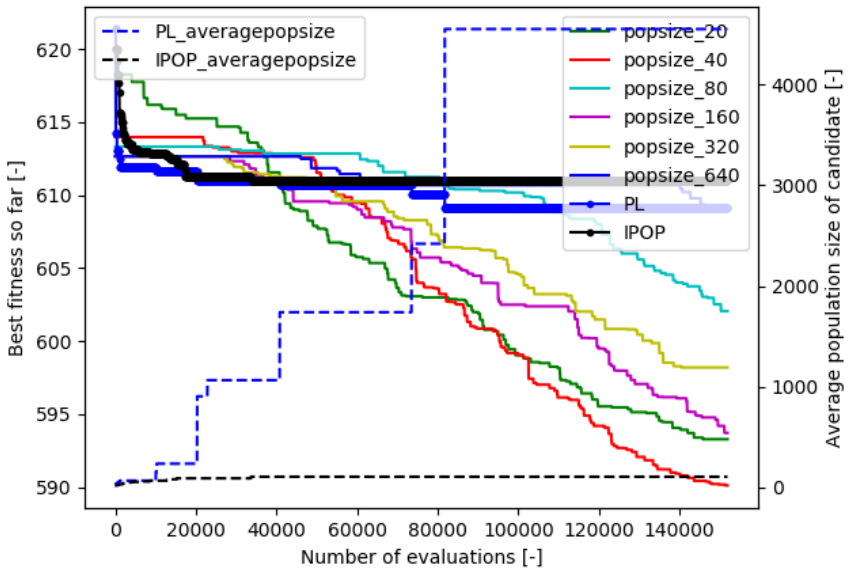

Figure 4.33: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP Nearest Neighbor of dimension 76 (filename 'eil76') with the use of the GA model.

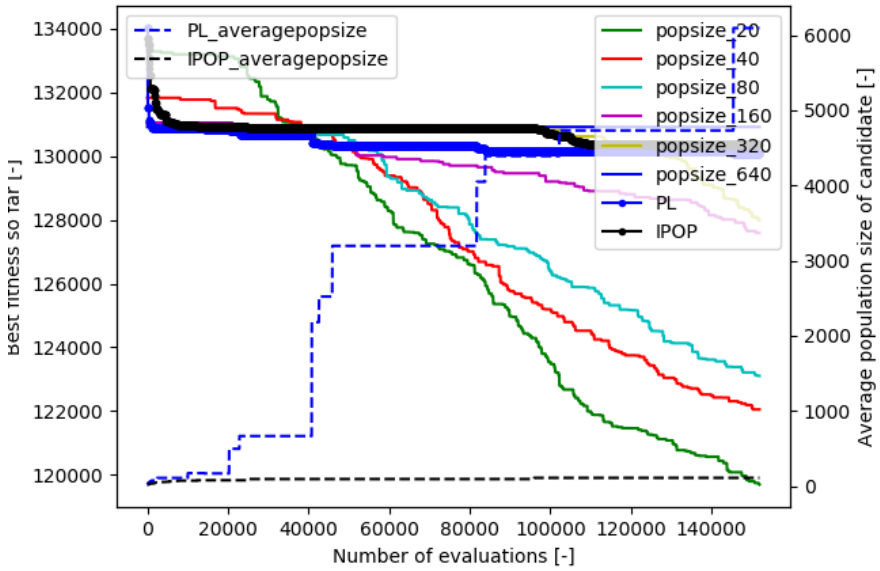

Figure 4.34: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP Nearest Neighbor of dimension 76 (filename 'pr76') with the use of the GA model.

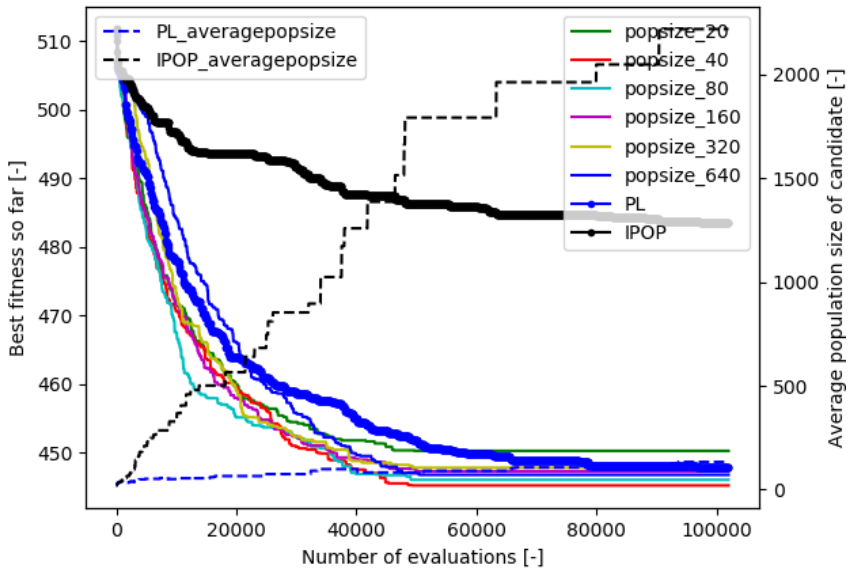

Figure 4.35: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP Nearest Neighbor of dimension 51 (filename 'eil51') with the use of the DEA model.

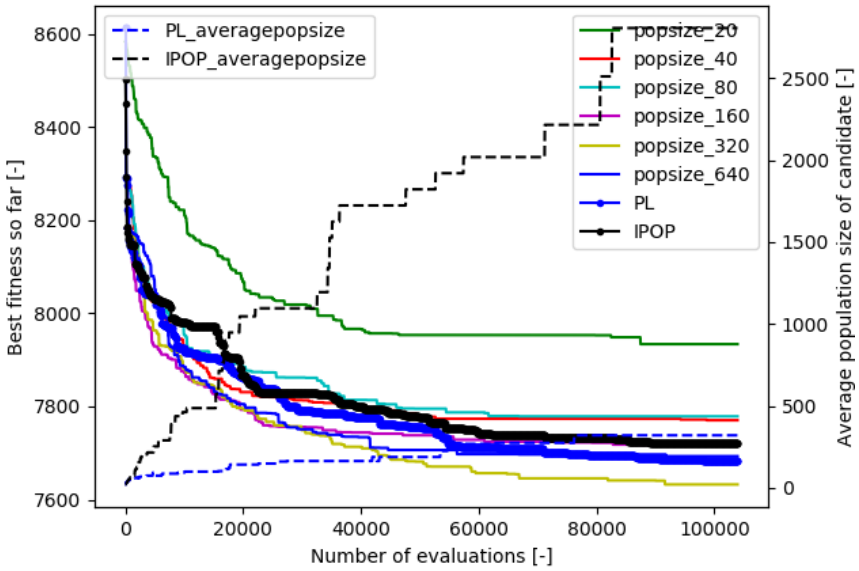

Figure 4.36: Comparison of the parameter-less EA and the IPOP algorithm on the background of the EAs with a constant population size on the TSP Nearest Neighbor of dimension 52 (filename 'berlin52') with the use of the DEA model.

The final section compares the behavior of the TSPs with the nearest

neighbor heuristics using the GA model (figures 4.32-4.34) and the DEA model (figures 4.35-4.36). Neither the parameter-less EA nor the IPOP performed well. They both ended with worse solutions than the EAs with a constant size of the population. This might be caused by the fact that the nearest neighbor heuristics is too greedy tactics, therefore both algorithms might have ended in a local optimum. It has been observed that the parameter-less EA did not behave as expected. When a new population was created, it had immediately a better average result than the smaller population thanks to the nearest neighbor heuristics. Therefore, the smaller population should have been erased. Consequently, only two populations were running simultaneously in the algorithm and large populations were created much earlier than in other experiments on the parameter-less EA. However, nothing similar has been observed with the IPOP, so its bad performance will stay unexplained.

4.6 Discussion

In order to describe the difference between the experiments where the GA model was used and the experiments where it was the DEA model, it must be mentioned that the DEA is a model where the size of the population is not a crucial parameter. The parameter-less EA and the IPOP algorithm using the DEA model showed a similar behavior as the algorithm with various constant sizes of the population using the same model (chapter 4.5.2 and 4.5.4).

Contrarily, the experiments testing the GA model for the parameter-less EA and the IPOP algorithms behaved as expected (chapter 4.5.1 and 4.5.3) from the chapter 4.4.1. It has been mentioned there that size of the population matters for the GA model. Therefore, these two algorithms - the parameter-less EA and the IPOP - approached a better solution than the algorithms with a constant size of the population.

The comparison of the parameter-less EA and the IPOP algorithm is a more demanding task. The tables 4.1 and 4.2 simplify the comparison of these algorithms. Both tables show the number of experiments where one algorithm provided a better result than the other one - meaning the parameter-less EA and the IPOP. In table 4.1 there are results of the experiments on GA model and in table 4.2 the results are from the use of the DEA. The results in the tables show that for real problems it is better to use the IPOP and the parameter-less EA provides a better results than the IPOP for TSPs.

	Real-valued functions	TSP
Parameter-less EA	0	9
IPOP	6	1

Table 4.1: Table shows the results of the paramete-less EA and the IPOP on the GA.

	Real-valued functions	TSP
Parameter-less EA	1	8
IPOP	5	2

Table 4.2: Table shows the results of the paramete-less EA and the IPOP on the DEA.

Chapter 5

Conclusion

To conclude the results of experiments conducted, it must be said that both algorithms - the parameter-less EA and the IPOP - behaved differently when solving various problems. Therefore, no complete and unitary conclusion can be drawn from their comparison. What the experiments with the use of the GA showed is that both algorithms - the parameter-less EA and the IPOP - gradually evolved and improved with the growth of the population size and their final results were better than the results of the GA with a constant size of the population. On the other hand, the experiments with the use of the DEA brought similar results both for the parameter-less EA and the IPOP as well as for the DEA with a constant size of the population. As emerged from the experiment results, when comparing only the parameter-less EA and the IPOP algorithm, it might be noticed that when testing the TSPs it was the parameter-less EA that approached a better solution than the IPOP algorithm, whereas IPOP provided better results for the real-valued functions.

Another important thing to be mentioned about the comparison of these two algorithms - the parameter-less EA and the IPOP - is that their implementation had a different level of difficulty. The correct setting of the IPOP restart conditions required a lot of experiments, as has been suggested in the chapters 4.4.3 and 4.5. Therefore, to find a set of its conditions that would provide the best results was a difficult and time-consuming task. On the contrary, the parameter-less EA is truly a parameter-less method whose using does not require a user to have any knowledge about the EAs.

5.1 Summary

This thesis presents the evolutionary algorithms with their advantages but also disadvantages, as the need for setting the population size parameter and the

mistakes related to the setting of this parameter. It describes the principles of behavior of the EAs along with the variants of the EAs. The thesis focuses mainly on the methods that avoid the setting of the population size and describes the algorithms that use these methods with a focus on the parameter-less EA and the IPOP algorithm that were implemented and chosen for the experiments. The first experiments concerned the setting of the parameters of these two algorithms. It was noted that the parameter-less EA has only one parameter which is the base of the counter. The experiments testing various base sizes were conducted, however the base that was proposed in the article [HL99] was proved to provide the best results. The IPOP required multiple parameters to be set, all of them had an influence on the behavior of this algorithm. Another parameter that both of these algorithms - the parameter-less EA and the IPOP - have in common is the initial population size. The experiments in the section 4.3.2 showed that the behavior of neither algorithms was affected by the change of this parameter. Another observation from the experiments is that the paramter-less EA as well as the IPOP behaved similarly to the algorithm with a constant population sizes when using the DEA model but they performed better than the algorithm with a constant population sizes with use of the GA. The last but not least important thing to be said is that the parameter-less EA provided better results than the IPOP on the TSPs. However, the IPOP algorithm approached the better solution than the parameter-less EA on real-valued problems.

Appendix A

Bibliography

[AH05] Anne Auger and Nikolaus Hansen, *A restart cma evolution strategy with increasing population size*, vol. 2, 01 2005, pp. 1769–1776.

[DC03] C. Darwin and J. Carroll, *On the origin of species*, Broadview Press, 2003.

[DJ06] Kenneth De Jong, *Evolutionary computation – a unified approach*, 01 2006.

[ES03] A. E. Eiben and J. Smith, *Introduction to evolutionary computing*, vol. 45, 01 2003.

[FOW66] L.J. Fogel, A.J. Owens, and M.J. Walsh, *Artificial intelligence through simulated evolution*, Wiley, 1966.

[Gar15] Aaron Garrett, *Examples - inspyred 1.0 documentation*, `https://pythonhosted.org/inspyred/examples.html#standard-algorithms`, 2015, [Online; accessed May-2019].

[GH88] David E. Goldberg and John H. Holland, *Genetic algorithms and machine learning*, Machine Learning **3** (1988), no. 2, 95–99.

[Han09] Nikolaus Hansen, *Benchmarking a bi-population cma-es on the bbob-2009 noisy testbed*.

[HAR+10] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík, *Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009*, Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation (New York, NY, USA), GECCO '10, ACM, 2010, pp. 1689–1696.

[HL99] Georges Harik and Fernando Lobo, *A parameter-less genetic algorithm*, 01 1999, p. 258–265.

[LG04] Fernando G. Lobo and David E. Goldberg, *The parameter-less genetic algorithm in practice*, Inf. Sci. **167** (2004), 217–232.

[Rei95] Gerhard Reinelt, *Tsplib95*, `https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95`, 1995, [Online; accessed May-2019].

[RK92] J R Koza, *Genetic programming: On the programming of computers by means of natural selection (complex adaptive systems)*, 01 1992.

[Sch75] Hans-Paul Schwefel, *Evolutionsstrategie und numerische optimierung*, Ph.D. thesis, 01 1975.

[SP95] Rainer Storn and Kenneth Price, *Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces*, Journal of Global Optimization **23** (1995).

[SS93] Robert Smith and Ellen Smuda, *Adaptively resizing populations: Algorithm, analysis, and first results.*

[WGP14] Brian W. Goldman and William Punch, *Parameter-less population pyramid*, GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference (2014).

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Šterberová Adéla** Personal ID number: **459187**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Population Size Adaptation in Evolutionary Algorithms

Bachelor's thesis title in Czech:

Adaptace velikosti populace v evolučních algoritmech

Guidelines:

Population size is one of the most important parameters of evolutionary algorithms, with a big influence on the algorithm performance. The literature contains several approaches how to adapt the population size during the algorithm run (Parameter-less GA, P3, IPOP, BIPOP, etc.). The goal of this project is to compare these methods for a chosen type of an evolutionary algorithm and for selected classes of problems, and possibly to propose a novel population size adaptation method and include it into the comparison.
1. Study the principles of the existing methods for population size adaptation (Parameter-less GA, P3, …).
2. Study the principles of methods for restarting an algorithm with a different population size (IPOP, BIPOP).
3. Choose at least one (ideally two) types of population optimization algorithms and test the adaptation mechanisms on selected problems.
4. (Optionally) propose your own adaptation method and include it into the comparison.
5. Evaluate the contributions of the adaptation mechanisms, compare them to each other and to the usage of a constant population size.

Bibliography / sources:

[1] G.R. Harik, F.G. Lobo, A parameter-less genetic algorithm, in: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, R.E. Smith (Eds.), GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, San Francisco, CA, 1999, pp. 258–267.
[2] F.G. Lobo, D.E. Goldberg, The parameter-less genetic algorithm in practice. Information Sciences, 167 (2004), 217—232.
[3] B.W. Goldman, W.F. Punch, Parameter-less Population Pyramid. In GECCO 2014: Proceedings of the Genetic and Evolutionary Computation Conference, 2014, pp. 758-792.
[4] Loshchilov I., Schoenauer M., Sebag M. (2012) Alternative Restart Strategies for CMA-ES. In: Coello C.A.C., Cutello V., Deb K., Forrest S., Nicosia G., Pavone M. (eds) Parallel Problem Solving from Nature - PPSN XII. PPSN 2012. Lecture Notes in Computer Science, vol 7491. Springer, Berlin, Heidelberg.

Name and workplace of bachelor's thesis supervisor:

Ing. Petr Pošík, Ph.D., Analysis and Interpretation of Biomedical Data, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **24.01.2019** Deadline for bachelor thesis submission: **24.05.2019**

Assignment valid until: **20.09.2020**

Ing. Petr Pošík, Ph.D.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. _____
Date of assignment receipt

_____
Student's signature