City Research Online

City St George's, University of London

Citation: Ter-Sarkisov, A. (2012). Computational complexity of elitist population-based evolutionary algorithms: a thesis presented in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science at Massey University, Palmerston North, New Zealand. (Unpublished Doctoral thesis, Massey University, Palmerston North, New Zealand.)

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/21445/

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to City Research Online policy.

City Research Online: http://openaccess.city.ac.uk/ publications@citystgeorges.ac.uk

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

COMPUTATIONAL COMPLEXITY OF ELITIST POPULATION-BASED EVOLUTIONARY ALGORITHMS

A thesis presented in partial fulfilment of the requirements for the degree of
Doctor of Philosophy
in
Computer Science
at Massey University, Palmerston North,
New Zealand.

Aram Ter-Sarkisov

2012

Contents

Acknowledgements **xi**

Abstract **xii**

List of Publications **xiv**

Notation **xvi**

1 Introduction **1**

1.1 Introduction to Evolutionary Computation 1

1.2 Motivation 3

1.3 Main results 3

1.4 Outline of the thesis 5

2 Related Work **7**

2.1 Schemata Theorem 7

2.1.1 Criticism of Schemata Theorem 9

2.1.2 Alternative explanation of EA efficiency 9

2.2 Convergence Analysis 10

2.3 Runtime Analysis 12

2.3.1 Runtime analysis of $(\mu + 1)$ and $(1 + \lambda)$ EAs 14

2.3.2 Runtime analysis of $(\mu + \lambda)$ EAs 15

2.4 Review of tools used for analyzing EAs 16

2.4.1 Fitness-Based Partition and Artificial Fitness Levels 17

2.4.2 Gambler's Ruin and Coupon Collector's Problem 17

2.4.3 Potential/Auxiliary Functions 19

2.4.4 Analysis of Typical runs 20
2.4.5 Structure of individuals in the population 20
2.5 Asymptotic notation 21
2.6 The No Free Lunch theorem and Analysis of computer algorithms . 22
2.7 Parallel computers 23

3 The K-Bit-Swap Genetic Operator 24
3.1 Explanation of the K-Bit-Swap Genetic Operator 24
3.2 Algorithms and Experimental setup 26
3.2.1 Problems selected for testing 27
3.3 Setup and Analysis of Statistical Tests 33
3.3.1 Statistical Analysis 37
3.4 Conclusions 38

4 Lower Bounds on the Runtime 40
4.1 Main results 40
4.2 Structure of the population and the recombination pool 41
4.3 Algorithms 43
4.4 Problems 44
4.4.1 OneMax 45
4.4.2 Royal Roads 45
4.5 Population-Based Evolutionary Algorithms and Distribution of Species 47
4.6 Runtime analysis of $(1+2)EA_{1BS}$ solving OneMax Problem 49
4.7 Main model of the $(\mu+\lambda)$ Algorithm on the OneMax Test Function 51
4.7.1 Runtime analysis of $(\mu+\lambda)EA_{1BS}$ on the OneMax problem 52
4.7.2 Asymptotic runtime of $(\mu+\lambda)\text{EA}_{1BS}$ on the OneMax Test function 56
4.7.3 Runtime analysis of $(\mu+\lambda)$RLS on the OneMax Test Function 57
4.7.4 Asymptotic runtime of $(\mu+\lambda)$RLS on the OneMax Test function 58
4.8 Main model of the $(\mu+\lambda)$ Algorithm on the Royal Roads Test Function 59
4.8.1 Runtime analysis of $(\mu+\lambda)\text{EA}_{1BS}$ on the RR Test Function 60

4.8.2 Asymptotic runtime of $(\mu+\lambda)$ EA_{1BS} on the RR Test Function 62
4.8.3 Runtime analysis of $(\mu+\lambda)$RLS on the RR Test Function . 64
4.8.4 Asymptotic runtime of $(\mu+\lambda)$RLS on the RR Test Function 65
4.9 Numerical results 66
4.10 Conclusions 84

5 Upper Bounds on the Runtime 86
5.1 Main results 87
5.2 The Elitism Levels Traverse Mechanism 88
5.3 Upper bounds on the OneMax test function 91
5.3.1 Simple upper bound on OneMax 92
5.3.2 Refined upper bounds on OneMax 96
5.3.3 Use of $<\alpha,\alpha>$ pair 101
5.3.4 Generations vs Function evaluations 101
5.3.5 Comparison to earlier results 102
5.4 Upper Bounds on the Royal Roads test function 102
5.4.1 The birth-and-death Markov Chain for Royal Roads 104
5.4.2 Upper bounds on the Royal Roads problem 106
5.4.3 Proof of the lower bound on the probability of advancing to the next artificial auxiliary level 116
5.4.4 Lower bounds on the probabilities involving η species in Phase 2 119
5.5 Approximation of the quasi-stationary distribution of super-elite species in Phase 1 121
5.5.1 Slow progress rate (Poisson approximation) 123
5.5.2 Fast progress rate (Normal approximation) 123
5.6 Conclusions 125

6 Summary, Conclusions and Future Work 128

A Results of Numerical Experiments 133

B Concepts from Probability Theory 147

List of Tables

1.1 Canonical Genetic Algorithm 2

3.1 The K-Bit-Swap Genetic Operator 25
3.2 Pseudocode of EAs in Chapter 3 27
3.3 Parameter settings for the problem set. 32
3.4 Benchmark settings 32
3.5 Estimate of the probability of failure, Equation 3.3 36
3.6 Estimate of the conditional probability of success, Equation 3.4 . . 36
3.7 Estimate of the conditional expectation, Equation 3.5 37

4.1 $(\mu + \lambda)\text{EA}_{1BS}$ 44
4.2 $(\mu + \lambda)$RLS 44
4.3 Selection Function 45
4.4 Set of parameters used for OneMax test function 66
4.5 Set of parameters used for the RR test function 67

List of Figures

3.1 Comparison of K-Bit-Swap to simple (segment) crossover, 2-point simple (segment) crossover and Uniform crossover 25
3.2 The global solution for the trivial k-means problem. Darker points are data, lighter are centroids 33

4.1 Distribution of the elite species in the population of $(\mu + \lambda)$ EA_{1BS} solving OneMax Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum 67
4.2 Distribution of the elite species in the population of $(\mu + \lambda)$RLS solving OneMax Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum 68
4.3 Probability of success of $(\mu + \lambda)$RLS solving OneMax Test Function. 69
4.4 Numerical runtime estimate for $(\mu+\lambda)\mathrm{EA}_{1BS}$ solving OneMax Test Function for different population sizes. 70
4.5 Numerical runtime estimate for $(\mu + \lambda)$RLS solving OneMax Test Function for different population sizes. 71
4.6 Theoretical and numerical estimate for $(\mu + \lambda)EA_{1BS}$ solving OneMax Test Function 72
4.7 Theoretical and numerical estimate for $(\mu+\lambda)RLS$ solving OneMax Test Function 73
4.8 Distribution of the elite species in the population of $(\mu + \lambda)$ EA_{1BS} solving Royal Roads Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum 74

4.9 Distribution of the elite species in the population of $(\mu + \lambda)$RLS solving Royal Roads Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum 75
4.10 Probability of success of $(\mu + \lambda)$ EA_{1BS} solving Royal Roads Test Function. For $n = 32, 64$ it is always almost 1 76
4.11 Probability of success of $(\mu+\lambda)$RLS solving Royal Roads Test Function.For $n = 32, 64$ it is always almost 1 77
4.12 Numerical runtime estimate for $(\mu + \lambda)\mathrm{EA}_{1BS}$ solving Royal Roads Test Function for different population sizes. The positive effect of the population size measured in the number of generations is obvious. 78
4.13 Numerical runtime estimate for $(\mu + \lambda)$RLS solving Royal Roads Test Function for different population sizes. The positive effect of the population size measured in the number of generations is obvious. 79
4.14 Theoretical and numerical bounds for $(\mu + \lambda)EA_{1BS}$ solving Royal Roads Test Function 80
4.15 Theoretical and numerical bounds for $(\mu + \lambda)$RLS solving Royal Roads Test Function 81

A.1 Conditional probability of success and runtime of $(\mu+\lambda)\mathrm{EA}_{KBS}$ vs $(\mu+\lambda)\mathrm{EA}_{-KBS}$ on the Rosenbrock test function 133
A.2 Conditional probability of success and runtime of $(\mu+\lambda)\mathrm{EA}_{KBS}$ vs $(\mu+\lambda)\mathrm{EA}_{-KBS}$ on the Rastrigin test function 134
A.3 Conditional probability of success and runtime of $(\mu+\lambda)\mathrm{EA}_{KBS}$ vs $(\mu+\lambda)\mathrm{EA}_{-KBS}$ on the Ackley test function 134
A.4 Conditional probability of success and runtime $(\mu + \lambda)\mathrm{EA}_{KBS}$ on the Royal Roads test function. Algorithms with other parameter settings do not solve the problem in the set number of generations. 135
A.5 Conditional probability of success and runtime of $(\mu+\lambda)\mathrm{EA}_{KBS}$ vs $(\mu+\lambda)\mathrm{EA}_{-KBS}$ on the Four Peaks test function 135
A.6 Conditional probability of success and runtime of $(\mu+\lambda)\mathrm{EA}_{KBS}$ vs $(\mu+\lambda)\mathrm{EA}_{-KBS}$ on the trivial TSP 136

A.7 Conditional probability of success and runtime of $(\mu + \lambda)\text{EA}_{-KBS}$ on the TSP on US Capital Cities. Algorithms with KBS do not solve the problem in the set number of generations 136
A.8 Probability of success $(\mu+\lambda)\text{EA}_{KBS}$ vs $(\mu+\lambda)\text{EA}_{-KBS}$ on the trivial k-means clustering problem 137
A.9 Probability of success $(\mu + \lambda)\text{EA}_{KBS}$ vs $(\mu + \lambda)\text{EA}_{-KBS}$ on the random k-means clustering problem 137
A.10 Histograms of bootstrap estimate of the difference in means for the Rosenbrock function: probability of failure, conditional probability of success, runtime 138
A.11 Histograms of bootstrap estimate of the difference in means for the Rastrigin function: probability of failure, conditional probability of success, runtime 139
A.12 Histograms of bootstrap estimate of the difference in means for the Ackley function: probability of failure, conditional probability of success, runtime 140
A.13 Histograms of bootstrap estimate of the difference in means for the Royal Roads function: probability of failure, and conditional probability of success 141
A.14 Histograms of bootstrap estimate of the difference in means for the Four Peaks function: probability of failure, conditional probability of success, runtime 142
A.15 Histograms of bootstrap estimate of the difference in means for the TSP on a circle: probability of failure, conditional probability of success, runtime 143
A.16 Histograms of bootstrap estimate of the difference in means for the TSP on US Cities: probability of failure and conditional probability of success 144
A.17 Histograms of bootstrap estimate of the difference in means for the trivial k-means clustering problem: probability of failure, conditional probability of success, runtime 145

A.18 Histograms of bootstrap estimate of the difference in means for the random k-means clustering problem: probability of failure, conditional probability of success, runtime 146

Für meine Großeltern, Bertha und Edward

Acknowledgements

I would like to thank first of all my supervisors, Associate Professor Stephen Marsland, Professor Chin-Diew Lai and Doctor Barbara Holland. Without their guidance and support this thesis would never be possible. I would like to thank everyone who was helping me in many ways throughout more than three past years, especially my parents, my girlfriend, my girlfriend's parents and grandparents.

Abstract

Evolutionary Algorithms (EAs) are a modern heuristic algorithm that have proven efficiency on a large number of real-life problems. Despite the rich history of applications understanding of both how and why EAs work is lagging far behind. This is especially true for one of the main components of EAs, that is hypothesized by many to underlie their efficiency: population.

The first problem considered in this thesis is the introduction of a recombination operator, K-Bit-Swap (KBS) and its comparison to mainstream operators, such as mutation and different types of crossover. A vast amount of statistical evidence is presented that shows that EAs using KBS outperform other algorithms on a whole range of problems. Two problems are selected for a deep theoretical analysis: OneMax and Royal Roads.

The main problem of modeling EAs that use both population and a pool of parents is the complexity of the structures that arise from the process of evolution. In most cases either one type of species is considered or certain simple assumptions are made about fitness of the species.

The main contribution of this thesis is the development of a new approach to modeling of EAs that is based on approximating the structure of the population and the evolution of subsets thereof. This approach lies at the core of the new tool presented here, the Elitism Levels Traverse Mechanism that was used to derive upper bounds on the runtime of EAs. In addition, lower bounds were found using simpler assumptions of the underlying distribution of species in the population.

The second important result of the approach is the derivation of limiting distributions of a subset of the population, a problem well-known in areas such as epidemiology. To the best of the author's knowledge, no such findings have been published in the EA community so far.

List of Publications

A. Ter-Sarkisov, S. Marsland, and B. Holland. The k-Bit-Swap: A New Genetic Algorithm Operator. In *Genetic and Evolutionary Computing Conference (GECCO) 2010*, pages 815–816, 2010

A. Ter-Sarkisov and S. Marsland. Convergence Properties of $(\mu + \lambda)$ Evolutionary Algorithms. In *25th AAAI Conference on Artificial Intelligence*, pages 1816–1817, 2011. Special Student Poster Session

A. Ter-Sarkisov and S. Marsland. Convergence Properties of Two $(\mu + \lambda)$ Evolutionary Algorithms on OneMax and Royal Roads Test Functions. In *International Conference on Evolutionary Computation Theorey and Applications (ECTA)*, pages 196–202, 2011

A. Ter-Sarkisov and S. Marsland. Convergence of a Recombination-Based Elitist Evolutionary Algorithm on the Royal Roads Test Function. In *24th Australasian Joint Conference on Artificial Intelligence*, pages 361–371, 2011

A. Ter-Sarkisov. Elitism Levels Traverse Mechanism For The Derivation of Upper Bounds on Unimodal Functions. In *WCCI 2012 IEEE World Congress on Computational Intelligence*, pages 2161–2168, 2012

A. Ter-Sarkisov and S. Marsland. Derivation of Upper Bounds on Optimization Time of Population-Based Evolutionary Algorithm on a Function with Fitness Plateaus Using Elitism Levels Traverse Mechanism. 2012. arXiv:1204.2321. To be submitted

Notation

The notation for species, $\alpha^* \dots \eta$ is used to denote both the type and the size of the type, i.e. instead of $|\alpha^*| \dots |\eta|$.

α	Elite species
α^*	Super-elite species
β	Non-elite species with the next-best fitness to α
β^*	Elite species with the next-best auxiliary value to α^*
γ	Non-elite species other than β
γ^*	Elite species other than α^* and β^*
δ	Proportion of elite species in the population
δ^*	Proportion of super-elite species in the population
η	All non-elite species in the population (both β and γ)
φ	Probability to swap bits between two parents in the recombination pool

λ	Size of the recombination pool
μ	Size of the population
K	Number of bins (plateaus of fitness) in a string
M	Size of the bin (length of the plateau of fitness)
M	Total number of types of infections in the population (only in Section 5.2)
m_j	Number of species with infection type j (only in Section 5.2)
$m_{1,\delta^*\alpha}$	Mean first hitting time of the absorbing state $\delta^*\alpha$ in a Markov Chain
n	Length of the string (total number of bits in the string)
N	Population size (only in Section 2.3.2)
(μ, λ)	Evolutionary Algorithm with population size μ and recombination pool size λ, no elitism
$(\mu + \lambda)$	Evolutionary Algorithm with population size μ and recombination pool size λ using some form of elitism
$P(H_j)$	Probability to select j pairs of elite parents (1BS) or j elite parents (RLS) into the recombination pool
$P(G_k)$	Probability to evolve at least one higher-ranked offspring given k improvements so far
$P(G_{0k})$	Probability to fail to evolve a higher-ranked offspring given k improvements so far

$P(\alpha)$	Probability to observe α elite parents in the population (Uniform)
$P_{sel,\alpha}$	Probability to select an elite pair (1BS) or species (RLS) into the recombination pool given α elite species in the population
P_{swap}	Probability to swap bits between parents using the KBS operator
P_{flip}	Probability to flip bits in a parent using RLS
rv	Random variable
S_{11}	The **first** expression in Phase 1
$S_{11}(\alpha^*)$	The summand in the **first** expression in Phase 1
S_{12}	The **second** expression in Phase 1
S_{21}	The **first** expression in Phase 2
S_{22}	The **second** expression in Phase 2
$\mathbf{E}\tau$	Mean first hitting time in a Markov Chain
s_k	k^{th} bin in the string
s	whole string
$V(s_k)$	Auxiliary value of k^{th} bin in the string (also V_k)
$V(s)$	Auxiliary value of the whole string (also V_s)

Chapter 1

Introduction

In this chapter, the motivation behind the thesis, its main results and outline are presented. In addition to this, a short introduction to Evolutionary Computation gives a quick overview of the area.

1.1 Introduction to Evolutionary Computation

Genetic Algorithms (GAs) were first introduced in 1975 book 'Adaptation in Natural and Artificial Systems' by J. Holland (see [Hol75]) and over time they grew to become some of the most popular heuristic optimization tools. Although since then a large number of new evolution-inspired tools has evolved, some of the most important features are still present in some form. A very general GA exhibits the routine detailed in Table 1.1. Hereinafter 'condition fulfilled' means that a pre-specified condition, e.g. the number of generations without improvement of fitness has bee fulfilled.

From now on the term Evolutionary Algorithms (EAs) instead of Genetic Algorithms (GAs) is used. EAs are a more general term that include, besides GA, Genetic Programming, Evolutionary Strategies (ES), Particle Swarm Optimization, Differential Evolution, Estimation of Distribution Algorithms, Covariance of Matrix Adaptation, Memetic Algorithms.

1	Generate starting population uniformly at random by producing a set of binary strings length n, $\{0,1\}^n$
	loop until condition fulfilled
2	Assign each species a problem-specific fitness value
3	Use a fitness-proportional selection function to select a subset of strings (not necessarily unique) into a recombination pool
4	Apply a recombination and mutation operators to each (or some) pair of parents in the recombination pool
5	Replace the old population with a new population of offsprings
	end loop

Table 1.1: Canonical Genetic Algorithm

As computational capacity grew, so did EAs' efficiency, especially on problems with complicated, poorly-understood landscapes, such as combinatorial problems that cannot be solved using deterministic approach based on deriving second-order derivatives. Problems such as Traveling salesman, knapsack, assignment, arise frequently in many areas: transport, biology, chemistry, manufacturing, etc. Also EAs are used frequently in data mining/machine learning in combination with classifiers (e.g. to optimize weights in the neural networks).

Gradually, alongside applications, theoretical investigation into efficiency of EAs on various functions evolved mainly along two lines: convergence (see Section 2.2) and runtime (see Section 2.3). The former concerns itself mostly with the probability that the algorithm finds the solution eventually and the rate of search. The latter studies the expected time, i.e. asymptotic bounds of time until the algorithm finds the solution.

So far all functions considered in the theoretical EA literature are test problems, i.e. they are either trivial or specially constructed to compare the working of different algorithms. Very generally, the set of problems considered can be divided into three subsets: 'easy' linear problems (e.g. OneMax), problems with plateaus (e.g. Royal Roads) and trap functions (e.g. TwoMax), i.e. functions with local optima. A number of toy combinatorial problems, such as vertex-cover, maximum matching, spanning trees are also frequently considered. Nevertheless, in-depth

theoretical analysis of 'real-life' problems still seems intractable.

1.2 Motivation

The main motivation and objective of this thesis is to study population-based EAs, that are often used in the real world applications, and extend numerous findings for EAs with trivial populations or recombination pools to certain more complicated algorithms. The evolution of the structure of the population and recombination pool on various problems is a matter of great interest, and one of the least-studied problems in the EA community, see e.g. [CHS+09, CTCY12].

Lack of this analysis comes from the fact that the results for simpler algorithms cannot be directly extended to more complicated ones, because they require a substantially different approach (see Section 2.3.2).

As a consequence, there is a lack of tools for the analysis of EAs that use both non-trivial population and recombination pool. One such tool, the Elitism Levels Traverse Mechanism is presented and applied to two test problems.

Another motivation is to study the performance of the K-Bit-Swap (KBS) operator, which combines features of both crossover and mutation. Analysis is restricted to 1BS (i.e. $K = 1$) in order to compare it to the mainstream Randomized Local Search (RLS) operator. Since most analysis in EA community is focused on mutation operators, this is a valuable extension.

1.3 Main results

The main results of this thesis can be subdivided into three parts:

- The introduction of a recombination operator K-Bit-Swap that is shown to have efficiency on various binary-encoded test functions,

- Analysis of dynamics and approximation of the structure of the population of EAs,
- Derivation of upper and lower bounds on the runtime of EAs

Although relevant results are summarized in their respective chapters, some of the most important ones are listed here:

1. The Elitism Levels Traverse Mechanism is a new tool that is designed to derive tight upper bounds on elitist EAs solving problems with and without plateaus. It was applied to the OneMax and Royal Roads, a test problem with plateaus. The following results evolved from this analysis:
   - proof that sub-elite species influence the performance of the algorithm
   - measured in the number of generations the order of convergence on OneMax is $O(\frac{\mu n \log n}{\lambda})$ so the performance is improved if run on parallel computers.
   - derivation of the upper bound on an arbitrary plateau function with K plateaus of the same length M. For a specific case of a Royal Road-type function with $K = M = \sqrt{n}$ this bound turns out to be $O(\frac{\mu n^{\frac{3}{2}} \log^2 n}{\lambda})$, which is a big improvement from the previous results. This is a strong argument in favor of using population-based algorithms for such functions.
   - when solving functions with plateaus, the probability of adding a super-elite species (the meaning of this term is explained in the Chapter 5) is lower-bounded by $1 - e^{-\frac{c}{8}} + o(1)$, c is a small constant.
2. In addition to the Elitism Levels Traverse Mechanism, the following properties of population-based EAs solving functions with plateaus have been derived:

- a birth-and-death Markov chain for the worst case, when no more than one (super-)elite species can be added or removed from the population

- an infinite-population approximation of the stationary distribution of (super-)elite species. It turns out to be distributed as Poisson for small improvement rates and Normal for large improvements rates.

3. Assuming Uniform distribution of elite species in the population one can derive lower bounds on runtime.

   - for $(\mu + \lambda)EA$ with 1BS solving OneMax problem it is of $\Omega(\frac{n \log n}{\lambda})$, for $(\mu + \lambda)$ Randomized Local Search (RLS) it is also $\Omega(\frac{n \log n}{\lambda})$

   - for $(\mu+\lambda)EA$ with 1BS solving Royal Roads problem it is of $\Omega(\frac{n^{\frac{3}{2}} \log^2 n}{\lambda})$, for $(\mu + \lambda)$ Randomized Local Search (RLS) it is $\Omega(\frac{n^{\frac{3}{2}} \log n}{\lambda})$
   - bounds for $(\mu+\lambda)EA$ with 1BS are asymptotically tight up to the order of μ
   - numerical results confirm the Uniform distribution assumption of the upper-bound on the probability of sampling of the elite species for Royal Roads, and reject it for OneMax

4. The results of computational experiments show the benefit of the K-Bit-Swap operator on various binary-encoded function compared to mainstream genetic operators, such as crossover, mutation, RLS, especially on Royal Roads, OneMax, k-means clustering, Rosenbrock, Ackley, Rastrigin test functions. Relevant statistical models are presented to prove this.

1.4 Outline of the thesis

In Chapter 2 past research relevant to this thesis is presented. It focuses mostly on results for population-based EAs in the past ten years. Also a quick review of mathematical tools used in this thesis is given.

In Chapter 3 the K-Bit-Swap operator is presented together with a large number of numerical results and statistical analysis. The results confirm the benefit of using KBS on a large number of functions.

In Chapter 4 the first attempt to analyze two algorithms is made (one using 1-Bit-Swap (1BS), the other RLS) on OneMax and Royal Roads test functions by making assumptions about the population structure. It is shown, both theoretically and numerically, that this approach yields the lower bound on the runtime of the algorithms. A large number of numerical results and some statistical analysis are presented that prove the validity of the models, distribution of elite species and the gap between theoretical and numerical results.

Chapter 5 presents a number of important results coming from the population partitioning approach. The Elitism Levels Traverse Mechanism is presented and a lower bound on the probability to add a (super-)elite species is derived. Using this tool, an upper bound on the runtime of the $(\mu + \lambda)\text{EA}_{1BS}$ algorithm on OneMax function is derived. A birth-and-death Markov chain is developed to use the Elitism Levels Traverse Mechanism to study a function with plateaus and derive the upper bound on the runtime. Additionally, approximations of stationary distributions of super-elite species are derived, perhaps for the first time in theoretical EA community.

The thesis concludes with the discussion of the results and possible extensions to them.

Chapter 2

Related Work

This chapter gives an in-depth review of past work in the analysis of EAs, focusing on runtime analysis.

2.1 Schemata Theorem

Most of this thesis is dedicated to answering the question 'how', i.e. how the EA works on some function. The earliest theoretical results on binary-encoded algorithms are aimed more at answering the question 'why', i.e. why EAs work. Some of the earliest findings in [Hol75, Gol89, MFH92, Mit96] that were summarized in [RR03], are dedicated to schemata theorem and building block hypothesis (BBH).

A schema H is a section of a string (substring) of type $H = x****y$ (in this case length 6) where defining bits $x, y \in \{0, 1\}, *$ means 'don't care', so e.g. substring 111111 would be an instance of H if $x = y = 1$. That is, a schema can be seen as a type of a template of genetic data.

Quite obviously, not all schemata are the same, since some templates have a higher-than-average fitness. Early theory states that EAs that use genetic operators such as biased selection, crossover and mutation implicitly estimate fitness of schemata (this phenomena is known as 'implicit parallelism', since schemata are processed in parallel) by explicitly recombining genetic information between

parents (see [Mit96], Chapter 1). To estimate the dynamics of schemata in the population, the following expression known as schemata theorem was derived. If H is a schema as defined above and $N(H,t)$ is the number of instances of schema H, then the change in the expectation of the number of instances over 1 time unit is (see in [RR03], Chapter 3.2 and [Gol89], Chapter 2)

$$\mathbf{E}[N(H,t+1)] = N(H,t)r(H,t)$$

where $r(H,t)$ is the ratio of fitness of the schema (measured as the average fitness of all strings containing H to the average fitness of the population, denoted $\frac{f(H)}{\bar{f}}$). If a certain schema H has some above-average fitness $f(H) = \bar{f} + c\bar{f} = (1+c)\bar{f}$ for some constant c, this expression can be solved recursively:

$$\mathbf{E}[N(H,t+1)] = N(H,0)r^t(H,t) = N(H,0)(1+c)^{t+1}$$

that is, the number of instances grows geometrically (see [Gol89], Chapter 2). If two other operators, crossover and mutation, are added, the following expression is obtained:

$$\begin{aligned}\mathbf{E}[N(H,t+1)] &\geq N(H,t)r(H,t)\left(1 - \frac{\delta(H)}{l-1}P_c - o(H)P_m\right) \\ &= N(H,0)(1+c)^{t+1}\left(1 - \frac{\delta(H)}{l-1}P_c - o(H)P_m\right)\end{aligned}$$

The expression in the brackets is the probability of survival of the schema, which is intuitively inverse-proportional to its length. P_c is the probability of crossover ($P_c = 1$ means it is applied to every pair of parents in the recombination pool), $\delta(H)$ is the defining length (distance, measured in bits, between the defining bits, in the example above $\delta(H) = 6 - 1 = 5$), l is the length of the string. P_m is the probability of flipping a bit, $o(H)$ is the order of schema (the number of defining bits, in the example above $o(H) = 2$).

2.1.1 Criticism of Schemata Theorem

A number of counterexamples to the schemata theorem have been suggested, e.g. a deceptive function (see [Gol89], Chapter 2 and [Mit96], Chapter 4). In [RR03] both numerical and analytical evidence was presented, as well as a compilation of counterexamples to the schemata theorem and implicit parallelism. For example, they argue that the focus on the number of instances of schemata are misplaced and in fact the theorem extends to arbitrary subsets of fitness landscape. This suggestion was made already in [Vos93, Vos99]. In addition to that, in [Vos99] it was shown that changing the mutation rate by a tiny value leads to a serious change in EA's trajectory.

In the next subsection a brief introduction to alternative theories of EA efficiency is provided, focusing on the idea of Markov chains.

2.1.2 Alternative explanation of EA efficiency

As an alternative to schemata theorem and BBH a number of theories have been suggested, among them Statistical mechanics (see [PB94, RS96]) and generative fixation hypothesis (see [Bur09]). In the remainder of the Section the focus is on the Markov Chains-based explanation, as it is the closest to the ideas discussed in this thesis.

Instead of analyzing the string structure, in [NV92] it was suggested to use a function $\mathcal{G}(x)$, a sampling distribution from current population x to determine the expected drift or direction of the population in the next generation. That is, $\mathcal{G}(x)$ is a probabilistic model that shows the drift of the population. The next task was to express the probability to obtain the population j from the current population i:

$$Q_{i,j} = n! \prod_{y=0}^{r-1} \frac{\left\{ \mathcal{M}\left(\frac{F\phi_i}{|F\phi_i|} \right)_y \right\}^{z_{y,j}}}{z_{y,j}!}$$

This equation is an exact transition matrix between populations i, j: if there exists a string length l, the total number of all possible strings is r, the populations size n, the total number of populations is $N = \binom{n+r-1}{r-1}$. The number of occurrences of string y in population P_j is $z_{y,j}$, $\mathcal{M}$ is a set of crossover and mutation operators, ϕ_i is an incidence vector of population i (number of strings y in population i), $\frac{F\phi_i}{|F\phi_i|}$ is the probability of selection of y for recombination.

To apply $\mathbf{Q}$, if there exists probability measure at generation k π_k over the set of populations, then

$$\pi_k = \pi_0 \mathbf{Q}^k$$

and

$$\lim_{k\to\infty} \pi_k = \pi$$

j^{th} element of the vector π is the limiting proportion of time the algorithm has population P_j. Due to ergodicity of $\mathbf{Q}$, no entries in this vector are equal to 0. Nevertheless, if the population size is infinite ($n \to \infty$), there exists a fixed collection of states (populations) π^* such that

$$\pi^* = \lim_{n\to\infty} \pi_n$$

Details of the derivation of these equations as well as properties of π^* can be found on [NV92, Vos93, Vos99], and an overview of approaches is in [RR03].

2.2 Convergence Analysis

Since the early 90s, attempts have been made to apply Markov chains and other probabilistic tools to model EAs, i.e. to find *how* they work (rather than *why*). Some of the earliest results can be found in [Rud94b, Rud94a, Rud96, Rud98], where a number of convergence rates were derived. In [Rud94a] it was proven that a canonical GA (see Table 1.1) with fitness-proportional selection, crossover and mutation does not converge asymptotically to the global optimum of a binary-encoded fitness function due to the ergodicity of the underlying Markov Chain, but the elitist one does (although later in [DJW02] it was proved that the upper

bound of runtime on any binary-encoded problem is $O(n^n)$, for the explanation of big-O notation see Section 2.5).

Test problems used in these proofs include a parabolic Sphere function ([Rud94b]) or some very generally defined functions, such as bounded from below([Rud96]) or convex functions ([Rud97]). Some of the important results from these publications are presented below.

In [Rud94a] the framework for analyzing EA convergence was presented. It is based on the transition matrix $\mathbf{P}$, defined on all possible populations (exactly like in [NV92]), which is a matrix product of three stochastic matrices:

$$\mathbf{P} = \mathbf{C} \cdot \mathbf{M} \cdot \mathbf{S}$$

where $\mathbf{C}$ is the one-step transition of EA population using crossover, $\mathbf{M}$ - mutation operator and $\mathbf{S}$ - selection. The author argues that $\mathbf{P}$ is ergodic, therefore regardless of the initial distribution all states (populations) have a nonzero limiting probability. Thus, an EA without elitism never converges to the global optimum (it finds and loses it an infinite number of times).

This argument is extended to EAs that maintain the best solution (elitism), and it is proven they are guaranteed to converge to the global solution. The question that naturally arises is the convergence rate, i.e. the rate at which the algorithm converges to the solution.

In [Rud94b] a non-elitist $(1, \lambda)$ algorithm (i.e. the one that does not keep the best solution found so far) solving a Sphere function was considered. First, conditions are derived under which it converges to the global optimum. Secondly, expression for the expectation of the maximal relative improvement ($V(\lambda)$):

$$\mathbf{E}[V(\lambda)] = \left(\frac{\lambda - 1}{\lambda + 1}\right)^2 \tag{2.1}$$

The results are true for functions with convex region around the global optimum. Also in [Rud96] it was proven for general search spaces (both binary-encoded and Euclidean) that if there exists A_ε, the set of $\varepsilon-$ optimal states and the transition kernel from non-optimal to optimal states in 1 generation $K(x, A_\varepsilon) \geq \delta > 0$, then regardless of initialization an EA converges as $t \to \infty$ almost surely since

$$K^{t+1}(x, A_\varepsilon) = 1 - (1 - \delta)^{t+1} \tag{2.2}$$

$$P(X_{t+1} \in A_\varepsilon) \to 1 \tag{2.3}$$

(by the Borel-Cantelli Lemma)

A large number of these and other important results (e.g. [Suz95, HK99]), concerning mainly convergence rates on general functions, convergence probabilities and selection of parameters were summarized in [Rud98, Rud99].

The remainder of this Chapter is dedicated to the overview of EA runtime analysis, which is more relevant to this thesis, and is an answer to the question 'how long it takes EAs to find solutions to different problems'.

2.3 Runtime Analysis

This area of EA analysis evolved in the late 90s. It concerns itself predominantly with the asymptotic properties of EAs (this term and its benefit is explained in great detail in Section 2.5), first of all upper and lower bounds on the expected runtime of the algorithm, understood in the same sense as the mean first hitting time in a Markov chain (see Appendix B). Tools and approaches in this area are summarized in [OHY07], some of them are:

1. (1+1) EA with mutation or Randomized Local Search Algorithm (the former flips each bit in the string with some probability, the latter an exact number of bits),

2. mutation rate $\frac{1}{n}$ is the predominant genetic operator (or flipping exactly 1 bit if RLS is used),

3. binary-encoded problems: OneMax, Leading Ones, Binary Value and generalizations to linear functions (see [DJW02, DJW10b, DJW10a, DFW11]),

4. also some combinatorial problems and trap functions are analyzed: Vertex cover ([OHY08]), Minimum Spanning Tree ([NW04, DJW10b]), Shortest-Path ([DHK11]), Maximum Matching ([OW11]), Mincut ([Sud08a])

5. mathematical tools include limit theorems from probability theory (Markov, Chebyshev, Chernoff inequalities, etc) and Drift analysis (see [HY01, DJW10b]), which is an adaptation of martingale theory from stochastic processes.

Using Landau notation (see e.g. in Section 2.1. in [Sud08a]) many results have been derived (some of them were mentioned already in [Rud98] and referenced to early papers), mostly for $(1+1)$ EA:

1. the bounds on $(1+1)$ with mutation rate $\frac{1}{n}$ and RLS are $\Theta(n \log n)$, see [DJW02],

2. recently the result for OneMax has been refined up to $0.982n \log n$ in [DFW11] using probability generating functions (PGFs),

3. for all linear functions, the expected optimization time is $\Theta(n \log n)$, see [DJW10a],

4. for LeadingOnes the expected optimization time is $\Theta(n^2)$

and many others. In [Wit04] problems that have runtime of $(1+1)EA$ $\omega(n \log n)$ are referred to as 'difficult'.

In [HY03] an analytic framework based on Markov chains was presented to analyze optimization time of EAs. This included:

1. definition of the absorbing MC (elitist EA) with the number of states corresponding to the number of artificial fitness levels (for this term see Subsection 2.4.1)

2. probabilistic nature of solution: the runtime is expressed through an expected time (upper bound on the expectation is the worst-case approach)

3. model of EA includes all mainstream genetic operators: fitness evaluation, selection, mutation and crossover

4. classification of problem complexity based on the expected time of solving the problem: polynomial and exponential

In [HY03] a whole range of algorithms using different genetic operators were analyzed. Expressions for some mean first hitting runtimes are found, but not solved, i.e. bounds are not expressed in the closed form or approximated.

2.3.1 Runtime analysis of $(\mu + 1)$ and $(1 + \lambda)$ EAs

Recently analysis of EAs that use some population and recombination operators has evolved. It is focused on the role of population in the evolution and effect of different sizes of population and offsprings. Some of the main results in this area are outlined below.

For a $(\mu + 1)$ EAs with the mutation operator in [Wit04] the upper bound on OneMax function was found to be $O(\mu n + n \log n)$ and lower bound $\Omega(\mu n + n \log n)$ for the same algorithm on any function with a unique global optimum. The latter was done using a family tree tool described in [Wit04] in great detail.

A $(1 + \lambda)$ EA with bitwise mutation operator and elitist selection function was found to solve OneMax in $O(\frac{n \log n}{\lambda} + n)$ generations or $O(n \log n + n\lambda)$ function evaluations (population size $\times$ order of runtime) in [JDJW05]. If measured in terms of function evaluations an increase in the offspring size degrades performance past a cut-off point $O(\frac{\log n \log \log n}{\log \log \log n})$. Small deviations from $O(\log n)$ can improve performance though.

In [He10] this result was generalized to all linear functions: for $\lambda < e^e$ expected runtime (measured in the number of function evaluations) is of the same order as $(1 + 1)$EA, $O(n \log n)$. Larger offspring size degrades performance.

An important consequence from this is that, since the term λ is in the denominator, if run on parallel computers, the increase in the offspring size actually improves performance (see [He10] for details).

2.3.2 Runtime analysis of $(\mu + \lambda)$ EAs

This area of research is the most relevant to this thesis. Its complexity arises, among other things, from the fact that results for $(\mu + 1)$ and $(1 + \lambda)$ algorithms are not directly extendible to the $(\mu + \lambda)$ one. This is easy to see by combining results for the bounds in [Wit04] for the $(\mu + 1)$ and in [JDJW05] for $(1 + \lambda)$. There is no way these yield the $O(n \log n + nN \log N)$ in [CHS$^+$09] (see below). This explains the demand in the development of a different approach to analyze $(\mu + \lambda)$ algorithms.

Some of the earliest publications in this area are [HY01, HY02, HY03, HY04]. Specifically, in [HY02] it was shown that for different $(N + N)$ EAs (N is the population size) with variants of tournament selections population effect is problem-specific. It was also shown that positive effect from the population tends to level out, although it is measured in the number of generations rather than function evaluations, so the result is true only for parallel computers. Tested on a number of trap-type functions, speed-ups in terms of time-complexity and increased convergence probability have been determined, as well as cases that reduce exponential-time complexity to polynomial time.

Later in [CHS$^+$09, CTCY12] the work of the $(N+N)$EA with mutation and fitness-proportional selection was modeled by tracking the progress of locally-optimal individuals (LOI), i.e. the currently elite species. The expected runtime of the OneMax problem (measured in the number of generations) is

$$\mathbf{E}\tau = O\left(n \log N + \frac{n \log n}{N}\right)$$

Compared to $(1+1)$ EA with mutation or RLS no advantage of population-based algorithms measured in the number of function evaluations was determined. Additionally, it is hard to tell the effect of the population from the recombination pool, since both are of the same size N.

In [CTCY12] on a TrapZeros test function, which is harder to optimize than OneMax or similar linear functions due to a basin of attraction that reduces the probability of finding the global optimum, a $(1+1)$EA outperformed two other algorithms both with population and recombination pool size $\omega(1)$: the first one $N = O(\log n)$, the second $N = O(\frac{n}{\log n})$. Although the runtime of the first algorithm is the same as the $(1+1)$ one, $O(n^2)$, the probability to find the optimum is $\frac{1}{poly(n)}$ (some polynomial function of n) in the former case and $\frac{1}{4}$ in the latter case. For the algorithm with the largest population size convergence time is superpolynomial.

Obviously it is of great interest to find out, for what problems the solution benefits from the increase in population size, both in terms of expected runtime, and the success probability and what are the cut-off points of this. Another question that, to the best of my knowledge, has only been touched upon in [CHS+09], is the structure of the population. Quite obviously the distribution of species in the population affects greatly the evolution. Structures arising from this process are both interesting and very complicated. These questions are the main areas of research in Chapters 4 and 5. Specifically, in Section 4.2 it is explained, why analyzing $(\mu+\lambda)$ algorithms is both interesting and complicated, and why the approach in this thesis is useful.

2.4 Review of tools used for analyzing EAs

In this section a comprehensive overview of approaches to EA analysis is presented.

2.4.1 Fitness-Based Partition and Artificial Fitness Levels

This is one of the earliest and most intuitive tools. It is closely related to the Coupon Collector's Problem (see below). The main idea is quite clear: $\exists$ a partition of the full set of possible binary strings size 2^n : $L_0, L_1, \ldots, L_n$ such that $L_i \subseteq \{0,1\}^n$, $L_i \cap L_j = \emptyset$ and $\cup_{i=1}^{n} L_i = \{0,1\}^n$. For many binary-encoded functions though the actual location of bits in the string is not important, therefore instead of having 2^n fitness levels, one can have $m < 2^n$ artificial fitness levels, where each string a, b are fully defined by their fitness, i.e. if $f(a) = f(b)$, they are on the same artificial fitness level (and vice versa).

The idea of artificial fitness levels easily extends to populations, which is very helpful for the purpose of this thesis. All sets of populations size μ are fully defined by the fitness of the best individual, thus reducing the search space from N defined above to n, the length of the binary string, in case this approach is used for functions that depend on the number of 1-bits in the string.

2.4.2 Gambler's Ruin and Coupon Collector's Problem

This is a pretty simple and intuitive approach to modeling EA or any stochastic algorithm that maintains the best solution (elitism). A good review of both can be found in [Weg03, DJW02, OHY07]. Approach here is based on probability theory and Markov chain. A good introduction to Markov chain theory is in [Ios80, Ros06, Shi07a, Shi07b].

Gambler's Ruin

A random process X_n is a sequence of independent and identically distributed random variables Z_i : $X_n = \sum_{i=0}^{n} Z_i$ such that $\mathbf{P}(Z_i = +1) = p$ and $\mathbf{P}(Z_i = -1) = q = 1 - p$ ($Z_0 = a > 0$). This can be viewed as a wealth of a gambler. One is interested in the limiting probability and expected time that the gambler loses all his money (if he is allowed to gamble forever). If $\alpha_n(x) = \mathbf{P}(X_n = 0)$, then

$$\lim_{n \to \infty} \alpha_n(x) = \alpha(x)$$

A more realistic and complicated variant of this game is when success and failure probabilities are state-dependent (i.e. $p_i \neq p_{i+1},\ q_i \neq q_{i-1}\ \forall\ i$ with some boundary conditions).

This approach is applied extensively in Chapter 5. The main idea is that the fitness function does not distinguish between species on a plateau, and therefore EA performs a random walk rather than biased search.

Coupon Collector's Problem

Assume $X \sim Geom(p)$. Then, using characteristic function of the Geometric random variable (rv) with parameter p:

$$\begin{aligned}
\phi_t(X) &= \lim_{n\to\infty} \sum_{k=1}^{n-1} e^{itk}(1-p)^{k-1}p = \frac{p}{1-p}\frac{1}{1-e^{it}(1-p)} \\
\frac{d\phi_t(X)}{dt} &= \frac{p}{1-p}\cdot\frac{i(1-p)e^{it}}{(1-e^{it}(1-p))^2} \\
\mathbf{E}X &= \left.\frac{d\phi_t(X)}{idt}\right|_{t=0} = \frac{p}{1-p}\cdot\frac{1-p}{p^2} = \frac{1}{p}
\end{aligned}$$

Applying this idea to EAs, if the algorithm starts with the string all set to 0 and flips exactly one bit randomly, as in the Randomized Local Search, the probability to select a 0-bit for flipping is $p = \frac{n-i}{n}$ (where i is the number of 1-bits so far). Since bits are selected randomly each generation, this can be seen as a sum of independent (but not identically distributed) Geometric rvs with expectation $\frac{1}{p} = \frac{n}{n-i}$. Therefore, since there are n bits to be flipped, the mean first hitting time becomes

$$\mathbf{E}Y = \mathbf{E}\sum_{i=0}^{n-1} X_i = \sum_{i=0}^{n-1}\frac{n}{n-i} = nH_n = O(n\log n)$$

where H_n is n^{th} Harmonic number. By the integral test, its value lies between $\log n$ and $\log n + 1$ (see [GKP95], Chapter 6).

2.4.3 Potential/Auxiliary Functions

This type of functions is closely associated with the drift analysis (a type of a martingale) extensively used in EA community. It was first introduced in [Haj82] and then in [HY01, HY03] for the first time in the EA community. Some recent use includes [DJW10b, DJW11, OW11] etc. The main idea of a potential function is to complement fitness, if the fitness-proportional selection function alone fails to capture certain important steps in the evolution. It has to posses some properties of the fitness function, but be easier to work with. For example, for a linear functions with weights:

$$f(x) = \sum_{k=1}^{n} w_k x_k \quad \text{s.t.}$$
$$x_k \in \{0,1\}, w_k > 0, w_k < w_{k+1} \ \forall k$$

the potential function is (c being a constant)

$$g(x) = \log\left(1 + \sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} x_j + \sum_{j=\lfloor \frac{n}{2} \rfloor + 1}^{n} c x_j\right)$$

which allows to bound the drift. In [Jag08] OneMax was selected as a potential function for all linear functions in order to recover the $\Theta(n \log n)$ bounds on all linear functions.

In this thesis the idea of potential/auxiliary function is used extensively in Chapters 4 and 5 to study convergence on functions with plateaus (Royal Roads). The main problem with this type of functions is that the fitness does not differentiate between species on the plateau with different distances to the next fitness level (plateau). In order to track the evolution of the population on each plateau, OneMax as a simple auxiliary function is used.

2.4.4 Analysis of Typical runs

Although much of EA analysis is focused on the derivation of upper bounds, i.e. worst-case scenario, it is sensible to study typical behavior of the algorithm, i.e. break down its work into phases that hold 'with high probability'. However, it seems that this definition is somewhat vague, since in [OHY07, Wit12] it is defined as $1 - o(1)$, but in [Sud08a] it is merely some 'high' probability, often as low as $\frac{1}{2}$.

For example, in [DJW02] the lower bound of $\Omega(n \log n)$ on $(1 + 1)$EA solving any linear function holds with probability at least $\frac{1}{2} \cdot (1 - e^{-\frac{1}{2}}) \approx 0.1976$, where $\frac{1}{2}$ is the probability that algorithm initializes with $\frac{n}{2}$ 1-bits. In [Sud08a] the same algorithm was found to solve OneMax with probability $1 - e^{-1} - e^{-\frac{n}{36}} \approx 0.6321$ within $(n - 1) \log \frac{n}{3}$ generations. By Chernoff bounds, the probability that the algorithm initializes with less than $\frac{n}{3}$ 0-bits is less than $e^{-\frac{n}{36}}$. Only in [Wit12] the upper bound on linear functions with mutation rate $O(\frac{1}{n})$ was shown to be $O(n \log n)$ with probability at least $1 - \frac{1}{n} = 1 - o(1)$. Also in [Sud08b] high probability was defined as $1 - n^{-\epsilon}$ for $\epsilon > 0$ and overwhelming probability $1 - 2^{-\Omega(n^{\epsilon})}$.

In fact, in most papers exponential (or just converging to 1) probability has been shown either only for the initialization stage (using Central Limit theorem or other limit theorems), e.g. for OneMax function in [Sud08a] or Needle function in [OW11] or for pretty rough bounds, e.g. for $(1 + 1)$ EA on Prefix Ones/Leading Suffix Ones (PO/LSO) problem in [Wit04] the probability of finding a solution within $2^{\Omega\left(\frac{\sqrt{n}}{\log n}\right)}$ generations was found to be at least $1 - 2^{-\Omega\left(\frac{\sqrt{n}}{\log n}\right)}$.

2.4.5 Structure of individuals in the population

This approach was mentioned in [Sud08a], but was mainly developed and applied in [CHS+09, CTCY12] to the runtime of $(\mu+\lambda)$ EAs. Some results in this thesis were inspired by findings in these articles. One of the main ideas is the accumulation of locally-optimal (currently elite) individuals (LOIs) of $(N + N)$ EAs that use some form of mutation. The optimization time is broken down into consecutive

phases. In the first phase the population accumulates LOIs until ϵN of it consists only of LOIs ((k, ϵ)-takeover time $\bar{\eta}, k$ is the fitness level, $\epsilon > 0$). The probability to improve fitness in this phase is 0. In the next phase the probability to improve fitness (evolve at least 1 advanced LOI, better individual) is defined as $p^k_{u,\frac{1}{2}}$. Mean first hitting time till the next fitness level given current population $\xi_k(t)$ is

$$\mathbf{E}\tau_k = \sum_{Y \in E_k} \mathbf{E}[\tau_k | \xi_{t_k} = Y] \mu_{t_k}(Y)$$

where $\mu(\cdot)$ is the distribution of population Y at time t_k. The upper bound on the expected optimization time of the algorithm is therefore

$$\mathbf{E}\tau = O\left(\sum_{k=1}^{n} \left(\bar{\eta}_{max,\frac{1}{2}} + \frac{1}{p^k_{u,\frac{1}{2}}} \right) \right)$$

Applying this equation to OneMax and LeadingOnes test function, upper bounds of respectively $O(nN \log N + n \log n)$ and $O(nN \log N + n^2)$ are derived.

2.5 Asymptotic notation

This Section is quite important, since the terms defined here are extensively applied throughout this thesis, e.g. $\mathbf{E}\tau = O(\cdot)$, $\Omega(\cdot)$. The main reason for this approach is that very often the obtained expression is either very complicated or does not exist in the closed form at all. The use of big-Oh notation, i.e. 'up to a normalizing constant' vastly reduces the efforts necessary to derive bounds on the runtime. It is expressed only through the arguments of the function (i.e. parameters of the algorithm)

Formally, for two functions, $f(n)$ and $g(n)$ $\exists N$ s.t. $\forall n >> N$ $f(n) \leq Cg(n)$, or

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = C > 0$$

In case the inequality above holds from below: $f(n) \geq Cg(n)$, then $f(n) = \Omega(g(n))$. In case $O(g(n)) = \Omega(g(n))$, then $f(n) = \Theta(g(n))$.

The second case is when

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

then the asymptotic order is $f(n) = o(g(n))$ and $o(1)$ denotes convergence to 0. The reciprocal is

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

that is, $f(n) = \omega(g(n))$.

2.6 The No Free Lunch theorem and Analysis of computer algorithms

In this section a brief overview of this very important theorem is given that justifies interest in the analysis of computer algorithms. One of the practical results of the ideas developed in this thesis (which follows from The No Free Lunch Theorem, or NFL) is the choice of the relevant algorithm that has a higher probability of finding the global optimum in shorter runtime.

In [WM97] NFL was presented for stochastic optimization algorithms, which naturally includes EAs. It levels out the efficiency of algorithms: if an algorithm is efficient on a class of problems, its performance is counterbalanced on another class. More formally, if the history of distinct function evaluations is denoted by d, the number of function evaluations m on a cost function f solved by algorithm a_k and $P(\cdot)$ performance of the algorithm then

$$\sum_f P(d|f, m, a_1) = \sum_f P(d|f, m, a_2)$$

that is, the performance of the algorithm is independent of a_k. This justifies search for such subsets of a class of cost functions on which algorithms a_1 and a_2 outperform each other. The proof of this is in Appendix A in [WM97].

There are three main ways to compare performance of the algorithms on a minimization problem:

- average probability of failure (best search point lies above some value ϵ),
- the fraction of algorithms which for a specific f and m have failed to find the global optimum,
- expression of the failure probability for a random algorithm that does not use information from d_m (past information)

The first criteria is widely used in EA community to estimate the efficiency. In the next Chapter is used too. In Chapters 4 and 5 though the main benchmark is the mean first hitting time (as in the MC, see Appendix B).

2.7 Parallel computers

Throughout the thesis the expression 'run on parallel computers' is often used. This means the following: if the runtime of the algorithm is $O\left(\frac{g(n)}{\lambda}\right)$, and the algorithm is run in parallel on $O(\lambda)$ computers, so the fitness values for the parents selected into the pool are found simultaneously, thus the performance is vastly improved if measured both in the number of generations and function evaluations. In a different situation, if the same algorithm is run on the same computer, this operation (unless parallelized) has to be repeated $O(\lambda)$ times. This means, that if measured in the number of function evaluations, performance has to be multiplied by $O(\lambda)$, and the runtime increases to $O(g(n))$.

Chapter 3

The K-Bit-Swap Genetic Operator

This chapter introduces the K-Bit-Swap (KBS) operator and tests it on a number of different functions. According to the No Free Lunch Theorem, if an algorithm overperforms on a class of functions, it underperforms elsewhere. This is an attempt to identify functions on which an EA using some form of KBS performs better than that without any form of it. In Section 3.3.1 the main benefit of the analysis based on these statistical results is presented: it allows one to understand which parameter settings to use to maximize the probability of solving the problem and reduce the runtime.

3.1 Explanation of the K-Bit-Swap Genetic Operator

This operator was first introduced in [TSMH10]. The motivation was to develop an operator that combines features of both uniform crossover and mutation or local search, but without some of their drawbacks. This adds greater flexibility to the search for the bits to be flipped. Its work is presented in Table 3.1 and, in order to compare to similar operators, in Figure 3.1. Operators that resemble KBS were proposed before (see e.g. in [SC99]).

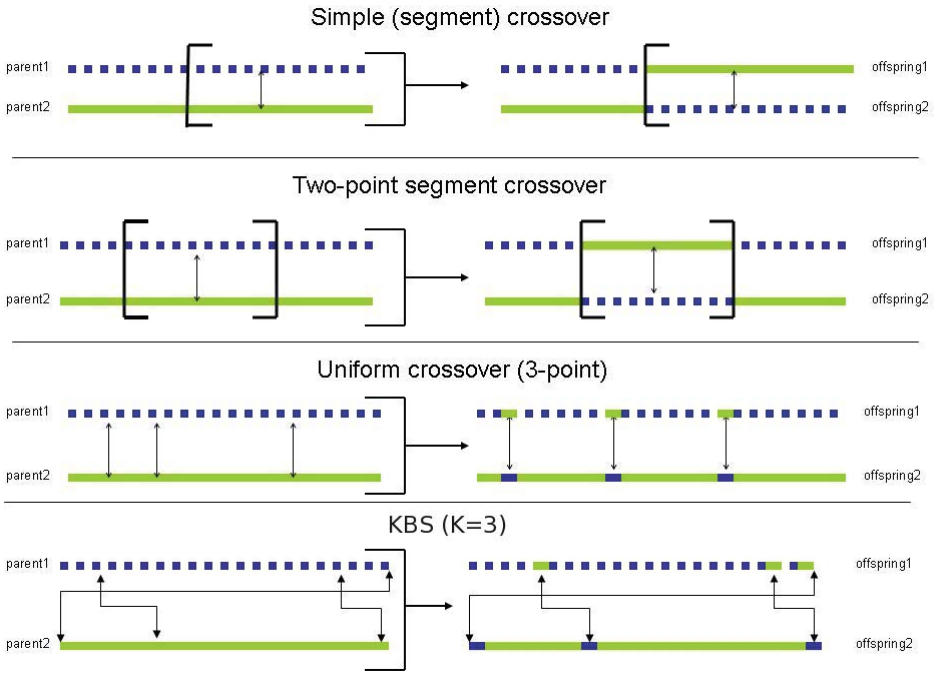


Figure 3.1: Comparison of K-Bit-Swap to simple (segment) crossover, 2-point simple (segment) crossover and Uniform crossover

	loop over the number of pairs in the recombination pool
1	select k bits in the first parent uniformly at random
2	select k bits in the second parent uniformly at random
3	swap (exchange) values in these bits
	end loop

Table 3.1: The K-Bit-Swap Genetic Operator

KBS resembles both Uniform crossover in the way that it selects bits in both parents uniformly at random and RLS/mutation, because it swaps an exact number of bits between them. This allows KBS to select the second bit independently (unlike uniform crossover), but it recombines information between two parents, unlike RLS/mutation. The main objective of this Chapter is to show its efficiency on a range of functions compared to mainstream genetic operators: crossover and mutation.

The author hypothesizes that KBS is likely to be more effective on problems where linkage between the neighbouring bits is not important, e.g. binary-encoded, multi-modal or monotone problems. On the other hand, combinatorial integer-encoded problems, such as the Travelling Salesman Problem, are problems with a large number of local solutions, where the linkage between neighbouring solutions can be quite important, since a minor change in the fitness (trips between two cities) can bring about a substantial change in the value of the fitness function.

Since KBS, unlike mutation, is not ergodic, i.e. it recombines existing information and cannot evolve solutions that cannot be constructed from the set of current solutions, it may under certain conditions impair the diversity in the population. For example, if the recombination pool contains only strings with 1-bits, it cannot evolve an offspring that has a 0-bit anywhere.

This hypothesis directly implies that in the real world it is probably best to combine KBS with other operators to maintain diversity and improve performance.

3.2 Algorithms and Experimental setup

The pseudocode of the algorithms compared in this chapter are presented in Table 3.2. They are the same as the canonical EA in Table 1.1, except adding KBS. Rates for all genetic operators as well as other parameters are specified in Table 3.3 (values in columns 3,4 are the number of bits crossed over/swapped, values in column 5 are probabilities to select a bit for mutation). As a fitness-proportional selection function, instead of linear (Equation 3.1) softmax function (Equation 3.2) is used to increase the probability of sampling species with higher fitness (for minimization problems the $f(x_i)$ in the exponential becomes $-f(x_i)$). All genetic operators have a 100% rate, i.e. they are applied to each string or pair of strings

in the recombination pool.

$$P_{sel}(x_i) = \frac{f(x_i)}{\sum_{i=1}^{\mu} f(x_i)} \tag{3.1}$$

$$P^*_{sel}(x_i) = \frac{e^{f(x_i)}}{\sum_{i=1}^{\mu} e^{f(x_i)}} \tag{3.2}$$

1	Initialize population of size μ
	loop over the number of generations
2	select $\frac{\lambda}{2}$ pairs of parents into the recombination pool using softmax selection function
3	apply crossover to each pair of parents
4	apply KBS to each pair of parents
5	apply mutation to each offspring
6	keep the specified number of the best species in the population, remove the same number of offsprings from the pool randomly, replace the rest of the population with the remaining offsprings
	end loop

Table 3.2: Pseudocode of EAs in Chapter 3

3.2.1 Problems selected for testing

Before analyzing the results, the details of each problem solved by the algorithm are presented.

Functional optimization (FuncOpt)

In all functional optimization problems, the number of dimensions is limited to two (i.e. they are functions of two arguments). The problems are binary-encoded, with each chromosome being 42 bits long, 20 bits for each value and 2 for the sign of the value, because all of the problems have support over negative values as well and standardized to keep the values within the designated bounds. Therefore, for example, for the Rastrigin function, 1111...111 (42 ones) stands for [-5.12,-5.12], 0111...0...11 (2 zeroes and 40 ones) stands for [5.12, 5.12], 000...000 (42 zeroes) stands for [0,0].

Rosenbrock, also known as a banana function (since the basin around the optima resembles a banana), is one of the most popular test problems for the verification of the efficiency of numerical and stochastic algorithms. This is due to the large number of local optima that are supposed to mislead the optimization algorithm and prevent it from finding the global solution. In the two variables case its equation is

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

where $-32 \leq x_1, x_2 \leq 32$ and the global solution is $[1, 1]$.

One of the most complicated local optima is the point $[0, 0]$ and many algorithms get stuck in it.

Another function with multiple local optima is **Ackley** function. For two variables its equation is

$$f(x_1, x_2) = 20 + e - 20e^{-0.2\sqrt{\frac{(x_1^2+x_2^2)}{2}}} - e^{\frac{(\cos(2\pi x_1)+\cos(2\pi x_2))}{2}}$$

where $-32 \leq x_1, x_2 \leq 32$ and the global solution is the point $[0, 0]$.

Rastrigin is also a minimization problem with a large number of local optima.

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos(2\pi x_1) + \cos(2\pi x_2))$$

where $-5.12 \leq x_1, x_2 \leq 5.12$ with the global solution in the same point as Ackley: $[0, 0]$.

EA-specific problems

These problems have been developed specifically to test EAs' ability to avoid premature convergence (Four peaks) or their capacity to detect and recombine

good schemata, and also traverse plateaus of fitness (Royal Roads). A large number of good heuristic solutions exist for both of them (including Hill Climbing and PBIL [Mit96, BC95, DBIJV97]).

Royal Roads

This problem (sometimes abbreviated as RR) is described in Section 4.4.2 in great detail. All that is needed to say here is that this is a function with plateaus of fitness that many EAs find hard to cross. The fitness is

$$f(x) = \sum_{k=0}^{K-1} c_k \prod_{j=0}^{M-1} x_{kj}$$
$$x_{kj} \in \{0,1\}, c_k = M \ \forall \ k$$

For the purpose of numerical analysis here $M = 8$, the same value as in [Mit96].

Four peaks

This is another EA-hard test problem. It is specifically designed to trick the EA into converging to a local minimum (see [DBIJV97]). The bonus (R) is set to 100 and the threshold (T) is set to 10. The main idea is that a certain ratio of 0-bits and 1-bits (threshold) has to be maintained in the string for the algorithm to achieve the global maximum (bonus). This is the objective function for Four peaks:

$$f(s) = \max\{\text{head}(0,s), \text{tail}(1,s)\} + R(s,T)$$

where

$$R(s,T) = \begin{cases} 100 & \text{if} \quad \text{head}(0,s) \quad \text{and} \quad \text{tail}(1,s) > 10 \\ 0 & \text{otherwise} \end{cases}$$

Head(0,s) means that the first s bits of the string have value 0, while tail(1,s) means that the last s bits have value 1.

Combinatorial optimization

This group of problems for the purpose of this thesis involves only one instance, the Traveling salesman problem (TSP), which is presented both as a trivial problem (cities on a circle) and a non-trivial one, on 48 US Capital cities. TSP is one of the best known problems in combinatorial computing, with a large number of applications (e.g. see [LK73]).

This is an unconstrained problem, so the fitness function for n cities is

$$D = \sum_{i=1}^{n-1} d(x_i, x_{i+1}) + d(x_n, x_1)$$

where x_i is the ith city of the tour, and $d(x, y)$ is the Euclidean distance between two cities.

All solutions considered by the EA are valid tours, i.e. every city appears in the string exactly once, except for the starting city, which is also the last one (this is reflected in the fitness function). The mutation operator can be applied in two different ways:

1. 2-p-opt. For this mutation type two cities are randomly selected and swapped to get a new solution. The order of cities between them is inverted.

2. 1-shift. A city in the tour is randomly selected and inserted anywhere in the tour, shifting the cities between the old and new position left or right. This approach has been found in [BKB05] to be more efficient for TSP and it is used here instead of 2-p-opt.

TSP on a circle is a trivial problem, since the global solution (circle) is obvious. For an EA it is a hard problem, especially as the number of cities gets large. The locations of the cities are restricted to the unit circle, so the shortest path for 26 cities is 6.2669 (this value converges to 2π as the number of cities tends to infinity). All solutions with $D < 7$ are considered to be within the vicinity of the global minimum.

TSP on 48 US cities is a non-trivial combinatorial problem for which the best solution has been found in [PR91, Ber].

This result is used as a benchmark against which findings here can be compared. Given the complexity of the problem, any tour with $D < 40,000$, which is more than 20% away from the global minimum, is within its vicinity.

k-means Clustering Problem

This sort of problem may arise in a large number of applications. This approach is a hybrid algorithm, with a k-means (here $k \equiv 4$) classifier and an EA optimizer. There are two variants of this problem: trivial (see Figure 3.2) and random. For the purpose of standardizing, the coordinates of data points lie in the [0,1] interval. The objective function is

$$D = \sum_{i=1}^{n} d(x_i, y)$$

where
x_i is the ith observation,
y is the nearest centroid for the i^{th} observation, and
$d(x, y)$ is the Euclidean distance between two points.
n number of data points (observations)

Trivial k-means problem Exactly as with the TSP, a trivial instance of the k-means problem is presented, with a set of observations located in the four 'corners' (see Figure 3.2).

Random k-means problem In this problem a set of 16 data points is generated randomly and the EA minimizes D by approximating the optimal locations for 4 centroids. To obtain comparable results, the randomly generated set of data points is fixed for the whole run of the algorithm.

Function Type	**PopSize**	**Crossover Rate**	**KBS**	**Mutation Rate**	**Chrom Length**	**Elitism**
FuncOpt	50,100	0,simple, 1,10,20	0,1,10,20	0, 0.025, 0.25, 0.5	40	1,50%
Royal roads	50,100	0,simple, 1,100,400	0, 1, 100, 400	0, 1/n, 100/n, 400/n	800	none
Four peaks	50,100	0,simple, 1,25,50	0,1,25,50	0, 0.01, 0.02,0.05	100	1,50 %
TSP	50,100	0,simple, 1,2,3	0,1,2,3	0, 1/n, 2/n, 3/n	26,48	1,50 %
k-means	50,100	0,simple, 1,5,10	0,1,5,10	0, 0.0125, 0.025, 0.05	160	1, 50 %

Table 3.3: Parameter settings for the problem set.

Function Type	**Global Optimum**	**Global optimum threshold**
FuncOpt	0	< 0.005
RoyalRoads	800	> 760
Four Peaks	189	> 122
TSP (trivial)	6.2669 ($\approx 2\pi$)	< 7.00
TSP (US Cities)	33524	< 40000
k-means (trivial)	1.1314	< 1.1880
k-means (random)	unknown	< 2.18

Table 3.4: Benchmark settings

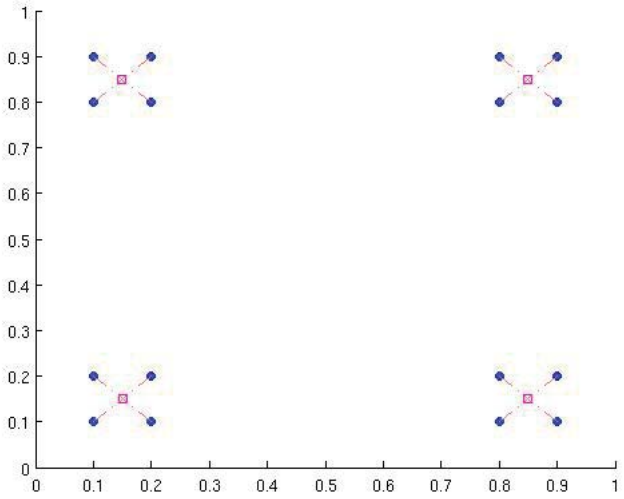


Figure 3.2: The global solution for the trivial k-means problem. Darker points are data, lighter are centroids

3.3 Setup and Analysis of Statistical Tests

The obtained datasets have to be analyzed and the results explained. To compare the efficiency of the algorithms and isolate the effect of KBS three test variables are introduced. The number of runs for each parameter setting is J, the number of parameter settings for each algorithm is R.

Estimate of the Probability of Failure. The indicator variable is defined as

$$I_{jr} = \begin{cases} 1 & \text{if } f_{Tj} \in X_\epsilon \text{ using parameter setting } r \\ 0 & \text{otherwise} \end{cases}$$

where f_{Tj} is the fitness of the best species in the population in $j^{textnormalth}$ run after a predefined number of generations (T) and X_ϵ the vicinity of the global solution defined in Table 3.4. For each parameter setting r another indicator variable is

$$F_r = \begin{cases} 1 & \text{if } \frac{\sum_{j=1}^{J} I_{jr}}{J} = 0 \\ 0 & \text{if otherwise} \end{cases}$$

Therefore, the probability of failure of algorithm A on problem P is defined as

$$P_F(A_P) = \frac{F_R}{R} = \frac{\sum_{r=1}^{R} F_r}{R} \tag{3.3}$$

Literally this means what proportion of parameter settings is completely useless (i.e. never finds A_ϵ). This value is found in the second and third columns of Table 3.5.

Estimate of the Conditional Probability of Success. Next thing one needs to know, is what algorithm performs best given that these bad parameter settings have been accounted for. Subset $R' \subset R$ is the set of all parameter settings such that $F_{r'} = 0$ for $r' \in R'$ (at least one successful run for this parameter setting). Hence the conditional probability of success is used:

$$P_S(A_P|F_{R'}) = \frac{\sum_{r'=1}^{R'} \sum_{j=1}^{J} I_{jr'}}{JR'} \tag{3.4}$$

By estimating the probability of success one gets the measure of how efficient 'good' parameter settings of the algorithm on some type of function are. These are the values in the second and third columns of Table 3.6, tested using bootstrap resampling technique. Boxplots of the conditional probabilities are on the left in Figures A.1 - A.9.

Estimate of the Conditional Expectation. Finally, one wants to know how long on the average it takes the algorithm to solve the problem. For this purpose another variable is introduced:

$$\tilde{T}_{jr'} = \min\{t : f(t)_{jr'} \in X_\epsilon\}$$

which immediately implies that $\tilde{T}_{jr'} \leq T$, the runtime of the algorithm (here $f(t)_{jr'}$ is used for the fitness of the best species at generation t on j^{th} run using r' parameter setting). From this the estimate of the runtime of the algorithm A on problem P is defined:

$$\mathbf{E}[A_P] = \frac{\sum_{r'=1}^{R'} \sum_{j=1}^{J} \tilde{T}_{jr'}}{JR'} \tag{3.5}$$

This variable are the second and third columns in Table 3.7 + bootstrap estimate of difference between algorithms with and without KBS. This parameter is of interest if one wants to find out which algorithm is the fastest on some problem. Boxplots of the estimates of conditional expectations are on the right in Figures A.1 - A.9.

Statistical Testing. The subset of algorithms that uses some form of KBS is denoted $(\mu + \lambda)\text{EA}_{KBS}$ or simply EA_{KBS}, the complement would be the subset without any KBS, $(\mu + \lambda)\text{EA}_{-KBS}$ to simply EA_{-KBS}. They are compared by using nonparametric bootstrap resampling. The main reasons for this are:

1. Violation of t-test assumptions, such as an underlying normal distribution
2. There is not enough information on how EA operators affect each other, so it can be expect that the assumption of independence are violated.
3. Unequal sample sizes, e.g. the sets of parameters where KBS=0 and KBS>0 are of size 80 and 240 respectively.

Nonparametric bootstrap sampling works the following way:

- for each function, divide the dataset into two segments: results for algorithm with KBS and without:
    1. repeat the resampling 5000 times:
        (a) from each segment S_1 and S_2 sample with replacement S_1^* and S_2^*, each size 5000
        (b) compute θ_1^* and θ_2^* for each sample
        (c) compute $d^* = \theta_1^* - \theta_2^*$
    2. find mean(d^*)-bootstrap mean and its 95% confidence interval

where θ_1^* and θ_2^* are parameters of interest for each sample(mean). If 0 is not in confidence interval, then the test is significant at 5% level of significance.

Function	$(\mu+\lambda)\text{EA}_{KBS}$	$(\mu+\lambda)\text{EA}_{-KBS}$	Bootstrap mean	95 % Bootstrap
Rosenbrock	**0**	0.2777	-0.2667	[-0.2790, -0.2542]
Rastrigin	**0.5417**	0.5750	-0.0332	[-0.0582, -0.0138]
Ackley	**0.6792**	0.6850	-0.096	[-0.1148, -0.074]
RR	**0.6850**	1	-0.3084	[-0.3212, -0.2958]
4Peaks	0.9542	**0.8875**	0.0666	[0.0562, 0.0771]
TSP (trivial)	0.9670	**0.5375**	0.4291	[0.4143, 0.4436]
TSP (US cities)	1	**0.9125**	0.0877	[0.0798, 0.0955]
k-means (trivial)	**0**	0.2500	-0.2501	[-0.2626, -0.2382]
k-means (random)	**0.0208**	0.2375	-0.2166	[-0.2292, -0.2044]

Table 3.5: Estimate of the probability of failure, Equation 3.3

Function	$(\mu+\lambda)\text{EA}_{KBS}$	$(\mu+\lambda)\text{EA}_{-KBS}$	Bootstrap mean	95 % Bootstrap CI
Rosenbrock	**0.6228**	0.3677	0.251	[0.2442, 0.2558]
Rastrigin	**0.7547**	0.3003	0.4544	[0.4452, 0.4662]
Ackley	**0.9326**	0.6281	0.3043	[0.2909, 0.3176]
RR	**0.5530**	0	0.5530	[0.5399, 0.5633]
4Peaks	0.08	**0.3111**	-0.2311	[-0.2395, -2231]
TSP (trivial)	0.03	**0.1368**	-0.1068	[-0.1090, -0.1046]
TSP (US cities)	0	**0.0268**	-0.0286	[-0.0290, -0.0282]
k-means (trivial)	0.7208	**0.8920**	-0.1713	[-0.1798, -0.1625]
k-means (random)	0.4205	**0.5169**	-0.0964	[-0.1049, -0.0878]

Table 3.6: Estimate of the conditional probability of success, Equation 3.4

Function	$(\mu+\lambda)\text{EA}_{KBS}$	$(\mu+\lambda)\text{EA}_{-KBS}$	Bootstrap mean	95 % Bootstrap CI
Rosenbrock	**309.62**	322.36	-12.6571	[-16.1195, -9.2113]
Rastrigin	**176.62**	200.40	-23.7983	[-28.5835,-18.9621]
Ackley	**170.09**	185.80	-15.6787	[-20.1500, -11.2800]
RR	**115.81**	NaN	NaN	[NaN, NaN]
4Peaks	**465.69**	480.26	-14.5716	[-15.0498,-14.0931]
TSP (trivial)	428.62	**407.56**	21.0330	[18.5931, 23.5135]
TSP (US cities)	NaN	**488.04**	NaN	[NaN, NaN]
k-means (trivial)	**459.46**	478.43	-18.9777	[-19.8190, -18.1151]
k-means (random)	**459.54**	471.95	-12.4229	[-14.0340, -10.7501]

Table 3.7: Estimate of the conditional expectation, Equation 3.5

3.3.1 Statistical Analysis

The selected test suite is quite diverse, so according to the No Free Lunch theorem, the probability that KBS performs equally well on all function is unlikely to be high. Since the results in the tables come from samples with violations of assumptions for standard statistical tests (size, distribution, independence), bootstrap resampling was used to determine the true difference between these values following the parameter-free bootstrap sampling detailed above. It is known that the empirical cumulative bootstrap distribution function $\hat{F}_n$ converges to the true CDF F almost surely: $\sup_n |\hat{F}_n - F| \to 0$ as $n \to \infty$.

Results of the bootstrap test for all three tables are presented in Figures A.10 - A.18.

In Tables 3.5 - 3.7 the columns are (left to right): test function, measured results (probability of failure, estimate of conditional probability of success, estimate of conditional expected runtime) for both EA with any form of KBS and EA without any KBS, bootstrap estimate of the mean of differences of results and 95 % confidence interval of the bootstrap estimate. All results are statistically significant at the 5 % level of significance.

The results in the tables can be interpreted in the following way (note all three parameters are mere estimates of the corresponding random variables):

Table 3.5 (Probability of failure): For all functions except TSP and 4Peaks an algorithm that uses some form of KBS (and anything else) is less likely to fail (i.e. never find a solution) than algorithms that do not use any form of KBS. The results are statistically significant, since 95% bootstrap confidence interval does not include 0. In fact on the trivial k-means clustering and Rosenbrock functions EA with KBS never fails at all, but this is counter-balanced by very high failure rates on Four Peaks and TSP. EA_{KBS} outperforms EA_{-KBS} strongly on Royal Roads (never solved by EA_{-KBS}), Rosenbrock and both instances of the k-means clustering problem (>20 percentage points fewer useless parameter settings)

Table 3.6 (Conditional probability of success): EA_{KBS} has an overwhelming probability of finding the solution on Royal Roads, which other algorithms fail at all. At the same time, in addition to 4Peaks and TSP, conditional probability of success using KBS is lower than without KBS in both k-means problems, especially for the trivial one. Nevertheless, many parameter settings of EA_{KBS} have a very high success rate (close to 1).

Table 3.7 (Conditional expectation): This last estimate demonstrates the best property of KBS so far: fast convergence. On all functions except TSP EA_{KBS} outperformed EA_{-KBS}. The advantage is especially clear on functions like Rastrigin and trivial k-means clustering. In combination with the results for the other two estimates, this gives very good information for applying KBS in real life, by selecting the set of parameters that both yield a high probability of success and competitive runtime.

3.4 Conclusions

In this chapter a number of important concepts was introduced that will be extensively used in the rest of the thesis (KBS, algorithms and test functions). A

large number of computational experiments showed that an algorithm using KBS has an advantage over other algorithms on many problems. Among other things, this allows one to select the optimal set of parameters, both for faster and more likely convergence, in the appropriate situation.

Using the terminology introduced here, the analysis will be focused on $\mathbf{E}[A_p]$, i.e. conditional expectation, albeit for a specific parameter setting. This estimate can be treated as a mean first hitting time in an absorbing MC, where the absorbing state is the artificial fitness level of the global solution (i.e. all populations that contain at least one species with the best solution).

Analysis will be focused on the algorithm that uses some form of KBS compared to the algorithm with some form of RLS. Test problems are Royal Roads and OneMax (not covered in this chapter, but explained in detail in Section 4.4.1).

Chapter 4

Lower Bounds on the Runtime

In this chapter the first attempt is made to analyze an EA with population and recombination pool by considering the population structure. The population is divided into two subsets, elite species and the rest of the population with an addition of a simple assumption about the distribution of elite species (probability to observe a certain number of them). The approach is applied to two algorithms solving two test problems. The result is the derivation of lower bounds on the runtime.

4.1 Main results

Some of the main results of this chapter are:

1. Lower bound of $(\mu + \lambda)\text{EA}_{1BS}$ on the OneMax test function is $\Omega\Big(\frac{n \log n}{\lambda}\Big)$,
2. Lower bound of $(\mu+\lambda)$RLS on the OneMax test function is $\Omega\Big(\frac{n \log n}{\lambda}\Big)$, which coincides with the previous result,
3. Lower bound of $(\mu+\lambda)\text{EA}_{1BS}$ on the Royal Roads test function with K bins length M each is $\Omega\Big(\frac{n^2 \log M \log(\frac{KM}{K+M})}{\lambda M}\Big)$
4. Lower bound of $(\mu + \lambda)$RLS on the Royal Roads test function with K bins length M each is $\Omega\Big(\frac{nK \log M}{\lambda}\Big)$

Results for OneMax confirm many previous findings in the EA community, and the results for the Royal Roads are an improvement.

4.2 Structure of the population and the recombination pool

Before going into detail on the approach in this chapter, a few words have to be said about population-based EAs and why they have seen less attention in EA community than they deserve. A brief history of $(\mu + \lambda)$EAs' analysis was given in Section 2.3.2. Here, certain important properties of these algorithms will be discussed.

It may seem strange that $(\mu + \lambda)$ algorithms (both μ and $\lambda > 1$), despite their widespread application in real life, are less popular in EA theory than $(1+1)$. The main reason is the complexity of the evolutionary process arising in connection with them. The main reason of this complexity is the structure of the population. Throughout the run of the algorithm both population and recombination pool consist of different types of species with different fitnesses. If a genetic operator that recombines information between parents is used (e.g. KBS), then pairs of parents have to be considered rather than single species. Quite obviously, the structure and prevalence of certain types of species in the population affect greatly the breeding process, and therefore, the probability of evolving a higher-ranked offspring.

It makes sense to give the overview of complexity arising from the population structure of $(\mu + \lambda)$EAs:

1. Distribution of species of different types in the population. This is probably the most obvious consequence of using $\mu > 1$, and one that is extensively applied in Sections 4.5- 4.8 and Chapter 5. Regardless of the problem considered (even with traps and/or local minima), species with different fitness do not have equal representation in the population. Among other things, it is important to distinguish species with the currently best fitness (elite) and the rest, especially for functions with plateaus. To the best of my knowledge, this

quite obvious and important feature has never attracted much interest in EA community, most likely due to the complexity of the dynamics of the structure of these subsets. This feature though is certainly well-known and widely studied in areas such as biology and epidemiology, see e.g. [Nas96, Nas99].

2. Fitness-proportional selection. Selection is one of the most important features of EAs. Obviously, it defines the structure of the recombination pool, and depends on the structure of the population. If a certain type of species dominates the population, there is a high probability that it would dominate the recombination pool too. At the same time, currently best (or next-best) species, even though they need not be abundant in the population, have a high probability of expanding their presence in the recombination pool and generate a higher-ranked offspring. This issue was touched upon in [CHS$^+$09], where the number of elite species in the recombination pool was found to exceed that in the population by at least 25%, although the selection process for any selection type is simplified to the proportion of LOI in the population and ignoring the upgrade of non-LOI to LOI, see also Section 2.3.2 in this thesis. One can also suspect that different selection functions may lead to different structures of recombination pools.

3. Pairing of parents. This point is valid for algorithms with recombination operators, e.g. crossover or KBS. The number of types of pairs greatly affects the structure of offsprings and the probability of evolving a higher-ranked one. If the algorithm uses recombination pool of size λ, there are $\frac{\lambda}{2}$ pairs of parents. Even for a simple case when the population has only two types of species, α and β, there are three types of pairs: $<\alpha,\alpha>, <\alpha,\beta>$ or $<\beta,\alpha>$ and $<\beta,\beta>$, each with its own properties and evolution probabilities that can differ to a greater degree than an order of a constant. For example, it can be $p_1 = \frac{1}{n}(1-\frac{1}{n}) = O(\frac{1}{n})$ and $p_2 = \frac{1}{n^2} = O(\frac{1}{n^2})$. Even in this simple case, using multinomial coefficients there are $\sum_{j=0}^{\frac{\lambda}{2}} \sum_{r=0}^{\frac{\lambda}{2}-j} \binom{\frac{\lambda}{2}}{j}\binom{\frac{\lambda}{2}-j}{r} = 3^{\frac{\lambda}{2}}$ possible combinations of pairs of parents in the recombination pool.

4. Exchange of genetic information between parents in the recombination pool. This point is also valid for algorithms with recombination operators. The majority of publications that consider populations with crossover (e.g. [OHY08]) find a simple bound on the probability of improvement using crossover (i.e. $\frac{i}{n} > \frac{1}{n}$). It is sensible to analyze crossover's capacity if different types of parents exchange genetic information (hence different probability of improvement). Even if this analysis does not improve the asymptotic runtime result, it may shed light on the working of the recombination process and the power behind EA, since it is hypothesized (see e.g. [Gol89, Mit96]) that EAs' efficiency is vastly boosted by recombination rather than mutation.

5. Rate of elitism. In [HY02, CHS+09] out of $\mu + \lambda$ population and recombination pool μ best species are selected to form a new population at the end of each generation. Nevertheless, in many applications different rates and/or types of elitism are used: saving only 1 best species, replace the rest with offsprings, or save a certain proportion of the best (not necessarily currently best), replace the rest with offsprings. There are very few theoretical investigations in how this affects the efficiency of the algorithm.

Most of these questions are usually either avoided or oversimplified in EA theoretical community. In the rest of this chapter a new approach is developed that takes on some of these issues. It is applied to four different cases, each also solved asymptotically. In the next chapter a whole new tool is developed and used to find upper bounds on the runtime of population-based EAs.

4.3 Algorithms

Although many findings here can be extended to similar algorithms, the analysis is restricted to just two: $(\mu+\lambda)\text{EA}_{1BS}$, i.e. an elitist EA with 1-Bit-Swap operator (see Table 4.1 and $(\mu+\lambda)$RLS, an elitist EA with Randomized Local Search(RLS), a form of stochastic hill-climber (see Table 4.2). The main difference from $\frac{1}{n}$ mutation is that RLS flips exactly 1 bit per string, not each bit with probability $\frac{1}{n}$. The KBS operator was introduced in [TSMH10] and described in great detail

in the previous chapter. Both algorithms use the same variant of Tournament selection function (see Table 4.3), and no other genetic operators (e.g. crossover, mutation).

1	create μ starting species at random
	loop until solution is found
2	select using a variant of fitness-proportional Tournament selection $\frac{\lambda}{2}$ pairs of parents into the pool
3	swap a pair of bits between each pair of parents
4	keep currently best species in the population, delete the same number of non-elite offsprings, replace the rest of the population with the remainder of the pool
	end loop

Table 4.1: $(\mu + \lambda)\text{EA}_{1BS}$

4.4 Problems

Two problems are analyzed in detail in this Chapter: OneMax and Royal Roads (RR)

1	create μ starting species at random
	loop until solution is found
2	select using a variant of fitness-proportional Tournament selection λ parents into the pool
3	flip exactly one bit in each parent (RLS)
4	keep currently best species in the population, delete the same number of non-elite offsprings, replace the rest of the population with the remainder of the pool
	end loop

Table 4.2: $(\mu + \lambda)$RLS

	loop over λ
1	select two candidates from the population at random
2	examine their fitness, place the better one in the pool
	discard the other one
	end loop

Table 4.3: Selection Function

4.4.1 OneMax

This is one of the simplest and frequently analyzed problems in EA community, also known as Counting Ones:

$$f(x) = \sum_{i=1}^{n} x_i$$
$$x_i \in \{0, 1\}$$

Although this is an easy problem with an obvious global solution n (all bits set to 1), it is of interest to compare the mathematical model of an algorithm with 1BS to many existing ones with crossover and mutation.

A whole range of bounds exist for various algorithms solving OneMax. Some of the best known are $\Theta(n \log n)$ for $(1+1)$EA with mutation, $O(n \log n + n\mu)$ for $(\mu+1)$EA with mutation in [Wit04], $O(n \log n + n\lambda)$ for $(1+\lambda)$EA in [JDJW05] and $O(n \log n + nN \log N)$ for $(N+N)$EA in [CHS+09] (all of these are measured in the number of function evaluations).

4.4.2 Royal Roads

This test function was presented in [MFH92] and analyzed in [Mit96] for comparison of EA to stochastic hill-climbers (e.g. Random-mutation hill climbing, also known as Randomized Local Search, RLS, considered in this thesis). For this

research it is important as an example of a function with plateaus of fitness.

$$f(x) = \sum_{k=0}^{K-1} c_k \prod_{j=0}^{M-1} x_{kj}$$
$$x_{kj} \in \{0,1\}, c_k = M \ \forall \ k$$

That is, a string is divided into K consecutive and disjoint subsets (bins) each of length M. Unless all bits in the bin are set to 1, its fitness is 0. Otherwise it 'jumps' to c_k. Since $c_k = M \ \forall \ k$ obviously $KM = n$ and the maximum fitness of the string is n. In Section 4.8 k^{th} bin is denoted as s_k or, for simplicity, k, and the whole string s.

There are two main reasons to study the RR function:

1. Complexity of the function, which comes from the fact that a plateau of fitness has to be overcome, i.e. selection function does not have an incentive to select species that are closer to the next fitness plateau. Hence the ability of algorithms to overcome plateaus can be tested,

2. Potential applications in real life, such as a model of DNA structure

Such functions were analyzed already in [Mit96] and later in [JW01, Sud08a] and some other and a $(1+1)$EA was found to have $\Theta(n^3)$ runtime. A similar LeadingOnes problem is solved by $(N+N)$EA in $O(n^2 + nN \log N)$ function evaluations (see [CHS+09]). It is of interest to see if population-powered EAs using 1BS can improve these bounds. The earliest result, in [Mit96] gives a bound of $O(2^M \log K)$, which does not use population size, although it is clear that population greatly affects the efficiency of the algorithm. Moreover this result would be quite loose, e.g. if $K = M = \sqrt{n}$ the runtime is $O(2^{\sqrt{n}} \log n)$ or if $M = 1, K = n$, then runtime is $O(\log n)$, which is impossible.

Also statistical dynamics of EA (evolutionary paths) was studied in [vNCM99, vN00] (does not involve runtime analysis).

4.5 Population-Based Evolutionary Algorithms and Distribution of Species

As discussed earlier, distribution of species with different properties in the population is a large and important area of research in biological sciences. So far in the EA community there has been almost no discussion of this, although quite clearly derivation of such distributions would contribute greatly to the understanding of the working of population-based EA.

In this Chapter the first attempt is made to at least assume (for now without any proof) that species in the population follow some distribution and use this assumption to derive lower bounds on the runtime. Since this feature is common to both algorithms, it is described here in greater detail.

Similar to the fitness-levels partition (see Section 2.4.1), a population of binary-encoded chromosomes (also referred to as species) is broken down into disjoint subsets based on their fitness. In case there are n levels of fitness and each level k has A_k representatives in the population, the total population can be represented as a union of these disjoint subsets: $\mu = A_1 \cup A_2 \cup \ldots \cup A_n$. Quite obviously, structures arising in the population dynamics of EA are very complicated, so it is necessary to approximate them by focusing on just a few subsets. Specifically, one distinguishes a subset α, which includes all currently best (elite) strings in the population. Next-best (species with the next-best fitness) are β, and the rest of the population are γ.

In this chapter the analysis is restricted to dynamics and evolution of α species, ignoring other subsets of the population. This serves the purpose of analyzing the two algorithms in question. They both use an operator that improves at most one bit in any parent, so the quality of the population cannot improve by more than 1 level of fitness each generation. This means lower-ranked species cannot 'jump' an evolution ladder to evolve better offsprings. Later in the chapter this idea is extended to account for plateaus of fitness and in the next chapter to include lower-ranked species, β and γ.

Since one needs to sample at least one α species into the recombination pool to have a non-zero probability of evolution, it is necessary to say something about the distribution of α species in the population. For various reasons: simplicity, expectation of $\frac{\mu-1}{2}$ species, empirical distribution of elite species in the population, it is assumed that this distribution is approximated by the Uniform distribution. For the verification of the empirical point see Figures 4.1, 4.2, 4.8, 4.9. The author argues that this assumption gives upper bound on the true probability to observe a 'large' number of elite species in the population, hence the resulting mean first hitting time is the lower bound on the true runtime of the algorithm.

There are certain similarities in the working and analysis of both algorithms that are pointed out here:

1. 1-Bit-Swap recombines exactly one bit from each parent, RLS flips exactly one bit in each parent

2. Analysis is restricted to elitist species in the population

3. Population size μ and recombination pool size λ are arbitrary unless specified (e.g. $\mu = \lambda$ or $\mu = 1$)

4. Rate of elitism is not specified: all elite species are saved, new elite offsprings are added from the recombination pool and the rest of the population is replaced with offsprings selected randomly.

5. Elite species in the population are assumed to follow Uniform distribution

6. The only variable analyzed in this Chapter is the expected runtime, e.g. the expected number of generations/function evaluations until the algorithm

achieves the highest artificial fitness level.

7. For the OneMax test function, the probability to generate a higher-ranked offspring serves as a parameter in the Geometric rv.

8. Expectations of the Geometric rv is the expected time until the event of evolving a higher-ranked offspring

9. The sum of independent (but not identically distributed, therefore this sum is not Negative Binomial) Geometric random variables is therefore the expected first hitting time of the whole algorithm

10. For the Royal Roads test function an auxiliary function tracking progress between fitness plateaus is used (OneMax for each plateau, see Section 4.8)

The main models are presented in the relevant sections, and later complemented with further analytical and numerical derivations. This allows one to make a very exact comparison between these two algorithms and find which one is a better optimizer. These findings are supplemented with numerical results.

The author starts though with a simple example that allows to present the efficiency of both tools used extensively later: Geometric rvs and Markov chains.

4.6 Runtime analysis of $(1+2)EA_{1BS}$ solving OneMax Problem

This is the simplest instance of $(\mu + \lambda)\text{EA}_{1BS}$ with $\mu = 1,\ \lambda = 2$. Since it is impossible have more than one 1-bit improvement each generation, the Coupon Collector's problem is directly applicable in this case. Pessimistically, assuming at

the beginning there is only one 1-bit in the string, the probability and the expected time of improving from k to $k+1$ bits are

$$p_{k,k+1} = \frac{2k}{n}\left(1 - \frac{k}{n}\right)$$
$$\mathbf{E}T_k = \frac{1}{p_{k,k+1}} = \frac{n^2}{2k(n-k)}$$

This comes from the fact that in order to improve fitness the algorithm must swap a 1-bit from any of two parents and 0-bit from the other one. The expected first hitting time of the algorithm on OneMax problem is

$$\begin{aligned}\mathbf{E}\tau_{(1+2)EA_{1BS}} &= \sum_{k=1}^{n-1} \mathbf{E}T_k = \frac{n^2}{2}\sum_{k=1}^{n-1}\frac{1}{k(n-k)} = \frac{n^2}{2}\cdot\frac{1}{n}\left(\sum_{k=1}^{n-1}\frac{1}{k} + \sum_{k=1}^{n-1}\frac{1}{n-k}\right) \\ &= \frac{n}{2}\cdot 2H_{n-1} = O(n\log n)\end{aligned}$$

The last step included expansion in partial fractions and approximation of n-th Harmonic sum: $\log n < H_n = \sum_{k=1}^{n}\frac{1}{k} < \log n + 1$. The solution is asymptotic in n, see Section 2.5

It is quite interesting that the same result can be obtained using the set of recurrent equations for finding the mean first hitting time in a Markov Chain. The finite set of states is of cardinality n of which all are transient (due to the elitism), except the last state s_n, which is absorbing. Also, transition is possible only from state s_k to the adjacent state s_{k+1}. This is essentially a pure birth Markov Chain. These properties follow directly from the definition of $(1+2)\mathrm{EA}_{1BS}$ with elitism. What one needs to derive is m_{1A}, i.e. the expected time of achieving the set of all populations that contain at least one chromosome with the global solution if the starting string has only one 1-bit. The probability of going one state up in the MC is the same as $p_{k,k+1}$ defined above.

It is quite easy to see that in fact, by the Kolmogorov-Chapman equations,

$$h_{k,n} = \sum_{A} p_{kA}h_{An} = 1$$

that is, the global optimum is achievable from any state with probability 1. Here A is the set of all states that do not include both the starting state and the global optimum, n is obviously the global optimum, $h_{k,n}$ is the probability to reach the state n eventually, starting from state k.

As mentioned above, due to the elitism property, this is a pure birth process, and the expected first hitting time for the state $k+1$ from the state k is

$$m_{k,k+1} = 1 + (1 - p_{k,k+1})m_{k,k+1} = \frac{1}{p_{k,k+1}}$$

Summing over all k, the expected first hitting time starting in state 1 is (given the boundary condition $m_{n,n} = 0$).

$$m_{1,n} = \sum_{k=1}^{n-1} m_{k,k+1} = \sum_{k=1}^{n-1} \frac{1}{p_{k,k+1}}$$

and, substituting in the expression for $p_{k,k+1}$:

$$m_{1,n} == \mathbf{E}\tau_{(1+2)EA_{1BS}} = \frac{n^2}{2} \sum_{k=1}^{n-1} \frac{1}{k(n-k)} = \frac{n^2}{2} \cdot \frac{1}{n} \left(\sum_{k=0}^{n-1} \frac{1}{k} + \sum_{k=0}^{n-1} \frac{1}{n-k} \right) = O(n \log n)$$

In fact, since $H_n = \Theta(\log n)$, $\mathbf{E}\tau_{(1+2)EA_{1BS}} = \Theta(n \log n)$, i.e. it is both upper- and lower bounded by $n \log n$ up to some constant. In the next section these ideas are extended to a more complicated case with both $\mu > 1$ and $\lambda > 2$.

4.7 Main model of the $(\mu + \lambda)$ Algorithm on the OneMax Test Function

The main model for both algorithms exhibits a number of similarities. The probability used in this model is the probability to evolve at least one higher-ranked offspring, which is the parameter in the Geometric random variable. It is defined as

1 - Probability of failing to evolve such an offspring (i.e. evolve 0 better offsprings):

$$P(G_k) = 1 - P(G_{0k})$$

where k is the number of improvements (so far). The expected runtime of the algorithm (measured in the number of generations) is found by summing over the total number of these improvements:

$$\mathbf{E}\tau = \sum_{k=1}^{n-1} \mathbf{E}T_k = \sum_{k=1}^{n-1} \frac{1}{P(G_k)} = \sum_{k=1}^{n-1} \frac{1}{1 - P(G_{0k})}$$

To find the main expression for the probability of failure $P(G_{0k})$, the law of total probability is used twice. First, one conditions on the number of occurrences of elite species in the recombination pool, which cannot be larger than λ. Second, one conditions on the number of elite species in the population, which is anywhere between 1 and μ:

$$P(G_{0k}) = \sum_{j=0}^{\lambda} P(G_{0k}|H_j)P(H_j) = \sum_{j=0}^{\lambda} P(G_{0k}|H_j) \sum_{\alpha=1}^{\mu} P(H_j|\alpha)P(\alpha) \tag{4.1}$$

$P(\alpha)$ is the probability to observe α elite species in the population (Uniform), $P(H_j)$ is the probability to obtain j elite parents in the recombination pool using a variant of tournament selection function. In the remainder of the Section, this model is applied to both algorithms. Also α denotes both elite species and their number, $|\alpha|$.

4.7.1 Runtime analysis of $(\mu + \lambda)EA_{1BS}$ on the OneMax problem

Since 1-Bit-Swap recombines information between two parents to evolve new offsprings, one needs to find the probability of selecting an elite pair $< \alpha, \alpha >$ into the recombination pool given there are α elite species in the population:

$$P_{sel,\alpha} = \Big(\frac{\alpha}{\mu} \cdot \frac{\alpha}{\mu} + \frac{\alpha}{\mu}\Big(1 - \frac{\alpha}{\mu}\Big)\Big)^2 = \Big(\frac{\alpha}{\mu}\Big)^2 \tag{4.2}$$

This comes from the fact that if an α candidate parent is paired with any lower-ranked candidate, it always wins, and if the other candidate is also α, either enters the pool. This is not to be confused with $P(\alpha)$, i.e. the probability to *observe* exactly α elite species in the population. Since the analysis is restricted only to elite pairs, the probability of evolution (generation of a better offspring as a result of 1-Bit-Swap) is

$$P_{swap} = \binom{2}{1}\frac{k}{n}\left(1-\frac{k}{n}\right) \tag{4.3}$$

since the algorithm needs to select a 0-bit in either parent and a 1-bit in the other one, with $k = 0 : n-1$. This is due to the pessimistic assumption that at the start of the algorithm the fitness of the best string is just 1.

Of interest is the probability of evolving at least 1 new elite offspring, i.e. of at least 1 successful swap in any elite pair. From the main model one obtains:

$$P(\text{at least 1 new elite offsping in the population at } t+1)$$

$$= 1 - P(\text{no new elite offspring in the population at } t+1)$$

The first expression is for convenience referred to as the 'probability of success', and the second one as the 'probability of failure'.

By assumption, elite species are distributed uniformly: $P(\alpha) = \frac{1}{\mu}$. The total number of elite pairs in the recombination pool is at most $\frac{\lambda}{2}$. The probability of failure given zero elite pair is

$$\begin{aligned} P(G_0|H_0)P(H_0) &= P(G_0|H_0)\sum_{\alpha=1}^{\mu} P(H_0|\alpha)P(\alpha) = \frac{1}{\mu}\sum_{\alpha=1}^{\mu} P(H_0|\alpha) \\ &= \frac{1}{\mu}\left(\left(1-P_{sel,1}\right)^{\frac{\lambda}{2}} + \left(1-P_{sel,2}\right)^{\frac{\lambda}{2}} + \ldots + \left(1-P_{sel,\mu}\right)^{\frac{\lambda}{2}}\right) \\ &= \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-P_{sel,\alpha}\right)^{\frac{\lambda}{2}} \end{aligned}$$

$P(G_0|H_0)$ is obviously equal to 1, since 0 elite pairs in the pool excludes any probability of evolution. Along these lines, to fail evolution given 1 elite pair in the recombination pool:

$$P(G_0|H_1)P(H_1) = \left(1 - \frac{2k}{n}\left(1 - \frac{k}{n}\right)\right)\frac{1}{\mu}\sum_{\alpha=1}^{\mu} P(H_1|\alpha)$$

and

$$\sum_{\alpha=1}^{\mu} P(H_1|\alpha) = \binom{\frac{\lambda}{2}}{1}\sum_{\alpha=1}^{\mu} P_{sel,\alpha}\Big(1 - P_{sel,\alpha}\Big)^{\frac{\lambda}{2}-1}$$

therefore, the probability of failure given 1 elite pair given k improvements so far is

$$P(G_{0k}|H_1)P(H_1) = \left(1 - \frac{2k}{n}\left(1 - \frac{k}{n}\right)\right)\binom{\frac{\lambda}{2}}{1}\frac{1}{\mu}\sum_{\alpha=1}^{\mu} P_{sel,\alpha}\Big(1 - P_{sel,\alpha}\Big)^{\frac{\lambda}{2}-1}$$

For cases $\{H_j : 2 \leq j \leq \frac{\lambda}{2}\}$ the logic is similar, so the full expression for the probability of failure is

$$\begin{aligned}
P(G_{0k}) &= \sum_{j=0}^{\frac{\lambda}{2}} P(G_0|H_j)P(H_j) \\
&= \frac{1}{\mu}\sum_{j=0}^{\frac{\lambda}{2}} \left(1 - \frac{2k}{n}\left(1 - \frac{k}{n}\right)\right)^j \binom{\frac{\lambda}{2}}{j} \sum_{\alpha=1}^{\mu} P_{sel}^j(\alpha)\Big(1 - P_{sel,alpha}\Big)^{\frac{\lambda}{2}-j} \\
&= \frac{1}{\mu}\sum_{j=0}^{\frac{\lambda}{2}} \left(1 - \frac{2k}{n}\left(1 - \frac{k}{n}\right)\right)^j \binom{\frac{\lambda}{2}}{j} \sum_{\alpha=1}^{\mu} \left(\frac{\alpha}{\mu}\right)^{2j}\left(1 - \left(\frac{\alpha}{\mu}\right)^2\right)^{\frac{\lambda}{2}-j}
\end{aligned}$$

Interchanging the sums and using the binomial identity $(s+t)^n = \sum_{k=0}^{n} \binom{n}{k} s^k t^{n-k}$, one gets:

$$\begin{aligned} P(G_{0k}) &= \frac{1}{\mu} \sum_{\alpha=1}^{\mu} \sum_{j=0}^{\frac{\lambda}{2}} \binom{\frac{\lambda}{2}}{j} \left(\left(1 - \frac{2k}{n}\left(1 - \frac{k}{n}\right)\right)\left(\frac{\alpha}{\mu}\right)^2 \right)^j \left(1 - \left(\frac{\alpha}{\mu}\right)^2\right)^{\frac{\lambda}{2}-j} \\ &= \frac{1}{\mu} \sum_{\alpha=1}^{\mu} \left(1 - \left(\frac{\alpha}{\mu}\right)^2 P_{swap}\right)^{\frac{\lambda}{2}} \end{aligned}$$

The probability of success is therefore

$$P(G_k) = 1 - P(G_{0k}) = 1 - \frac{1}{\mu} \sum_{\alpha=1}^{\mu} \left(1 - \left(\frac{\alpha}{\mu}\right)^2 P_{swap}\right)^{\frac{\lambda}{2}}$$

For each $1 \leq k \leq n-1$:

$$\mathbf{E}T_k = \frac{1}{1 - \frac{1}{\mu} \sum_{\alpha=1}^{\mu} (1 - (\frac{\alpha}{\mu})^2 P_{swap})^{\frac{\lambda}{2}}}$$

and therefore the expected first hitting time for the algorithm is

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \sum_{k=0}^{n-1} \mathbf{E}T_k = \mu \sum_{k=0}^{n-1} \frac{1}{\mu - \sum_{\alpha=1}^{\mu} (1 - (\frac{\alpha}{\mu})^2 P_{swap})^{\frac{\lambda}{2}}} \tag{4.4}$$

Unfortunately, this quantity does not seem to exist in a closed form. Therefore, the asymptotic approximation is derived and checked numerically by comparing Equation 4.4 to computational experiments for different values of μ, λ, n.

4.7.2 Asymptotic runtime of $(\mu+\lambda)\mathbf{EA}_{1BS}$ on the OneMax Test function

The author starts with approximating the probability of failure:

$$P(G_{0k}) = \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\left(\frac{\alpha}{\mu}\right)^2 P_{swap}\right)^{\frac{\lambda}{2}} \geq \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\lambda}{2}\left(\frac{\alpha}{\mu}\right)^2 P_{swap}\right)$$
$$= 1-\sum_{\alpha=1}^{\mu}\frac{\lambda}{2}\frac{\alpha^2}{\mu^3}P_{swap} > 1-\frac{\lambda P_{swap}}{2}$$

The last step comes from the well-known identity:

$$\sum_{\alpha=1}^{\mu}\alpha^2 = \frac{\mu(2\mu+1)(\mu+1)}{6} \leq \mu^3$$

since $P(G_k) = 1 - P(G_{0k})$ one can easily get the lower bound on the expression for the whole algorithm:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} \geq \frac{2}{\lambda}\sum_{k=1}^{n-1}\frac{1}{P_{swap}} = \frac{2n^2}{\lambda}\sum_{k=1}^{n-1}\frac{1}{2k(n-k)} = \frac{2n^2}{\lambda}\cdot\frac{1}{n}\left(\sum_{k=1}^{n-1}\frac{1}{k}+\frac{1}{2}\cdot\sum_{k=1}^{n-1}\frac{1}{n-k}\right)$$
$$> \frac{3n\log(n-1)}{\lambda}$$

The third step is due to the partial fraction expansion. The last step is due to the bounds on the Harmonic sum. Therefore, the lower asymptotic bound on runtime of this algorithm is

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \Omega\left(\frac{n\log n}{\lambda}\right) \tag{4.5}$$

or, if measured in the number of function evaluations,

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \Omega(n\log n) \tag{4.6}$$

which is a well-known runtime bound on the OneMax function. In other words, if elite species in the population are distributed uniformly, expected runtime of $(\mu+\lambda)\mathrm{EA}_{1BS}$ is asymptotically the same as any mainstream $(1+1)$ Evolutionary Algorithm and gets improved if run on parallel computers.

4.7.3 Runtime analysis of $(\mu+\lambda)$RLS on the OneMax Test Function

For the comparison, $\mathbf{E}\tau$ for $(\mu+\lambda)$RLS using a similar approach (law of total probability + sum of independent Geometric rvs) is derived. Results will be compared to Equation 4.4. Changes apply mostly to the selection and flipping probabilities, as there are no pairs to form:

$$P_{sel,\alpha} = \frac{\alpha}{\mu}\left(\frac{\alpha}{\mu} + 1 - \frac{\alpha}{\mu}\right) = \frac{\alpha}{\mu}$$

This comes from the fact that regardless of what candidate is selected by the tournament selection function, an α species enters the pool. The probability to flip the correct bit is the same as for any other RLS-type algorithm:

$$P_{flip} = 1 - \frac{k}{n}$$

The same assumptions of uniform distribution of elite species in the population are used, as with the $(\mu+\lambda)\text{EA}_{1BS}$.

Failure event G_0 is defined in the same way: no successful flips in the recombination pool, so the probability thereof is defined in a similar way to the one in Equation 4.1.

$$P(G_0) = \sum_{j=0}^{\lambda} P(G_0|H_j) \sum_{\alpha=1}^{\mu} P(H_j|\alpha)P(\alpha) \tag{4.7}$$

Only in this case, of course, j is the number of elite parents in the pool and goes from 0 to λ.

$$P(H_j|\alpha)P(\alpha) = \binom{\lambda}{j} P_{sel,\alpha}^{j}(1 - P_{sel,\alpha})^{\lambda-j}$$

and therefore (using the same idea with the binomial theorem)

$$P(G_{0k}) = \frac{1}{\mu}\sum_{j=0}^{\lambda}\left(\frac{k}{n}\right)^j\binom{\lambda}{j}\sum_{\alpha=1}^{\mu}P_{sel,\alpha}^j(1-P_{sel,\alpha})^{\lambda-j} = \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\sum_{j=0}^{\lambda}\binom{\lambda}{j}\left(\frac{k}{n}P_{sel,\alpha}\right)^j(1-P_{sel,\alpha})^{\lambda-j}$$
$$= \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\alpha}{\mu}P_{flip}\right)^{\lambda}$$

Unfortunately, the closed expression for this sum does not seem to exist either, so one remains with the following expression for the expected runtime of the algorithm:

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} \geq \sum_{k=1}^{n-1}\mathbf{E}T_k = \mu\sum_{k=1}^{n-1}\frac{1}{\mu-\sum_{\alpha=1}^{\mu}(1-\frac{\alpha}{\mu}(1-\frac{k}{n}))^{\lambda}} \tag{4.8}$$

Just as with the previous algorithm, asymptotic approximation is derived and this expression is compared to the numerical results in Section 4.9

4.7.4 Asymptotic runtime of $(\mu+\lambda)$RLS on the OneMax Test function

It is quite straightforward to find the lower bound on $P(G_{0k})$:

$$P(G_{0k}) = \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\alpha}{\mu}\left(1-\frac{k}{n}\right)\right)^{\lambda} \geq \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\lambda\alpha}{\mu}\left(1-\frac{k}{n}\right)\right) = 1-\lambda\left(1-\frac{k}{n}\right)$$

where once again Bernoulli inequality was found useful for the lower bound on the expression. Therefore the probability of success is

$$P(G_k) \leq \lambda\left(1-\frac{k}{n}\right)$$

and, to find the expected runtime of the whole algorithm one needs to sum expectations of Geometric rvs:

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} \geq \sum_{k=1}^{n-1}\frac{1}{G_k} = \frac{n}{\lambda}\sum_{k=1}^{n-1}\frac{1}{n-k} \geq \frac{n\log(n-1)}{\lambda}$$

In other words, the lower bound on expected runtime is asymptotically

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} = \Omega\left(\frac{n \log n}{\lambda}\right) \tag{4.9}$$

or, measured in number of function evaluations,

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} = \Omega(n \log n) \tag{4.10}$$

which is the same as the lower bound for $(\mu + \lambda)\text{EA}_{1BS}$. This means that asymptotically these two algorithms perform the same (with the benefit from parallelization).

4.8 Main model of the $(\mu + \lambda)$ Algorithm on the Royal Roads Test Function

The setup for the RR problem is along the lines of [Mit96] (referred to as R_1 in the book). If the length of the chromosome is n, it is split into K consecutive disjoint subsets (also referred to as segments or bins), length of each bin is M (so that $n = KM$). Originally this problem was designed to test EA's capacity for recombining building blocks compared to other heuristics (for details see [Mit96]). It can also be seen as EA's capacity to traverse the fitness plateau. For the details of RR see Section 4.4.2.

In [JW01] efficiency of two different algorithms on functions with plateaus of constant fitness was compared and it was proven that for some functions an algorithm that accepts a new solution if it is at least as good (same fitness) can outperform the one that accepts only better solutions. Here only the second case is considered (success is defined as improving the auxiliary function, see below).

An auxiliary function is introduced to track progress between improvements in the fitness (the idea is similar to that in, e.g. [DJW10a, HY04]), which in this case is $V(s_k) = V_k = \text{OneMax}(s_k)$ since both functions achieve have $\max V(s_k) =$

$\max f(s_k) = M$ and $\max f(s) = \max V_s = \max \sum_k V_k = n$.

The motivation behind this auxiliary function is quite obvious: it is necessary to track progress of the algorithms between jumps of the fitness value. OneMax is a suitable candidate for this purpose as the number of 1-bits in the bin gives a good measure on the distance to the next fitness level. Therefore, in addition to artificial fitness levels one can also use artificial auxiliary levels, i.e. all populations that have at least one species with the auxiliary value V_k.

There is one important feature of the evolutionary process to see here: when parents exchange genetic information, it doesn't matter where the information comes from (which segment of the parent). What matters, is where it is inserted, because it may mean that the fitness of the recipient segment has reached M, and therefore the fitness of the offspring has increased.

Since both 1BS and RLS bins swap/flip exactly 1 bit, bins in the string evolve in an arbitrary sequence (i.e. there are $M!K!$ ways of evolution, since it does not matter, in which order bits in the bin and bins in the string evolve). This directly means that no two different bins can evolve simultaneously. It is also pessimistically assumed that the best auxiliary function value in the first generation is 1 (and hence the auxiliary value of the whole string is K) and the fitness function is 0. The bin that is currently being 'processed' is referred to as 'active'. Again, the assumption about the distribution of elite species in the population is made (Uniform).

4.8.1 Runtime analysis of $(\mu + \lambda)\mathbf{EA}_{1BS}$ on the RR Test Function

As with OneMax, the author starts with introducing the probability of failure:

$$P(G_0) = \sum_{j=0}^{\frac{\lambda}{2}} P(G_0|H_j) \sum_{\alpha=1}^{\mu} P(H_j|\alpha)P(\alpha) \tag{4.11}$$

where all variables are the same as in $(\mu+\lambda)\mathrm{EA}_{1BS}$ solving OneMax: H_j is j^{th} elite pair in the recombination pool λ, α is the number of elite species in the population μ with both highest fitness and auxiliary function values. The selection function is the same as Equation 4.2:

$$P_{sel,\alpha} = \left(\frac{\alpha}{\mu}\right)^2$$

Having selected the pair, the probability that as the result of swapping bits between them, a better offspring evolves is:

$$P_{swap} = 2 \cdot \frac{M-l}{n} \cdot \frac{K-k+l+kM}{n} = \frac{2(M-l)(K+l+k(M-1))}{n^2}$$

This probability comes from the fact that one wants to select any 0 in the active bin in one of the parents and a 1 anywhere in the other parent. Obviously, as the number of 1-bits in both parents grows, so does this quantity. K comes from the pessimistic assumption that the algorithm is initialized with one 1-bit in each bin. Again, the probability of failure is also used:

$$P_{\mathrm{F}} = 1 - P_{swap}$$

The same assumption about the Uniform distribution of elite species is made as in the OneMax Section. The main model (Equation 4.11) is the application of the law of total probability twice. Once the probability of success has been found, one can treat it as a parameter in a Geometric rv: the number of failures before the first success. The probability of failure given l bits set to 1 in the active bin is

$$P(G_{0l}) = \frac{1}{\mu}\sum_{\alpha=1}^{\mu}(1 - P_{sel,\alpha}P_{swap})^{\frac{\lambda}{2}}$$

Therefore, the probability of breeding at least one offspring with higher auxiliary function value is

$$P(G_l) = 1 - P(G_{0l}) = 1 - \frac{1}{\mu}\sum_{\alpha=1}^{\mu}(1 - P_{sel,\alpha}P_{swap})^{\frac{\lambda}{2}}$$

and the expected time until the next improvement of the auxiliary function of the active bin is

$$\mathbf{E}T_k = \sum_{l=0}^{M-1} \frac{1}{P(G_l)} \tag{4.12}$$

Finally, summing over all k from 0 to $K-1$(since G depends on both l and k):

$$\begin{aligned}\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} &= \sum_{k=0}^{K-1}\sum_{l=0}^{M-1} \frac{1}{P(G_{l,k})} = \sum_{k=0}^{K-1}\sum_{l=1}^{M-1} \frac{1}{1-\frac{1}{\mu}\sum_{\alpha=1}^{\mu}(1-P_{sel,\alpha}P_{swap})^{\frac{\lambda}{2}}} \\ &= \sum_{k=0}^{K-1}\sum_{l=1}^{M-1} \frac{1}{1-\frac{1}{\mu}\sum_{\alpha=1}^{\mu}(1-\frac{2\alpha^2(M-l)(K+l+k(M-1))}{\mu^2n^2})^{\frac{\lambda}{2}}}\end{aligned} \tag{4.13}$$

This expression is tested numerically for different values of n, μ, λ. Unfortunately, it does not seem to exist in the closed form, so instead asymptotic approximation is derived in the next subsection.

4.8.2 Asymptotic runtime of $(\mu+\lambda)$ $\mathbf{EA}_{1BS}$ on the RR Test Function

This derivation is in many ways similar to that for the OneMax test function.

$$\begin{aligned}P(G_{0l}) &= \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-P_{sel,\alpha}P_{swap}\right)^{\frac{\lambda}{2}} \geq \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\lambda}{2}P_{sel,\alpha}P_{swap}\right) \\ &= 1-\frac{\lambda P_{swap}}{2\mu^3}\sum_{\alpha=1}^{\mu}\alpha^2 \geq 1-\frac{3\lambda P_{swap}}{2}\end{aligned}$$

Here the Bernoulli inequality $(1-y)^n > 1-ny$ was used since obviously $0 < P_{sel,\alpha}P_{swap} < 1$ (product of the probabilities) and $\frac{\lambda}{2} > 1$. Then also

$$\sum_{\alpha=1}^{\mu}\alpha^2 = \frac{(2\mu+1)(\mu+1)\mu}{6} < 3\mu^3 = O(\mu^3)$$

Since $P(G_l) = 1 - P(G_{0l})$, the expected time until solving an active bin (using the expression for P_{swap}) is

$$\begin{aligned}
\mathbf{E}T_l &\geq \sum_{l=1}^{M-1} \frac{1}{P(G_l)} = \frac{n^2}{3\lambda} \sum_{l=1}^{M-1} \frac{1}{(M-l)(K+l+k(M-1))} \\
&= \frac{n^2}{3\lambda(M+K+k(M-1))} \left(\sum_{l=1}^{M-1} \frac{1}{M-l} + \sum_{l=1}^{M-1} \frac{1}{K+k(M-1)+l} \right) \\
&\geq \frac{n^2}{3\lambda(M+K+k(M-1))} \big(\log M + o(1) \big)
\end{aligned}$$

Here again the bounds on the M^{th} Harmonic number:

$$\log M \leq \sum_{l=1}^{M} \frac{1}{l} \leq \log M + 1$$

The $o(1)$ of the second term comes from the following derivation (using the definition $n = KM$) and asymptotically as $K \to \infty$:

$$\begin{aligned}
\sum_{l=1}^{M-1} \frac{1}{K+k(M-1)+l} &= \log(K+k(M-1)+M) - \log(K+k(M-1)) \\
&= \log\left(1 + \frac{M}{K+k(M-1)}\right) \geq \log\left(1+\frac{M}{n}\right) = \log\left(1+\frac{1}{K}\right) = o(1)
\end{aligned}$$

Finally, one is able to sum over the number of bins in the string, K and obtain the asymptotic expression for the whole algorithm:

$$\begin{aligned}
\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} &\geq \sum_{k=0}^{K-1} \frac{n^2 \log M}{3\lambda(M+K+k(M-1))} = \frac{n^2 \log M}{3\lambda} \sum_{k=0}^{K-1} \frac{1}{K+M+k(M-1)} \\
&= \frac{n^2 \log M}{3\lambda M} \log(1 + \frac{KM}{K+M}) = \Omega\left(\frac{n^2 \log M \log(\frac{KM}{K+M})}{\lambda M} \right) \qquad (4.14)
\end{aligned}$$

This result is true if run on theoretical parallel computers. To measure this bound in the number of function evaluations, it is multiplied by $\Omega(\lambda)$:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \Omega\left(\frac{n^2 \log M \log(\frac{KM}{K+M})}{M}\right) \tag{4.15}$$

For a specific case of the Royal Roads Test function when $K = M = \sqrt{n}$ this bound becomes

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \Omega\left(n^{\frac{3}{2}} \log^2 n\right) \tag{4.16}$$

which seems to be a big improvement compared to the RR function [WJ07, SW03], where the bound is $\Omega(n^2)$ and LeadingOnes in [CHS+09], where the bound is $O(n^2 + nN \log N)$.

4.8.3 Runtime analysis of $(\mu+\lambda)$RLS on the RR Test Function

The main difference of this model from the previous one is the genetic operator. Since bits are flipped in a single parent, they do not depend on the number of 1-bits in other strings in the pool. Therefore the index of the active bin does not really matter. This simplifies the analysis quite a bit.

$$P_{sel,\alpha} = \frac{\alpha}{\mu}\left(\frac{\alpha}{\mu} + 1 - \frac{\alpha}{\mu}\right) = \frac{\alpha}{\mu}$$

The same assumption of the Uniform distribution of elite species is used. The flip probability, assuming each bin has only 1 bit at the beginning of the run (therefore the auxiliary value of the string is K) is just

$$P_{flip} = \frac{M - l}{n}$$

since it is sufficient to select any one of 0-bits from the active bin regardless of values in other bins or in the other parents.

Probability of failure given l successful flips so far (using Equation 4.1) is

$$P(G_{0l}) = \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\alpha P_{flip}}{\mu}\right)^{\lambda}$$

and the runtime of the algorithm is

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} = \sum_{k=0}^{K-1}\sum_{l=0}^{M-1}\frac{1}{1-\frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\alpha P_{flip}}{\mu}\right)^{\lambda}} \tag{4.17}$$

which is also tested numerically.

4.8.4 Asymptotic runtime of $(\mu+\lambda)$RLS on the RR Test Function

Approximating the probability of failure given l bits set to 1,

$$\begin{aligned} P(G_{0l}) &= \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\alpha P_{flip}}{\mu}\right)^{\lambda} \geq \frac{1}{\mu}\sum_{\alpha=1}^{\mu}\left(1-\frac{\lambda\alpha P_{flip}}{\mu}\right) = 1-\sum_{\alpha=1}^{\mu}\frac{\lambda\alpha P_{flip}}{\mu^2} \\ &= 1-\lambda P_{flip} \end{aligned}$$

and the expected time to solve the active bin is therefore

$$\mathbf{E}T_k = \sum_{l=1}^{M-1}\frac{1}{\lambda P_{flip}} = \frac{n}{\lambda}\sum_{l=1}^{M-1}\frac{1}{M-l} \geq \frac{n\log M}{\lambda}$$

Since there are K such bins and the probability of successful sampling does not depend on it (unlike $(\mu+\lambda)\mathrm{EA}_{1BS}$), the expected first hitting time for the algorithm on the RR test function is obtained:

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} = \sum_{k=1}^{K-1}\mathbf{E}T_k = \Omega\left(\frac{nK\log M}{\lambda}\right) \tag{4.18}$$

or, measured in the number of function evaluations,

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} = \sum_{k=1}^{K-1} \mathbf{E}T_k = \Omega(nK \log M) \tag{4.19}$$

If for example $K = M = \sqrt{n}$, the bounds become

$$\mathbf{E}\tau_{(\mu+\lambda)RLS} = \Omega(n^{\frac{3}{2}} \log n)$$

which is only up to an order $\Omega(\log n)$ faster than EA using 1-Bit-Swap operator.

4.9 Numerical results

To test four equations that do not exist in the closed form, a large number of numerical experiments was performed. For simplicity, $\mu \equiv \lambda$ and $M \equiv 8$ for the RR test function. In Tables 4.4 and 4.5 the set of parameters for which the experiments were run is presented for both OneMax and Royal Roads test functions.

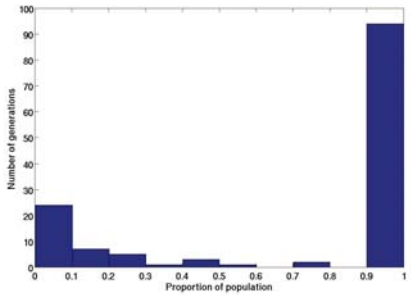
n	μ	Length of run (generations)
50	20,30,40,50	500
100	30,40,50,60,70,80,90,100	1000
500	20,50,100,150,200 250,300,350,400,450,500	3000
1000	50,100,150,200 250,300,350,400,450,500	5000

Table 4.4: Set of parameters used for OneMax test function

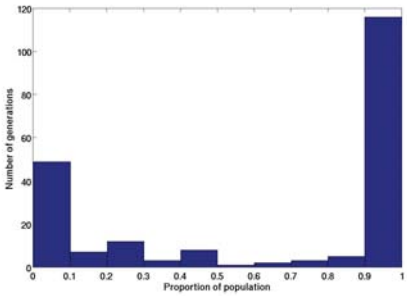
Each parameter tuple (μ, n) was run 50 times (the length of run is given in the respective table). The author compares the average generation at which the global optimum (or to relax this condition, its $\epsilon-$basin) was reached to the theoretical estimate in the respective equation. In case the basin was not reached, this run is ignored. Plots of the probability of success of $(\mu+\lambda)\mathrm{EA}_{1BS}$ on OneMax is missing

n	μ	Length of run (generations)
32	40,50,60,70,80,90,100	1000
64	40,50,60,70,80,90,100,110,120	1000
128	40,50,60,70,80,90,100,110,120 130,140,150,160,170,180,190,200	1000/2000
256	40,60,80,100,120,140,160 180,200,220,240,260,280	3000/4000
512	30,60,90,120,150,180,210,240 270,300,330,360,390,420,450,480,510,540	5000/6000

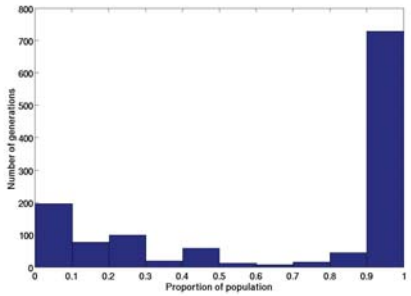
Table 4.5: Set of parameters used for the RR test function



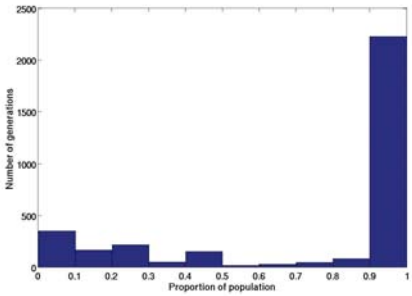
(a)
n=50



(b)
n=100

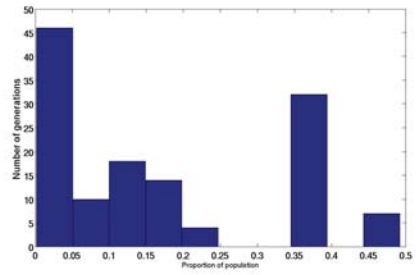


(c)
n=500

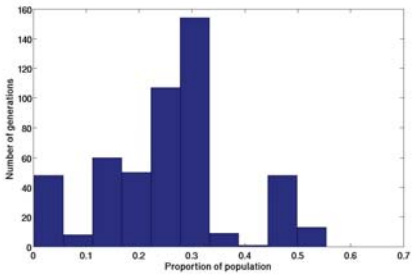


(d)
n=1000

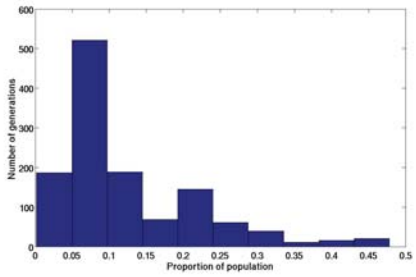
Figure 4.1: Distribution of the elite species in the population of $(\mu + \lambda)$ EA_{1BS} solving OneMax Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum



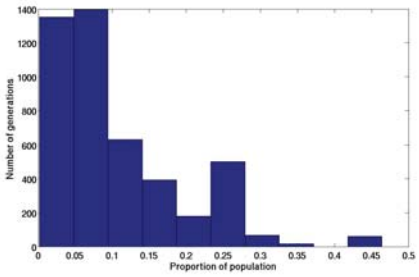
(a)
n=50



(b)
n=100

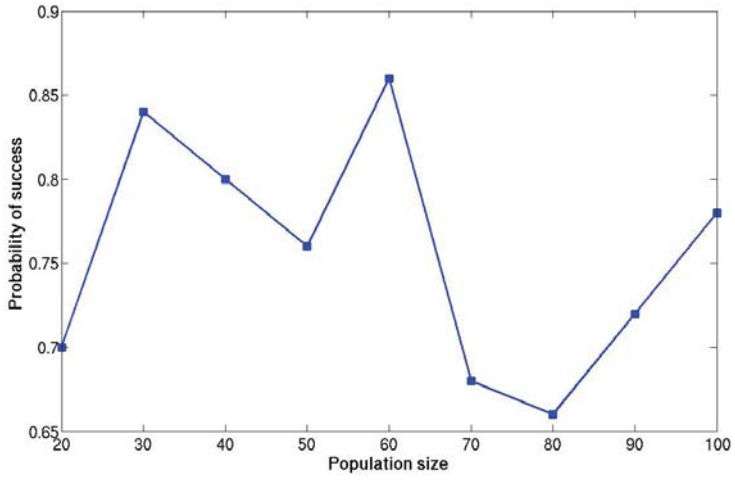


(c)
n=500

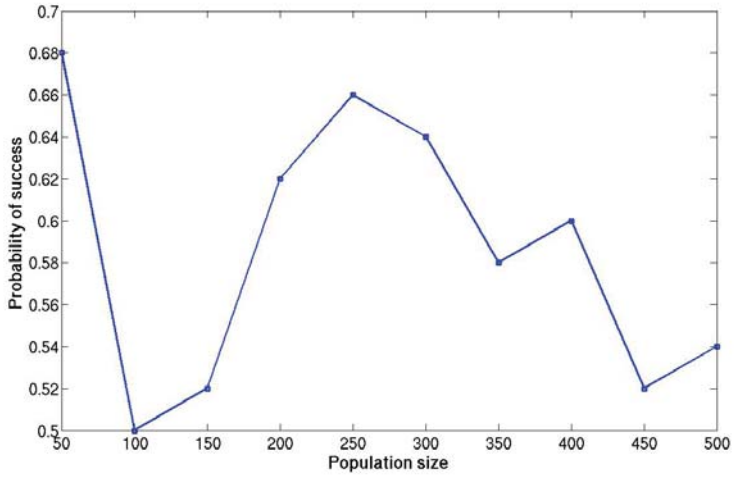


(d)
n=1000

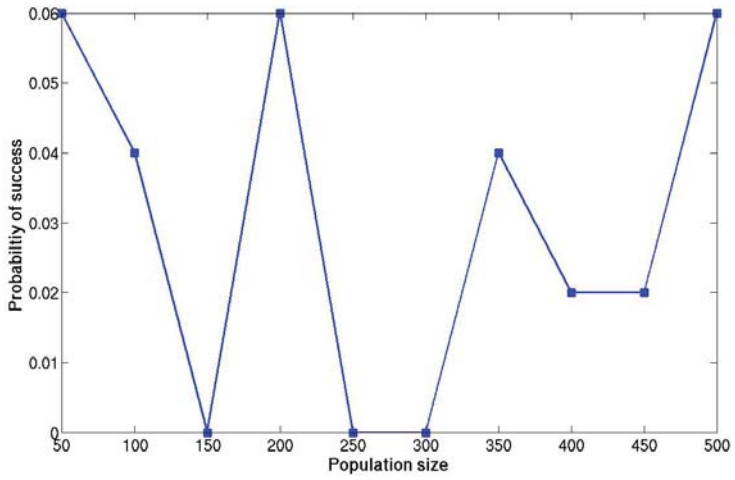
Figure 4.2: Distribution of the elite species in the population of $(\mu+\lambda)$RLS solving OneMax Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum



(a)
n=100

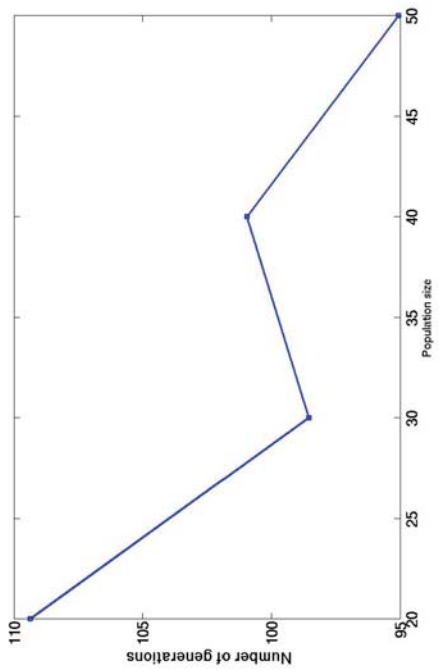


(b)
n=500

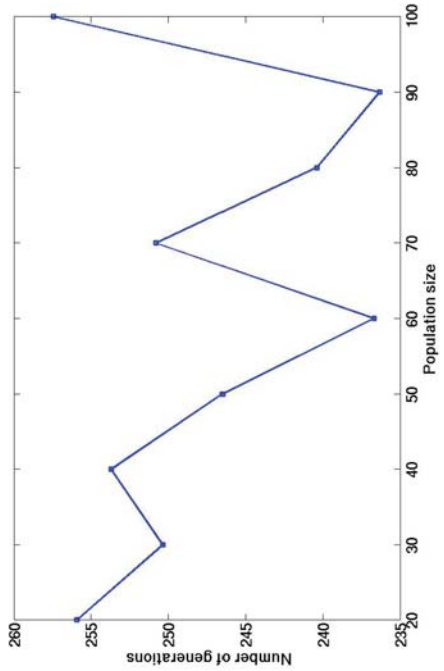


(c)
n=1000

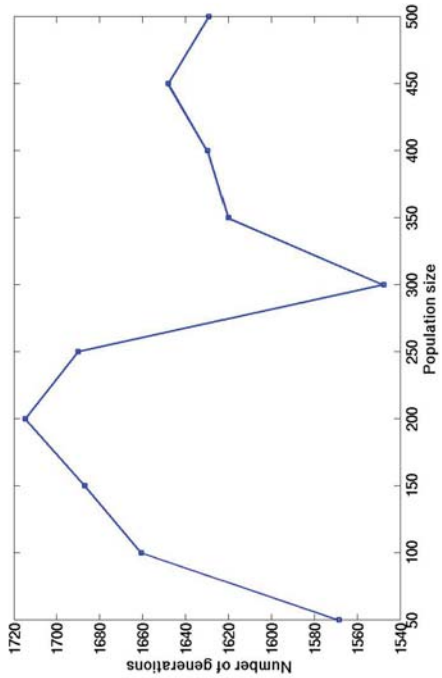
Figure 4.3: Probability of success of $(\mu + \lambda)$RLS solving OneMax Test Function.



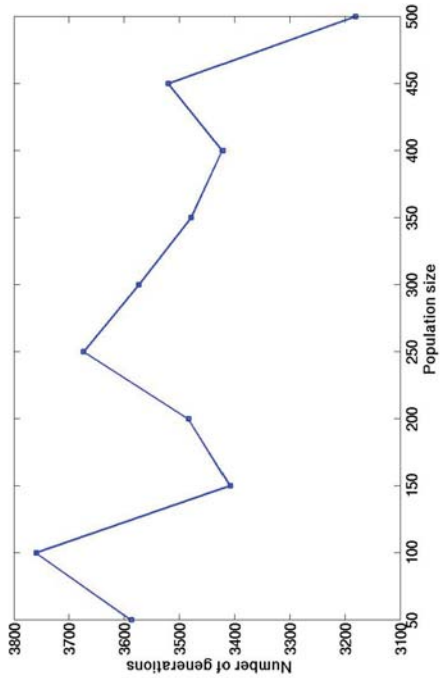
(a)
n=50



(b)
n=100

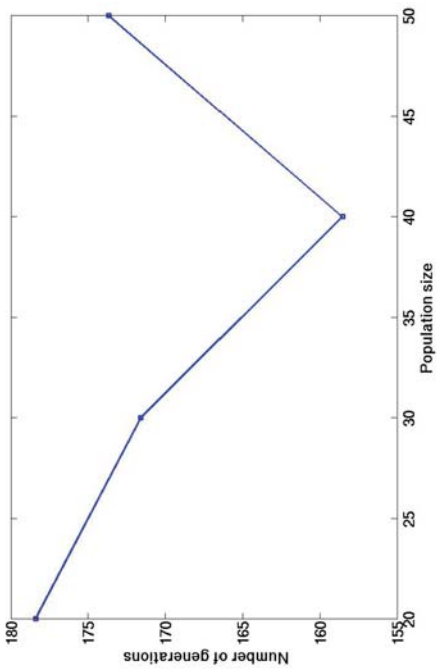


(c)
n=500

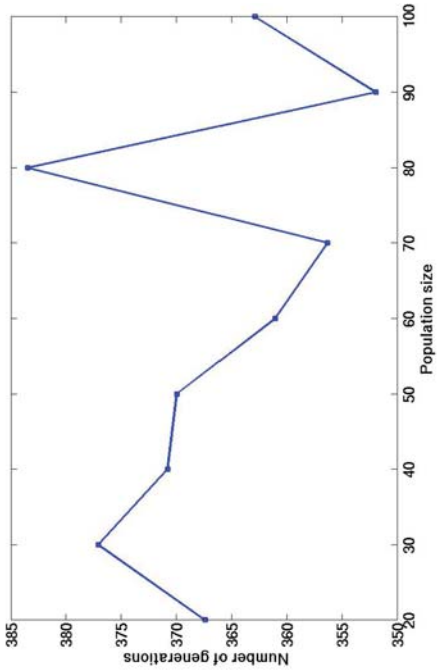


(d)
n=1000

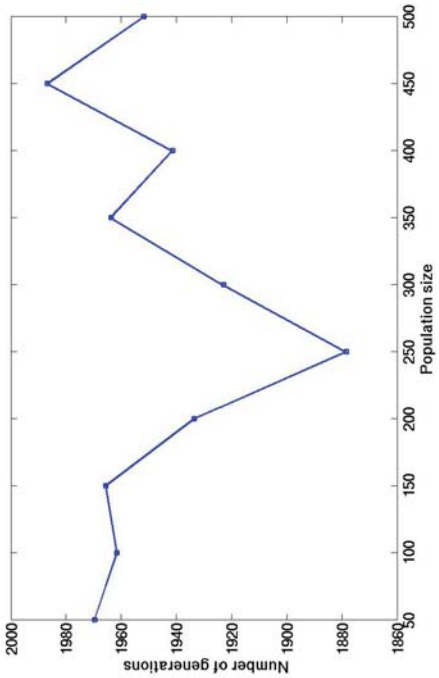
Figure 4.4: Numerical runtime estimate for $(\mu + \lambda)\text{EA}_{1BS}$ solving OneMax Test Function for different population sizes.



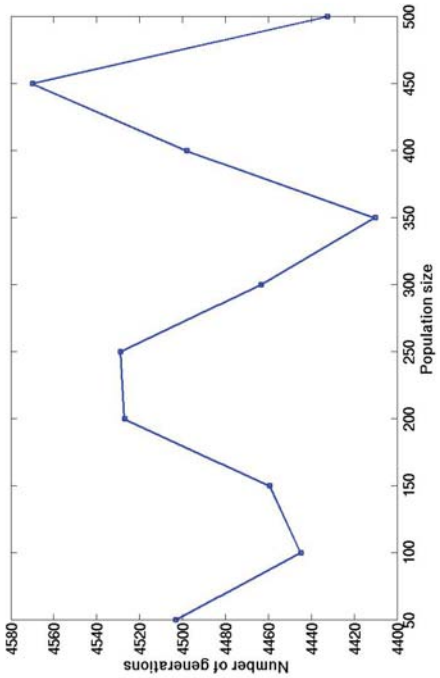
(a)
n=50



(b)
n=100

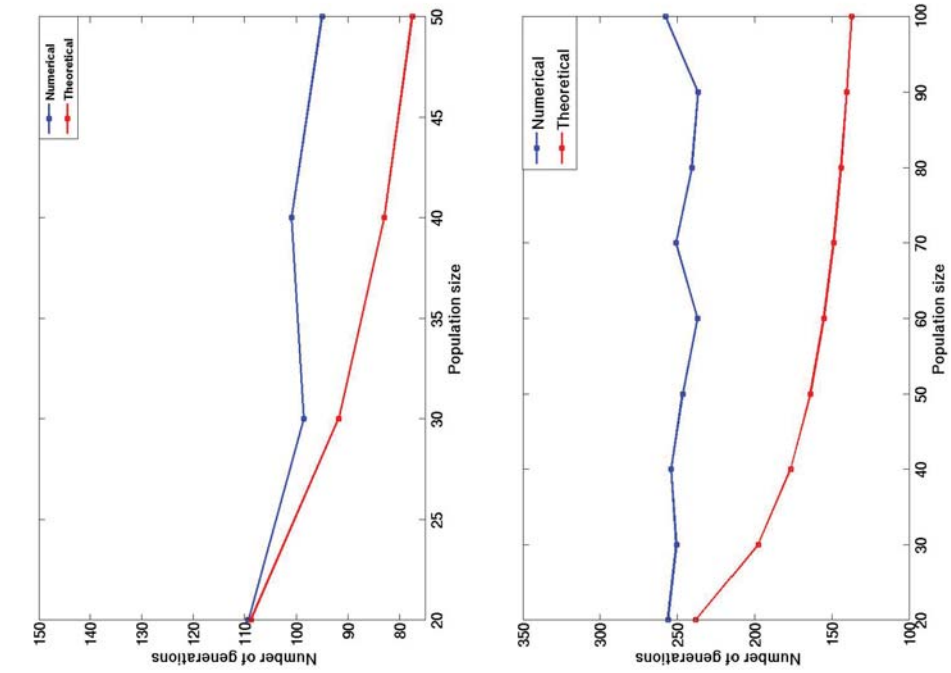


(c)
n=500



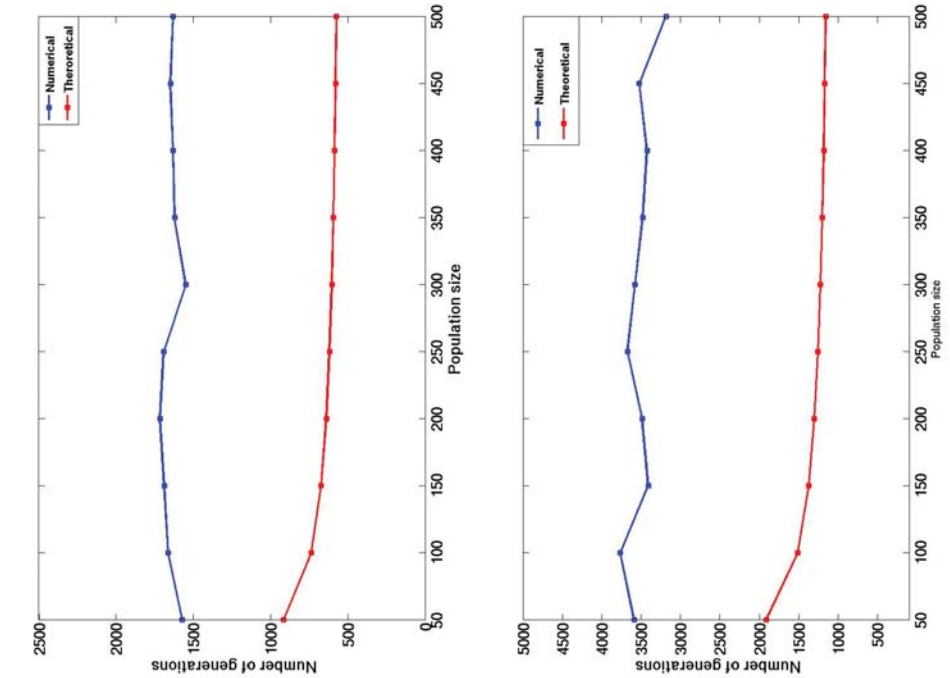
(d)
n=1000

Figure 4.5: Numerical runtime estimate for $(\mu + \lambda)$RLS solving OneMax Test Function for different population sizes.



(a)
n=50

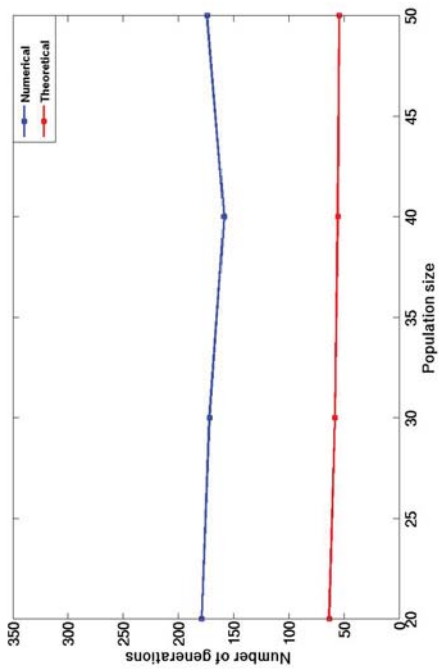
(b)
n=100



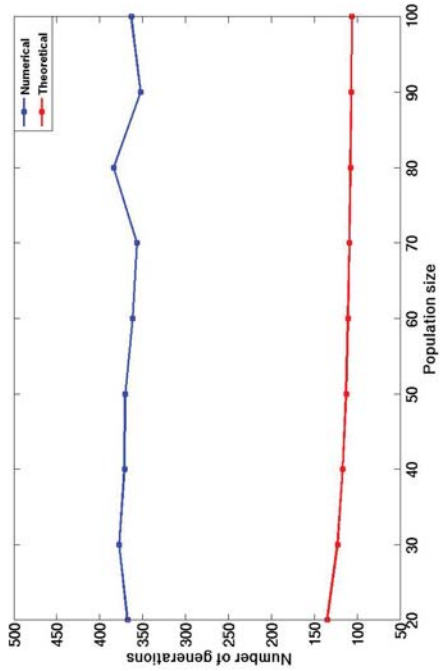
(c)
n=500

(d)
n=1000

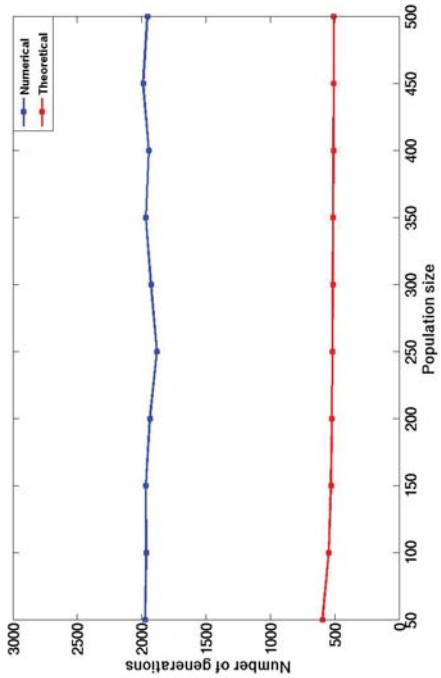
Figure 4.6: Theoretical and numerical estimate for $(\mu+\lambda)EA_{1BS}$ solving OneMax Test Function



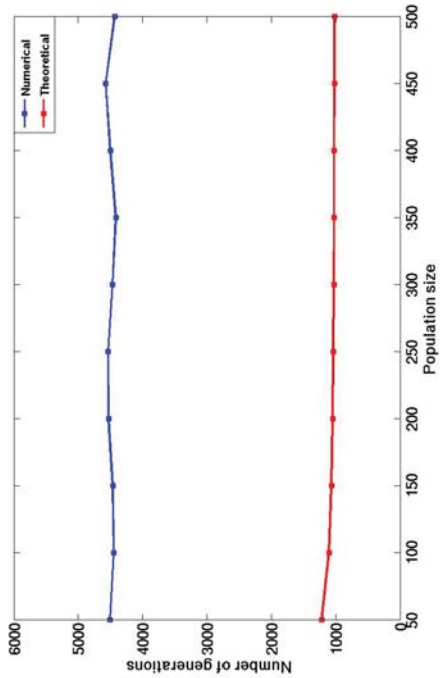
(a)
n=50



(b)
n=100

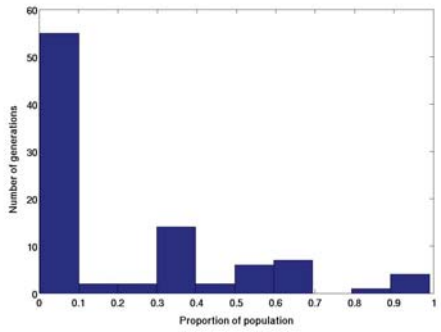


(c)
n=500

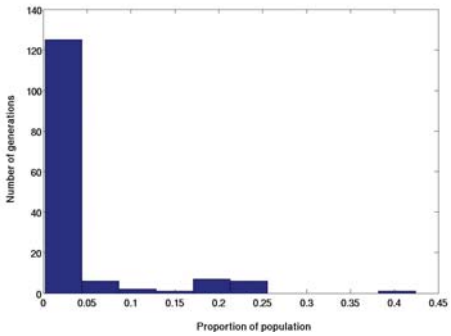


(d)
n=1000

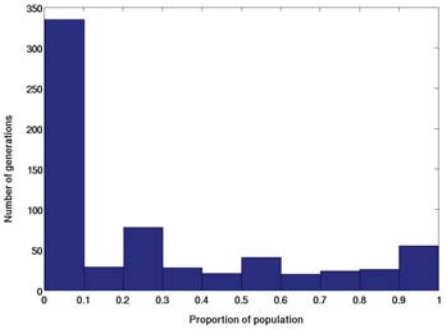
Figure 4.7: Theoretical and numerical estimate for $(\mu + \lambda)RLS$ solving OneMax Test Function



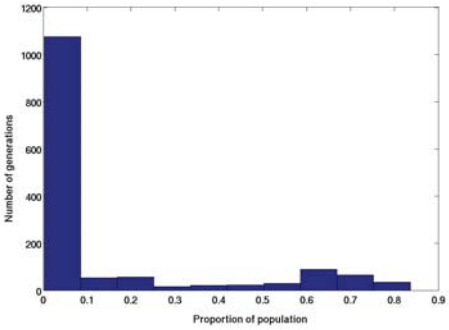
(a)
n=32



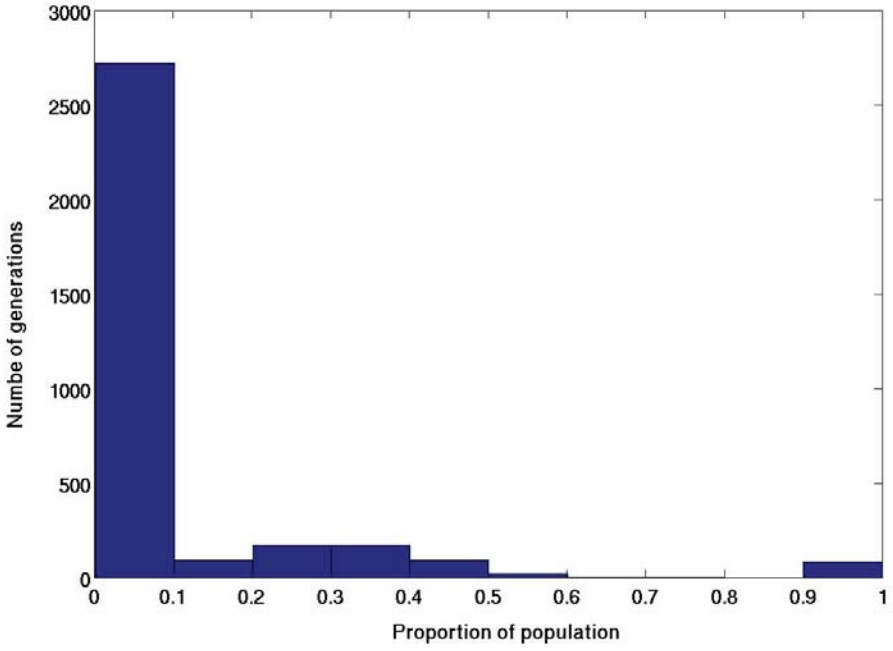
(b)
n=64



(c)
n=128

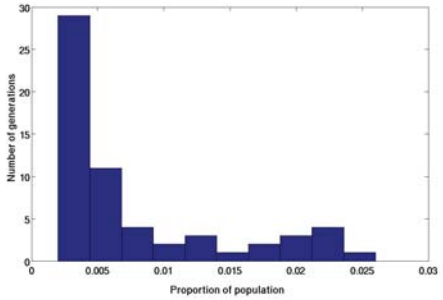


(d)
n=256

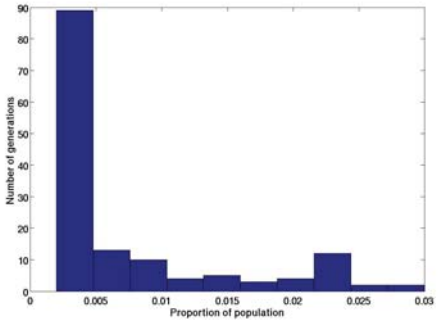


(e)
n=512

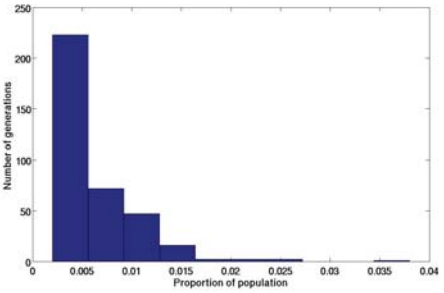
Figure 4.8: Distribution of the elite species in the population of $(\mu + \lambda)$ EA_{1BS} solving Royal Roads Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum



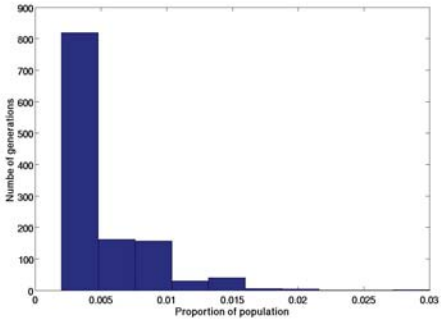
(a)
n=32



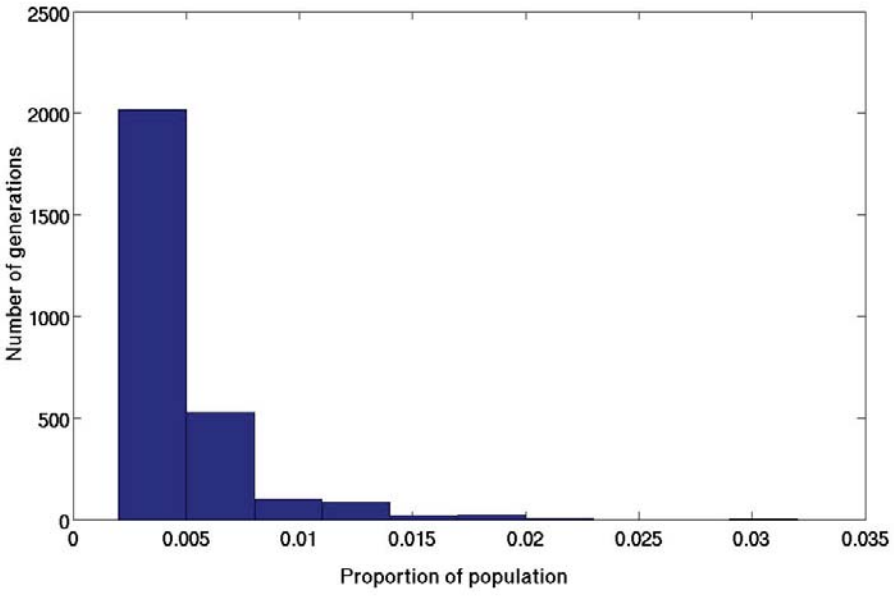
(b)
n=64



(c)
n=128

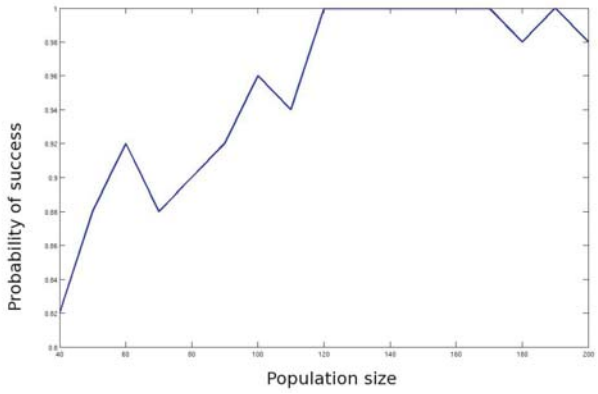


(d)
n=256

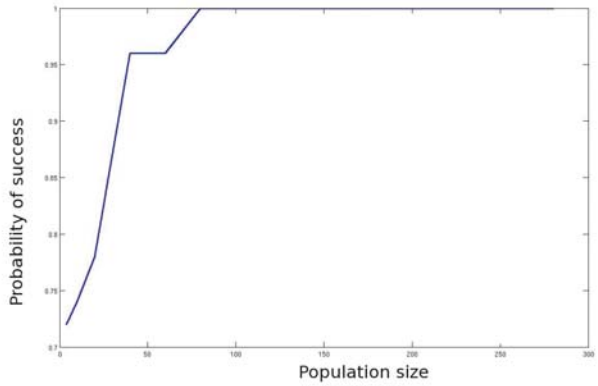


(e)
n=512

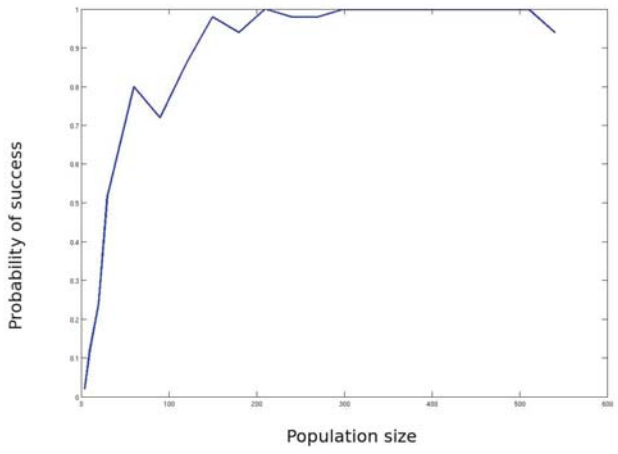
Figure 4.9: Distribution of the elite species in the population of $(\mu+\lambda)$RLS solving Royal Roads Test Function for $\mu = \lambda = 500$ and stopped at the achievement of the global optimum



(a)
n=128

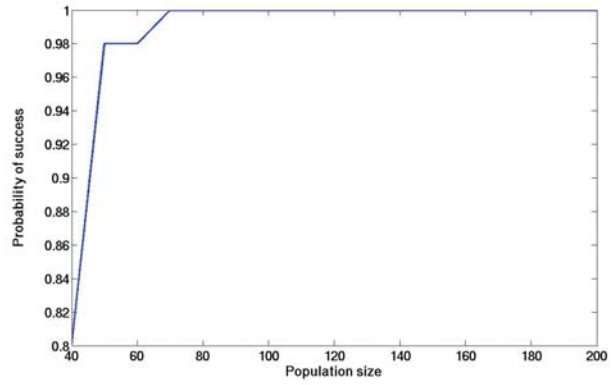


(b)
n=256

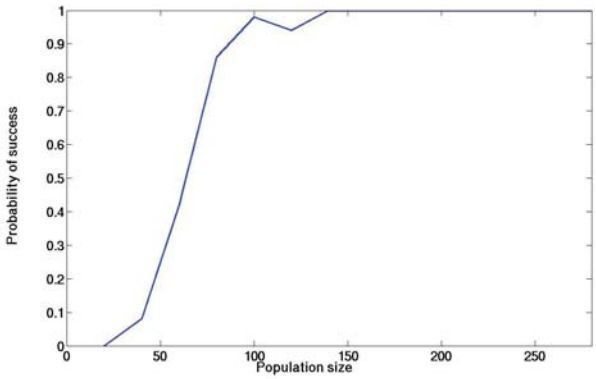


(c)
n=512

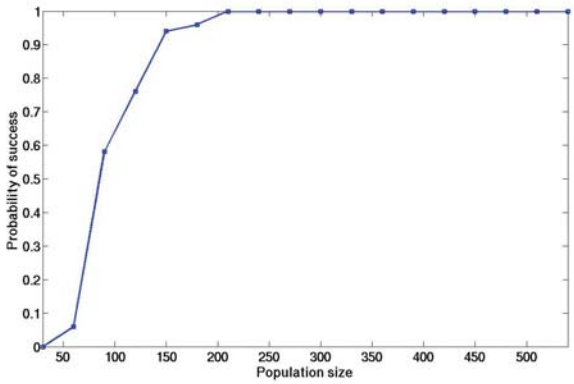
Figure 4.10: Probability of success of $(\mu + \lambda)$ EA_{1BS} solving Royal Roads Test Function. For $n = 32, 64$ it is always almost 1



(a)
n=128

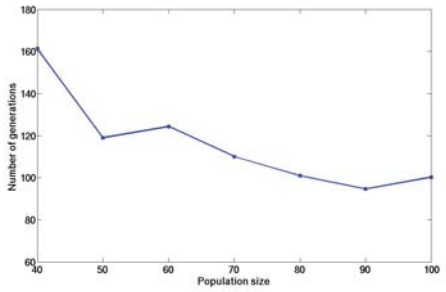


(b)
n=256

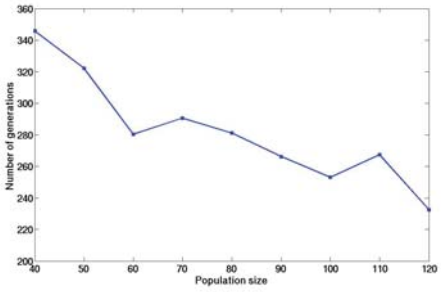


(c)
n=512

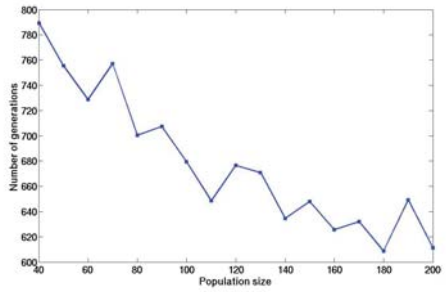
Figure 4.11: Probability of success of $(\mu + \lambda)$RLS solving Royal Roads Test Function.For $n = 32, 64$ it is always almost 1



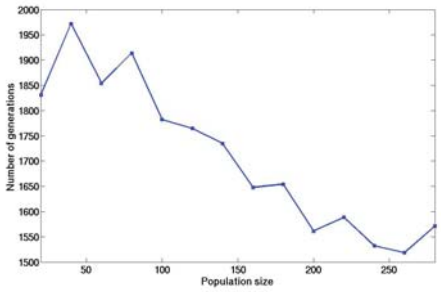
(a)
n=32



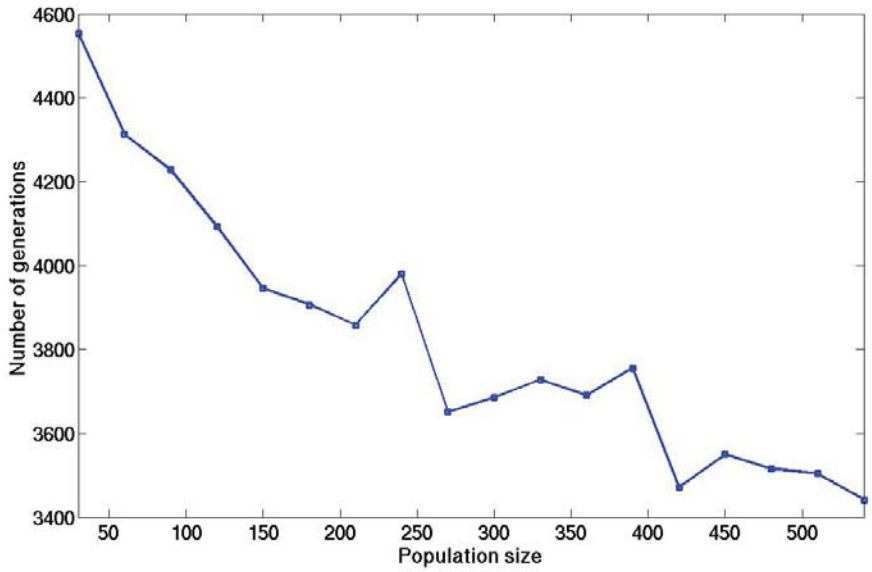
(b)
n=64



(c)
n=128

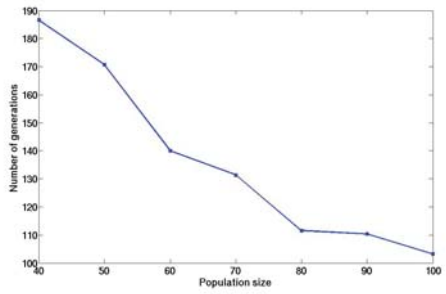


(d)
n=256

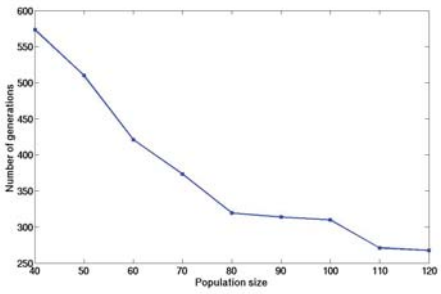


(e)
n=512

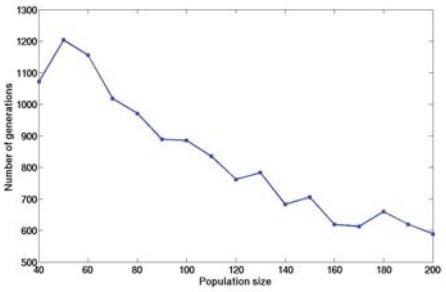
Figure 4.12: Numerical runtime estimate for $(\mu + \lambda)\text{EA}_{1BS}$ solving Royal Roads Test Function for different population sizes. The positive effect of the population size measured in the number of generations is obvious.



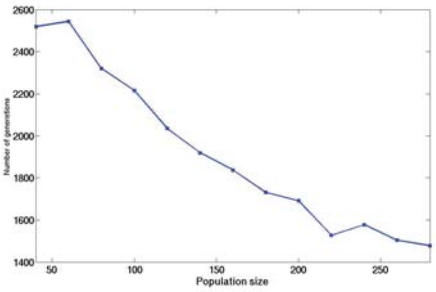
(a)
n=32



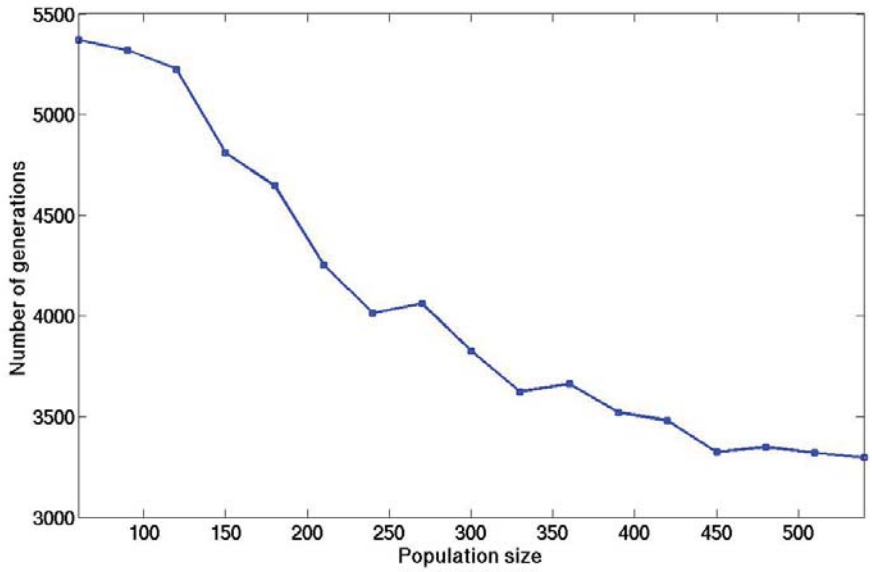
(b)
n=64



(c)
n=128

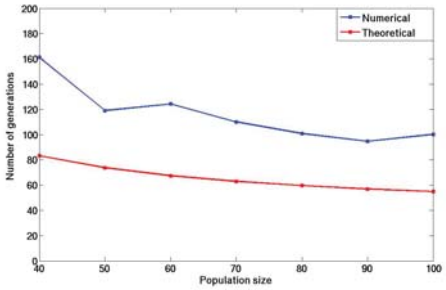


(d)
n=256

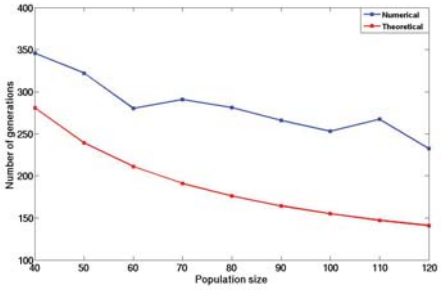


(e)
n=512

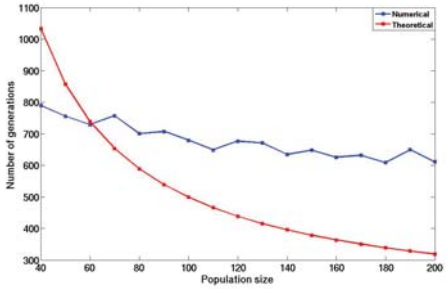
Figure 4.13: Numerical runtime estimate for $(\mu+\lambda)$RLS solving Royal Roads Test Function for different population sizes. The positive effect of the population size measured in the number of generations is obvious.



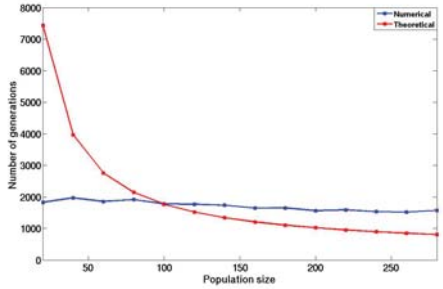
(a)
n=32



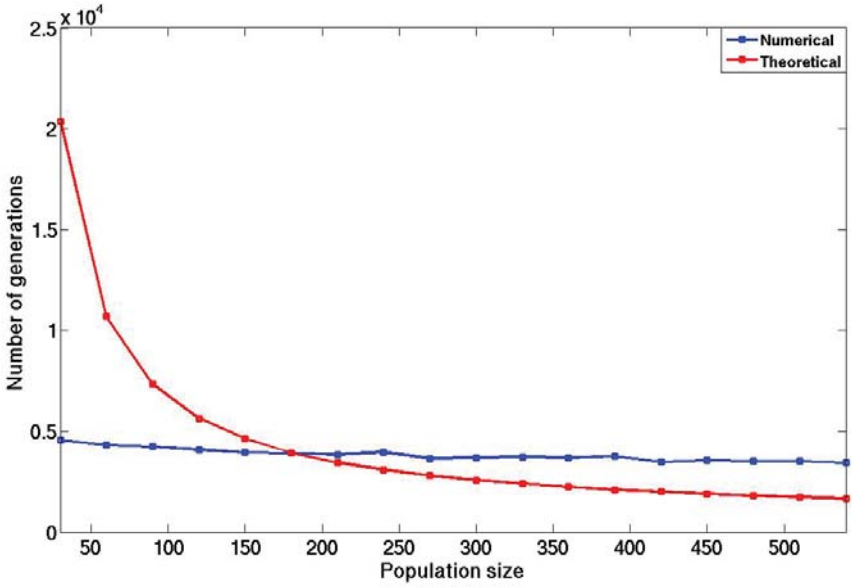
(b)
n=64



(c)
n=128

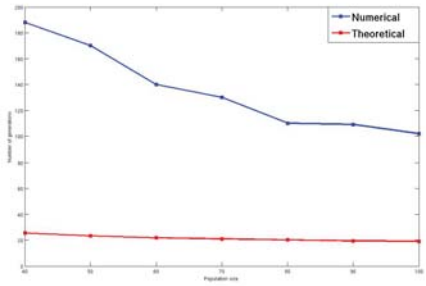


(d)
n=256



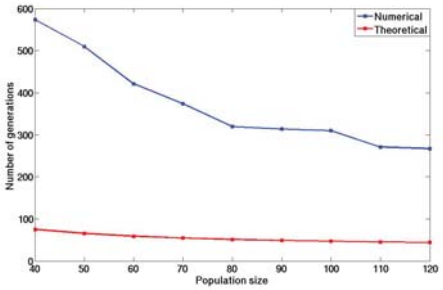
(e)
n=512

Figure 4.14: Theoretical and numerical bounds for $(\mu + \lambda)EA_{1BS}$ solving Royal Roads Test Function

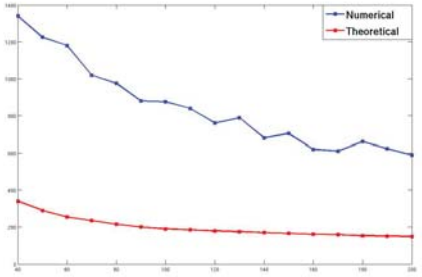


s

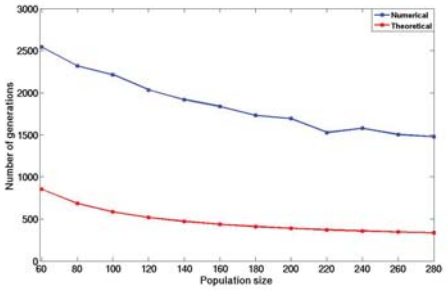
(a)
n=32



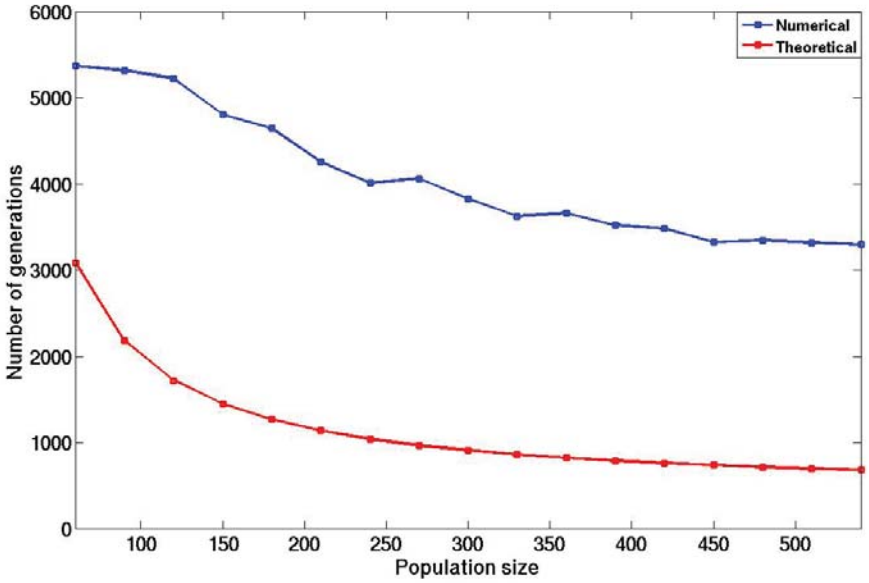
(b)
n=64



(c)
n=128



(d)
n=256



(e)
n=512

Figure 4.15: Theoretical and numerical bounds for $(\mu+\lambda)$RLS solving Royal Roads Test Function

since it is always 1.

Certain conclusions can be draw from these results.

1-Bit-Swap vs Randomized Local Search Results for EA with 1BS are consistently better than those for RLS for most instances of both functions, both in terms of the higher probability of success (1BS always finds the solutions, hence the respective plots are missing) and shorter runtime. This results is especially strong for all instances of OneMax and small populations on the RR. For large populations on RR the results for RLS are similar to those for 1BS. One doesn't fail to notice that theoretically RLS performs up to an order $\Omega(\log n)$ faster than EA with 1BS. More investigation is needed in this area to clarify this result. One may suggest that if tested on even large populations sizes, RLS may outperform 1BS.

Distribution of Elite Species (OneMax Test Function) Numerically for OneMax problem (see Figures 4.1, 4.2) it is obvious that the assumption of Uniform distribution of elite species as an upper bound on the probability to observe 'large' numbers of elite species does not hold. In fact, EA with 1BS exhibits a completely different pattern of elite species than RLS. It seems to accumulate a large number of elite species fast and thus improve the probability of finding a better solution. RLS on the other hand does not seem to exhibit any consistent pattern at all, although the upper bound assumption seems to hold.

Distribution of Elite Species (Royal Roads Function) By looking at figures 4.8, 4.9 it becomes pretty clear that elite species in both algorithms (or, more correctly, super-elite, those with the highest auxiliary OneMax value) follow some form of an exponential decay. This justifies the use of Uniform distribution because it gives an upper bound on the probability to observe 'large' numbers of α species. This phenomena will be scrutinized in the next chapter.

Effect of the Population size (OneMax Function) On Figures 4.3, 4.4 and 4.5 it is quite obvious, that the increase in the population size does not bring about

any positive changes in the efficiency of both algorithm, neither for the probability of finding the solution, nor the runtime. Obviously, if measured in the number of function evaluations (multiplied by 2λ) effect of the population will be strictly degrading. This confirms many previous findings (e.g. in [He10, Wit04]) that have showed the negative effect of the population for algorithms solving OneMax or other easy (runtime at most $O(n \log n)$) problems.

Effect of the Population size (Royal Roads Function) On Figures 4.10 and 4.11 the positive effect on the probability of finding the global optimum by both algorithms is quite obvious, unlike OneMax. Another thing is the leveling out effect, that was noticed already in [HY02]: as the population size increases, the marginal effect reduces. On Figures 4.12 and 4.13 another positive effect of the population is obvious, i.e. faster convergence: as the population increases, runtime, measured in generations, decreases. This benefit is achieved though if run on parallel computers. On a single computer, if measures in the number of function evaluations, in some of these cases increase in the population size degrades performance.

Numerical vs Theoretical Results (OneMax Test Function) Finally, it is possible to compare numerical and theoretical findings (see Figures 4.6 and 4.7). Theoretical findings (Equations 4.4 and 4.8) yield lower bounds despite the clear violation of the Uniform distribution assumption as shown above. As a result, apart from the $(\mu + \lambda)\text{EA}_{1BS}$ algorithm with $n = 50$ all other bounds seem quite loose. Results for the other three $(\mu + \lambda)\text{EA}_{1BS}$ algorithms are tighter than for all four $(\mu + \lambda)$RLS, although even in the worst case theoretical results are different by the factor of 4, which still qualifies for a 'small constant'.

Numerical vs Theoretical Results (Royal Roads Test Function) For Royal Roads (Figures 4.14, 4.15 and Equations 4.13, 4.17) the results are substantially more consistent than for OneMax. Numerical and theoretical curves have the same shape, demonstrating the leveling-out effect of the population. Since the Uniform distribution assumption for the RR function was verified, lower bounds are much tighter than for OneMax. For the $(\mu + \lambda)\text{EA}_{1BS}$ algorithm it is tight at

most up to a very small constant (for $n = 128$ it is ≈ 2). For larger n, e.g. $n = 512$ theoretical result is asymptotically tight as $\mu \to \infty$. For $(\mu + \lambda)$RLS the results are more loose, but for large n they again get tight asymptotically (the constant is ≈ 3).

4.10 Conclusions

As mentioned at the beginning of the chapter, there have been very few results on $(\mu + \lambda)$ population-based EAs, and even fewer on the structure of the population. In this chapter an attempt was made to answer some of the most basic questions arising in connection with population-based elitist EAs, e.g. the distribution of elite species. As a result of this approach, a number of lower bounds on the runtime were derived and the results were tested numerically.

- Both asymptotic solutions for OneMax function are of the same order $\Omega(\frac{n \log n}{\lambda})$ and are comparable to the results for $(\mu + 1)$ and $(1 + \lambda)$ EAs available in the literature. Also this results demonstrates the benefit of using parallel computers by increasing the size of the recombination pool

- The asymptotic order for the Royal Roads function depends on the number of bits in a bin M and the number of bins K. For $K = M = \sqrt{n}$ RLS outperforms 1BS by an order of $\Omega(\log n)$. The result comes from the linear expression for the probability of selecting the correct bit for RLS and quadratic for 1BS. Since theoretical results contradict this finding (1BS greatly outperforms RLS for most sizes of the problem and the populations), one can infer that that additional investigation is needed (see next chapter)

- The positive effect of the population (measured as the probability to find the global solution and the runtime till this event) is evident on the Royal Roads function. Numerically it is shown that for OneMax, as the population size increases, neither the probability of finding the global solution nor the expected time (measured in the number of generations) improve. For the Royal Roads the situation is the opposite-the first benchmark grows, the

second one reduces, clearly demonstrating the benefit of large populations (at least when run on parallel computers).

- All theoretical results obtained from respective equations are tight up to a small constant. Nevertheless, it is quite obvious that for the Royal Roads test function they are much more consistent with the numerical results, i.e. they both reduce and level out (i.e. the efficiency of adding more species to the population drops). For the 1BS, results are much tighter, sometimes the multiplying constant can be as small as 1.5 (e.g. for $n = 256, \mu = 280$). Results for RLS are more loose, sometimes up to the factor of 6 (e.g. for $n = 32, \mu = 100$). This directly implies that the approximation of the distribution of elite species with Uniform distribution is much more relevant for 1BS than RLS. This is indirectly confirmed by the empirical distribution of elite species- they resemble some form of an exponential decay for both algorithms, but the right tail is much longer for 1BS than RLS.

- The distribution of elite species in both algorithms on the OneMax function does not seem to follow any distribution, which may well explain the lower consistency of theoretical results: the shapes of the theoretical curves are different to those of numerical on Figures 4.6 and 4.7. Although they are never too loose, the assumption of the Uniform distribution is clearly violated.

- The main point is that, when analyzing population-based EAs, making an assumption about the distribution of species to construct the model is a straightforward solution, but it may lead to worse-than-expected results. The next Chapter is a further step in the direction of the analysis of the subsets of the population and their effect on the runtime without such assumptions.

Chapter 5

Upper Bounds on the Runtime

In the Chapter 4 an attempt was made to understand some of the processes affecting the structure of the population and the recombination pool, and apply this understanding to the derivation of the bounds on runtime. Although some of the results were quite encouraging, numerical experiments revealed a number of drawbacks, among them:

1. Assumption of the Uniform distribution as an approximation of the sampling probability is clearly violated in the OneMax Problem.

2. Numerical results demonstrated that some bounds were quite loose (e.g. RLS solving OneMax problem).

3. The adopted approach does not distinguish between species of the same fitness level, but different distance to the next fitness level on the Royal Roads problem (or on any other problem with fitness plateaus)

It is therefore necessary to design a method that would take these points into consideration. More specifically, it should:

1. Avoid making assumptions about the distribution of species in the population,

2. Give an asymptotically tight upper bound that can be compared to the findings in Chapter 4 and results in the literature,

3. Distinguish between species on the plateau of fitness (Royal Roads), which is crucial to the derivation of the runtime

For this purpose, in this chapter a tool called the Elitism Levels Traverse Mechanism is developed and applied to both OneMax and Royal Roads (or, more generally, function with fitness plateaus). This is a novel flexible method that allows analysis of population-based algorithms solving different functions.

5.1 Main results

Some of the main results of this Chapter are:

1. Development of a new tool for the analysis of population-based Evolutionary Algorithms

2. Upper bound on the runtime of $(\mu + \lambda)\text{EA}_{1BS}$ on OneMax is $O(\frac{\mu n \log n}{\lambda})$

3. Upper bound on the runtime of $(\mu + \lambda)\text{EA}_{1BS}$ on Royal Roads with $K = M = \sqrt{n}$ is $O(\frac{\mu n^{\frac{3}{2}} \log^2 n}{\lambda})$

4. There exists a lower bound on the probability to add 1 elite offspring to the population that allows only to consider types of pairs, rather than the number of each type in the recombination pool

5. Probability to evolve a higher-ranked offspring solving a function with plateaus is lower-bounded by $1 - e^{-\frac{c}{8}} + o(1)$ for some small constant c if $\lambda = \mu$.

6. Limiting distribution of super-elite species, when solving the first bin in the Royal Roads problem converges to truncated Poisson distribution if the rate of progress is slow and Normal if it is fast.

The recovery of the results for OneMax and improvement of the results for Royal Roads verifies the validity of this approach and its potential for solving similar problems.

5.2 The Elitism Levels Traverse Mechanism

In this section a new approach to modeling $(\mu + \lambda)$ algorithms, the Elitism Levels Traverse Mechanism, is presented. It is loosely based on the ideas of fitness levels traversal (see [JW01]) and locally-optimal individuals (see [CHS+09, CTCY12]). This approach addresses some of the main problems identified in Chapter 4 and suggests a different way to derive bounds on mean first hitting times of global solutions:

1. It replaces the assumption of the species distribution in the population with the mechanism that tracks the change in their number by identifying the ways of adding certain types of species (e.g. elite) to the population,

2. Instead of using the probability of advancing a *level of fitness*, it makes use of the probability of advancing a *level of elitism*, e.g. adding an elite species up to a certain proportion δ such that the probability of evolution is arbitrarily close to 1, i.e. $1 - o(1)$,

3. Tracks the change in the number of elite species on plateaus of fitness that enables the derivation of sharper results for the Royal Roads function (see Subsection 5.4.1).

The working of the Elitism Levels Traverse Mechanism can be illustrated by an example from epidemiology (similar to the approach in [Nas96, Nas99]).

Suppose that there exists a population of species of size N, which is susceptible to M types of infection, which are mutually exclusive, i.e. a species cannot be infected by more than one infection at the same time. The size of each type of infected species cannot be larger than m_j. A sick species can infect exactly one healthy individual. An event E^*_{jr} that $r < m_j$ infected species of type j are observed in the population one of which infects exactly one healthy species. Since the sets of infected species are mutually exclusive, by additivity, the probability that exactly one healthy species gets any of the M infections is obtained:

$$P\left(\bigcup_{j=1}^{M} E^*_j\right) = P\left(\bigcup_{j=1}^{M}\bigcup_{r=1}^{m_j} E^*_{jr}\right) = \sum_{j=1}^{M}\sum_{r=1}^{m_j} P(E^*_{jr}) = \sum_{j=1}^{M} P(E^*_j) = P(E^*)$$

This expression needs to be simplified for a number of reasons, e.g. the knowledge of m_j. Although one can find bounds on the partial sum of rows of Pascal triangle (since m_j is clearly less than N), it is guaranteed to make the derivation messy. Therefore, only one infected species of each type is considered rather than r, and the event of infecting exactly one healthy species with infection type j is therefore E_j. This yields the lower bound on the total probability of adding exactly one infected offspring:

$$P(E_j^*) \geq P(E_j) \leftrightarrow P(E^*) = \sum_{j=1}^{M} P(E_j^*) > \sum_{j=1}^{M} P(E_j) = P(E) \tag{5.1}$$

In the language of Evolutionary Computation, infected species of any type are elite species, M are the types of pairs that are able to produce at most one elite offspring, m_j is the number of pairs of type j in the recombination pool, N is the size of the recombination pool. The new idea is the $\delta\mu$ (for $0 < \delta < 1$), which is the number of elite individuals in the population that ensures the $1 - o(1)$ probability of advancing to the next level of evolution. If there are n such levels, the expected runtime of EA can be expressed as

$$\begin{aligned} \mathbf{E}\tau &= \sum_{k=1}^{n} \sum_{\alpha=1}^{\delta\mu} \frac{1}{P(E^*(\alpha,k))} = \sum_{k=1}^{n} \sum_{\alpha=1}^{\delta\mu} \frac{1}{\sum_{j=1}^{M} P(E_j^*(\alpha,k))} \\ &< \sum_{k=1}^{n} \sum_{\alpha=1}^{\delta\mu} \frac{1}{\sum_{j=1}^{M} P(E_j(\alpha,k))} = \sum_{k=1}^{n} \sum_{\alpha=1}^{\delta\mu} \frac{1}{P(E(\alpha,k))} \end{aligned} \tag{5.2}$$

Derivation of the upper bound on the runtime using Equation 5.2 is rather flexible. One needs to identify pairs of possible parents $< p_1, p_2 >$ such that there exists some probability of swapping (or otherwise exchanging) bits $\varphi(k) > 0$ that, as a result, a new elite offspring evolves.

Before this tool is applied to OneMax and Royal Roads problems, Equation 5.1 (and, hence, Equation 5.2) is proved for an arbitrary type of pairs (infection) j (this lower bound is not to be confused with a trivial one of the form $\sum_{k \geq r} \binom{n}{k} p^k (1 -$

$p)^k > \binom{n}{r}p^r(1-p)^{n-r}$):

$$P(E_j) = \binom{\frac{\lambda}{2}}{1} P_{sel} P_{swap} = \binom{\frac{\lambda}{2}}{1} P_{sel}\varphi_j(k) \tag{5.3}$$

$$\begin{aligned} P(E_j^*) &= P\left(\bigcup_{r=1}^{m_j} E_{jr}^*\right) \\ &= \sum_{r=1}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-P_{sel})^{\frac{\lambda}{2}-r} \binom{r}{1} \varphi_j(k)(1-\varphi_j(k))^{r-1} \end{aligned} \tag{5.4}$$

As mentioned before, P_{swap} is the probability to swap bits such that a new elite offspring evolves. Since all the terms in the sum are positive, the lower bound on this expression can be used:

$$\begin{aligned} P(E_j^*) &\geq \sum_{r=1}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-P_{sel})^{\frac{\lambda}{2}-r} \varphi_j(k)(1-\varphi_j(k))^{r-1} \\ &\geq \frac{\varphi_j(k)}{(1-\varphi_j(k))} \left(\sum_{r=0}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-\varphi_j(k))^r (1-P_{sel})^{\frac{\lambda}{2}-r} - (1-P_{sel})^{\frac{\lambda}{2}} \right) \\ &\geq \varphi_j(k) \left(\sum_{r=0}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-\varphi_j(k))^r (1-P_{sel})^{\frac{\lambda}{2}-r} - (1-P_{sel})^{\frac{\lambda}{2}} \right) \end{aligned}$$

Canceling out $\varphi_j(k)$ and moving the term $e^{-1} \leq (1-P_{sel})^{\frac{\lambda}{2}} \leq \frac{1}{\sqrt{e}} < 1$ to the other side, the LHS of the inequality becomes

$$\begin{aligned} P(E_j^*) &\geq \sum_{r=0}^{m_j} \binom{\frac{\lambda}{2}}{r} P_{sel}^r (1-\varphi_j(k))^r (1-P_{sel})^{\frac{\lambda}{2}-r} \\ &\geq (P_{sel}(1-\varphi_j(k)) + 1 - P_{sel})^{\frac{\lambda}{2}} \\ &= (1 - P_{sel}\varphi_j(k))^{\frac{\lambda}{2}} \end{aligned}$$

and the RHS is upper-bounded by

$$\frac{1}{\sqrt{e}} + \frac{\lambda P_{sel}}{2} = \frac{1}{\sqrt{e}} + o(\lambda^{c-1}) \text{by the argument below}$$

The LHS is lower-bounded by (using Bernoulli inequality for $\frac{\lambda}{2} \geq 1$):

$$P(E_j^*) \geq (1 - P_{sel}\varphi_j(k))^{\frac{\lambda}{2}} \geq 1 - \frac{\lambda P_{sel}\varphi_j(k)}{2}$$

Since one can select $P_{sel} = O(\lambda^{-c})$ and $\varphi_j(k) = O(n^{-c}), c \in \mathbb{Z}$, the expression is

$$P(E_j^*) = 1 - o(1) > \frac{1}{\sqrt{e}} + o(1) = P(E_j) \tag{5.5}$$

thus proving the lower bound on the probability of evolving exactly one more elite species for an arbitrary subset j. This logic applies for each of the M subsets (types of pairs) of the recombination pool, and the inequality becomes

$$P(E^*) = P\left(\bigcup_{j=1}^{M} E_j^*\right) > \sum_{j=1}^{M} P(E_j) = P(E) \tag{5.6}$$

The upper bound in Equation 5.2 follows directly.

5.3 Upper bounds on the OneMax test function

In this section the assumption is made that the starting fitness of strings is at most 3 (to avoid division by 0). This comes from considering three types of species in the population, unlike the approach in Chapter 4.

α : currently best species, i.e. species with the highest fitness so far

β : species with the next-best fitness value

γ : the remainder of the population

The Elitism Levels Traverse Mechanism requires identifications of only those pairs that can evolve a currently elite offspring, therefore pairs like $< \gamma, \gamma >$ are not considered. In fact, one of the questions that is addressed in this section, is whether the use of γ species is necessary at all.

5.3.1 Simple upper bound on OneMax

To derive the simple upper bound, first only two pairs are considered, neither of which makes use of γ species:

$$E_1 :< \alpha, \beta >$$
$$E_2 :< \beta, \beta >$$

Note the obvious pair $< \alpha, \alpha >$ as in Chapter 4 is not used since if bits are swapped successfully (i.e. in both strings either two 0- or 1-bits are selected), two currently elite offsprings are generated. In case in both parents bits with different values are selected, a better offspring evolves. Both of these cases are outside of the scope of the approach considered here. At the end of this Section though the possible extension of this approach to include $< \alpha, \alpha >$ pairs is discussed.

The probabilities of events $E_1,\ E_2$ are

$$P(E_1) = 2\binom{\frac{\lambda}{2}}{1}\varphi_1(k) \cdot \frac{\alpha}{\mu} \cdot \frac{\beta}{\mu}\left(1 - \frac{\alpha}{\mu}\right) = \frac{\alpha\beta\lambda(\mu-\alpha)\varphi_1(k)}{\mu^3}$$

$$P(E_2) = \binom{\frac{\lambda}{2}}{1}\varphi_2(k)\left(\frac{\beta}{\mu}\left(1 - \frac{\alpha}{\mu}\right)\right)^2 = \frac{\lambda\varphi_2(k)\beta^2(\mu-\alpha)^2}{2\mu^4}$$

Both equations yield the lower bound on the true probability, as per Equation 5.1. $P(E_1)$ is due to selecting exactly 1 α parent and 1 β parent (the order in the pair does not matter). $P(E_2)$ comes from selecting β parents in both cases. In the tournament selection this means that a β candidate has to be paired either with another β, or γ to enter the pool.

The probability of at least 1 of these events is

$$P(E(\alpha, k)) \geq P(E_1) + P(E_2) = \frac{2\varphi_1(k)\alpha\beta\lambda(\mu-\alpha)}{\mu^3} + \frac{\varphi_2(k)\lambda\beta^2\varphi_1(k)(\mu-\alpha)^2}{2\mu^4}$$

and, since $P(E(alpha, k))$ is the lower bound on the probability, the upper bound on the expected time to traverse enough levels of elitism (i.e. add enough currently elite species to the population) to evolve from fitness levels k to $k+1$ is

$$\mathbf{E}\tilde{T}_{\alpha,k} \leq \sum_{\alpha=1}^{\delta\mu} \frac{1}{P(E(\alpha,k))}$$

The following auxiliary notation is uses here:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \sum_{k=1}^{n-1} \mathbf{E}\tilde{T}_{\mu,k} = \sum_{k=1}^{n-1}\sum_{\alpha=1}^{\delta\mu} \mathbf{E}\tilde{T}_{\alpha,k} = \sum_{k=1}^{n-1}\sum_{\alpha=1}^{\delta\mu} \frac{1}{P(S_{\alpha,k})}$$

The expression for the mean time to observe $\delta\mu$ elite species in the population is obtained as a result of this setup (at this point β is pessimistically set to 1 to simplify the derivation):

$$\begin{aligned}
\mathbf{E}\tilde{T}_{\alpha,k} &\leq 2\mu^4 \sum_{\alpha=1}^{\delta\mu} \frac{1}{\beta\lambda(\mu-\alpha)(2\alpha\mu\varphi_1(k) - \alpha\beta\varphi_2(k) + \beta\mu\varphi_2(k))} \\
&< \frac{2\mu^4}{\lambda} \\
&\times \sum_{\alpha=1}^{\delta\mu} \frac{1}{(\varphi_2(k) - 2\mu\varphi_1(k))\alpha^2 + (2\mu^2\varphi_1(k) - 2\mu\varphi_2(k))\alpha + \mu^2\varphi_2(k)} \\
&= \frac{2\mu^4}{\lambda(\varphi_2(k) - 2\mu\varphi_1(k))} \sum_{\alpha=1}^{\delta\mu} \frac{1}{\alpha^2 + b_1\alpha + b_0}
\end{aligned} \tag{5.7}$$

where

$$\begin{aligned}
b_0 &= \frac{\mu^2\varphi_2(k)}{\varphi_2(k) - 2\mu\varphi_1(k)} \\
b_1 &= \frac{2\mu(\mu\varphi_1(k) - \varphi_2(k))}{\varphi_2(k) - 2\mu\varphi_1(k)}
\end{aligned}$$

In order to simplify the already complicated derivation, the summand in the expression for $\mathbf{E}\tilde{T}_{\alpha,k}$ is written in the form

$$\tilde{S}(\alpha,k)=\frac{1}{(\alpha+r)^2}=\frac{1}{(\alpha^2+2r\alpha+r^2)}$$

for some r. From equating coefficients it becomes clear that

$$r=\sqrt{b_0} \text{ or } r=\frac{b_1}{2}$$

and so, using the first root

$$\tilde{S}(\mu,k)=\sum_{\alpha=1}^{\delta\mu}\frac{1}{(\alpha+r)^2}=\psi_1(\sqrt{b_0})-\psi_1(\sqrt{b_0}+\delta\mu+1)$$

For large b_0 these expressions involving a digamma function can be expanded asymptotically in the Taylor series (only the first two terms are used):

$$\begin{aligned}\tilde{S}(\mu,k)&\approx\left(\frac{1}{b_0}-\frac{1}{2b_0}\right)-\left(\frac{1}{b_0}-\frac{\delta\mu}{b_0}-\frac{1}{2b_0}\right)=\frac{\delta\mu}{b_0}\\&=\frac{\delta\mu(\varphi_2(k)-2\mu\varphi_1(k))}{\mu^2\varphi_2(k)}=\frac{\delta(\varphi_2(k)-2\mu\varphi_1(k))}{\mu\varphi_2(k)}\end{aligned}$$

and therefore the expected time to traverse enough levels of elitism to improve 1 bit of the string is (substituting this expression into Equation 5.7)

$$\begin{aligned}\mathbf{E}\tilde{T}_{\mu,k}&=\frac{2\mu^4}{\lambda(\varphi_2(k)-2\mu\varphi_1(k))}\cdot\frac{\delta(\varphi_2(k)-2\mu\varphi_1(k))}{\mu\varphi_2(k)}\\&=\frac{2\mu^3\delta}{\lambda\varphi_2(k)}\end{aligned}$$

To improve the pair $<\beta,\beta>$ one needs to either swap 1 from the first parent and 0 from the second, or the other way around (any other outcome just keeps the current number of bits in each parent):

$$\varphi_2(k)=2\cdot\frac{k-1}{n}\cdot\frac{n-k+1}{n}=\frac{2(k-1)(n-k+1)}{n^2}$$

Substituting this into the expression for $\mathbf{E}\tilde{T}_{\mu,k}$, one obtains the expected optimization time of the algorithm, pessimistically assuming that at the beginning of the run the best species has only 2 1-bits anywhere in the string.

$$\begin{aligned}\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} &\leq \frac{\mu^3 n^2 \delta}{\lambda} \sum_{k=2}^{n-2} \frac{1}{(k-1)(n-k+1)} \\ &= \frac{\delta\mu^3 n^2}{\lambda} \cdot \frac{1}{n} \left(\sum_{k=2}^{n-2} \frac{1}{k-1} + \sum_{k=2}^{n-2} \frac{1}{n-k+1} \right) \\ &= \frac{\delta\mu^3 n}{\lambda} \left(\log(n-1) + O(1) \right) \end{aligned} \tag{5.8}$$

The second step is due to the partial fraction expansion. Although this seems quite a loose bound given cubic in μ, one can take $\mu = O(\lambda)$ so all it is left to establish is δ.

Obviously $0 < \delta < 1$, but one needs to select it s.t. summation over α makes sense. $\delta = \mu^{-\varepsilon_1}$ is set for an arbitrary $\varepsilon_1 > 0$ s.t. $\delta\mu = \mu^{1-\varepsilon_1} > 1$. Then $\delta\mu^2 = \mu^{2-\varepsilon_1} = \mu^{1+\varepsilon_2}$. For example, $\varepsilon_2 = \frac{1}{2}$ yields $\delta\mu = \sqrt{\mu}$ and $\delta\mu^2 = \sqrt{\mu^3}$.

The optimization time is

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\mu^{1+\varepsilon_2} n \log n) \tag{5.9}$$

or, in the number of function evaluations

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\lambda\mu^{1+\varepsilon_2} n \log n) \tag{5.10}$$

That is, runtime grows both in μ and λ. Nevertheless, if $\delta\mu = c = O(1)$, this bound reduces to linear in μ:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = O(\lambda\mu n \log n) \tag{5.11}$$

Hence, it is of interest to see if these bounds improve if all pairs of parents able to breed a currently elite offspring are considered.

5.3.2 Refined upper bounds on OneMax

In addition to pairs E_1 and E_2 in Subsection 5.3.1, two more are added, both involve the 'primitive' species γ:

$$E_1 :< \alpha, \beta >$$
$$E_2 :< \beta, \beta >$$
$$E_3 :< \alpha, \gamma >$$
$$E_4 :< \beta, \gamma >$$

The corresponding probabilities of evolving an α offspring are

$$P(E_1) = 2\binom{\frac{\lambda}{2}}{1}\varphi_1(k) \cdot \frac{\alpha}{\mu} \cdot \frac{\beta}{\mu}\left(1 - \frac{\alpha}{\mu}\right) \geq \frac{\alpha\lambda(\mu-\alpha)\varphi_1(k)}{\mu^3}$$
$$P(E_2) = \binom{\frac{\lambda}{2}}{1}\varphi_2(k)\left(\frac{\beta}{\mu}\left(1 - \frac{\alpha}{\mu}\right)\right)^2 \geq \frac{\lambda\varphi_2(k)(\mu-\alpha)^2}{2\mu^4}$$
$$P(E_3) = 2\binom{\frac{\lambda}{2}}{1}\frac{\alpha}{\mu} \cdot \left(1 - \frac{\alpha+\beta}{\mu}\right)^2\varphi_3(k) \approx \frac{\lambda\alpha}{\mu}\left(1 - \frac{\alpha}{\mu}\right)^2\varphi_3(k)$$
$$P(E_4) = 2\binom{\frac{\lambda}{2}}{1}\frac{\beta}{\mu}\left(1 - \frac{\alpha}{\mu}\right)\left(1 - \frac{\alpha+\beta}{\mu}\right)^2\varphi_4(k) \approx \frac{\lambda}{\mu}\left(1 - \frac{\alpha}{\mu}\right)^3\varphi_4(k)$$

$P(E_3)$ is due to selecting a γ species in addition of α: it is possible only if both candidates are γ. $P(E_4)$ is similar, but to select a β parent it must not compete against α. Again, to avoid overcomplications, only the lower bound on the number of the next-best species is considered, $\beta \geq 1$. The combined probability of evolution is ($P(E_1), P(E_2)$ are the same as in the previous derivation):

$$P(E(\alpha, k)) = P(E_1) + P(E_2) + P(E_3) + P(E_4)$$

and the expected time until there are $\delta\mu$ elite strings in the population:

$$\begin{aligned}\mathbf{E}\tilde{T}_{\mu,k} &\leq \sum_{\alpha=1}^{\delta\mu} \frac{1}{P(E(\alpha,k))} = 2\mu^4 \sum_{\alpha=1}^{\delta\mu} \frac{1}{b_3\alpha^3 + b_2\alpha^2 + b_1\alpha + b_0} \\ &= \frac{2\mu^4}{b_3} \sum_{\alpha=1}^{\delta\mu} \frac{1}{\alpha^3 + \frac{b_2}{b_3}\alpha^2 + \frac{b_1}{b_3}\alpha + \frac{b_0}{b_3}} \end{aligned} \tag{5.12}$$

where

$$\begin{aligned} b_0 &= \lambda\mu^2(2\mu\varphi_4(k) + \varphi_2(k)) \\ b_1 &= 2\lambda\mu(\mu\varphi_1(k) + \mu^2\varphi_3(k) - 3\mu\varphi_4(k) - \varphi_2(k)) \\ b_2 &= \lambda(\varphi_2(k) - 4\mu^2\varphi_3(k) - 2\mu\varphi_1(k) - 6\mu\varphi_4(k)) \\ b_3 &= 2\lambda(\mu\varphi_3(k) - \varphi_4(k)) \end{aligned}$$

One needs a solution to the cubic (in α) equation of the form

$$S(\mu,k) = \sum_{\alpha} S(\alpha,k) = \sum_{\alpha=1}^{\delta\mu} \frac{1}{\alpha^3 + b_2'\alpha_2 + b_1'\alpha + b_0}$$

where

$$\begin{aligned} b_2' &= \frac{\varphi_2(k) - 4\mu^2\varphi_3(k) - 2\mu\varphi_1(k) - 6\mu\varphi_4(k)}{2(\mu\varphi_3(k) - \varphi_4(k))} \\ b_1' &= \frac{\mu(\mu\varphi_1(k) + \mu^2\varphi_3(k) - 3\mu\varphi_4(k) - \varphi_2(k))}{2(\mu\varphi_3(k) - \varphi_4(k))} \\ b_0' &= \frac{\mu^2(2\mu\varphi_4(k) + \varphi_2(k))}{2(\mu\varphi_3(k) - \varphi_4(k))} \end{aligned}$$

Solution to $S(\mu,k)$ is of the form

$$S(\mu,k) = \sum_{\alpha} \frac{1}{(\alpha+\rho)^3} = \sum_{\alpha} \frac{1}{\alpha^3 + 3\alpha^2\rho + 3\alpha\rho^2 + \rho^3}$$

Equating the coefficients three roots for ρ are obtained:

$$\rho = \frac{b_2'}{3}$$
$$\rho = \pm\frac{\sqrt{b_1'}}{3}$$
$$\rho = \sqrt[3]{b_0'}$$

To simplify the increasingly complicated notation, only the last root is selected:

$$\begin{aligned} S(\mu,k) &= \sum_{\alpha=1}^{\delta\mu} \frac{1}{(\alpha+\sqrt[3]{b_0'})^3} = \frac{\psi_2(\sqrt[3]{b_0'}+\delta\mu+1)-\psi_2(\sqrt[3]{b_0'}+1)}{2} \\ &= \frac{1}{2}\left(\left(-\frac{1}{\sqrt[3]{b_0'^2}}+\frac{2\delta\mu+1}{b_0'}\right)-\left(-\frac{1}{\sqrt[3]{b_0'^2}}+\frac{1}{b_0'}\right)\right) \\ &= \frac{1}{2}\cdot\frac{2\delta\mu}{b_0'} = \frac{\delta\mu}{b_0'} \end{aligned}$$

The second line in the derivation was obtained by expanding both second-order polygamma functions in Taylor series as $b_0' \to \infty$ and taking the first two terms of each function. The front term in Equation 5.12 is combined with this derivation to obtain the expression of the upper bound on the expected time until the number of elite species in the population in k^{th} fitness level reaches $\delta\mu$:

$$\mathbf{E}\tilde{T}_{\mu,k} \leq \frac{2\mu^5\delta}{\lambda b_3 b_0'} = \frac{2\delta\mu^3}{\lambda(2\mu\varphi_3(k)+\varphi_4(k))}$$

since

$$b_3 b_0' = 2(\mu\varphi_3(k)-\varphi_4(k))\cdot\frac{\mu^2(2\mu\varphi_3(k)+\varphi_4(k))}{2(\mu\varphi_3(k)-\varphi_4(k))} = \mu^2(2\mu\varphi_3(k)+\varphi_4(k))$$

Finally, it is possible to find the upper bound on the expected optimization time of the algorithm:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} \leq \sum_{k=3}^{n-3}\mathbf{E}\tilde{T}_{\mu,k} = \frac{2\delta\mu^3}{\lambda}\sum_{k=3}^{n-3}\frac{1}{2\mu\varphi_3(k)+\varphi_4(k)} \tag{5.13}$$

Here again a pessimistic assumption is made that the best species at the start of the run has a fitness of 3, since in such case the fitness of γ has minimal fitness of 1 to avoid inconsistencies of the form $\frac{1}{0}$. Two probabilities are considered for the two new types of pairs:

$$\begin{aligned}\varphi_3(k) &= \frac{k}{n} \cdot \frac{k-2}{n} + \frac{n-k}{n} \cdot \frac{n-k+2}{n} \\ &= \frac{k(k-2) + (n-k)(n-k+2)}{n^2}\end{aligned}$$

The expression comes from the fact that it is necessary to preserve the 1-bits in the better parent in order to add its offspring to the population, so one needs to either select 1-bits in each parent or 0-bits in each parent. For the last swap probability, $\varphi_4(k)$, one needs only to select a 0-bit in the β parent and a 1-bit in γ parent, other options either degrade the better parent or leave the current fitness.

$$\varphi_4(k) = \frac{(n-k+1)(k-2)}{n^2}$$

Manipulating the summand over k:

$$\begin{aligned}S(\mu, k) &= \frac{1}{2\mu\varphi_3(k) + \varphi_4(k)} \\ &= \frac{n^2}{(4\mu-1)k^2 + (n - 8\mu - 4\mu n + 3)k + 4\mu n - 2n + 2\mu n^2 - 2} \\ &\leq \frac{n^2}{\mu} \cdot \frac{1}{k^2 - 4(n+2)k + 2n(n+1)}\end{aligned}$$

The first fraction is left out, and the denominator is factored in the form $(k - s)(k - r)$, s.t. s, r are solutions to the set of equations:

$$\begin{cases} s + r &= 4(n+2) \\ sr &= 2n(n+1) \end{cases}$$

The resulting solution (only the larger of the two roots that are symmetric around $2n$ is used) is:

$$\begin{cases} s &= 2n + \sqrt{2}\sqrt{n^2 + 7n + 8} + 4 \\ r &= 2n - \sqrt{2}\sqrt{n^2 + 7n + 8} + 4 \end{cases}$$

The value under the root can be bounded by

$$n+2 \leq \sqrt{n^2+7n+8} \leq n+4$$

So the expression becomes upper-bounded by

$$S(\mu,n) \leq \frac{n^2}{\mu} \frac{1}{(k-(2n+\sqrt{2}(n+2)))(k-(2n-\sqrt{2}(n+4)))}$$

Expanding this in partial fractions, two sums over k are obtained:

$$\begin{aligned} S_1(\mu,n) &= \sum_{k=3}^{n-3} \frac{1}{k-(2n+\sqrt{2}(n+2))} \approx \psi_0(n-2n-\sqrt{2}n) - \psi_0(3-2n-\sqrt{2}n) \\ &= \psi_0(-(1-\sqrt{2})n) - \psi_0(3-(2+\sqrt{2})n) = O(1) - O(1) = -O(1) \end{aligned}$$

The result of $-O(1)$ is due to the fact that any n can be selected, for which the values of digamma function are small negative constants. For the second sum, the upper bound on the value in the denominator, since $2-\sqrt{2} \approx 0.58 < 1$, is:

$$\begin{aligned} S_2(\mu,n) &= \sum_{k=3}^{n-3} \frac{1}{k-(2n-\sqrt{2}(n+2))} \leq \sum_{k=3}^{n-3} \frac{1}{k-n} = -\sum_{k=3}^{n-3} \frac{1}{n-k} \\ &\approx -\log(n-3) + O(1) \end{aligned}$$

the minus sign in front of the expression cancels out by multiplying $S_1(\mu,k)$ by $S_2(\mu,k)$ and the upper bound for $S(\mu,n)$ is obtained:

$$S(\mu,n) \leq \frac{n^2(\log(n-3)-O(1))}{\mu n}$$

and the upper bound on the expected first hitting time:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} \leq \frac{2\delta\mu^2 n(\log(n-3)-O(1))}{\lambda} \tag{5.14}$$

This equation is up to an order μ tighter than Equation 5.8, demonstrating the benefit of using 'primitive' species to traverse levels of elitism. Setting $\delta\mu = c =$

$O(1)$, the expression becomes (measured in the number of generations, for $c > 0$)

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \frac{c\mu n \log n}{\lambda} - O\left(\frac{\mu n}{\lambda}\right) \tag{5.15}$$

or, in the number of function evaluations,

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = c\mu n \log n - O(\mu n) \tag{5.16}$$

In addition to being tighter than the bound with two terms in subsection 5.3.1, it is also comparable to the results in [CHS+09, JDJW05, He10] (see below).

5.3.3 Use of $< \alpha, \alpha >$ pair

At the beginning of this section it was mentioned that this type of pairs, considered in Chapter 4 is excluded from analysis since it does not follow the idea of the worst upper bound in the Elitism Levels Traverse Mechanism. Although this approach is probably worth additional scrutiny, there is one thing to notice here.

In this chapter the worst upper bound for OneMax was derived, and turned out to be $O(\mu n \log n)$. The lower bound in Chapter 4 was found to be $\Omega(n \log n)$. Trivially, adding the new pair can only improve the upper bound, but it certainly cannot be better that the lower bound. And since the only difference is the μ term, improvement is only possible up to $O(\mu)$.

5.3.4 Generations vs Function evaluations

Tournament selection has a property that one does not need to evaluate every species, but it is necessary to make 2λ evaluations (since two species compete for 1 slot in the recombination pool, so the number of evaluations each generation is $O(\lambda)$. Therefore, in terms of the number of function evaluations the simple bound in Subsection 5.3.1 becomes $O(\mu\lambda n \log n)$ and the refined one in Subsection 5.3.2 $O(\mu n \log n)$. If $\mu = \lambda = O(1)$ this reduces to the well-known result of $O(n \log n)$ for OneMax function. The λ term in the denominator means that if the algorithm is run on parallel computers, the increase in the recombination pool size improves

the performance.

Another interesting feature, is that the refined case demonstrates the importance of making use of γ species, that turn out to improve the bounds.

5.3.5 Comparison to earlier results

The results in this Section can be compared to those in [CHS$^+$09] for $(N+N)$EA with mutation and a variant of tournament selection function, $O(nN\log N + n\log n)$ if measured in the number of function evaluations (Proposition 4). By setting $N = O(1) = c \geq 1$ this bound becomes $n\log n + O(n)$, which is larger than $cn\log n - O(n)$ in this section. If instead population size is set to $\mu = N = O(\sqrt{\log n})$ or $O(\frac{\log n}{\log\log n})$ the result in [CHS$^+$09] is sharper than here. For populations $\Omega(\frac{\sqrt{n}}{\log n})$ the bound in this section becomes sharper again, e.g., for $\mu = N = O(\sqrt{n})$ it is $cn^{\frac{3}{2}}\log n - n^{\frac{3}{2}}$, and in [CHS$^+$09] it is $0.5n^{\frac{3}{2}}\log n + O(n\log n)$.

The models in this section concerned only OneMax, a function without plateaus of fitness. In the remainder of this chapter it will be shown that the Elitism Levels Traverse Mechanism can be effectively applied to functions with fitness plateaus. It is also possible to approximate the limiting distribution of the number of super-elite species in the population.

5.4 Upper Bounds on the Royal Roads test function

The remainder of this chapter is dedicated to the upper bound on the runtime of $(\mu+\lambda)$EA$_{1BS}$ solving Royal Roads and stationary properties of the distribution of species. The definition of the problem and the approach in the part of using active bins are the same as in Chapter 4. Analysis here though has a number of substantial differences from both OneMax in Section 5.3 and Royal Roads in Section 4.8:

1. One of the most important differences is the random walk on the plateau

of fitness. As mentioned several times in Chapter 4, fitness-proportional selection function does not differentiate between species of the same fitness, but different distance to the next fitness level ('plateau of fitness'). Therefore, population performs random walk on this plateau. Hence the idea of artificial auxiliary levels has to be expanded to construct a better model.

2. Elitism that simplified the analysis on OneMax by considering only pure birth Markov Chain, is not straightforwardly applicable in this case. An additional measure has to be introduced to account for the difference in the number of bits set to 1 among all elite individuals.

3. Therefore, a birth-and-death MC has to be constructed with the number of states equal to the number of those species that are needed to evolve an offspring with a higher number of 1-bits in the active bin.

In essence, this new approach is a combination of a birth-and-death Markov Chain that tracks the progress of the population between jumps in the artificial auxiliary levels and the Elitism Levels Traverse Mechanism that determines the parameters of this MC. Therefore, it is necessary to introduce additional notation. If the current fitness of the best species in the population is k, then all α species can be partitioned into the following subsets:

α^* : species with the currently highest auxiliary value

β^* : species with the next highest auxiliary value

γ^* : the remainder of the elite subset

η : union of the sets β and γ

In the rest of the chapter α^* species are referred to as 'super-elite'. The explanation of the birth-and-death Markov chain follows in the next subsection. The last new definition necessary to add here is, similar to $\delta\mu$ for the OneMax function, $\delta^*\alpha$, i.e. the number of super-elite species necessary to advance to the next artificial auxiliary level with probability $1 - o(1)$.

To simplify the derivation, it is assumed that at least 1 super-elite species always remains in the population, i.e. the artificial auxiliary level does not degrade, unlike the number of α^* species. This assumption can be viewed in the following way: if the auxiliary value of the last super-elite species degrades (which is perfectly possible in real life), the auxiliary value of the population degrades with it to that of β^*. By this time, the number of the next-best species is large enough to regenerate an α^* species with probability $1 - o(1)$. This approach is used, for example, in epidemiology by having a constant rate ν that reintroduces virus into the population.

5.4.1 The birth-and-death Markov Chain for Royal Roads

For an introduction, good explanation and some advanced features of birth-and-death MCs see [Shi07a, Shi07b, Doo90, KS76].

Birth-and-death MC is defined for each artificial auxiliary level and has 1 absorbing state ($\delta^*\alpha$). By the assumption above, state 0 is excluded. The $\delta^*\alpha \times \delta^*\alpha$ transition matrix for this MC is

$$\mathbf{P} = \begin{pmatrix} r_{1,1} & p_{1,2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ q_{2,1} & r_{2,2} & p_{2,3} & 0 & \cdots & 0 & 0 & 0 \\ 0 & q_{3,2} & r_{3,3} & p_{3,4} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & q_{\delta^*\alpha-1,\delta^*\alpha-2} & r_{\delta^*\alpha-1,\delta^*\alpha-1} & p_{\delta^*\alpha-1,\delta^*\alpha} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

The dimensionality follows from the pessimistic assumption that each artificial auxiliary level starts with only one super-elite parent. The mean first hitting time of the absorbing state from any state in the MC is

$$m_{\alpha^*,\delta^*\alpha} = 1 + q_{\alpha^*,\alpha^*-1} m_{\alpha^*-1,\delta^*\alpha} + r_{\alpha^*,\alpha^*} m_{\alpha^*,\delta^*\alpha} + p_{\alpha^*,\alpha^*+1} m_{\alpha^*+1,\delta^*\alpha} \tag{5.17}$$

due to the assumption the boundary conditions are

$$m_{1,\delta^*\alpha} = 1 + r_{1,1}m_{1,\delta^*\alpha} + p_{1,2}m_{2,\delta^*\alpha} \tag{5.18}$$

and $m_{\delta^*\alpha,\delta^*\alpha} = 0$. A new recurrence expression is defined:

$$M_{\alpha^*} = m_{\alpha^*,\delta^*\alpha} - m_{\alpha^*+1,\delta^*\alpha}$$

This quantity is obviously nonnegative and the telescoping sum is

$$\sum_{\alpha^*=1}^{\delta^*\alpha-1} M_{\alpha^*} = m_{1,\delta^*\alpha}$$

Solving the recurrence equation one gets:

$$\begin{aligned}
&(p_{\alpha^*,\alpha^*+1} + q_{\alpha^*,\alpha^*-1})m_{\alpha^*,\delta^*\alpha} = 1 + q_{\alpha^*,\alpha^*-1}m_{\alpha^*-1,\delta^*\alpha} + p_{\alpha^*,\alpha^*+1}m_{\alpha^*+1,\delta^*\alpha} \\
&p_{\alpha^*,\alpha^*+1}(m_{\alpha^*,\delta^*\alpha} - m_{\alpha^*+1,\delta^*\alpha}) = 1 + q_{\alpha^*,\alpha^*-1}(m_{\alpha^*-1,\delta^*\alpha} - m_{\alpha^*,\delta^*\alpha}) \\
&p_{\alpha^*,\alpha^*+1}M_{\alpha^*} = 1 + q_{\alpha^*,\alpha^*-1}M_{\alpha^*-1} \\
&M_{\alpha^*} = \frac{1}{p_{\alpha^*,\alpha^*+1}} + \frac{q_{\alpha^*,\alpha^*-1}}{p_{\alpha^*,\alpha^*+1}}M_{\alpha^*-1}
\end{aligned} \tag{5.19}$$

Also:

$$m_{1,\delta^*\alpha} = \frac{1}{p_{1,2}} + m_{2,\delta^*\alpha}$$

and, therefore,

$$M_1 = \frac{1}{p_{1,2}}$$

Recurrently substituting in terms in the RHS of Equation 5.19, one obtains the expression for the general term M_{α^*}:

$$\begin{aligned}
M_{\alpha^*} &= \frac{1}{p_{\alpha^*,\alpha^*+1}}\left(1 + \frac{q_{\alpha^*,\alpha^*-1}}{p_{\alpha^*-1,\alpha^*}} + \cdots + \frac{q_{\alpha^*,\alpha^*-1} \cdot q_{\alpha^*-1,\alpha^*-2} \cdots q_{2,1}}{p_{\alpha^*-1,\alpha^*} \cdot p_{\alpha^*-2,\alpha^*-1} \cdot p_{1,2}}\right) \\
&= \frac{1}{p_{\alpha^*,\alpha^*+1}}\left(1 + \sum_{m=2}^{\alpha^*}\prod_{l=0}^{\alpha^*-m}\frac{q_{\alpha^*-l,\alpha^*-l-1}}{p_{\alpha^*-l-1,\alpha^*-l}}\right)
\end{aligned} \tag{5.20}$$

Summing on α^*, the LHS is simply the desired quantity, $m_{1,\delta^*\alpha}$:

$$m_{1,\delta^*\alpha} = \sum_{\alpha^*=1}^{\delta^*\alpha-1} \frac{1}{p_{\alpha^*,\alpha^*+1}} + \sum_{\alpha^*=1}^{\delta^*\alpha-1} \frac{1}{p_{\alpha^*,\alpha^*+1}} \sum_{m=2}^{\alpha^*} \prod_{l=0}^{\alpha^*-m} \frac{q_{\alpha^*-l,\alpha^*-l-1}}{p_{\alpha^*-l-1,\alpha^*-l}} \tag{5.21}$$

In the next subsection the RHS will be approximated, but first some simplification is necessary. The numerator of each fraction is a product of probabilities, so each of them is upper-bounded by q_{α^*,α^*-1}. The denominator can be simplified in the following way: if the probability to increase the number of super-elite species grows (which is proven at the beginning of the next subsection) for any $1 \leq \alpha^* \leq \delta^*\alpha$, the set of inequalities $p_{\delta^*\alpha-1,\delta^*\alpha} \geq p_{\delta^*\alpha-2,\delta^*\alpha-1} \geq \ldots \geq p_{1,2} \geq p_{1,2}^{\delta^*\alpha}$ can be used. Thus the upper bound bound on the first hitting time of $\delta^*\alpha$ super-elite species in the population is

$$m_{1,\delta^*\alpha} \leq \sum_{\alpha^*=1}^{\delta^*\alpha-1} \frac{1}{p_{\alpha^*,\alpha^*+1}} + \frac{\delta^*\alpha}{p_{1,2}^{\delta^*\alpha}} \sum_{\alpha^*=2}^{\delta^*\alpha-1} q_{\alpha^*,\alpha^*-1} \tag{5.22}$$

From this, the upper bound on the expected first hitting time of the algorithm will be derived:

$$\mathbf{E}\tau_{(\mu+\lambda)EA_{1BS}} = \sum_{k=0}^{K-1} \sum_{j=3}^{M-1} m_{1,\delta^*\alpha}(j,k) \tag{5.23}$$

For simplicity $\sum_k \sum_j \sum_{\alpha^*=1}^{\delta^*\alpha-1} \frac{1}{p_{\alpha^*,\alpha^*+1}}$ is referred to as the 'first expression' and $\sum_k \sum_j \frac{\delta^*\alpha}{p_{1,2}^{\delta^*\alpha}} \sum_{\alpha^*=2}^{\delta^*\alpha-1} q_{\alpha^*,\alpha^*-1}$ as the 'second expression'.

5.4.2 Upper bounds on the Royal Roads problem

The runtime of the algorithm is broken down into two phases. The first one begins after the random initialization of the population and ends when fitness of one of the offspring improves. The second phase begins immediately after this and ends when one of the offspring has maximal fitness. The rationale behind this is that the first bin is especially hard to solve, since all strings have fitness 0 and the fitness-proportional selection function is completely unguided.

Before the model is solved though, it is necessary to show the monotonicity of

p_{α^*,α^*+1} in α^* that is used extensively here. Since all swap probabilities $\varphi_1 \ldots \varphi_4 = O(1)$, one needs to show

$$\frac{p_{\alpha^*,\alpha^*+1}}{p_{\alpha^*-1,\alpha^*}} \geq 1 \; \forall \; \alpha^*$$

The numerator and denominator for all four types of probabilities are (since binomial coefficient $\binom{\frac{\lambda}{2}}{1}$ cancels out):

$$p_{\alpha^*,\alpha^*+1} = \frac{2\alpha^*}{4\mu^2} + \frac{2\alpha^*(\mu-\alpha^*-1)}{4\mu^2} + \frac{1}{4\mu^2} + \frac{2(\mu-\alpha^*-1)}{4\mu^2}$$

$$p_{\alpha^*-1,\alpha^*} = \frac{2(\alpha^*-1)}{4\mu^2} + \frac{2(\alpha^*-1)(\mu-\alpha^*)}{4\mu^2} + \frac{1}{4\mu^2} + \frac{2(\mu-\alpha^*)}{4\mu^2}$$

$$\frac{p_{\alpha^*,\alpha^*+1}}{p_{\alpha^*-1,\alpha^*}} = \frac{2\mu - 2\alpha^* + 2\mu\alpha^* - 2\alpha^{*2} - 1}{2\alpha^* + 2\mu\alpha^* - 2\alpha^{*2}} - 1 \geq 1 \text{ if } \alpha^* \leq \frac{\mu}{2}$$

It is shown in Section 5.4.3 that one needs only a relatively small number of super-elite species (independent of μ for $\lambda = \mu$ and certainly less than $\frac{\mu}{2}$) to produce an offspring with a higher auxiliary value.

Phase 1

As with OneMax, it is assumed that the auxiliary value of the first active bin is 3. This is done to avoid division by 0 in case of γ^* parents.

To use the Elitism Levels Traverse Mechanism, one needs to identify all pairs of parents that may add exactly one super-elite string to the population. These are: $< \alpha^*, \beta^* >, < \alpha^*, \gamma^* >, < \beta^*, \beta^* >, < \beta^*, \gamma^* >$. As the author is primarily concerned with the evolution of super-elite species, the trivial lower bound of $\beta^* \geq 1$ is used. Therefore the number of γ^* species is upper-bounded by $\mu - \alpha^* - 1$.

If a super-elite parent competes with another parent from the same level, the probability to get selected into the pool is halved (since fitnesses of α^*, β^* and γ^* are the same). In the first phase all species in the population have the same fitness, they differ only in the auxiliary function, thus super-elite species are relatively unlikely to be selected into the recombination pool. This is in contrast with OneMax, where α species are much more likely to get selected into the recombination pool.

The swapping probabilities for the respective type of pair are defined as $\varphi_1, \varphi_2, \varphi_3, \varphi_4$. The first sum in the expression becomes (first, second and fourth expression are multiplied by two since parents in the pair can be in any order). Here it also used $\mu - 1 > \frac{\mu}{2} \; \forall \; \mu \geq 2$.

$$p_{\alpha^*,\alpha^*+1} = \binom{\frac{\lambda}{2}}{1}\left(\frac{2\alpha^*}{4\mu^2}\varphi_1 + \frac{2\alpha^*}{2\mu}\left(\frac{\mu-1-\alpha^*}{2\mu}\right)\varphi_2 + \frac{1}{4\mu^2}\varphi_3 + \frac{2}{2\mu}\left(\frac{\mu-1-\alpha^*}{2\mu}\right)\varphi_4\right)$$

Many of the algebraic manipulations here were done in MATLAB, MATLAB SYMBOLIC TOOLBOX and WOLFRAM ALPHA.

The probabilities are defined as following:

$$\begin{aligned}
\varphi_1 &= \frac{j}{n^2}(2K-3+j) \\
\varphi_2 &= \frac{j}{n^2}(K-3+j) \\
\varphi_3 &= \frac{2(M+1-j)(2K-3+j)}{n^2} \\
\varphi_4 &= \frac{(M+1-j)(K-3+j)}{n^2}
\end{aligned}$$

The **first** expression is due to selecting a 1-bit in the active bin in α^* and a 1 anywhere in the second parent β^*. This uses the pessimistic assumption that in all bins β^* parents have only two 1-bits. The second one is the same, but the second parent is γ^*. The third one is due to selecting a 0-bit in the β^* species and a 1 anywhere in the second parent (also β^*, therefore multiplied by 2). The fourth swap probability is selecting a 0-bit in the β^* parent's active bin and 1 anywhere

from γ^*. Therefore,

$$\begin{aligned}
p_{\alpha^*,\alpha^*+1} &= \frac{\lambda}{8\mu^2}(2\alpha^*(\varphi_1 + (\mu - 1 - \alpha^*)\varphi_2) + \varphi_3 + (\mu - 1 - \alpha^*)\varphi_4) \\
&= \frac{\lambda}{8\mu^2 n^2}(j(2K - 3 + j) - \alpha^* j(K - 3 + j) + (\mu - 1)j(K - 3 + j)) \\
&+ (M + 1 - j)(2(2K - 3 + j) - 2\alpha^*(K - 3 + j) + 2(\mu - 1)(K - 3 + j)) \\
&= \frac{\lambda}{8\mu^2 n^2}(2j(\alpha^* - 1)(K + (K - 3 + j)(1 - \alpha^*) + 2j(\mu - 1)(K - 3 + j)(\alpha^* + 1))) \\
&\approx \frac{2\lambda j\alpha^*(K + (K - 3 + j)(\mu + 1 - \alpha^*))}{8\mu^2 n^2}
\end{aligned}$$

The **first** expression of the expectation of Phase 1 is

$$S_{11} = \sum_{j=3}^{M} \sum_{\alpha^*=1}^{\delta^*\alpha-1} \frac{1}{p_{\alpha^*,\alpha^*+1}(j)} = \frac{8\mu^2 n^2}{\lambda} \sum_{j=3}^{M} \sum_{\alpha^*=1}^{\delta^*\alpha-1} \frac{1}{2j\alpha^*(K + (K - 3 + j)(\mu + 1 - \alpha^*))}$$

The summand above is denoted $S_{11}(\alpha^*)$ (also $\sum_{\alpha^*=1}^{\delta^*\mu} S_{11}(\alpha^*) = S_{11}(\mu)$) and expanded in partial fractions w.r.t α^*:

$$\begin{aligned}
S_{11}(\alpha^*) &= \frac{1}{2j\alpha^*(K + (K - 3 + j)(\mu + 1 - \alpha^*))} \\
&\approx \frac{1}{2j(K + K\mu + j\mu)} \left(\frac{1}{\alpha^*} + \frac{K + j}{K + K\mu + j\mu - \alpha^*(K + j)} \right)
\end{aligned}$$

Sum over α^* of the first fraction in the brackets is of course $\log(\delta^*\alpha)$. The second fraction can be approximated in the following way (up to a constant):

$$\frac{K + j}{K + K\mu + j\mu - \alpha^*(K + j)} \approx \frac{1}{\mu - \alpha^*}$$

and the expression becomes

$$S_{11}(\mu) \approx \frac{1}{2j(K + K\mu + j\mu)} \left(\log \frac{\delta^*\alpha\mu}{\mu - \delta^*\alpha} \right)$$

Again, expanding the expression in partial fractions w.r.t. j, one gets

$$S_{11}(\mu) \approx \log\left(\frac{\delta^*\alpha\mu}{\mu-\delta^*\alpha}\right)\left(\frac{1}{2(K+K\mu)j} - \frac{\mu}{2(K+K\mu)(K+K\mu+j\mu)}\right)$$

Summing the first term w.r.t. j one gets $H_M \approx \log(M+1)$. The second term is at most $O(1)$. The upper bound for the **first** expression is (since $\frac{\mu}{\mu+1} \approx 1$ for large μ)

$$S_{11} < \frac{4\mu n^2 \log\left(\frac{\delta^*\alpha\mu}{\mu-\delta^*\alpha}\right)\log(M+1)}{\lambda K}$$

The **second** expression is

$$S_{12} = \frac{\delta^*\alpha}{p_{1,2}^{\delta^*\alpha}} \sum_{\alpha^*=2}^{\delta^*\alpha-1} q_{\alpha^*,\alpha^*-1}$$

The cumbersome term in the denominator in the **second** expression (assuming $r_{1,1} \leq p_{1,2}$ and using Bernoulli inequality) is simplified:

$$\frac{\delta^*\alpha}{p_{1,2}^{\delta^*\alpha}} = \frac{\delta^*\alpha}{(1-(1-p_{1,2}))^{\delta^*\alpha}} = \frac{\delta^*\alpha}{(1-r_{1,1})^{\delta^*\alpha}} \leq \frac{\delta^*\alpha}{1-\delta^*\alpha p_{1,2}}$$

The expression for $p_{1,2}$ can be upper-bounded in the following way:

$$\begin{aligned}
\delta^*\alpha p_{1,2} &= \frac{\lambda\delta^*\alpha}{2}\left(\frac{2j(j-1+2(K-1))}{4\mu^2 n2} + \frac{2(\mu-2)j(K-3+j)}{4\mu^2 n^2}\right. \\
&\quad + \frac{2(M+1-j)(2K-3+j)}{4\mu^2 n^2} \\
&\quad \left. + \frac{2(\mu-2)(M+1-j)(K-3+j)}{4\mu^2 n^2}\right) \\
&\leq \frac{\lambda\delta^*\alpha}{4\mu^2 n^2}\left(j(2K+j) + \mu j(2K+j) + (2M-j)(2K+j)\right. \\
&\quad \left. + \mu(2M-j)(2K+j)\right) \leq \frac{\lambda\delta^*\alpha}{4\mu^2 n^2}\left(2\mu j(2K+j) + 2\mu(2M-j)(2K+j)\right) \\
&= \frac{\lambda\delta^*\alpha M(2K+j)}{\mu n^2}
\end{aligned}$$

Trivially $j < M$, so the whole expression becomes (since there are at most $M-3$ bits to flip:

$$\sum_{j=1}^{M-3} \frac{\delta^*\alpha}{p_{1,2}^{\delta^*\alpha}} \leq \frac{\delta^*\alpha(M-4)}{1-\frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}}$$

Thus the front term in the **second** expression was simplified. Obviously, if the super-elite species degrades, it is replaced by its offspring (β^*). The lower bound on the probability of degrading is by pairing $< \alpha^*, \gamma^* >$.

$$\begin{aligned}
P_{sel} &= \frac{\alpha^*(\mu-1-\alpha^*)}{4\mu^2} \\
q_{flip} &= \frac{j}{n} \cdot \frac{n-(j-2)-(K-1)}{n} = \frac{j(n-K+3-j)}{n^2} \\
q'_{flip} &= 1 - q_{flip} \\
a &= \frac{\lambda j(n-K+3-j)}{n^2}
\end{aligned}$$

$$\begin{aligned}
\sum_{\alpha^*=2}^{\delta^*\alpha-1} q_{\alpha^*,\alpha^*-1} &= \sum_{\alpha^*=2}^{\delta^*\alpha-1} \binom{\frac{\lambda}{2}}{1} P_{sel}(1-P_{sel})^{(\frac{\lambda}{2}-1)} q_{flip} \\
&\leq \sum_{\alpha^*=2}^{\delta^*\alpha-1} \sum_{l=0}^{\frac{\lambda}{2}} \binom{\frac{\lambda}{2}}{l} P_{sel}^l (1-P_{sel})^{(\frac{\lambda}{2}-l)} q_{flip}^l \binom{l}{l} \\
&= \sum_{\alpha^*=2}^{\delta^*\alpha-1} (1-P_{sel}q'_{flip})^{\frac{\lambda}{2}} = \sum_{\alpha^*=2}^{\delta^*\alpha-1} (1-\frac{\alpha^*(\mu-1-\alpha^*)}{4\mu^2} q'_{flip})^{\frac{4\mu^2}{4\mu^2}\frac{\lambda}{2}} \\
&\leq \sum_{\alpha^*=2}^{\delta^*\alpha-1} e^{-\frac{\alpha^*(\mu-1-\alpha^*)\lambda q'_{flip}}{8\mu^2}} \approx \delta^*\alpha \int_0^1 e^{\frac{x(\mu-1)(\mu-1-(\mu-1)x)\lambda q'_{flip}}{8\mu^2}} dx \\
&\leq \delta^*\alpha \int_0^1 e^{-\frac{\mu^2 x(1-x)\lambda q'_{flip}}{32\mu^2}} dx = \delta^*\alpha \int_0^1 e^{-\frac{x(1-x)\lambda q'_{flip}}{32}} dx \\
&= \frac{\delta^*\alpha 2\sqrt{2} F\left(\frac{\sqrt{a}}{8\sqrt{2}}\right)}{\sqrt{a}}
\end{aligned}$$

The third step is due to the Binomial theorem $(a+b)^n = \sum_k \binom{n}{k} a^k b^{(n-k)}$ and the fourth one is the definition of the exponential function: $(1-\frac{a}{n})^n \leq e^{-a}$. The sum in the third line is approximated with an integral using Riemann sums. $F(\cdot)$ in the

last line is Dawson's integral. For small a, which are expected given that $j \leq M$, $a = O(\frac{1}{n})$ and Dawson's integral can be expanded in Taylor series:

$$\frac{\delta^*\alpha 2\sqrt{2}F\left(\frac{\sqrt{a}}{8\sqrt{2}}\right)}{\sqrt{a}} \approx \delta^*\alpha\left(1 - \frac{a}{192}\right) \leq \delta^*\alpha$$

Summing over j this becomes

$$\sum_{j=1}^{M-3}\sum_{\alpha^*=2}^{\delta^*\alpha-1} q_{\alpha^*,\alpha^*-1} = \delta^*\alpha(M-4)$$

and so the upper bound on the **second** expression is

$$S_{12} < \frac{\delta^*\alpha(M-4)}{1 - \frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}} \cdot \delta^*\alpha(M-4) = \frac{(\delta^*\alpha(M-4))^2}{1 - \frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}} \tag{5.24}$$

Therefore the upper bound on the mean first hitting time of Phase 1 becomes

$$\mathbf{E}T_1 \leq \frac{4\mu n^2 \log\left(\frac{\delta^*\alpha\mu}{\mu-\delta^*\alpha}\right)\log(M+1)}{\lambda K} + \frac{(\delta^*\alpha(M-4))^2}{1 - \frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}} \tag{5.25}$$

Phase 2

In this phase the remaining $K-1$ bins are solved. In addition to the four pairs in Phase 1, there are two more: $<\alpha^*, \eta>, <\beta^*, \eta>$. η are any species that are not elite, i.e. $\beta \cup \gamma$ using the terminology established before. Since all the expressions are already quite messy, the author simplifies further. First, one needs to reduce the number of types of parents considered.

In subsection 5.4.4 it is shown how both probabilities involving η can be lower-bounded by respective probabilities on pairs $<\alpha^*, \gamma^*>, <\beta^*, \gamma^*>$. Therefore in the expression for p_{α^*,α^*+1} these probabilities are multiplied by 2.

$$p_{\alpha^*,\alpha^*+1} \geq \binom{\frac{\lambda}{2}}{1}\left(\frac{2\alpha^*}{4\mu^2}\varphi_1 + \frac{4\alpha^*}{2\mu}\left(\frac{\mu-1-\alpha^*}{2\mu}\right)\varphi_2 + \frac{1}{4\mu^2}\varphi_3 + \frac{4}{2\mu}\left(\frac{\mu-1-\alpha^*}{2\mu}\right)\varphi_4\right)$$

The swap probabilities are

$$\begin{aligned}\varphi_1 &= \frac{j(kM+2K+j-3)}{n^2}\\ \varphi_2 &= \frac{j(kM+K+j-2)}{n^2}\\ \varphi_3 &= \frac{2(M-j+1)(kM+2K+j-3)}{n^2}\\ \varphi_4 &= \frac{(M-j+1)(kM+K+j-2)}{n^2}\end{aligned}$$

So p_{α^*,α^*+1} can be transformed accordingly:

$$\begin{aligned}p_{\alpha^*,\alpha^*+1} &\geq \frac{\lambda}{8\mu^2}(2\alpha^*\varphi_1+\varphi_3+4(\mu-1-\alpha^*)(\alpha^*\varphi_2+\varphi_4))\\ &= \frac{\lambda}{8\mu^2n^2}(2(kM+2K+j-3)(M+1+j(\alpha^*-1)\\ &+4(\mu-1-\alpha^*)(kM+K+j-2)(M+1+j(\alpha^*-1))))\\ &= \frac{\lambda}{4\mu^2n^2}(M+1+j(\alpha^*-1))(K+(kM+k+j-3)(2\mu+3-\alpha^*))\end{aligned}$$

The **first** expression in Phase 2 is

$$S_{21} = \frac{4\mu^2n^2}{\lambda}\sum_{k=1}^{K-1}\sum_{j=3}^{M}\sum_{\alpha^*=1}^{\delta^*\alpha-1}\frac{1}{(M+1+j(\alpha^*-1))(K+(kM+k+j-3)(2\mu+3-\alpha^*))}$$

since the rest of the calculations are quite similar to Phase 1, except for the straightforward sum over k, only the main result on the upper bound on the **first** term is stated here:

$$S_{21} < \frac{8\mu n^2\log\delta^*\alpha\log K\log(M+1)}{\lambda M}$$

Now for the **second** expression:

$$S_{22} = \sum_k \sum_j \frac{\delta^*\alpha - 1}{p_{1,2}^{\delta^*\alpha}} \cdot \sum_{\alpha^*} q_{\alpha^*,\alpha^*-1}$$

For the front term the similar trivial upper bound as in Phase 1, but also for the K term:

$$\sum_{k=1}^{K-1} \frac{\delta^*\alpha(M-4)}{1-\frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}} = \frac{\delta^*\alpha(M-4)(K-2)}{1-\frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}}$$

There are two ways to lose a super-elite species now, when pairing it with either γ^* or η. The probability to select a 0-bit in γ^* parent is

$$\frac{n-(K-1)-k(M-1)-(j-2)}{n} = \frac{n-K+3-k(M-1)-j}{n}$$

for η it is

$$\frac{n-(K-1)-(k-1)(M-1)}{n}$$

which is larger than the one for γ^*. The probability to select a 1-bit in the active bin in α^* parent is always $\frac{j}{n}$:

$$\begin{aligned}
S_{22} &= \sum_k \sum_j \sum_{\alpha^*=2}^{\delta^*\alpha-1} q_{\alpha^*,\alpha^*-1} = \binom{\frac{\lambda}{2}}{1} P_{sel1} q_{swap1} + \binom{\frac{\lambda}{2}}{1} P_{sel2} q_{swap2} \\
&\leq \sum_k \sum_j \sum_{\alpha^*=2}^{\delta^*\alpha-1} \lambda P_{sel1} q_{swap2} = \frac{\lambda}{\mu^2} \sum_{k=1}^{K-1} \frac{(n-K-1-(k-1)(M-1))}{n^2} \\
&\times \sum_{j=3}^{M-1} \frac{j}{4} \sum_{\alpha^*=1}^{\delta^*\alpha-1} \alpha^*(\mu-\alpha^*) \\
&= \frac{\lambda}{4\mu^2 n^2} \cdot \frac{M^2-M-6}{2} \cdot \frac{(3\mu-2\delta^*\alpha-1)\delta^*\alpha(\delta^*\alpha-1)}{6} \\
&\times \frac{(K-2)(2n-K-KM+M-3)}{2} \\
&\leq \frac{\lambda(M^2-M-6)(3\mu-2\delta^*\alpha-1)(\delta^*\alpha)^2(K-2)(2n-K-KM+M-3)}{96\mu^2 n^2}
\end{aligned}$$

The **second** expression for Phase 2 is

$$S_{22} < \frac{(M-4)(K-2)^2\lambda(M^2-M-6)}{\left(1-\frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}\right)} \times \frac{(3\mu-2\delta^*\alpha-1)(\delta^*\alpha)^3(2n-K-KM+M-3)}{96\mu^2n^2}$$

Combining both expression, the expected first hitting time for Phase 2 is upper-bounded by

$$\mathbf{E}T_2 < \frac{8\mu n^2\log\delta^*\alpha\log K\log(M+1)}{\lambda M} + \frac{(M-4)(K-2)^2\lambda(M^2-M-6)(3\mu-2\delta^*\alpha-1)(\delta^*\alpha)^3(2n-K-KM+M-3)}{\left(1-\frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}\right)96\mu^2n^2} \tag{5.26}$$

Combined runtime of the algorithm

Combining the expressions in Equations 5.25 and 5.26, the upper bound on the expected first hitting time for the whole algorithm is obtained:

$$\begin{aligned}\mathbf{E}\tau = \mathbf{E}T_1 + \mathbf{E}T_2 < {} & \frac{4\mu n^2\log\left(\frac{\delta^*\alpha\mu}{\mu-\delta^*\alpha}\right)\log(M+1)}{\lambda K} + \frac{(\delta^*\alpha(M-4))^2}{1-\frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}} \\ & + \frac{8\mu n^2\log\delta^*\alpha\log K\log(M+1)}{\lambda M} \\ & + \frac{(M-4)(K-2)^2\lambda(M^2-M-6)(3\mu-2\delta^*\alpha-1)(\delta^*\alpha)^3}{1-\frac{\lambda\delta^*\alpha M(2K+M)}{\mu n^2}} \\ & \times \frac{2n-K-KM+M-3}{96\mu^2n^2}\end{aligned} \tag{5.27}$$

Equation 5.27 looks quite cumbersome, but it can be reduced to same case with as in Chapter 4. By setting $K = M = \sqrt{n}$ and taking $\delta^*\alpha = O(1)$, as was the case

with $\delta\mu$ for OneMax, it is obtained:

$$\begin{aligned}\mathbf{E}\tau &\leq \frac{4\mu n^{\frac{3}{2}}\log n}{\lambda}\cdot O(1) + n\cdot O(1) + \frac{8\mu n^{\frac{3}{2}}\log^2 n}{\lambda}\cdot O(1) + \frac{\lambda n}{\mu}\cdot O(1)\\ &= O\left(\frac{\mu n^{\frac{3}{2}}\log^2 n}{\lambda}\right)\end{aligned} \tag{5.28}$$

or, measured in the number of function evaluations,

$$\mathbf{E}\tau = O(\mu n^{\frac{3}{2}}\log^2 n) \tag{5.29}$$

Comparing this result to similar ones in EA literature, including $2^{\sqrt{n}}\log n$ in [Mit96] and $O(\mu n^2)$ in [CHS+09], it is easy to notice that $(\mu+\lambda)\text{EA}_{1BS}$ outperforms other algorithms.

No less important, upper bounds need to be compared to the lower bounds in Chapter 4. It is easy to see that they are tight up to the μ term in the numerator, which can probably be explained by the (fairly loose) assumption of Uniform distribution. It is an encouraging result, and allows for the hypothesis that there is a different way to approach derivation of the lower bound to obtain $\Theta(\mu n^{\frac{3}{2}}\log^2 n)$ bound on RR, if measured in the number of function evaluations.

5.4.3 Proof of the lower bound on the probability of advancing to the next artificial auxiliary level

In this Chapter derivations of the runtime were based on the assumption that $\delta^*\alpha = O(1)$ and $\delta\mu = O(1)$, i.e. for some constant c that does not depend on α or μ. Here the bound on the probability to evolve a higher-ranked offspring is derived. The attention is restricted to Phase 1 only. Analysis of Phase 2 is similar.

Using the law of total probability on the probability of failure (F), i.e. probability that a species with higher auxiliary value does not evolve if the super-elite species haver already reached $O(1) = c$. There are three types of pairs that can

evolve: $< \alpha^*, \alpha^* >, < \alpha^*, \beta^* >, < \alpha^*, \gamma^* >$. Event A is defined that none of the three pairs get selected into the pool and event B that at least one of any pair gets selected. Since obviously $P(F|A) = 1$,

$$P(F) = P(F|A)P(A) + P(F|B)P(B) = P(A) + P(F|B)P(B)$$

Since there are c super-elite species in the population, the probability not to select any $< \alpha^*, \alpha^* >$ pairs is simply $(1 - \frac{c^2}{4\mu^2})^{\frac{\lambda}{2}}$. In a similar way, the probability not to select any of the other two types of pairs is, respectively, $(1-\frac{c}{4\mu^2})^{\frac{\lambda}{2}}$ and $(1-\frac{\mu-c-1}{4\mu^2})^{\frac{\lambda}{2}}$ using the trivial lower bound on the number of β^* parents (> 1). The product of these probabilities is

$$\begin{aligned} P(A) &\leq \left(1 - \frac{c^2}{4\mu^2}\right)^{\frac{\lambda}{2}} \left(1 - \frac{c}{4\mu^2}\right)^{\frac{\lambda}{2}} \left(1 - \frac{c(\mu - c - 1)}{4\mu^2}\right)^{\frac{\lambda}{2}} \\ &\leq e^{-\frac{c^2\lambda}{8\mu^2}} e^{-\frac{c\lambda}{8\mu^2}} e^{-\frac{c\lambda(\mu-c-1)}{8\mu^2}} = e^{-\frac{\lambda\mu c}{8\mu^2}} \end{aligned}$$

For $\lambda = \mu$, which is the usual choice in many applications, the probability of this event becomes upper bounded by:

$$P(A) \leq e^{-\frac{c}{8}}$$

the number of pairs of each type in the recombination pool is upper-bounded by (respectively) m_1, m_2, m_3, so the second part of the expression is the probability to select p pairs into the recombination pool and flip the bits unsuccessfully, which

is denoted by $\varphi_1^{'}, \varphi_2^{'}, \varphi_3^{'}$.

$$
\begin{aligned}
P(F|B)P(B) = &\sum_{p=1}^{m_1} \binom{\frac{\lambda}{2}}{p} \left(\frac{c^2\varphi_1^{'}}{4\mu^2}\right)^p \left(1-\frac{c^2}{4\mu^2}\right)^{\frac{\lambda}{2}-p} \sum_{p=1}^{m_2} \binom{\frac{\lambda}{2}}{p} \left(\frac{c\varphi_2^{'}}{4\mu^2}\right)^p \left(1-\frac{c}{4\mu^2}\right)^{\frac{\lambda}{2}-p} \\
&\times \sum_{p=1}^{m_3} \binom{\frac{\lambda}{2}}{p} \left(\frac{c(\mu-c-1)\varphi_3^{'}}{4\mu^2}\right)^p \left(1-\frac{c(\mu-c-1)}{4\mu^2}\right)^{\frac{\lambda}{2}-p} \\
&\leq \left(1-\frac{c^2}{4\mu^2}\right)^{\frac{\lambda}{2}} \left(1-\frac{c}{4\mu^2}\right)^{\frac{\lambda}{2}} \left(1-\frac{c(\mu-c-1)}{4\mu^2}\right)^{\frac{\lambda}{2}} \\
&\times \left(\sum_{p=0}^{m_1} \frac{(\frac{\lambda}{2})^p}{p!} \left(\frac{c^2\varphi_1^{'}}{4\mu^2-c^2}\right)^p - 1\right) \\
&\times \left(\sum_{p=0}^{m_2} \frac{(\frac{\lambda}{2})^p}{p!} \left(\frac{c\varphi_2^{'}}{4\mu^2-c}\right)^p - 1\right) \left(\sum_{p=0}^{m_3} \frac{(\frac{\lambda}{2})^p}{p!} \left(\frac{c(\mu-c-1)\varphi_3^{'}}{4\mu^2-c}\right)^p - 1\right) \\
&\leq e^{-\frac{c}{8}} \left(e^{\frac{\lambda c^2\varphi_1^{'}}{2(4\mu^2-c^2)}} - 1\right) \left(e^{\frac{\lambda c\varphi_2^{'}}{2(4\mu^2-c)}} - 1\right) \left(e^{\frac{\lambda c(\mu-c-1)\varphi_3^{'}}{2(4\mu^2-c(\mu-c-1))}} - 1\right)
\end{aligned}
$$

The front term is O(1). Taking $\max\{\varphi_1', \varphi_2', \varphi_3'\} = \varphi' \leq 1 - O(\frac{1}{n})$ one obtains the upper bound for the exponential term in the first bracket $e^{\frac{\lambda c^2}{2(4\mu^2-c^2)} - \frac{\lambda c^2}{2n(4\mu^2-c^2)}}$ and $e^{\frac{\lambda c}{2(4\mu^2-c)} - \frac{\lambda c}{2n(4\mu^2-c)}}$ in the second bracket. Asymptotically for $\mu, n \to \infty$ and $\mu = \lambda$ both of these terms converge to 1, so expressions in both brackets are $o(1)$. Following the same ideas, the exponential term in the last bracket is $O(1)$, so the whole expression is

$$P(F|B)P(B) = O(1) \cdot O(1) \cdot o(1) \cdot o(1) = o(1)$$

Combining the results above, one obtains the upper bound on the probability of failing to advance to the next artificial auxiliary level given c super-elite species in the population:

$$P(F) \leq e^{-\frac{c}{8}} + o(1) \tag{5.30}$$

And, therefore with probability of at least $1 - e^{-\frac{c}{8}} + o(1)$ the population progresses to a higher artificial auxiliary level by generating a higher-ranked offspring. Numerically it can be easily shown that for $c = 1$ this probability is roughly 0.1175

and for $c = 6 \approx 0.5276$. Obviously for $c = 8$ it is ≈ 0.6321.

5.4.4 Lower bounds on the probabilities involving η species in Phase 2

In Subsection 5.4.2 for lower bounds on probabilities for pairs $< \alpha^*, \eta >$ and $< \beta^*, \eta >$ in Phase 2 (where $\eta = \beta \cup \gamma$) the values for$< \alpha^*, \gamma^* >$ and $< \beta^*, \gamma^* >$ were used. Here it is shown more rigorously, for what values of α, the number of elite species, this bound is correct. First, the following expressions are compared:

$$P(< \alpha^*, \gamma^* >) = \binom{\frac{\lambda}{2}}{1} \frac{\alpha^*(\mu - 1 - \alpha^*)}{4\mu^2} \varphi_2$$
$$P(< \alpha^*, \eta >) = \binom{\frac{\lambda}{2}}{1} \frac{\alpha^*(\mu - \alpha)^2}{2\mu^3} \varphi_5$$

The swap probability $\varphi_5 > \varphi_2$ since to evolve an α^* offspring, one only needs to select a 0 in α^* active bin and a 0 in the second parent, and obviously there are more 0-bits in η than in γ^* and terms $\binom{\frac{\lambda}{2}}{1}$ and $\frac{\alpha^*}{2\mu}$ cancel out. What remains to be shown is for what $\alpha \geq \alpha^*$

$$\frac{(\mu - \alpha)^2}{\mu^2} \geq \frac{(\mu - 1 - \alpha^*)}{2\mu}$$

This is a quadratic inequality in α, so it is expressed as (set $\mu^2 + \mu + \mu\alpha^* = t$):

$$2\alpha^2 - 4\mu\alpha + t > 0$$

Since $t > 0, 4\mu > 0$ and $0 < 2 < \frac{16\mu^2}{4(\mu^2+\mu+\mu\alpha^*)}$, the only sensible solution is

$$1 < \alpha \leq \frac{4\mu - \sqrt{8}\mu}{4} \leq \frac{4\mu - \sqrt{16\mu^2 - 4 \cdot 2(\mu^2 + \mu + \mu\alpha^*)}}{4} \approx .29\mu$$

So for $1 < \alpha < .29\mu$ the inequality holds:

$$P(<\alpha^*, \eta>) > P(<\alpha^*, \gamma^*>)$$

The second expression,

$$P(<\beta^*, \gamma^*>) = \binom{\frac{\lambda}{2}}{1} \frac{1 \cdot (\mu - 1 - \alpha^*)}{4\mu^2} \cdot \frac{(M - j + 1)(kM + j - 2)}{n^2}$$
$$P(<\beta^*, \eta>) \geq \binom{\frac{\lambda}{2}}{1} \frac{1 \cdot (\mu - \alpha)^2}{2\mu^3} \cdot \frac{(M - j - 1)K}{n^2}$$

This is a different situation to the previous case, since $\varphi_6 \leq \varphi_4$ in this case, as it is obviously more likely to sample a 1-bit from γ^* rather than η. Note an extreme lower bound on the probability of sampling a 1-bit from η, $\frac{K}{n}$ is considered, which amounts to saying that at the initialization of the algorithm each bin starts with one 1-bit. Canceling out terms, a quadratic inequality in α is obtained:

$$6K\alpha^2 - 12\mu K\alpha + \mu(kM + j - 2)(1 + \alpha^* - \mu) + 6\mu^2 K > 0$$

Solving this, as before, there exists only one sensible solution:

$$1 < \alpha < \mu - \frac{\mu}{\sqrt{6}}\sqrt{\frac{kM + j - 2}{K}} \leq \mu - \frac{\sqrt{\frac{\mu(kM+j-2)(\mu-\alpha^*-1)}{K}}}{\sqrt{6}}$$

By taking $k = K, j = M - 1$ the worst upper bound on α is

$$1 < \alpha < \mu(1 - .4\sqrt{M + \frac{M}{K} - \frac{3}{K}})$$

which is valid for

$$1 - .4\sqrt{M + \frac{M}{K} - \frac{3}{K}} > 0 \Leftrightarrow K \geq \frac{M - 3}{6.25 - M}$$

that is, the derivations here are valid for only $M \leq 6$. Of course, most γ parents have more than K 1-bits, but neither assumptions about their distribution are

made, nor the change in their number is tracked (unlike super-elite species). If instead the number of 1-bits in η parents is equal to $\sqrt{MK}$, the bounds become valid for $M \leq 39$. More investigation is needed to make this approach more general.

5.5 Approximation of the quasi-stationary distribution of super-elite species in Phase 1

Throughout this thesis the question of the dynamics of the population, demonstrated by its elite subset, has played a substantial role. In Chapter 4 it was assumed to follow some distribution, in Chapter 5 the dynamics of its size was tracked. Finally, the question considered here can be formulated as the proof of some of the assumptions made before, e.g. distribution of super-elite species, thus in some way combining ideas from both of these Chapters.

In several areas of science, such as epidemiology and the study of computer viruses, Markov Chain models are widely applied to the study of the evolution and extinction of processes, e.g. spread of viral diseases (see [Nas99]) and the distribution of uninfected computers in a computer network (see [WM04]). These ideas were already used in Section 5.2 to explain the working of the Elitism Levels Traverse Mechanism. One of the best known models in this area is Susceptible-Infected-Susceptible (SIS), see [Nas99, WM04], which roughly corresponds to the birth-and-death MC that was constructed in Section 5.4.1. To the best of the author's knowledge this is the first application of this approach in EA community, although statistical distribution of the first hitting times was analyzed in [GKS99].

In this section the limiting distribution of the number of super-elite species in the population is approximated as $\delta^*\alpha, \mu \to \infty$ and $\delta^*\alpha = o(\mu)$. Markov chain is the same as in Section 5.4.1, which is aperiodic and time-homogeneous. Since there are no transitive states, presence of the absorbing state makes the stationary distribution trivial with the full mass of the limiting probability set to state $\delta^*\alpha$. The MC is transformed by adding the probability of moving from state $\delta^*\alpha$ back

to state 1 equal to 1 along the lines of the model in [WM04]. In the light of the set-up of the whole model this makes sense, since after the improvement in the auxiliary function the new number of super-elite species reduces back to 1 on the new auxiliary level.

It is also assumed that the MC is reversible (otherwise a set of recurrent equations similar to those in Section 5.4.1 have to be derived). The stationary distribution of an MC is defined as the limiting proportion of time spent by the stochastic process X_t in a state s_k:

$$\pi_k = \lim_{t\to\infty} P(X_t = s_k)$$

From this, using the set of detailed balance equations one can derive the expressions for stationary distributions:

$$\begin{aligned}
p_{1,2}\pi_1 &= q_{2,1}\pi_2 \\
p_{2,3}\pi_2 &= q_{3,2}\pi_3 \\
&\cdots\cdots \\
p_{\delta^*\alpha-2,\delta^*\alpha-1}\pi_{\delta^*\alpha-2} &= q_{\delta^*\alpha-1,\delta^*\alpha-2}\pi_{\delta^*\alpha-1} \\
\pi_2 &= \frac{p_{1,2}}{q_{2,1}}\pi_1 \\
\pi_3 &= \frac{p_{1,2}p_{2,3}}{q_{2,1}q_{3,2}}\pi_1 \\
&\cdots\cdots \\
\pi_{\alpha^*} &= \frac{p_{1,2}p_{2,3}\cdots p_{\alpha^*-1.\alpha^*}}{q_{2,1}q_{3,2}\cdots q_{\alpha^*,\alpha^*-1}}\pi_1
\end{aligned}$$

and, since π_{α^*} is a probability distribution:

$$\sum_{\alpha^*=1}^{\delta^*\alpha} \pi_{\alpha^*} = 1$$

which enables us to find the expression for π_1:

$$\pi_1 = \frac{1}{1 + \sum_{m=1}^{\delta^*\alpha-1} \prod_{l=1}^{m} \frac{p_{l,l+1}}{q_{l+1,l}}}$$

Ratios of probabilities serve as progress rates here. Two cases are considered: slow and fast progress rates. All derivations are for Phase 1. Findings for Phase 2 are similar.

5.5.1 Slow progress rate (Poisson approximation)

Derivation of π_{α^*} depends on the ratio $\frac{p_{l-1,l}}{q_{l,l-1}}$. This ratio can be written as a product

$$\frac{p_{l-1,l}}{q_{l,l-1}} = \frac{r(l)\theta_1}{s(l)\theta_2} \approx \frac{\rho}{l}$$

where θ_1, θ_2 do not depends on l. If the approximation above holds, and $\frac{\theta_1}{\theta_2} = \rho$ is not very large, the following limiting distribution of super-elite species can be derived.

$$\pi_1 = \frac{1}{1 + \sum_{m=2}^{\delta^*\alpha} \prod_{l=2}^{m} \frac{p_{l-1,l}}{q_{l,l-1}}} \approx \frac{1}{1 + \sum_{l=1}^{\delta^*\alpha} \frac{\rho^l}{l!}} = c^* e^{-\rho}$$

for some constant c^*. The ratio in the denominator accounts for quasi-stationarity: $p_{\delta^*\alpha,1} = 1$. Therefore, the quasi-stationary distribution of the super-elite species is

$$\pi_{\alpha^*} = c^* e^{-\rho} \frac{\rho^{\alpha^*}}{\alpha^*!}$$

which is a form of truncated Poisson distribution with removed origin (0) and the upper tail $(\delta^*\alpha + 1, \delta^*\alpha + 2, ...)$. If $\delta^*\alpha \to \infty$, c^* can be found:

$$c^* = \frac{1}{\sum_{\alpha^*=1}^{\infty} \pi_{\alpha^*}} = \frac{1}{1 - e^{-\rho}}$$

5.5.2 Fast progress rate (Normal approximation)

Here the limiting distribution of super-elite species as $\rho \to \infty$ is considered. A^* is used as a random variable for the super-elite species:

$$\pi_{\alpha^*} = \mathbf{P}(A^* = \alpha^*) = \frac{c^* e^{-\rho} \rho^{\alpha^*}}{\alpha^*!} = \frac{\rho^{\alpha^*}}{\alpha^*!} \pi_1$$

which, as mentioned before, is a form of truncated Poisson distribution with the removed origin (for comparison see [Nas96], Section 3). Characteristic function is

used to derive its expectation and variance:

$$\phi_{A^*}(t) = \mathbf{E}e^{itA^*} = \sum_{\alpha^*=1}^{\delta^*\alpha} e^{it\alpha^*}\pi_{\alpha^*} = \pi_1 \sum_{\alpha^*=1}^{\delta^*\alpha} e^{it\alpha^*}\frac{\rho^{\alpha^*}}{\alpha^*!} = \pi_1 e^{\rho e^{it}}$$

For $\delta^*\alpha \to \infty$. Standardizing constant is c^* (to account for the removed origin). By taking a derivative w.r.t t and setting $t = 0$:

$$\mathbf{E}A^* = \frac{1}{i}\phi'_{A^*}(0) = \left.\frac{d\mathbf{E}e^{itA^*}}{idt}\right|_{t=0} = \left.\pi_1\rho e^{\rho e^{it}}e^{it}\right|_{t=0} = \pi_1 e^{\rho}\rho = c^*\rho$$

In a similar way the asymptotic expression for the variance, $\mathbf{Var}A^*$ can be found:

$$\mathbf{Var}A^* = \mathbf{E}A^{*2} - (\mathbf{E}A^*)^2 = -\phi''(0) + (\phi'(0))^2$$

This uses the fact that $i^2 = -1$ and $\frac{1}{i} = -i$. Therefore,

$$\phi''(t) = i\pi_1\rho(ie^{it+\rho e^{it}} + i\rho e^{2it+\rho e^{it}}) = -\pi_1\rho(e^{it+\rho e^{it}} + \rho e^{2it+\rho e^{it}})$$
$$\phi''(0) = -\pi_1\rho(\rho+1)e^{\rho}, (\phi'(0))^2 = -\pi_1^2\rho^2 e^{2\rho}$$

Hence,

$$\mathbf{Var}A^* = \pi_1\rho(\rho+1)e^{\rho} - \pi_1^2\rho^2 e^{2\rho} = c^*\rho(1+\rho(1-c^*))$$

Since both expectation and variance of this random variable have been found, one can derive the limiting distribution of standardized A^*:

$$A' = \frac{A^* - \mathbf{E}A^*}{\sqrt{\mathbf{Var}A^*}}$$

Using the sum of Poisson random variables, one gets $\mathbf{E}A' = 0,\ \mathbf{Var}A' = 1$. Therefore:

$$A' = \frac{\sum_l A_l^* - (\mathbf{E}A_1^* + \mathbf{E}A_1^* + \ldots \mathbf{E}A_\rho^*)}{\sqrt{\rho\mathbf{Var}A_l^*}} = \frac{A_1^* - \mathbf{E}A_1^*}{\sqrt{\rho\mathbf{Var}A_l^*}} + \frac{A_2^* - \mathbf{E}A_2^*}{\sqrt{\rho\mathbf{Var}A_l^*}} + \ldots \frac{A_\rho^* - \mathbf{E}A_\rho^*}{\sqrt{\rho\mathbf{Var}A_l^*}}$$
$$= A'_1 + A'_2 + \ldots A'_\rho$$

where all A'_l are iid. The characteristic function of A' is

$$\begin{aligned}\phi_{A'}(t) &= \mathbf{E}e^{itA'} = \mathbf{E}e^{it\sum_l A'_l} = \mathbf{E}\prod_{l=1}^{\rho} e^{itA'_l} = (\mathbf{E}e^{itA'_l})^{\rho} \\ &= \left(\mathbf{E}\left(1 + itA'_l + \left(\frac{itA'_l}{2!}\right)^2 + O\left(\frac{1}{\rho^2}\right)\right)\right)^{\rho} \\ &= \left(1 - \frac{t^2}{2\rho}\right)^{\rho} \to e^{-\frac{t^2}{2}}\end{aligned}$$

which is a characteristic function of standard Normal distribution with parameters 0 and 1. The derivation was due to Taylor series expansion of the exponential function around 0 (since the function is dominated by ρ). Since A'_l are all identically distributed, their expectation is 0. Additionally,

$$\begin{aligned}\mathbf{E}A'_l &< \mathbf{E}A^* < \infty \\ \mathbf{E}(A'_l)^2 &= \frac{\mathbf{E}(A^*_l - \mathbf{E}A^*_l)^2}{\rho\mathbf{Var}A^*_l} = \frac{\mathbf{Var}A^*_l}{\rho\mathbf{Var}A^*_l} = \frac{1}{\rho} < \infty\end{aligned}$$

So the conditions for the Central Limit Theorem are fulfilled, and super-elite species converge to Normal distribution if the progress rate is high. As it turned out, to prove convergence of the transformed truncated Poisson distribution to Normal distribution, its expectation and second moment were not used.

5.6 Conclusions

A new tool for the analysis of population-based elitist EAs, the Elitism Levels Traverse Mechanism, was presented in this chapter, which was used to derive an upper bound on runtime of two $(\mu + \lambda)$ EAs with a recombination operator and a variant of tournament selection. The main idea of the tool is to identify pairs or species that are able to breed at most one currently elite offspring. The lower bound on this probability was proven, which enabled the derivation of the upper bound on the expected optimization time.

The Mechanism that was designed in this Chapter proved to be quite efficient in

deriving sharp upper bounds (worst case) for functions with and without plateaus and it is quite possible it can also yield tight upper bounds on other population-driven algorithms solving more complicated problems (e.g. with local optima).

Bounds on OneMax function were found to be $O(\frac{\mu n \log n}{\lambda})$, on an instance of Royal Roads with the size of the plateau equal to the number of bins, i.e. $K = M = \sqrt{n}\ O(\frac{\mu n^{\frac{3}{2}} \log^2 n}{\lambda})$. The former is a well-known bound, the latter is an improvement compared to similar algorithms. Additionally, it was shown that the probability to generate a higher-ranked offspring is lower-bounded by $1 - e^{-\frac{c}{8}} + o(1)$ if $\mu = \lambda$.

These results used the assumption that to advance to the next artificial fitness level with probability $1 - o(1)$ $\delta\mu$ elite species are needed in case of OneMax. For RR, $\delta^*\alpha$ super-elite species are needed to advance to the next artificial auxiliary level with the same probability.

It was shown that if the number of super-elite species is less than $\frac{\mu}{2}$, the positive effect from adding them continues to grow. What has not been shown for $(\mu + \lambda)$ algorithms yet, is the effect of the population size. The result for OneMax reduces to $O(\mu n \log n)$ if measured in the number of function evaluations that reduces to $O(n \log n)$ for $\mu = 1$. This also confirms many previous findings.

Perhaps for the first time in theoretical EA community limiting distributions of super-elite species were derived. For slow rates of progress ρ, it was approximated with truncated Poisson distribution (origin removed). For large ρ it converges to Normal distribution. In both cases the infinite-population approximation (i.e. $\delta^*\alpha \to \infty$) and quasi-stationarity of the underlying MC were used. It also turned out that the truncated Poisson distribution has expectation $c^*\rho$ and variance $c^*\rho(1 + \rho(1 - c^*))$ for some constant c^*.

Findings in this chapter can be extended in many ways:

1. Sharper approximation of the population structure. So far limiting distribution only of super-elite species in one algorithm on one function was derived,

but other subsets of the population, algorithms and functions are of interest too. It is also beneficial to derive the stationary distribution without the infinite population approximation.

2. Analysis of functions with traps and local optima. One can expect that such functions benefit from the population diversity more than those analyzed here, i.e. without local solutions.

3. Elitism rates comparison. In this chapter the rates of elitism were never really considered, i.e. the actual number of species saved in the population each generation, although numerical computation shows that it has a strong effect on the runtime. So far the author used the fact that all elite species are saved each generation, thus accumulating over time till $\delta\mu$ for OneMax or $\delta^*\alpha$ for Royal Roads. It would be interesting to compare elitism level 1 to 50% (as implemented in Chapter 3), i.e. if there is any difference if only 1 species is saved compared to half of the population.

4. Comparison of the structure of the population and recombination pool (which subsets are more likely to evolve).

5. Derivation of δ and δ^* to find the proportion of elite species that yields a $1 - o(1)$ probability of evolution. Quite obviously it is different for functions with plateaus and without.

6. Effect of the population size. It is of major interest to develop theoretical foundations for numerical results in Chapters 3 and 4 by showing a positive effect of the population size for algorithms solving functions with plateaus, at least when measured in the number of generations. So far only the positive effect of elite and super-elite species was shown.

Chapter 6

Summary, Conclusions and Future Work

There are many reasons to use EAs with population, recombination pools and recombination operators rather than $(1+1)$ algorithms with mutation. Among them are diversity and lower probability of premature convergence. Unfortunately, due to the complexity of analysis, most population-powered EAs have seen less attention in the theoretical EA community than they deserve. Most importantly, the knowledge of the structure of the population and recombination pool are almost non-existent.

Summary In this thesis an attempt was made to understand some of the processes observed in the population-based EAs by modeling and approximating the structure of the population. This knowledge was applied to derive certain properties of population-based algorithms. Some of these findings are recovery of known results using completely new approaches, some are improvements of past known results and some results are completely new. Most of findings here are bounds on runtime, but some also concern limiting distributions of species.

Most EAs analyzed in the theoretical community use only mutation-type genetic operators. Much of analysis in this thesis is dedicated to K-Bit-Swap (mostly for

K=1), which recombines genetic information between two parents in the recombination pool. In Chapter 3, by performing a large number of numerical experiments on different problems, it was shown to be efficient on many of them, such as Royal Roads. In terms of probability of finding the global solution, it outperformed many other EAs with crossover and mutation. At the same time, it was not very efficient on the Traveling Salesman Problem. The intuition behind this efficiency, is the uniform sampling of bits in each parent and the bias towards larger number of highly-fit parents in the recombination pool. In Section 3.3 these results were validated statistically using the bootstrap resampling technique. This motivated to study the runtime of EAs with KBS in greater depth.

In Chapter 4 EA using 1BS was compared to Randomized Local Search (RLS), which flips exactly one bit in each parent. Assuming that the unknown probability to observe a large number of elite species in the population is upper-bounded by the Uniform distribution, both algorithms were found to have the same lower bound on OneMax. On Royal Roads though RLS outperforms 1BS by a factor or $\Omega(\log n)$. In Section 4.9 four models (two algorithms on two test functions) were tested numerically. It was shown that the assumption of the Uniform upper bound is violated for OneMax and holds for the Royal Roads. For OneMax the increase in the population size does not improve the probability of finding the global solution within a set number of generations, thus confirming the well-known result for $(\mu + 1)$ algorithms. It also does not improve runtime (measured in the number of generations). Derived bounds for EA with 1BS on OneMax are better than for RLS, which are still tight up to a small constant (e.g. ≈ 4.5 for $n = 1000$).

Results for the Royal Roads are much more consistent: population size greatly improves performance (probability of finding the global maximum), distribution of elite species exhibits a form of exponential decay, which suggests that the Uniform distribution gives an upper bound on the probability of observing large number of elite species. In terms of the runtime, theoretical and numerical results are much more consistent: both exhibit improvement of performance and reduction of the population effect, i.e. as the population size grows larger the positive effect levels out. Analysis of the Royal Roads makes use of artificial auxiliary levels.

Such results motivated the study of the effect and evolution of elite species in Chapter 5. In Section 5.2 the Elitism Levels Traverse Mechanism was introduced and used to derive upper bounds for both test functions. It works by identifying pairs of parents that are able to breed at most one new currently elite offspring that is added to the population. Next, the expected time until a certain proportion is full of such species is derived, after which any of them improves to the next level of fitness with probability $1 - o(1)$. Although it sounds like a pessimistic approach that can give a loose bound, the bounds derived were quite sharp. In Section 5.4 the concept of super-elite species was introduced and developed in depth to derive a complicated expression on the runtime of $(\mu + \lambda)\text{EA}_{1BS}$ on the Royal Roads function, which improved the previously-known results and was consistent with the results in Chapter 4.

Finally, in Section 5.5 certain limiting distributions of super-elite species were considered that don't seem to have attracted much attention in EA community before. Applying ideas well-known in other areas of science, e.g. epidemiology, approximations of the limiting distributions of the super-elite species were derived, both for the slow and fast progress rates. The results in this chapter to some extent verified numerical results for the Royal Roads function in Chapters 4 and 5 by showing that often only a small number of super-elite species is needed for the evolution with high probability.

Conclusions The Elitism Levels Traverse Mechanism is a powerful and flexible tool for analyzing EAs that use population and recombination pool larger than 1. It can incorporate various genetic operators, although in this thesis only 1BS was considered. Since it focuses on the lower bound of the probability of evolving an additional currently best offspring, worst-case analysis can be easily extended to any function with single and multiple optima.

Perhaps the most important contribution of this work is the attempt to solve

some of the known problems in the area of analysis of EAs using tools and approaches that have not seen much attention before: structure of the population, recombination of information between individuals in the recombination pool, limiting distribution of the size of subsets. Results have either been improved, or confirmed, which, in the author's opinion, validates the adopted approach.

In this thesis the author was able to construct an approximation to some processes underlying the efficiency and success of population-based EAs. The complex area of the dynamics of the structure of EA populations is in its infancy, and although both theoretical and numerical results are very encouraging, during this work more questions have arisen that are discussed below. Among them are the dynamics and limiting distribution of subsets of the population, effect of the rates of elitism, comparison of the recombination pool and population structures, sharper approximation of the population structure based on its fitness or other benchmarks (in this thesis only three types of species have been analyzed based on artificial fitness and auxiliary levels).

Implications of Results and Future Work Of all possible ways to expand the results in this thesis the author suggests that the following problems may of great interest.

1. Limiting distribution of different types of species in the population. As shown above, even primitive γ species can affect performance, so it is interesting to see if their presence in the population converges asymptotically to any distribution at all.

2. Extension of findings to KBS with $K > 1$. Numerical results show that often higher values of K improve performance, so logically one would expect that theoretical findings should confirm this, at least for some type of problems.

3. Extension of findings to different rates of elitism. Numerical results also show

that the change in the rate of elitism from 1 to 50 % improve the performance.

4. Results in Chapter 5 can be proven more rigorously. For example, for the Royal Roads function one can derive the number of super-elite individuals that ensures evolution of a higher-ranked offspring w.p. $1-o(1)$, rather than $1-O(1)$.

5. The approach using the Elitism Levels Traverse Mechanism can be adopted to find sharper lower bounds, as it clear that both the assumption of Uniform rv and the missing μ term in the main results of Chapter 4 can be improved.

6. Of course, the ideas behind the Elitism Levels Traverse Mechanism can be applied to other genetic operators and other problems, including problems with different encoding and local solutions, e.g. TSP, TwoMax and trap functions.

The author is convinced that better understanding of the population structure and dynamics will shed light on the work and efficiency of Evolutionary Algorithms and their application in real life.

Appendix A

Results of Numerical Experiments

In this appendix plots of numerical results of experiments in Chapter 3 are presented.

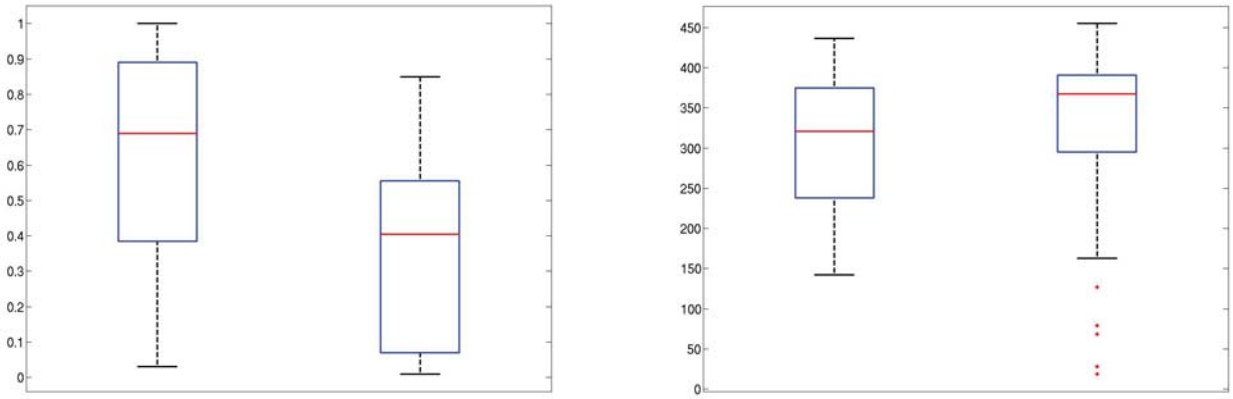


Figure A.1: Conditional probability of success and runtime of $(\mu + \lambda)\text{EA}_{KBS}$ vs $(\mu + \lambda)\text{EA}_{-KBS}$ on the Rosenbrock test function

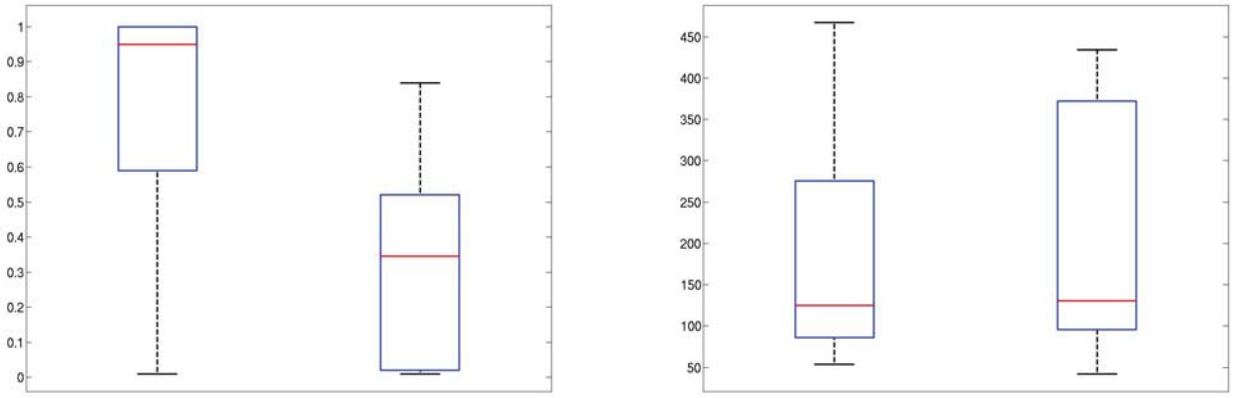


Figure A.2: Conditional probability of success and runtime of $(\mu + \lambda)\text{EA}_{KBS}$ vs $(\mu + \lambda)\text{EA}_{-KBS}$ on the Rastrigin test function

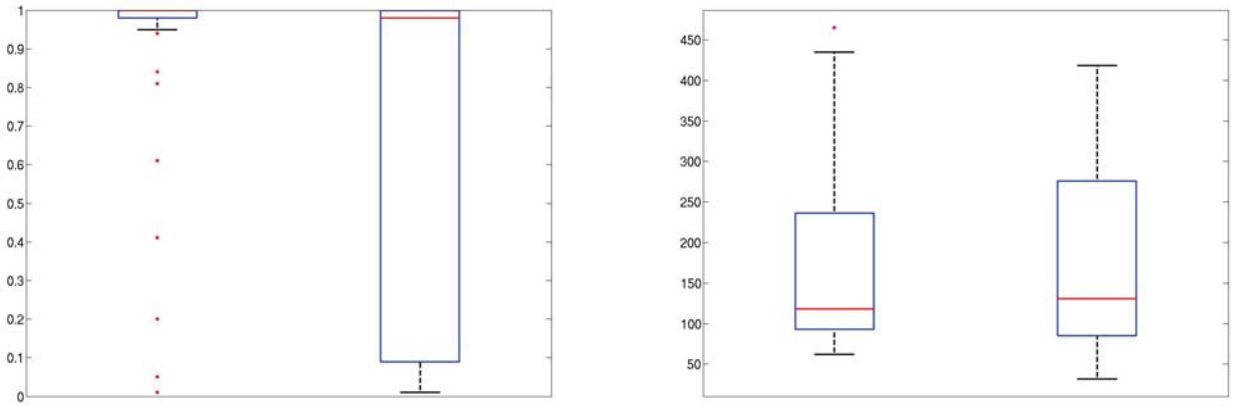


Figure A.3: Conditional probability of success and runtime of $(\mu + \lambda)\text{EA}_{KBS}$ vs $(\mu + \lambda)\text{EA}_{-KBS}$ on the Ackley test function

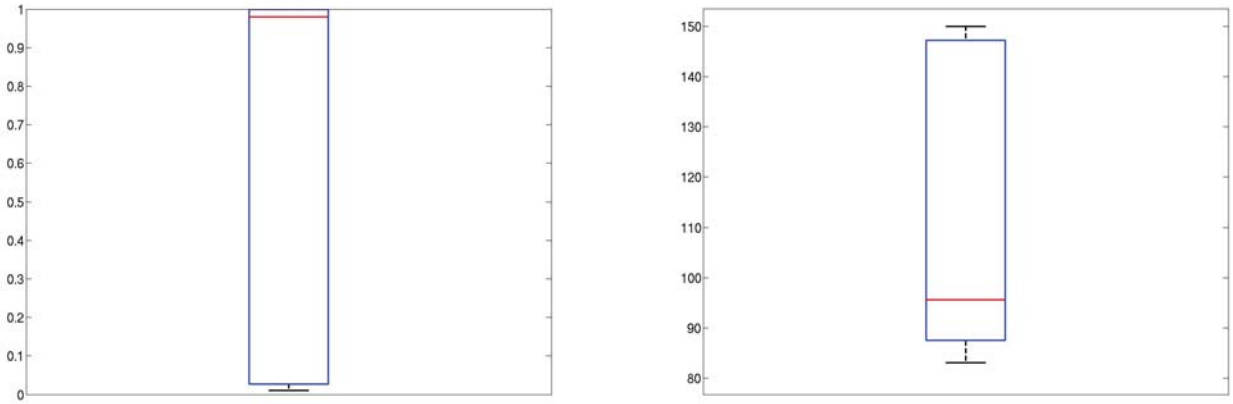


Figure A.4: Conditional probability of success and runtime $(\mu + \lambda)\text{EA}_{KBS}$ on the Royal Roads test function. Algorithms with other parameter settings do not solve the problem in the set number of generations.

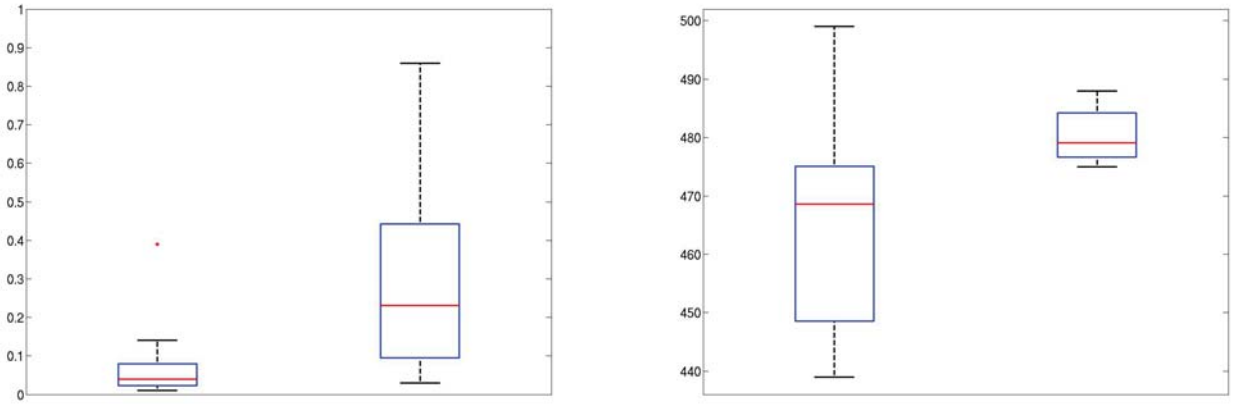


Figure A.5: Conditional probability of success and runtime of $(\mu + \lambda)\text{EA}_{KBS}$ vs $(\mu + \lambda)\text{EA}_{-KBS}$ on the Four Peaks test function

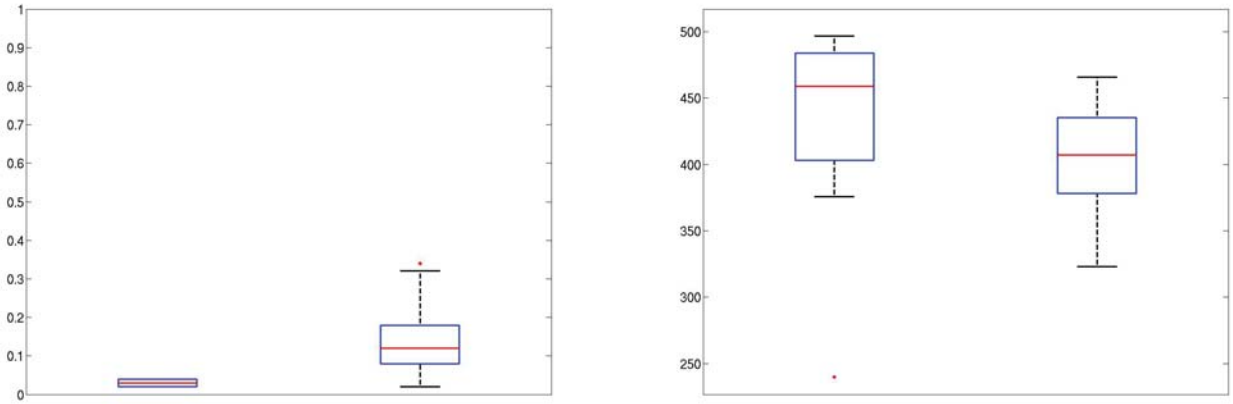


Figure A.6: Conditional probability of success and runtime of $(\mu + \lambda)\text{EA}_{KBS}$ vs $(\mu + \lambda)\text{EA}_{-KBS}$ on the trivial TSP

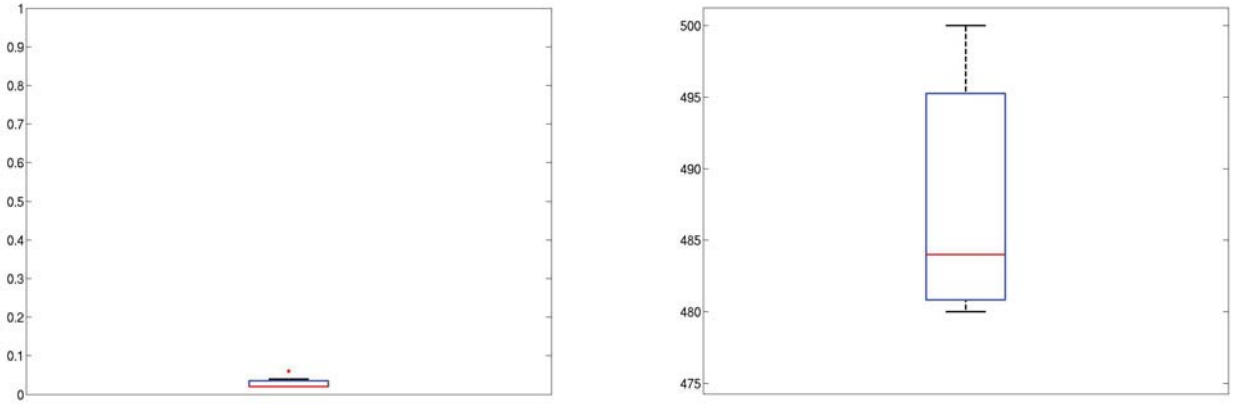


Figure A.7: Conditional probability of success and runtime of $(\mu + \lambda)\text{EA}_{-KBS}$ on the TSP on US Capital Cities. Algorithms with KBS do not solve the problem in the set number of generations

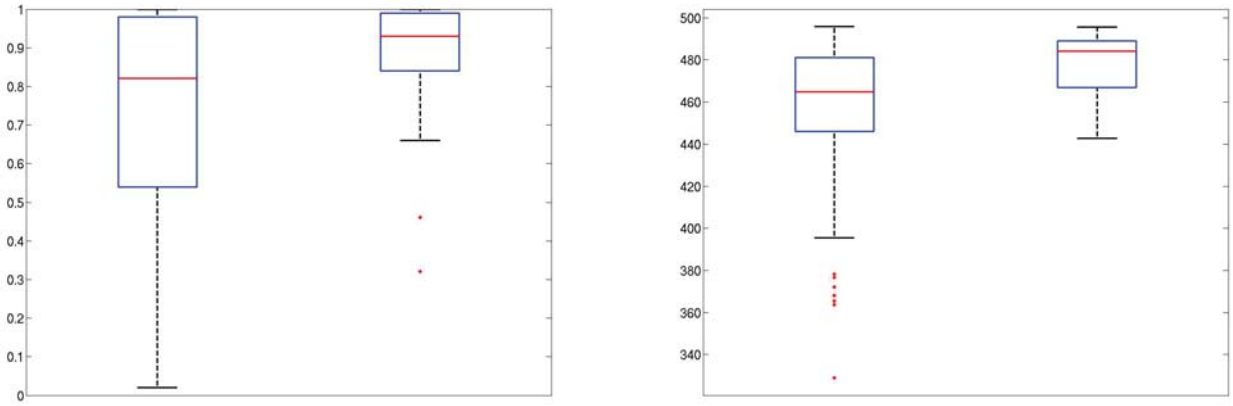


Figure A.8: Probability of success $(\mu+\lambda)\text{EA}_{KBS}$ vs $(\mu+\lambda)\text{EA}_{-KBS}$ on the trivial k-means clustering problem

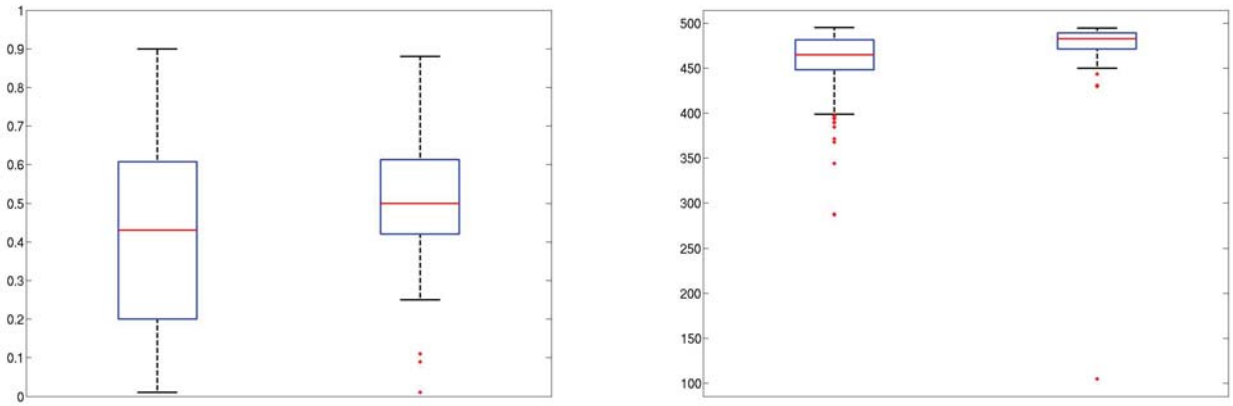


Figure A.9: Probability of success $(\mu+\lambda)\text{EA}_{KBS}$ vs $(\mu+\lambda)\text{EA}_{-KBS}$ on the random k-means clustering problem

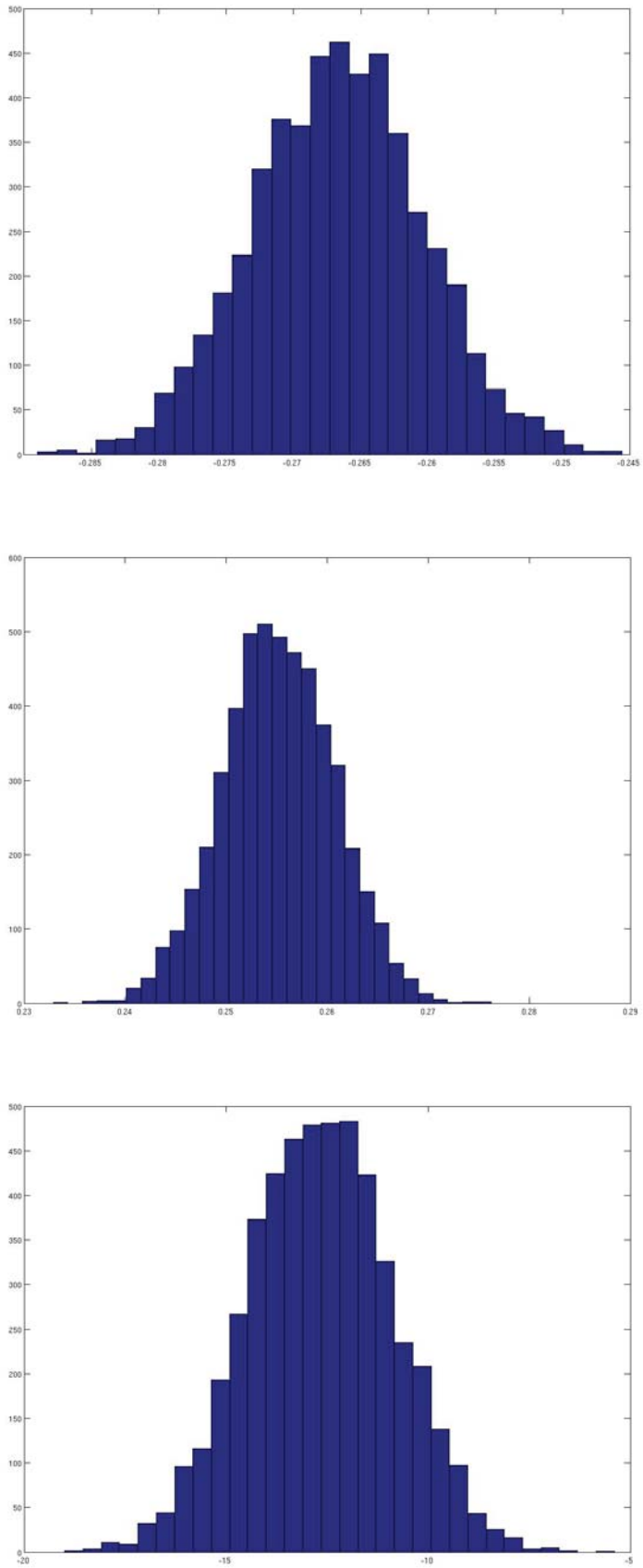


Figure A.10: Histograms of bootstrap estimate of the difference in means for the Rosenbrock function: probability of failure, conditional probability of success, runtime

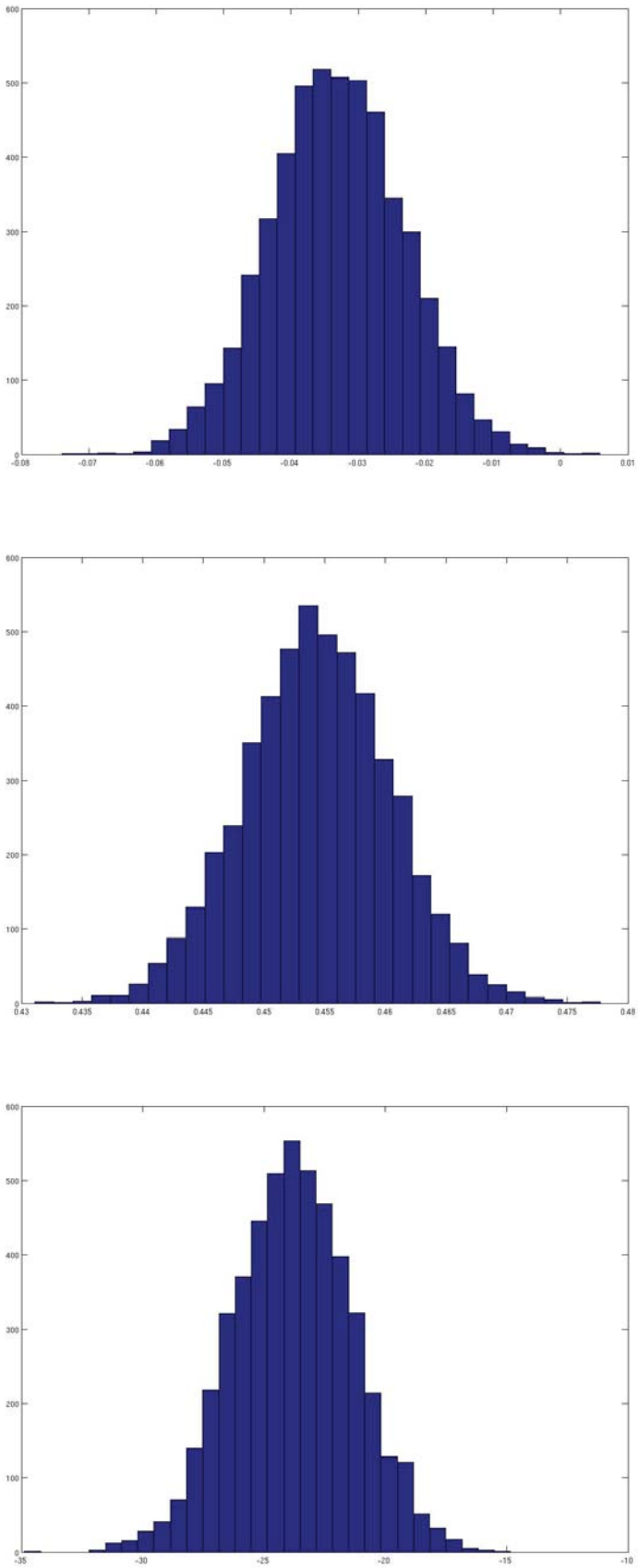


Figure A.11: Histograms of bootstrap estimate of the difference in means for the Rastrigin function: probability of failure, conditional probability of success, runtime

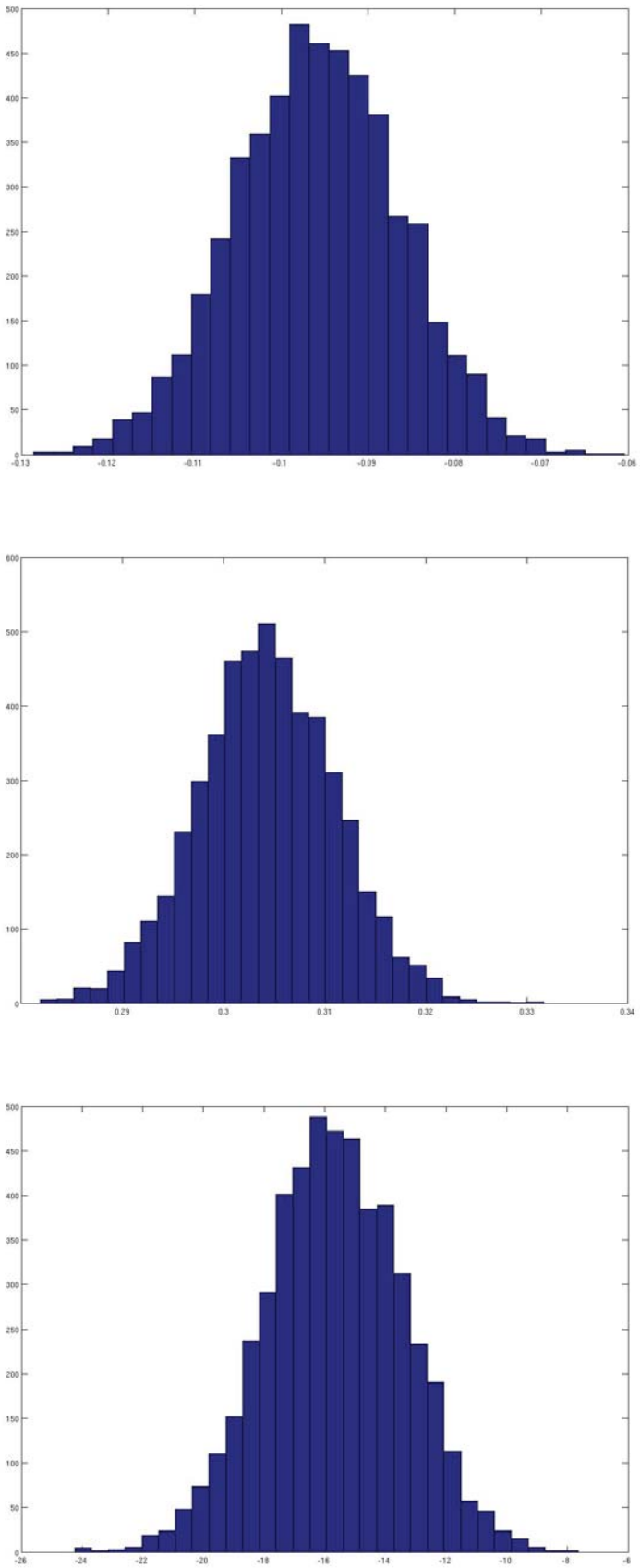


Figure A.12: Histograms of bootstrap estimate of the difference in means for the Ackley function: probability of failure, conditional probability of success, runtime

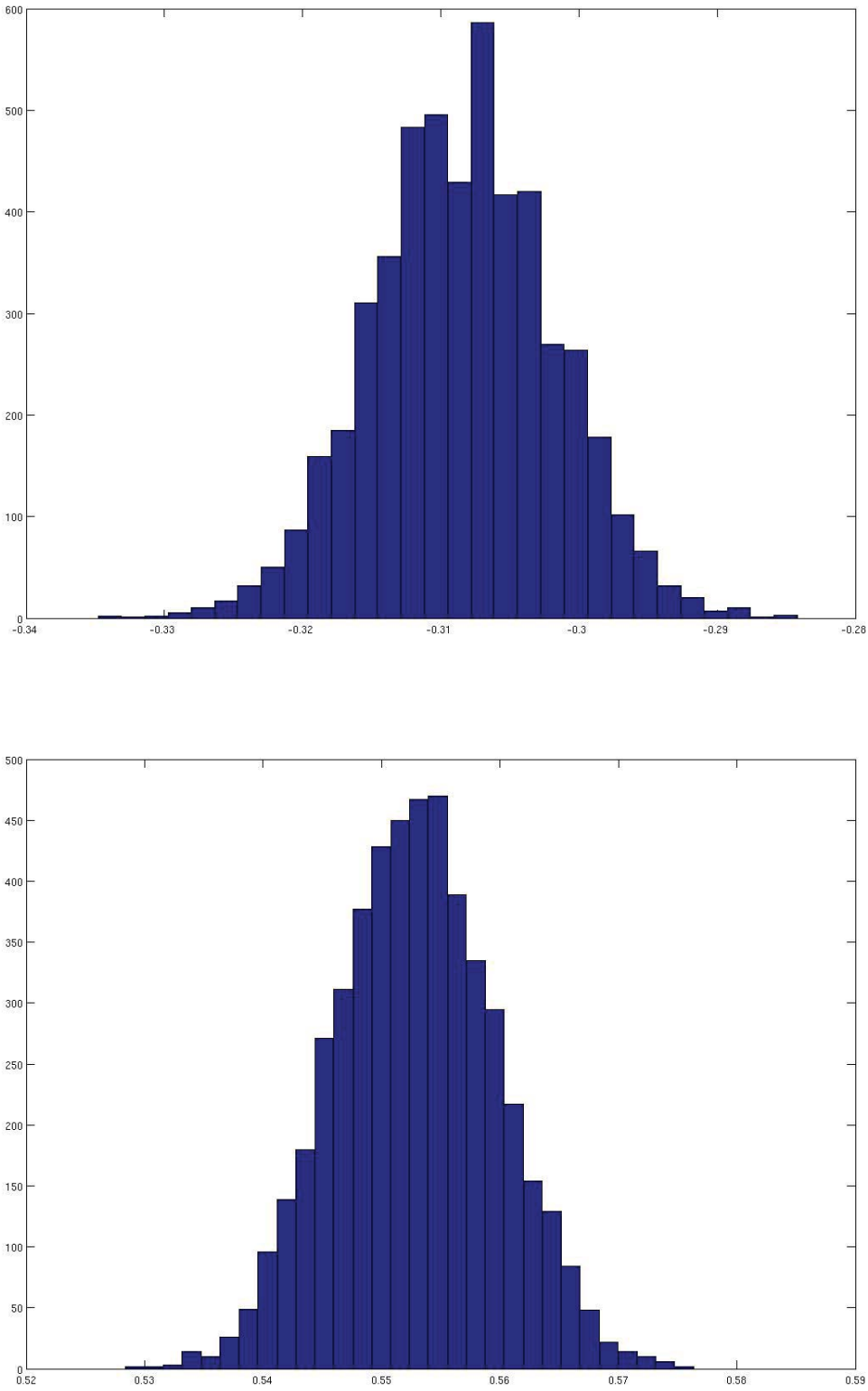


Figure A.13: Histograms of bootstrap estimate of the difference in means for the Royal Roads function: probability of failure, and conditional probability of success

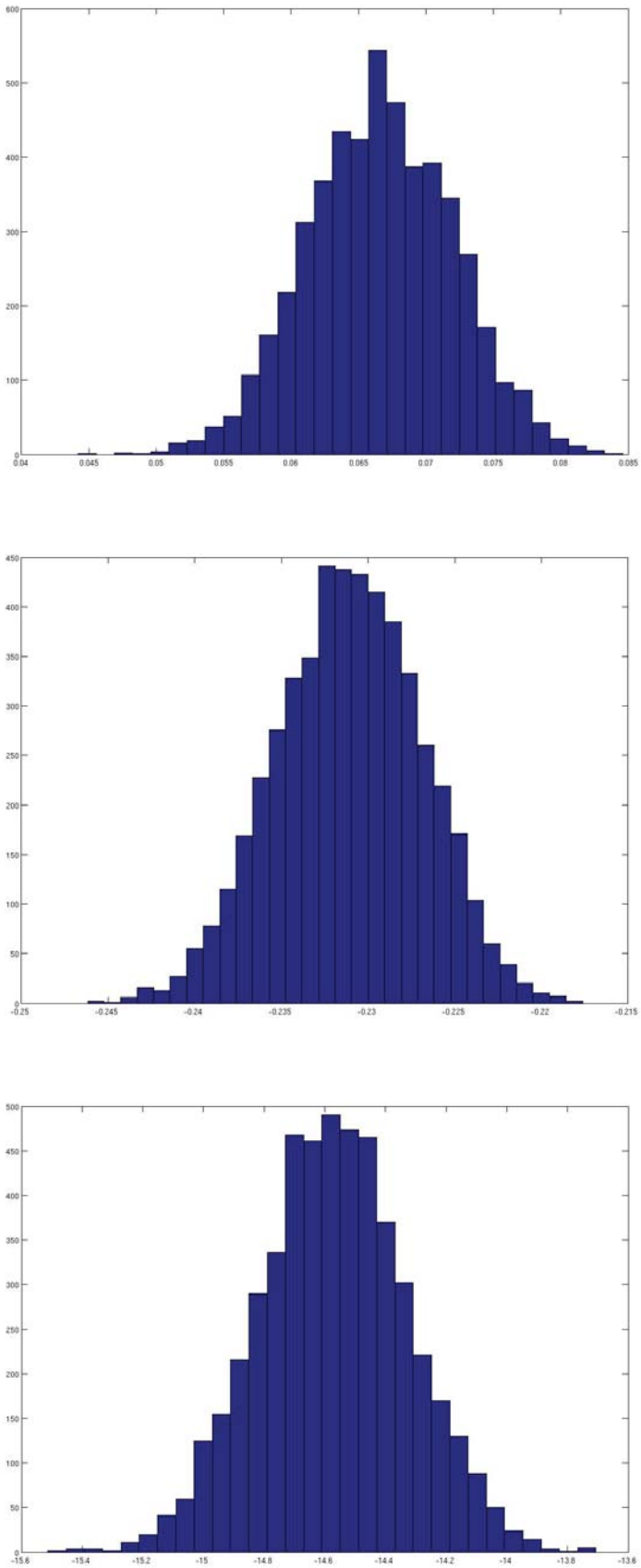


Figure A.14: Histograms of bootstrap estimate of the difference in means for the Four Peaks function: probability of failure, conditional probability of success, runtime

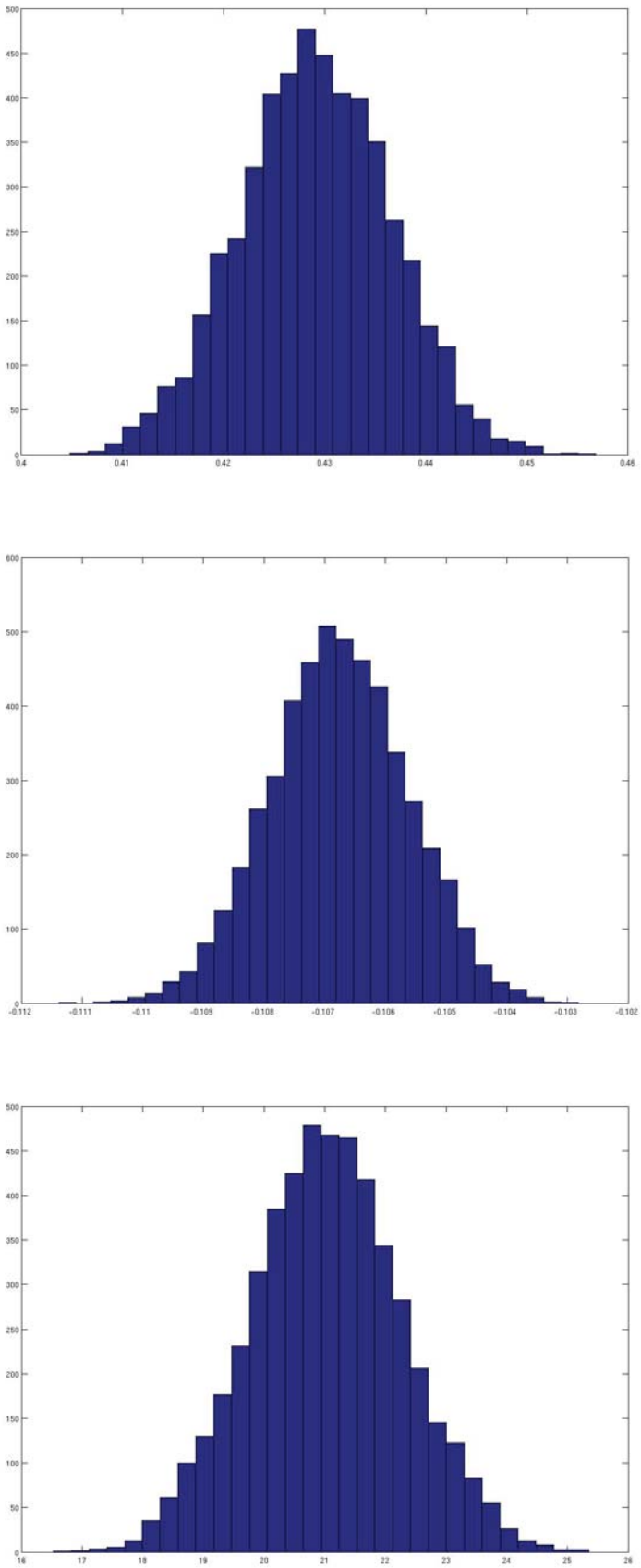


Figure A.15: Histograms of bootstrap estimate of the difference in means for the TSP on a circle: probability of failure, conditional probability of success, runtime

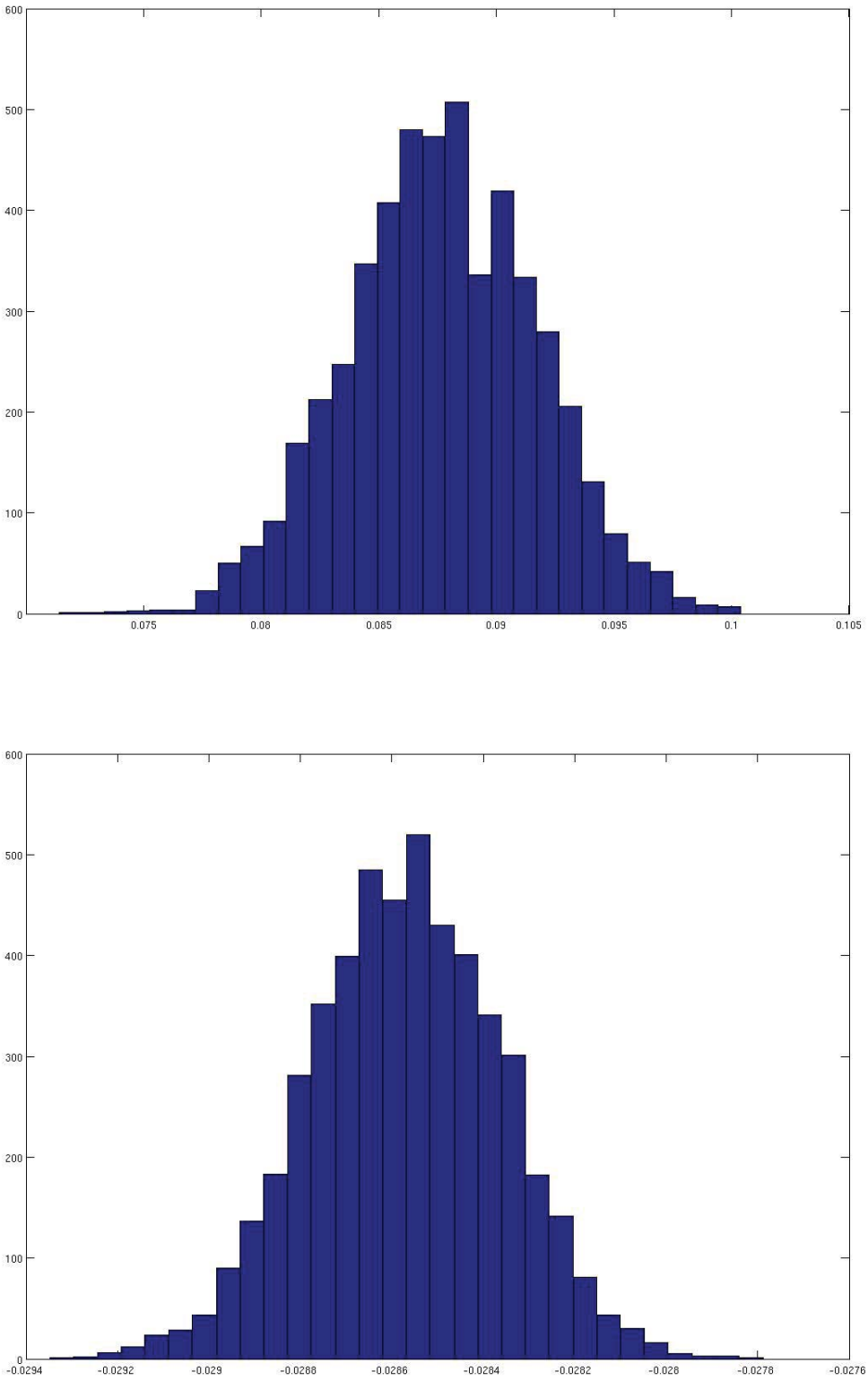


Figure A.16: Histograms of bootstrap estimate of the difference in means for the TSP on US Cities: probability of failure and conditional probability of success

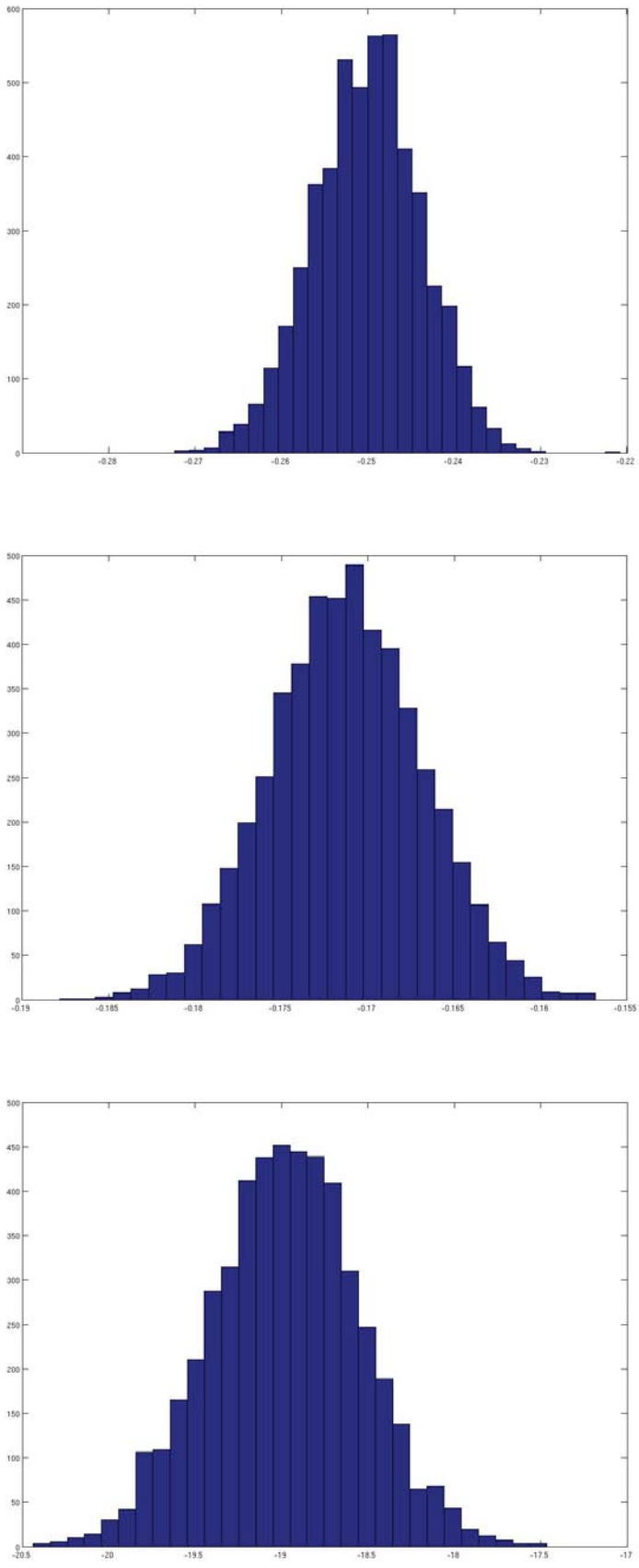


Figure A.17: Histograms of bootstrap estimate of the difference in means for the trivial k-means clustering problem: probability of failure, conditional probability of success, runtime

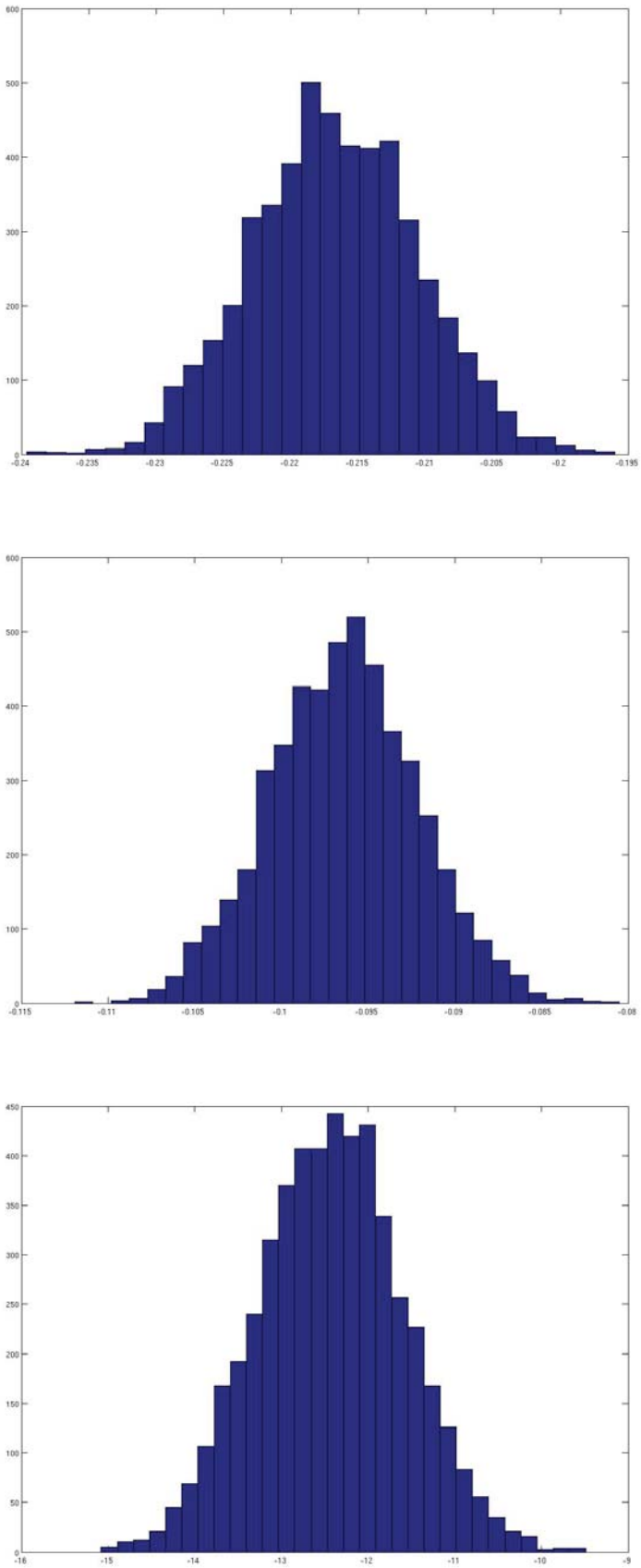


Figure A.18: Histograms of bootstrap estimate of the difference in means for the random k-means clustering problem: probability of failure, conditional probability of success, runtime

Appendix B

Concepts from Probability Theory

Concepts in this Appendix were used throughout the thesis: the law of total probability in Chapter 4 to derive the probability of failure, Bernoulli, Binomial and CLT in Chapter 5. Ideas from Markov Chain theory were used extensively in both of these chapters.

Law of Total Probability For an event A and a set of events B_i s.t. $\cup_i B_i = \Omega$, $\forall i, j \ B_i \cup B_j = \emptyset$ where Ω is the sample space the following is true:

$$P(A) = \sum_{i=1}^{n} P(A|B_i)P(B_i) > P(A|B_i)P(B_i)$$

Bernoulli and Binomial probability if $X_k \sim Bernoulli(p)$, then $Y = \sum_{k=1}^{n} X_k \sim Binomial(n, p)$ with $\mathbf{E}Y = np, \mathbf{Var}Y = np(1-p)$ and

$$P(Y \leq r) = \sum_{k=0}^{r} \binom{n}{k} p^k (1-p)^{n-k}$$

Since this expression does not exist in closed form, it can be either approximated using mathematical tools, as in e.g. [Wor94], which can get quite messy, or using the Central Limit Theorem.

Central Limit Theorem (CLT) If $X_i \sim F$ with $\mathbf{E}X_i = \mu < \infty, \mathbf{Var}X_i = \sigma^2 < \infty$, then

$$\lim_{n\to\infty}\left(\frac{\sum_{i=1}^{n} X_i - n\mu}{\sigma\sqrt{n}} \leq x\right) = \Phi(x)$$

where $\Phi(x)$ is Standard Normal distribution with mean 0 and variance 1.

Applications of CLT Getting back to the example with the Binomial distribution above, the cumulative probability can be derived in the following way:

$$\sum_{j=0}^{r}\binom{n}{j}p^j(1-p)^{n-j} = P(Y \leq r) = P\left(\frac{Y-n\mu}{\sigma\sqrt{n}} \leq \frac{r-n\mu}{\sigma\sqrt{n}}\right) \to_n \Phi\left(\frac{r-n\mu}{\sigma\sqrt{n}}\right)$$

Given that for the Bernoulli rv $\mu = p, \sigma = \sqrt{p(1-p)}$ the above result reduces to

$$P(Y \leq r) \to_n \Phi\left(\frac{r-np}{\sqrt{np(1-p)}}\right)$$

Another useful application of CLT in this thesis is approximation of cumulative distribution function of Poisson random variable. If $X \sim Poisson(\lambda)$, then $X = \sum_{i=1}^{\lambda} X_i$, where $X_i \sim Poisson(1)$ with $\mathbf{E}X = \mathbf{E}\sum_{i=1}^{\lambda} X_i = \lambda, \mathbf{Var}X = \mathbf{Var}\sum_{i=1}^{\lambda} X_i = \lambda$ the, for large λ:

$$S_\lambda = \sum_{k=0}^{\lambda}\frac{\lambda^k}{k!} = e^\lambda\sum_{k=1}^{\lambda}\frac{e^{-\lambda}\lambda^k}{k!} = e^\lambda\mathbf{P}(X \leq \lambda) = e^\lambda\mathbf{P}\left(\frac{\sum_{k=1}^{n} X_k - \lambda}{\sqrt{\lambda}} \leq 0\right) \approx \frac{e^\lambda}{2}$$

since $\Phi(0) = \frac{1}{2}$.

Absorbing states in a Markov Chain Throughout the thesis, the concept of absorbing states was used to support the fact that once an elitist EA has found the global solution, it always maintains it, i.e. the artificial fitness level has the same property as an absorbing state in a Markov chain:

If a stochastic process $X(t)$ is defined on a discrete set of states S, a state s_i with the property

$$P(X(t+1) = s_i | X(t) = s_i) = 1 \; \forall \; t$$

is called absorbing.

Mean first hitting time in a Markov Chain The main focus of the thesis, runtime of EAs, roughly corresponds to the concept of the mean first hitting time in an MC. Since the objective is the expected time to find the global solution, this can be seen as the problem of finding the mean first hitting time of the absorbing state A in a system where all other states are transient (i.e. $\lim_{t\to\infty} p_{ii}^t = 0$, the probability to return to the state after t steps):

$$m_{1,A} = 1 + \sum_{j=1}^{n} m_{j,A} p_{1,j}$$

Stationary distribution in a Markov Chain In Section 5.5 an absorbing MC was approximated by transforming the probability distribution in the state $\delta^*\alpha$. Here the idea behind this transformation is explained.

Stationary distribution in an MC is defined as

$$\pi = \pi \lim_{n\to\infty} \mathbf{P}^n$$

where π is the vector or stationary distribution, $\mathbf{P}^n$ is the transition matrix defined recurrently as $\mathbf{P}^n = \mathbf{P}^{n-1}\mathbf{P}$. Intuitively, π is the limiting proportion of time spent by $X(t)$ in each state (so the values in the vector sum to 1). Obviously, if the state s_i is transient, then, by Borel-Cantelli lemma $P(X(t) = s_i \text{ i.o.}) = 0$, and the respective entry is 0. If the MC consists of all but one state transitive and one state absorbing, the values in π trivially converge to 0 for all entries except this last one and 1 for the absorbing state. Transformation in Section 5.5 makes the MC ergodic and all states positive-recurrent, i.e. all entries in π are strictly positive.

Bibliography

[BC95] S. Baluja and R. Caruana. Removing the Genetics from the Standard Genetic Algorithm. Technical report, Carnegie Mellon University, School of Comupter Science, 1995.

[Ber] Zuse Institut Berlin. GA Playground - Travelling Salesman Problem. `http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/att48.opt.tour`.

[BKB05] L. Bianchi, J. Knowles, and N.l Bowler. Local search for the probabilistic travelling salesman problem. Correction to the 2-p-opt and 1-shift algorithms. *European Journal of Operations Research*, 162:206–219, 2005.

[Bur09] K. Burjorjee. *Generative Fixation: A Unified Explanation of Adaptive Capacity of Simple Recombinative Genetic Algorithms*. PhD thesis, Brandeis University, 2009.

[CHS+09] T. Chen, J. He, G. Sun, G. Chen, and X. Yao. A New Approach for Analyzing Average Time Complexity of Population-Based Evolutionary Algorithms on Unimodal Problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 39(5):1092–1106, 2009.

[CTCY12] T. Chen, K. Tang, G. Chen, and X. Yao. A large population size can be unhelpful in evolutionary algorithms. *Theoretical Computer Science*, 436(2012):54–70, 2012.

[DBIJV97] J. S. De Bonet, C. L. Isbell Jr., and P. Viola. MIMIC: Finding Optima

by Estimating Probability Densities. *Advances in Neural Information Processing Systems (NIPS97)*, 9:424, 1997.

[DFW11] B. Doerr, M. Fouz, and C. Witt. Sharp Bounds by Probability-Generating Functions and Variable Drift. In *Genetic and Evolutionary Computing Conference (GECCO) 2011*, pages 2083–2090, 2011.

[DHK11] B. Doerr, E. Happ, and C. Klein. Crossover Can Provably be Useful in Evolutionary Computation. In press. 2011.

[DJW02] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[DJW10a] B. Doerr, D. Johannsen, and C. Winzen. Drift Analysis and Linear Functions Revisited. In *Genetic and Evolutionary Computing Conference (GECCO) 2010*, pages 1–8, 2010.

[DJW10b] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative Drift Analysis. In *Genetic and Evolutionary Computing Conference (GECCO) 2010*, pages 1449–1456, 2010.

[DJW11] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative Drift Analysis. In press. 2011. arXiv:1101:0776.

[Doo90] J.L. Doob. *Stochastic Processes*. John Wiley & Sons, 1990.

[GKP95] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics:A Foundation for Computer Science*. Addison-Wesley Publishing Company, 1995.

[GKS99] J. Garnier, L. Kallel, and M. Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7(1):45–68, 1999.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Massachussetts, 1989.

[Haj82] B. Hajek. Hitting-Times and Occupation-Time Bounds Implied by Drift Analysis with Applications. *Advanced Applied Probability*, 14:502–525, 1982.

[He10] J. He. A Note on the First Hitting Time of $(1 + \lambda)$ Evolutionary Algorithm for Linear Functions with Boolean Inputs. In *Conference on Evolutionary Computation (CEC)*, pages 1–6, 2010.

[HK99] J. He and L. Kang. On The Convergence Rates of Genetic Algorithms. *Theoretical Computer Science*, 229(1999):23–39, 1999.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[HY01] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(2001):57–85, 2001.

[HY02] J. He and X. Yao. From an Individual to a Population: An Analysis of the First Hitting Time of Population-Based Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6-5, October 2002:495–511, 2002.

[HY03] J. He and X. Yao. Towards an analytic framework for analysing the computational time of evolutionary algorithm. *Artificial Intelligence*, 145(2003):59–97, 2003.

[HY04] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithm. *Natural Computing*, 3(2004):21–35, 2004.

[Ios80] M. Iosifescu. *Finite Markov Processes and Their Applications*. John Wiley and Sons, 1980.

[Jag08] J. Jagerskupper. A Blend of Markov Chains and Drift Analysis. *Parallel Problem Solving from Nature X*, 5199/2008:41–51, 2008.

[JDJW05] T. Jansen, K. A. De Jong, and I. Wegener. On the Choice of the Offspring Population Size in Evolutionary Algorithm. *Evolutionary Computation*, 13(4):413–440, 2005.

[JW01] T. Jansen and I. Wegener. Evolutionary Algorithms-How to Cope With Plateuas of Constant Fitness and When to Reject Strings of

The Same Fitness. *IEEE Transactions on Evolutionary Computation*, 5(6):589–599, 2001.

[KS76] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer Verlag, 1976.

[LK73] S. Lin and B.W. Kernighan. An Effective Heuristic Algorithm for the Travelling Salesman Problem. *Operations Research*, 21-2:498–516, 1973.

[MFH92] M. Mitchell, S. Forrest, and John H. Holland. The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. *European Conference on Artificial Life*, 1:245–254, 1992.

[Mit96] M. Mitchell. *Introduction to Genetic Algorithm*. Kluwer Academic Publishers, 1996.

[Nas96] I. Nasell. On the quasi-stationary distribution of the closed endemic SIS model. *Advances in Applied Probability*, 28(1996):895–932, 1996.

[Nas99] I. Nasell. On the quasi-stationary distribution of the stochastic logistic epidemic. *Mathematical Biosciences*, 156(1999):21–40, 1999.

[NV92] AE Nix and MD Vose. Modeling Genetic Algrotihms with Markov chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.

[NW04] F. Neumann and I. Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. In *Genetic and Evolutionary Computing Conference (GECCO) 2004*, pages 953–960, 2004.

[OHY07] P. S. Oliveto, J. He, and Xin Y. Time Complexity of Evolutionary Algorithms for Combinatorial Optimization: A Decade of Results. *International Journal of Automation anc Computing*, 04(3):281–293, 2007.

[OHY08] P. S. Oliveto, J. He, and X. Yao. Analysis of Population-based Evolutionary Algorithms for the Vertex Cover Problem. In *Congress of Evolutionary Computation(CEC)*, pages 1563–1570, 2008.

[OW11] P. S. Oliveto and C. Witt. Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *Algorithmica*, 2011/59:369–386, 2011.

[PB94] A. Prugel-Bennet. Analysis of Genetic Algorithms Using Statistical Mechanics. *Physical Review Letters*, 72-9:1305–1309, 1994.

[PR91] M. Padberg and G. Rinaldi. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Travelling Salesman Problem. *SIAM Review*, 33-1:60–100, 1991.

[Ros06] Jeffrey S. Rosenthal. *A First Look at Rigorous Probability Theory.* World Scientific, 2006.

[RR03] Colin R. Reeves and Johnathan E. Rowe. *Genetic algorithms - Principles and Perspectives:A Guide to GA Theory.* Kluwer Academic Publishers, 2003.

[RS96] M. Rattary and J. Shapiro. The Dynamics of a Genetic Algorithm for a Simple Learning Problem. *Journal of Physics A:Mathematical and General*, 29-23:7451–7473, 1996.

[Rud94a] G. Rudolph. Convergence analysis of canonical genetic algorithms. In *IEEE Transactions on Neural Networks*, volume 5, pages 96–101, 1994.

[Rud94b] G. Rudolph. Convergence of non-elitist strategies. In *The 1st IEEE Conference on Evolutionary Computation*, pages 65–68, 1994.

[Rud96] G. Rudolph. Convergence of Evolutionary Algorithms in General Search Spaces. In *The IEEE International Conference on Evolutionary Computation*, pages 50–54, 1996.

[Rud97] G. Rudolph. Convergence Rates of Evolutionary Algorithms for a Class of Convex Objective Functions. *Control and Cybernetics*, 26:375–390, 1997.

[Rud98] G. Rudolph. Finite Markov Chain results in evolutionary computation: A tour d'horizon. In *Fundamenta Informaticæ*, volume 1-22, 1998.

[Rud99] G. Rudolph. Theory of Evolutionary Algorithms:A Bird's Eye View. *Theoretical Computer Science*, 1999(229):3–9, 1999.

[SC99] A. Simoes and E. Costa. Transposition: A Biologically Inspired Mechanism To Use With Genetic Algorithms. In *Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms*. Springer-Verlag, 1999.

[Shi07a] A.N. Shiryaev. *Veroyatnost I (Probability I)*. MTsNMO, 2007.

[Shi07b] A.N. Shiryaev. *Veroyatnost II (Probability II)*. MTsNMO, 2007.

[Sud08a] D. Sudholt. *Computational Complexity of Evolutionary Algorithms, Hybridization and Swarm Intelligence*. PhD thesis, Dortmund Technical University, 2008.

[Sud08b] D. Sudholt. The Impact of Parametrization in Memetic Evolutionary Algorithms. *Theoretical Computer Science*, 2008:1–30, 2008.

[Suz95] J. Suzuki. A Markov Chain Analysis on Simple Genetic Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 25-4:655–659, 1995.

[SW03] T. Storch and I. Wegener. Real Royal Road Function for Constant Population Size. In *Genetic and Evolutionary Computing Conference (GECCO) 2003*, pages 1406–1417, 2003.

[TS12] A. Ter-Sarkisov. Elitism Levels Traverse Mechanism For The Derivation of Upper Bounds on Unimodal Functions. In *WCCI 2012 IEEE World Congress on Computational Intelligence*, pages 2161–2168, 2012.

[TSM11a] A. Ter-Sarkisov and S. Marsland. Convergence of a Recombination-Based Elitist Evolutionary Algorithm on the Royal Roads Test Function. In *24th Australasian Joint Conference on Artificial Intelligence*, pages 361–371, 2011.

[TSM11b] A. Ter-Sarkisov and S. Marsland. Convergence Properties of $(\mu + \lambda)$ Evolutionary Algorithms. In *25th AAAI Conference on Artificial Intelligence*, pages 1816–1817, 2011. Special Student Poster Session.

[TSM11c] A. Ter-Sarkisov and S. Marsland. Convergence Properties of Two $(\mu + \lambda)$ Evolutionary Algorithms on OneMax and Royal Roads Test Functions. In *International Conference on Evolutionary Computation Theorey and Applications (ECTA)*, pages 196–202, 2011.

[TSM12] A. Ter-Sarkisov and S. Marsland. Derivation of Upper Bounds on Optimization Time of Population-Based Evolutionary Algorithm on a Function with Fitness Plateaus Using Elitism Levels Traverse Mechanism. 2012. arXiv:1204.2321. To be submitted.

[TSMH10] A. Ter-Sarkisov, S. Marsland, and B. Holland. The k-Bit-Swap: A New Genetic Algorithm Operator. In *Genetic and Evolutionary Computing Conference (GECCO) 2010*, pages 815–816, 2010.

[vN00] E. van Nimwegen. Metastabe Evolutionary Dynamics: Crossing Fitness Barriers or Escaping via Neutral Paths. *Bulletin of Mathematical Biology*, 62(5):799–848, 2000.

[vNCM99] E. van Nimwegen, J. P. Crutchfield, and M. Mitchell. Statistical Dynamics of the Royal Road Genetic Algorithm. *Theoretical Computer Science*, 229-1999:41–102, 1999.

[Vos93] M. Vose. A Critical Examination of Schema Theorem. Technical report, University of Tennessee, 1993.

[Vos99] M. Vose. *The Simple Genetic Algorithm: Foundation and Theory*. MIT Press, 1999.

[Weg03] I. Wegener. Methods for the analysis of EAs on pseudo-boolean functions. *International Series in Operations Research and Management Science*, 48, VII:349–369, 2003.

[Wit04] C. Witt. An Analysis of the (μ+1)EA on Simple Pseudo-Boolean Functions. In *Genetic and Evolutionary Computing Conference (GECCO) 2004*, pages 761–773, 2004.

[Wit12] C. Witt. Tight Bounds on The Optimization Time of The (1+1)EA on Linear Functions. In *arXiv Abstract:1108.4386*, 2012.

[WJ07] R.A. Watson and T. Jansen. A Building Block Royal Road Where Crossover is Provably Essential. In *Genetic and Evolutionary Computing Conference (GECCO) 2007*, pages 1452–1459, 2007.

[WM97] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions On Evolutionary Computation*, 1997/1/1:67–82, 1997.

[WM04] John C. Wierman and David J. Marchette. Modeling computer virus prevalence with a susceptible-infected-susceptible model with reintroduction. *Computational Statistics and Data Analysis*, 45(2004):3–23, 2004.

[Wor94] T. Worsch. Lower and upper bounds for (sums of) binomial coefficients. Technical report, University of Karlsruhe, 1994. Report 31/94.