Analysis of Ant Colony Optimization and Population-Based Evolutionary Algorithms on Dynamic Problems

Lissovoi, Andrei

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

Analysis of Ant Colony Optimization and Population-Based Evolutionary Algorithms on Dynamic Problems

Andrei Lissovoi

Kongens Lyngby 2016
PhD-2016-404

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

PhD-2016-404
ISSN: 0909-3192

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfilment of the requirements for acquiring a Ph.D. in Computer Science. The results presented in this thesis were obtained from October 2012 to January 2016 under supervision of Associate Professor Carsten Witt. The thesis is comprised of four joint peer-reviewed publications and an unpublished result.

Acknowledgements I am grateful to my advisor, Carsten Witt, who has provided excellent guidance and support throughout my PhD studies, and has been an great source of inspiration in both. The three years have been made considerably more enjoyable by the great atmosphere at the ALGOLOG section at DTU. I also feel fortunate to have spent three months at an external stay at the University of Nottingham, where Per Kristian Lehre, as well as Duc-Cuong, Dogan, and Fawaz, have all been amazing hosts. I would additionally like to thank Timo Kötzing and Dirk Sudholt, whose work has laid a foundation for much of the results presented in this thesis, as well as the broader EC theory community, for always suggesting interesting directions of research to consider.

Andrei Lissovoi
Kongens Lyngby, January 2016

Abstract

This thesis presents new running time analyses of nature-inspired algorithms on various dynamic problems. It aims to identify and analyse the features of algorithms and problem classes which allow efficient optimization to occur in the presence of dynamic behaviour. We consider the following settings:

λ-MMAS on Dynamic Shortest Path Problems We investigate how increasing the number of ants simulated per iteration may help an ACO algorithm to track optimum in a dynamic problem. It is shown that while a constant number of ants per-vertex is sufficient to track some oscillations, there also exist more complex oscillations that cannot be tracked with a polynomial-size colony.

MMAS and (μ+1) EA on Maze We analyse the behaviour of a $(\mu + 1)$ EA with genotype diversity on a dynamic fitness function Maze, extended to a finite-alphabet search space. We prove that the $(\mu + 1)$ EA is able to track the dynamic optimum for finite alphabets up to size μ, while MMAS is able to do so for any finite alphabet size.

Parallel Evolutionary Algorithms on Maze We prove that while a $(1 + \lambda)$ EA is unable to track the optimum of the dynamic fitness function Maze for offspring population size up to $\lambda = O(n^{1-\varepsilon})$, a simple island model with $\Omega(\log n)$ islands is able to do so if the migration interval is chosen appropriately.

Migration Topology in Island Models We investigate the impact of the migration topology on the performance of an island model optimizing a Maze-like dynamic function, demonstrating that in some cases, a less-dense migration topology is preferable to a complete migration topology.

(1+1) EA on Generalized Dynamic OneMax We analyze the $(1 + 1)$ EA on dynamically changing OneMax, re-proving known results on first hitting times using modern drift analysis, and providing a new anytime analysis showing how closely the EA can track the dynamically moving optimum over time. These results are also extended to a finite-alphabet search space.

Danish Abstract

Analyse af Ant Colony Optimization og Population-based Evolutionary Algorithms algoritmer på dynamiske optimeringsproblemer

Denne afhandling præsenterer nye køretidsanalyser af natur-inspirerede algoritmer på forskellige dynamiske optimeringsproblemer. Den har til formål at identificere og analysere de egenskaber af algoritmer og problem-klasser, der tillader effektiv optimering til at finde sted selv ved forekomsten af dynamiske ændringer. Vi betrager følgende kombinationer af problemer og algoritmer:

λ-MMAS på Dynamiske Korteste Stier Vi undersøger hvordan en øgelse af antal myrer simuleret per iteration per knudepunkt kan hjælpe en myrekolonibaseret algoritme til at følge optimalløsningen i et dynamisk problem. Vi viser at mens et konstant antal af myrer per knudepunkt er tilstrækkelig til at følge nogle oscillationer, der findes også mere indviklede oscillationer som kan ikke følges med kolonier af enhver polynomial størrelse.

MMAS og (μ+1) EA på Maze Vi analyserer hvordan en $(\mu + 1)$ evolutionær algoritme (EA) med genotype-mangfoldighed optimerer Maze, en dynamisk fitness-funktion udvidet over et alfabet af endelig størrelse. Vi beviser at $(\mu + 1)$ EA kan følge den dynamiske optimum på alfabeter med op til μ tegn, mens myrekoloni-algoritme MMAS kan gøre det samme for enhver endelig alfabet.

Parallele EA'er på Maze Vi beviser at mens en $(1 + \lambda)$ EA kan ikke følge optimalløsningen af den dynamiske fitness-funktion Maze med en afkom-population på op til $\lambda = O(n^{1-\varepsilon})$ individer, en enkel ø-model kan klare dette med kun $\lambda = \Omega(\log n)$ øer, hvis intervalen mellem migrationer er valgt på en passende måde.

Migration Topologi i Ø-modeller Vi undersøger hvordan valget af migration topologi påvirker en ø-models evne til at følge optimalløsningen af en Maze-lignede funktion. Vi beviser at i nogle tilfælder, en mindre tæt topologi kan føre til bedre evne til at følge optimalløsningen end en komplet topologi.

(1+1) EA på Generaliseret Dynamisk OneMax Vi analyserer en $(1 + 1)$ EA på en dynamisk version af OneMax, genbeviser kendte grænser for køretiden det tager algoritmen til at evaluere optimalløsningen for første gang, og supplerer med en ny når-som-helst analyse, der går ud på at bevise hvor tæt EA kan komme til optimalløsningen i et givet tidsbudget. Disse resultater er også udvidet til et søgerum over et endelig alfabet.

Contents

Chapter 1

Introduction

Optimization problems are wide-spread in both natural and human-defined settings. For many such problems, we do not yet have a classical, deterministic algorithm capable of finding the optimal solution in polynomial time, or even know whether such an algorithm can exist. For others, it may be too computationally demanding to solve the full optimization problem directly. Yet natural processes often depend on finding good solutions to such optimization problems, even in settings where their parameters are allowed to change with time.

Nature-inspired algorithms incorporate models of natural behaviours, such as the use of pheromone trails for pathfinding, or mutation and natural selection, into randomized algorithms, aiming to find good, if not optimal, solutions in reasonable time. While such algorithms have been applied in practice with some success, few theoretical analysis results are available for practical problems, although advances in the theoretic approaches have been made recently.

In this thesis, we consider how variants of two nature-inspired algorithms, Ant Colony Optimization, and Population-based Evolutionary Algorithms, optimize dynamic problems, whose conditions, and hence also optimal solutions, are allowed to change over time. We aim to identify which aspects of the problems make optimization difficult, and which algorithm features make efficient optimization of some classes of problems possible.

The work in this thesis appears in the following papers:

- **Runtime Analysis of Ant Colony Optimization on Dynamic Shortest Path Problems**
  Andrei Lissovoi, Carsten Witt.
  In Theoretical Computer Science, Vol. 561 (2015) [LW15c]. A preliminary version appeared in the Proceedings of the Genetic and Evolutionary Computation Conference (2013) [LW13b]. We note that while this article was composed during the Ph.D. studies, some of the results also appeared in preliminary form in the MSc thesis [Lis12].

- **MMAS Versus Population-Based EA on a Family of Dynamic Fitness Functions**
  Andrei Lissovoi, Carsten Witt.
  In Algorithmica, currently in print; online in 2015 [LW15a]. A preliminary version appeared in the Proceedings of the Genetic and Evolutionary Computation Conference (2014) [LW14b].

- **A Runtime Analysis of Parallel Evolutionary Algorithms in Dynamic Optimization**
  Andrei Lissovoi, Carsten Witt.
  The version presented here is a journal extension currently in review. A previous version appeared in the Proceedings of the Genetic and Evolutionary Computation Conference (2015) [LW15b].

- **(1+1) EA on Generalized Dynamic OneMax**
  Timo Kötzing, Andrei Lissovoi, and Carsten Witt.
  In the Proceedings of Foundations of Genetic Algorithms Workshop (2015) [KLW15].

- **The Impact of Migration Topology on the Runtime of Island Models in Dynamic Optimization**
  Andrei Lissovoi, Carsten Witt.
  Unpublished. Will be extended and submitted for publication in 2016.

In this chapter, we will provide an introduction to the prerequisites for the results presented in subsequent chapters, as well as a high-level summary of our contributions. Chapters 2 through 6 present the results independently, in a form that closely matches the original publications.

1.1 Nature-inspired algorithms

In this section, we will introduce the two main metaheuristics upon which the algorithms considered in Chapters 2 through 6 are based: the Population-Based Evolutionary Algorithm metaheuristic is based on natural selection – mutation and selection steps are used to produce solutions with higher fitness; while the Ant Colony Optimization metaheuristic is based on the foraging behaviour of ants, who rely on pheromone trails to steer exploration towards solutions that have been observed to be good in the past.

1.1.1 Population-based evolutionary algorithms

The $(1 + 1)$ EA is a population-based evolutionary algorithm, shown as Algorithm 1.1 below. In this formulation, the algorithm operates on n-bit strings representing solutions to some optimization problem, which the fitness function $f(x, t)$ evaluates the quality of. In each iteration, the algorithm produces an offspring individual x' by applying a mutation operator (the commonly used standard bit mutation operator is shown as Definition 1.1) to the current-best solution x^*, and chooses whichever of x' and x^* has a higher fitness value according to $f(x, t)$ to be used as x^* in a future iteration. We note that for static problems, $f(x, t) = f(x, t')$ for any two iterations (t, t'), while in dynamic optimization problems, we allow the fitness value of individuals to change from one iteration to another.

Algorithm 1.1 $(1+1)$ EA, a simple Population-based Evolutionary Algorithm

Choose $x^* \in \{0, 1\}^n$ uniformly at random.
for $t = 1, 2, \ldots$ **do**
  $x' \leftarrow \text{mutate}(x^*)$
  if $f(x', t) \geq f(x^*, t)$ **then**
    $x^* \leftarrow x'$

Definition 1.1 (Standard bit mutation) The mutation operator mutate(x) creates an image $y \in \{0, 1\}^n$ from $x \in \{0, 1\}^n$ by independently replacing each bit x_i of x $(1 \leq i \leq n)$ with $1 - x_i$ with probability $1/n$.

As previously mentioned, this approach mimics natural selection: the mutation operator tends to make small changes to the current-best individual, which are accepted if they represent an improvement (or at least not a worsening) of the ancestor solution. In situations where there is a sequence of simple changes

leading to the optimal solution, the $(1 + 1)$ EA can be an efficient hill-climber, finding the optimum in expected polynomial time.

We note that the n-bit string (or n-character string) representation is a natural choice for the optimization problems considered in further chapters, as well as some combinatorial optimization problems (such as Independent Set or Maximal Clique, which require involve the maximal set of vertices in a graph which either have no edges between them, or form a complete subgraph). For problems where this is not the case, this framework may still be applied by adjusting the way $f(x, t)$ interprets the solution, or extending the mutation operator and fitness function to operate on a more convenient solution representation.

Notably, the sizes of both the ancestor and offspring population may be increased while using this framework: a $(\mu + \lambda)$ EA is able to store μ individuals as potential ancestors, and generates λ new individuals in each iteration. Should the size of the ancestor population increase, various diversity mechanisms could be applied to prevent the ancestor population from being filled by a single individual; the benefits of using such measures to ensure population diversity is illustrated in Chapter 3.

1.1.2 Ant Colony Optimization algorithms

ACO is a metaheuristic based on the foraging behaviour of ants: as ants move around to locate a food source (or return to the colony after locating a food source), they leave pheromone trails in the environment. When exploring the environment, ants tend to prefer to follow these pheromone trails, strengthening them further if they eventually find a food source; while unreinforced pheromone trails tend to evaporate over time. This allows the colony to maintain a "memory" of which choices have proven to be good in the past, eventually optimizing their foraging behaviour to use more efficient routes to food sources.

The Max-Min Ant System (MMAS) algorithm, shown as Algorithm 1.2 below, operates by constructing a path through a graph G, guided by the per-arc pheromone values τ. The variant shown here simulates the path of a single ant, starting at a distinguished source vertex s, through the graph G, guided by the pheromone values τ. If G has $n + 1$ vertices, arranged in a line, such that there are always two directed edges from one vertex to the next, the path the ant traces through the graph can be converted to an n-bit string, where picking a particular edge between two vertices corresponds to setting a particular bit to a 1, allowing this algorithm to be used in similar settings to the Evolutionary Algorithms introduced in the previous section.

Algorithm 1.2 Max-Min Ant System (MMAS) on a graph $G = (V, \mathcal{A})$

Initialize $\tau_a \leftarrow 1/\deg^+(v)$ for all $a = (v, v') \in \mathcal{A}$
for $i \leftarrow 1, 2, \ldots$ **do**
  Let x' be an empty path starting at vertex $s \in V$
  $p \leftarrow s,\ S \leftarrow \{(p, v') \in \mathcal{A} \mid v' \notin x'\}$
  while S is not empty and $p \neq t$ **do**
    Pick arc $a = (p, h')$ from S with probability:
      $p_a = \tau_a / \sum_{s \in S} \tau_s$
    Append a to x'
    $p \leftarrow h',\ S \leftarrow \{(p, v') \in \mathcal{A} \mid v' \notin x'\}$
  if $i = 1$ **or** $f(x') < f(x^*)$ **then**
    $x^* \leftarrow x'$
  for each $a = (v, v') \in \mathcal{A}$ **do**

$$\tau_a \leftarrow \begin{cases} \min(\tau_{\max}, (1-\rho)\tau_a + \rho) & \text{if } a \in x^* \\ \max(\tau_{\min}, (1-\rho)\tau_a) & \text{otherwise} \end{cases}$$

The core of the algorithm lies in the pheromone memory τ: after computing a path through G, the fitness value (or quality of the solution represented by that path) is compared against the quality of the best-so-far solution x^* to determine the new best-so-far solution. Then, pheromone values are updated to guide the path construction, and hence the solutions constructed by the algorithm, toward the best-so-far solution: arcs that are on the best-so-far path x^* have their pheromone values reinforced (increased), while arcs that are not part of x^* only have their pheromone values evaporate (decrease), according to the evaporation rate ρ (with greater ρ values increasing the impact each iteration has on the pheromone values, while lower ρ values allow the algorithm to take more of the past history into account when constructing new paths). Over time, this increases the probability of constructing solutions similar to x^*. To avoid the algorithm settling on constructing only x^*, the range of pheromone values are constrained to $[\tau_{\min}, \tau_{\max}]$.

We note that the pheromone memory allows MMAS to easily represent oscillations between two choices, potentially even identifying the frequency at which particular solutions are favoured in an oscillation. This is used in particular in Chapter 3.

In other settings, such as the one in Chapter 2, it may be beneficial to start an ant (or more than one ant) at each vertex of the graph, and have the ants constructing paths from a particular vertex bear the sole responsibility for updating the pheromone values on arcs leaving that vertex. This variation is typically called λ-MMAS, where λ is the number of ants started at each vertex.

1.2 Efficient optimization

In this section, we elaborate on how we evaluate performance of nature-inspired algorithms.

In general, when performing runtime analysis on evolutionary or nature-inspired algorithms on static problems, we consider their performance in terms of only the number of fitness function evaluations performed before the global optimum solution is found, reasoning that evaluating the fitness function is likely to be considerably more computationally expensive than all other operations performed by the algorithm. As nature-inspired algorithms are randomized algorithms, we typically consider the expected runtime, i. e. the expectation of the number of fitness function evaluations performed before the global optimum is found, and, in some cases, tail bounds on the probability that a solution of a particular quality is found within a certain number of iterations.

Generally, we consider "efficient optimization" to be possible for a nature-inspired algorithm whenever its expected number of fitness function evaluations is polynomial with respect to the problem size, as opposed to super-polynomial or exponential runtimes. This term should not be taken to imply that nature-inspired algorithms are the most efficient approach to a particular problem – there may well exist other algorithms with better properties.

In some cases, if the work performed by a nature-inspired algorithm during a single iteration is trivially parallelizable, we may instead consider the number of *iterations* that pass before the global optimum is found, reasoning that this bound better corresponds to the time required to solve the problem when sufficient hardware is available.

For dynamic optimization problems, our focus shifts to considering the algorithms' ability to track the optimum as it moves around the search space: thus, instead of a runtime bound on the expected time to find the global optimum, results focus on demonstrating that the algorithm remains close to the optimum (or reaches it periodically) for some duration while the optimum is moving. This is particularly important if the optimum moves frequently, for instance in oscillating patterns, rather than moving slowly enough to allow the algorithms to always find the new optimum before it moves again.

1.3 Our contributions

This section introduces the problems considered in the following chapters, and provides a brief summary of our results.

1.3.1 Chapter 2: Runtime Analysis of Ant Colony Optimization on Dynamic Shortest Path Problems

We consider the behaviour of a population-enhanced variant of the Max-Min Ant System algorithm, λ-MMAS, on a dynamic version of the Single-Destination Shortest Path Problem, where the goal is to find the shortest path to some sink vertex in a directed graph where the weight function is allowed to change over time. This builds on the previous analysis of 1-MMAS on the static SDSP in [ST12a], using similar techniques to demonstrate that if the changes in the weight function are not too frequent, λ-MMAS is able to keep track of the optimum by continually recovering as if following a random initialization in the static setting. Notably, minute changes in the weight function (such as modifying the weight) of a single arc can require a large number of iterations for λ-MMAS to rediscover all of the shortest paths, and conversely, large changes might not affect the optimum shortest paths at all.

We prove that as some amount of time is spent waiting for particular paths to be constructed, there is a straightforward improvement to be gained via parallelization by increasing λ, the number of ants simulated from each vertex in a single iteration, to ensure that such waiting times are reduced to an expected constant number of iterations, yielding a modest improvement in the expected number of iterations required to recover from a one-time change.

We then consider the behaviour of λ-MMAS in several settings where the optimum oscillates quickly between two solutions. When such oscillation is limited to a single triangle in the graph (and hence impacts only a single choice), we prove that $\lambda = 4$ ants are sufficient to keep the pheromones in a state which allows the optimum solution to be constructed with at least a constant probability in every iteration for an exponential number of iterations. At the opposite extreme, if the oscillation affects many choices in the graph, we show that even any polynomial λ is not sufficient to achieve a constant probability of constructing the optimum.

The considered settings illustrate that while the pheromone memory is able to store that a dynamic optimum oscillates between several solutions, it is only able to do so usefully while the difference between the solutions is minimal.

1.3.2 Chapter 3: MMAS versus Population-Based EA on a Family of Dynamic Fitness Functions

The MAZE fitness function, defined over n bit strings in [KM12], moves the global optimum from the all-ones bit string to the all-zeroes bit string in $n+1$ long phases, in which the optimum solution oscillates between $0^{i-1}1^{n-i+1}$ and 0^i1^{n-i} (favouring the latter solution two iterations out of three), while all other individuals have lower fitness values forming a gradual slope to the all-ones bit string. In [KM12], this function is used to illustrate that MMAS is able to adapt its pheromone values to reflect the oscillations, and thereby track the global optimum through the n oscillating phases, while the $(1+1)$ EA with high probability loses track of the optimum before the n phases are over, and requires an exponential number of iterations to find the all-zeroes optimum at the end of the maze.

We prove that by increasing the size of the ancestor population and requiring genotype diversity, a (2+1) EA is able to track the optimum of the original MAZE function. Then, we extend the MAZE fitness function over a finite alphabet, proving that while a $(\mu+1)$ EA can track the optimum of the MAZE defined over a μ-symbol alphabet, it with high probability fails to track the optimum of the MAZE defined over alphabets of $\mu+1$ characters or more.

Finally, we return to the MMAS algorithm, and prove that it is able to track the optimum of the finite-alphabet MAZE function without requiring additional modifications. Here, we refine the analysis of [KM12] to allow for shorter MAZE phase lengths, as long as the finite alphabet is not too large.

This work serves to illustrate that increasing the ancestor population size, and requiring genotype diversity may allow population-based algorithms to track the optimum of some rapidly-oscillating dynamic functions by enabling them to store each individual solution the optimum oscillates between – but even with genotype diversity, it is important to select a sufficiently large ancestor population size, which is difficult to do in practice without insight into the problem structure. On the other hand, this oscillation pattern, where a small part of the optimal solution oscillates between multiple solutions, is the ideal case for the pheromone memory used in MMAS, which is able to encode the relative frequency of the oscillating solutions in pheromone values, and thereby track the optimum of the considered fitness function.

1.3.3 Chapter 4: A Runtime Analysis of Parallel Evolutionary Algorithms in Dynamic Optimization

We consider a simple island model optimizing the MAZE dynamic fitness function. In this model, λ islands each run a $(1 + 1)$ EA independently, and periodically-occurring migration copies the best solution found by any island to all other islands. We investigate how τ, the interval between migrations, affects whether the island model is able to track the dynamic optimum of MAZE.

We prove that when $\tau = 1$, the island model's behaviour is equivalent to that of a $(1 + \lambda)$ EA, and it is not able to track the MAZE optimum even for increased offspring population size up to $\lambda = O(n^{1-\varepsilon})$. On the other hand, if the migration interval τ is increased sufficiently, even $\lambda = \Omega(\log n)$ islands are sufficient to track the optimum of the MAZE function. Finally, we investigate additional restrictions on migration that would allow the algorithm to track the MAZE optimum, concluding that as long as no migrations occur within $\Omega(n)$ iterations prior to a MAZE phase transition, and migrations occur more often than once in every $O(\log \lambda)$ phases, the island model is able to track the optimum.

In this work, we replace the genotype diversity mechanism of Chapter 3 with a different mechanism: allowing several independent $(1 + 1)$ EAs to operate in parallel, and periodically migrating the best solution to all islands. Without migration, each island has a constant probability of failing to track the optimum through each phase (as in the previous chapter); by performing migration, the island model is able to return all islands to tracking the oscillating optimum as long as at least one island has not yet failed when migration occurs. On the other hand, migration proves to be a double-edged sword as a diversity mechanism: when performed on a complete migration topology (i. e. migrating the best solution to *all* islands), it actively removes diversity from the population. In this work, we point out that this can be managed by carefully controlling when migration occurs, but this requires some knowledge of the oscillation pattern and phase length, a topic that is addressed further in Chapter 6.

1.3.4 Chapter 6: The Impact of Migration Topology on the Runtime of Island Models in Dynamic Optimization

In this work, we consider the impact of migration topology on the behaviour of a simplified version of the island model considered in Chapter 4.

We prove that using a unidirectional ring as the migration topology (i. e. having each island only compare its current-best individual with that of its predecessor in the ring), allows the simplified model to track the optimum of the Maze-like fitness function in settings where using a complete graph as the migration topology (i. e. having each island compare its current-best individual with that of all other islands, as in Chapter 4) results in a failure to track the dynamic optimum.

This demonstrates the importance of being able to preserve population diversity when performing migration in an island model, providing an approach to Maze-like oscillating functions that requires less problem-specific knowledge than the approach that has been analysed in Chapter 4.

1.3.5 Chapter 5: (1+1) EA on Generalized Dynamic OneMax

We consider the $(1 + 1)$ EA on a dynamic version of OneMax introduced in [Dro03], which we extend to deal with finite alphabets of r symbols. In dynamic OneMax, the optimum string is allowed to mutate over time, with each bit having a probability p of being toggled (or, in the case of the finite-alphabet extension, changed to an adjacent value, chosen uniformly at random).

We re-prove previous results for the hitting-time of the optimum (i.e. the number of iterations before the optimum individual is evaluated for the first time) using modern drift analysis tools, proving that as long as the optimum does not move too quickly, or, more specifically, $p \leq \frac{c \ln n}{\min\{r, \ln n\} n^2}$, the expected first hitting time of the optimum bit string is polynomial in n, while if $p \in \omega(\log n / n^2)$, the first-hitting time is polynomial only with polynomially small probability.

Additionally, we provide an anytime analysis, as suggested by [JZ14], analyzing how close the $(1 + 1)$ EA can get to the dynamically moving optimum in a specified time budget.

In this work, we apply modern drift theorems to previously-considered settings, yielding cleaner and more general proofs, as well as contribute a new version of a variable drift theorem, allowing for a small region of negative drift close to the optimum, and showing that stochastic processes can bridge such an area of headwind. The anytime analysis builds on the tools used in the analysis of MMAS in Chapter 3.

Chapter 2

Runtime Analysis of Ant Colony Optimization on Dynamic Shortest Path Problems

Andrei Lissovoi Carsten Witt

DTU Compute, Technical University of Denmark

A simple ACO algorithm called λ-MMAS for dynamic variants of the single-destination shortest paths problem is studied by rigorous runtime analyses. Building upon previous results for the special case of 1-MMAS, it is studied to what extent an enlarged colony using λ ants per vertex helps in tracking an oscillating optimum. It is shown that easy cases of oscillations can be tracked by a constant number of ants. However, the paper also identifies more involved oscillations that with overwhelming probability cannot be tracked with any polynomial-size colony. Finally, parameters of dynamic shortest-path problems which make the optimum difficult to track are discussed. Experiments illustrate theoretical findings and conjectures.

A preliminary version of this work appeared in *GECCO '13: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, pages 1605-1612, ACM, New York, USA, 2013.

2.1 Introduction

Ant colony optimization (ACO) is a class of nature-inspired algorithms that is mostly used to solve combinatorial optimization problems. In recent years, run-

time analysis of nature-inspired algorithms has advanced considerably [AD11, Jan13, NW10b]. Even though the majority of results still apply to simple evolutionary algorithms, a lot of progress has also been made in the analysis of ACO. With respect to problems from combinatorial optimization, which is the classical domain of application for ACO, there are results on shortest paths [AF08, ST12a], minimum spanning trees [NW10a], minimum cuts [KLNO10] and the traveling salesperson problem [Zho09, KNRW12].

Real-world optimization problems are not always static in nature. Often problem and goal of optimization are dynamic, i. e., they change over time. In these cases, it is important to find a solution that is "good" with respect to the current goal of optimization. Nature-inspired algorithms are often considered to be "robust" optimizers which can adapt to such dynamic problems if the underlying optimal solution does not change too quickly or extremely. In particular, many applications of evolutionary algorithms, as well as ACO, on dynamic problems are reported in the literature [GB02, XL08, NYB12].

In this paper, we are concerned with ACO on dynamic problems. Our aim is to understand, using rigorous runtime analyses, the conditions under which ACO algorithms are able to track the optimum of a dynamically changing problem, i.e. maintain the ability to construct optimum, or close-to-optimum solutions while the fitness function changes. Such runtime analyses are motivated by related theoretical studies of evolutionary algorithms for dynamic problems [RLY09, JS05, Dro03]. We have chosen the single-destination shortest path problem (SDSP) as object of our analysis as this is probably the combinatorial optimization problem that ACO has been understood best on. There are even runtime analyses of ACO on stochastic optimization problems [ST12b, DHK12, FK13], which, together with dynamic problems, can be subsumed under the term "optimization under uncertainty". However, methods for the analysis of stochastic optimization problems are not directly applicable to dynamic problems.

So far, there is only a single runtime analysis of ACO on dynamic problems. [KM12] compare a simple ACO algorithm and a simple evolutionary algorithm on a dynamic pseudo-boolean problem and show that the ACO algorithm can outperform the evolutionary one. To the best of our knowledge, our work is the first runtime analysis of ACO on a dynamic combinatorial optimization problem.

Our findings can be summarized as follows. A simple ACO algorithm based on the Max-Min Ant System [SH00] is studied on dynamic shortest path problems with increasing amount of dynamics. First it is analyzed how long it takes the system to adapt to a one-time modification of the graph, observing that if modifications to the weight functions occur slower than this, the dynamic optimization process can be treated as a series of adaptations to one-time changes.

It is proved that two extreme cases can happen: only a single pheromone value or all pheromone values might need to be updated to adapt. Upper and lower bounds on the time required are proved. Then, more rapid periodic changes are studied by considering changing between two different functions in every iteration as an extreme case. Examples are shown where the changes exhibit enough locality for the system to track them reliably by updating independent components, maintaining at least a constant probability of constructing the optimum solution in any given iteration. Interestingly, this is possible by increasing the size of the ant colony moderately. The utility of a population for tracking problems was studied in evolutionary computation by [JS05], but our result seems the first of this kind in the runtime analysis of ACO. It is then proved that a single ant started at each vertex is sufficient to track a slower oscillation in the same setting. Finally, an example is given where the problem changes globally, which makes it very unlikely that changes can be tracked quickly. Experiments supplement the theoretical findings, and we discuss properties of the dynamics that are related to the difficulty of the tracking problem.

The paper is structured as follows. Section 2.2 introduces notation and the algorithm λ-MMAS, which generalizes previously studied MMAS by introducing a larger population. Section 2.3 proves polynomial lower and upper bounds on the time for the system to adapt to one-time changes. Periodic changes with locality are studied in Section 2.4, where it is shown that the oscillating optimum can be tracked by the ant colony for a super-polynomial number of iterations, both for rapid oscillations using a larger colony, and for slower oscillations using a single ant. Section 2.6 describes a globally changing example that is conjectured to be difficult to track. Experiments are described in Section 2.7. We finish with some conclusions.

2.2 Preliminaries

The λ-MMAS algorithm (Algorithm 2.1) is a generalized version of the $\text{MMAS}_{\text{SDSP}}$ analyzed on shortest path problems in [ST12a], where the generalization is due to a use of a population (see NSW10 for a related algorithm in pseudo-boolean optimization). The algorithm allows for negative weights, but requires that all cycles in the graph are of strictly positive weight. We note that graphs with $\Delta < 2$ are not interesting for shortest path problems, and assume that $\Delta \geq 2$ throughout the rest of the paper: otherwise, each vertex has at most one outgoing edge, and finding the shortest paths is trivial.

Every iteration of the algorithm starts λ ants at each vertex, and each ant constructs a simple path through the graph, following arcs randomly with prob-

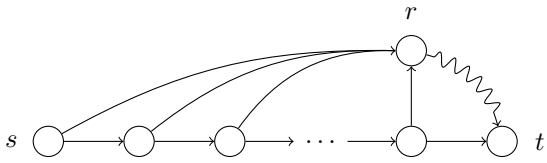

Figure 2.1: An example graph; the weight on the (r,t) arc affects whether the vertex r is visited or avoided by the shortest paths to t.

ability proportional to their pheromone values until it reaches the destination vertex t, or until it reaches a vertex from which no arcs lead to a vertex that has not already been visited. For each vertex v, the best-so-far path x_v^* is updated to the path of least weight of the λ paths constructed from v in the current iteration and the previous x_v^* path, and is then used to update the pheromone values on arcs leaving v.

The algorithm needs to start at least one ant at each vertex, even if the goal is to find the shortest path between a specific pair of vertices, in order to ensure that the shortest paths can be found in expected polynomial time. The issue is illustrated in [ST12a] using the graph shown in Figure 2.1: if the weight on the (r,t) arc is $n = |V|$ and all other arcs have unit weights, the vertex r is avoided by the s-t shortest path. An ant started at s will, with probability $1-(1/2)^{n/2}$, make no more than $n/2$ steps towards t (each taken with probability $1/2$ as the pheromone values are initialized to $1/deg^+(v) = 1/2$, and sum to 1 for each vertex) before visiting r. Subsequent iterations will accept a path that makes additional steps towards t only if it reaches t without visiting r, which occurs with probability at most $(1/2)^{n/2}$ (as each of the additional $n/2$ steps toward t is made with probability at most $1/2$); discovery, acceptance, and pheromone updates based on paths that take even fewer steps towards t before visiting r will only decrease the probability of reaching t without visiting r over time.

Three parameters affect the behavior of the algorithm: the evaporation rate ρ controls the speed with which the pheromone values are updated, while the pheromone bounds $\tau_{\min}$ and $\tau_{\max}$ control how likely an ant is to deviate from a pheromone trail, balancing exploration with the ability to follow reinforced paths. As in [ST12a], we choose the pheromone bounds are chosen based on Δ, the maximum vertex out-degree, and ℓ, the maximum number of arcs in any shortest path in the graph:

$$\tau_{\min} = 1/(\Delta\ell) \qquad\qquad \tau_{\max} = 1 - \tau_{\min}$$

where $n = |V|$ can be used in place of Δ or ℓ if either value is unknown. With this choice of pheromone values, an ant is able to follow a reinforced path of

Algorithm 2.1 The λ-MMAS algorithm on a directed graph $G = (V, \mathcal{A})$, with pheromone bounds $\tau_{\min}$ and $\tau_{\max}$, and evaporation rate ρ, where $t \in V$ is the destination vertex, $\deg^+(v)$ is the outdegree of vertex v, and $f(x_v)$ is the fitness value of a given path (sum of its arc weights if the path terminates at t, ∞ if it does not).

Initialize $\tau_a \leftarrow 1/\deg^+(v)$ for all $a = (v, v') \in \mathcal{A}$
for $i \leftarrow 1, 2, \ldots$ **do**
  for each $v \in V$ **do**
    for $j = 1, 2, \ldots, \lambda$ **do**
      Let $x_{v,j}$ be an empty path starting at v
      $p \leftarrow v$, $S \leftarrow \{(p, v') \in \mathcal{A} \mid v' \notin x_{v,j}\}$
      while S is not empty and $p \neq t$ **do**
        Pick arc $a = (p, h')$ from S with probability:
          $p_a = \tau_a / \sum_{s \in S} \tau_s$
        Append a to $x_{v,j}$
        $p \leftarrow h'$, $S \leftarrow \{(p, v') \in \mathcal{A} \mid v' \notin x_{v,j}\}$
    $x_v \leftarrow \arg\min_{x_{v,j}} f(x_{v,j})$
    if $i = 1$ **or** $f(x_v) < f(x_v^*)$ **then**
      $x_v^* \leftarrow x_v$
  for each $a = (v, v') \in \mathcal{A}$ **do**

$$\tau_a \leftarrow \begin{cases} \min(\tau_{\max}, (1-\rho)\tau_a + \rho) & \text{if } a \in x_v^* \\ \max(\tau_{\min}, (1-\rho)\tau_a) & \text{otherwise} \end{cases}$$

$\ell - 1$ arcs (where the pheromone values on the arcs in the path are all $\tau_{\max}$, and on all other outgoing arcs from vertices visited by the path are $\tau_{\min}$) with probability at least $1/e$. This is proved in [ST12a] by considering the probability of not deviating from the path at each vertex, which is at least $(1 - \Delta\tau_{\min})^{\ell-1} \geq (1 - 1/\ell)^{\ell-1} \geq 1/e$, as the pheromones sum to at least $\tau_{\min} + \tau_{\max} = 1$ whenever there's an opportunity to deviate from the reinforced path.

The Lemma 2.1 is heavily used to study the probabilistic pheromone model, and is proved in [ST12a]. As λ-MMAS selects the next arcs with probability equal to the arc's pheromone value divided by the sum of pheromone values on viable arcs leaving the vertex, the lemma allows us to bound the probability that an ant will follow any specific outgoing arc to a vertex it has not visited before by at least $\tau_{\min}/2$.

Lemma 2.1 *The sum of pheromone values on all outgoing arcs from any vertex is always at most 2.*

When the first shortest path from vertex v is constructed by an ant, it will be used as the best-so-far path x_v^* in every subsequent pheromone update. The following Lemma from [AF08] bounds the freezing time – the number of iterations of reinforcing a single arc before the pheromone values reach the pheromone bounds, at which point they will remain unchanged by subsequent pheromone updates until a new best-so-far path, using a different arc to leave v, is discovered. The freezing time is also a bound on the number of iterations between the discovery of a shortest path and the pheromone value on its first arc reaching $\tau_{\max}$.

Lemma 2.2 *If x_v^* is unchanged for $\ln(\tau_{max}/\tau_{min})/\rho$ iterations, the pheromone value on the first arc of x_v^* is τ_{max}, and equal to τ_{min} on all other arcs leaving v.*

Proof. Consider the effect of $\ln(\tau_{\max}/\tau_{\min})/\rho$ pheromone updates on the pheromone value on an arc. It is easy to see that this number of pheromone updates is sufficient to reduce a pheromone value to $\tau_{\min}$, even if it was originally $\tau_{\max}$:

$$\tau_{\max} \cdot (1-\rho)^{\ln(\tau_{\max}/\tau_{\min})/\rho} \leq \tau_{\max} \cdot e^{-\ln(\tau_{\max}/\tau_{\min})} = \tau_{\min}$$

Consider two pheromone values, initially set to $\tau_{\max}$ and $\tau_{\min}$; while the first evaporates, the second is reinforced. The sum of the two values is initially 1, and remains 1 as multiplying both values by $(1-\rho)$ is exactly balanced by adding ρ to the second value. Thus the number of iterations required to increase a pheromone value from $\tau_{\min}$ to $\tau_{\max}$ is equal to the number of iterations required to reduce it from $\tau_{\max}$ to $\tau_{\min}$.

Therefore, $\ln(\tau_{\max}/\tau_{\min})/\rho$ iterations of reinforcing a single arc is sufficient to increase its pheromone value to $\tau_{\max}$, while also reducing the pheromone values on all other arcs exiting its source vertex to $\tau_{\min}$.

The next section examines how λ-MMAS is able to handle a one-time change to the weight function, and motivates using $\lambda > 1$ ants to reduce the expected number of iterations needed to discover the shortest paths after a one-time change. Following sections illustrate the benefits and limitations of pheromone memory in a setting where the weight function is changed more frequently: it allows a relatively small number of ants to keep track of shortest paths if the differences between the weight functions are relatively minor, but requires a super-polynomial number of ants if the changes are more significant.

2.3 A one-time change

If the oscillation is sufficiently slow, λ-MMAS may be able to rediscover and re-freeze all of the shortest paths before the next change to the weight function occurs. If this is the case, the process can be treated as a series of one-time modifications to the weight function, with each modification followed by a number iterations during which λ-MMAS rediscovers the shortest paths, and a number of iterations during which the pheromones are frozen to favor the shortest paths, resulting in each ant having at least a constant probability of constructing the shortest path from its starting vertex.

The expected number of iterations λ-MMAS needs to discover the shortest paths of the new weight function depends both on how similar these shortest paths are to those of the previous weight function, and on the maximum number of arcs in any shortest path in the graph using the new weight function. The more similar the shortest paths, the fewer pheromone values need to be changed; and the fewer arcs in the longest shortest path, the more shortest paths can be discovered in parallel.

The proof of the following theorem is based on the analysis in [ST12a]. The presentation here is simplified; a finer analysis in [ST12a] demonstrates that it is not necessary to wait for the full freezing time, removing the log factor from the freezing time component of the expected number of iterations.

Theorem 2.3 *After $O(\ell^*/\tau_{min} + \ell \ln(\tau_{max}/\tau_{min})/\rho)$ iterations 1-MMAS will have, with high probability, discovered all shortest paths after a one-time change to the weight function given that such shortest paths are unique, where $\ell^* = \max(\ell, \log n)$ and ℓ is the maximum number of arcs in any shortest path to t in the new graph. This is also the expected number of iterations before all the shortest paths are rediscovered after the weight function is altered.*

Proof. Consider an arbitrary vertex v: the shortest path from v to t has at most $\ell \leq \ell^*$ arcs. λ-MMAS can discover the shortest path from v to t by constructing all of its subpaths (ending at t) in the order of increasing number of arcs. This allows shortest paths to be discovered by only constructing subpaths with one arc with a sub-$\tau_{\max}$ pheromone value at a time, waiting for the pheromones to freeze, and repeating the process for the next subpath.

Let p_{path} be the probability that a specific ant constructs a specific k-arc path containing a single sub-$\tau_{\max}$ arc. In order to discover the v-t shortest path, at most ℓ^* subpaths, all with $k \leq \ell^*$, need to be constructed (and frozen). Assuming the pheromone values on the shorter subpaths are frozen at $\tau_{\max}$, the

probability of discovering a new shortest path in this fashion is at least p_{path}:

$$p_{\text{path}} \geq \frac{\tau_{\min}}{2} \cdot (1 - \Delta\tau_{\min})^{k-1} \geq \frac{\tau_{\min}}{2e}$$

where $1 - \Delta\tau_{\min} \geq 1 - 1/\ell$ is the probability of selecting the arc with pheromone value $\tau_{\max}$ at any frozen vertex, recalling that the pheromone bounds $\tau_{\min}$ and $\tau_{\max}$ were chosen so as to ensure that the probability of following $\ell - 1$ such arcs was at least $1/e$.

The number of the desired subpaths discovered in this fashion can be bounded using a Chernoff bound: if every considered iteration is able to discover the next subpath with probability p_{path}, the ℓ^* desired subpaths are discovered in $t = 10e\ell^*/\tau_{\min}$ iterations with high probability. Let N_t be the number of subpath discoveries in t iterations, then $\mu = E(N_t) = t \cdot p_{\text{path}} \geq 5\ell^*$ (treating N_t as binomially distributed), and apply the Chernoff bound:

$$\begin{aligned} P(N_t < \ell^*) &= P(N_t < (1 - 4/5) \cdot \mu) \\ &< e^{-5\ell^* \cdot (4/5)^2/3} = O\Big(n^{-16/15}\Big) \end{aligned}$$

inserting $\ell^* \geq \log n$ in the last step.

At most $\ell \ln(\tau_{\max}/\tau_{\min})/\rho$ additional iterations are required to freeze the pheromone values after each discovery to preserve the $p_{\min}$ lower bound on the probability of discovering the next subpath in each iteration. Thus a shortest path from an arbitrary v to t is discovered in at most $10e\ell^*/\tau_{\min} + \ell \ln(\tau_{\max}/\tau_{\min})/\rho$ iterations with high probability. Shortest paths from all of the n vertices in the graph are also found with high probability, which can be shown by applying a union bound:

$$\begin{aligned} P(\min_{v \in V} N_t \geq \ell^*) &\geq 1 - (1 - P(N_t < \ell^*))^n \\ &\geq 1 - nP(N_t < \ell^*) \\ &= 1 - O\Big(n^{-1/15}\Big) \end{aligned}$$

Treating this as a success probability in a geometric distribution, it is clear that, in expectation, no more than $O(1)$ phases of $10\ell^*/\tau_{\min} + \ell \ln(\tau_{\max}/\tau_{\min})/\rho$ iterations each are required. Thus, the expected number of iterations before all shortest paths are rediscovered after the weight function changed is $O(\ell^*/\tau_{\min} + \ell \ln(\tau_{\max}/\tau_{\min})/\rho)$.

As remarked in [ST12a], the unique shortest paths constraint can be dropped at the cost of introducing a $\log n$ factor to the first component of the bound, by requiring in the analysis that all shortest paths of length i are found and frozen

before λ-MMAS is considered to have a change to discover a shortest path of length $i+1$; the expected time until all shortest paths of a particular length are found (given that all shorter paths have been found and frozen) is dominated by the coupon collector problem, and is therefore $O(\log n/\tau_{\min})$; in total, there are no more than ℓ^* such phases.

To illustrate the potential effects of a one time change, consider the graph in Figure 2.1 and the weight functions w_1 and w_2 shown below; the shortest paths of w_1 visit r immediately, while w_2 avoids r if possible:

$$w_1(\alpha) = \begin{cases} -1 & \text{if } \alpha = (r,t) \\ 1 & \text{otherwise} \end{cases} \qquad w_2(\alpha) = \begin{cases} n & \text{if } \alpha = (r,t) \\ 1 & \text{otherwise} \end{cases}$$

If the pheromone values are frozen to the shortest paths using w_1, λ-MMAS will require a large number of iterations to rediscover all of the shortest paths when the weight function is changed to w_2. The proof of the following lower bound is also inspired by a related result in [ST12a], which allows for even finer bounds, but considers a uniform initialization of pheromone values, which is not given after λ-MMAS has frozen the pheromones to favor specific shortest paths.

Theorem 2.4 *One-time changes to the weight function may require an expected $\Omega(\ell/\tau_{min})$ iterations for 1-MMAS with $\rho = 1$ to discover all shortest paths, where ℓ is the maximum number of arcs in any shortest path to t in the new graph.*

Proof. The theorem is proved by example: we will show that if, on the graph of Figure 2.1, all shortest paths under w_1 are discovered and frozen, $\Omega(\ell/\tau_{\min})$ iterations are required to rediscover the shortest paths after the weight function is switched to w_2. Given the high evaporation rate ($\rho = 1$), after the first iteration, all pheromone values are at the pheromone bounds, and switch between those instantly whenever new shortest paths are discovered by λ-MMAS. Assume for a moment that the optimization proceeds as in the proof of Theorem 2.3: only shortest paths with a single non-$\tau_{\max}$ arc are discovered until shortest paths from all vertices have been found.

The probability that the next subpath (with only one non-$\tau_{\max}$ arc) is discovered in a single iteration is at most $\tau_{\min}$. Let T_i be the number of iterations between the discoveries of subpaths $i-1$ and i; as $\Delta \geq 2$ and hence $\tau_{\min} \leq 1/2$, the probability that the next subpath is not discovered for at least $1/(2\tau_{\min})$ iterations is:

$$\begin{aligned} P(T_i > 1/(2\tau_{\min})) &\geq (1-\tau_{\min})^{1/(2\tau_{\min})} \\ &= \sqrt{(1-\tau_{\min})^{1/\tau_{\min}}} \geq 0.5 \end{aligned}$$

When the weight function is changed to w_2, the entire ℓ-arc longest shortest path needs to be rediscovered, changing the pheromone values on the outgoing arcs from each of the ℓ non-trivial vertices. Let N be the number of subpaths for which $T_i > 1/(2\tau_{\min})$, and $\mu = E(N) \geq \ell/2$. A Chernoff bound shows that at least a quarter of the subpaths will require more than $1/(2\tau_{\min})$ iterations each with overwhelming probability:

$$\begin{aligned} P(N \geq \ell/4) &= 1 - P(N < (1-1/2)\mu) \\ &> 1 - e^{-\mu/2^2/3} \geq 1 - e^{-\ell/24} \end{aligned}$$

Thus, if no shortest paths with more than one non-$\tau_{\max}$ arc are discovered, $\Omega(\ell/\tau_{\min})$ iterations are required with overwhelming probability.

How likely is λ-MMAS not to find any shortest paths with more than one non-$\tau_{\max}$ arc? The probability p_{f} of discovering such a path is greatest at the beginning of the process, when shortest paths with 2, 3, ..., ℓ non-$\tau_{\max}$ arcs can be discovered; and the probability that no such paths are found for ${1/\tau_{\min}}^2$ iterations is at least a constant:

$$\begin{aligned} p_{\mathrm{f}} &\leq {\tau_{\min}}^2 \sum_{i=2}^{\ell} {\tau_{\min}}^{i-2} < 2 \cdot {\tau_{\min}}^2 \\ (1-p_{\mathrm{f}})^{1/{\tau_{\min}}^2} &\geq 1/16 \end{aligned}$$

assuming $\tau_{\min} \leq 1/2$ as before.

Thus, with at least constant probability, λ-MMAS does not discover any shortest paths with more than one non-$\tau_{\max}$ arc in ${1/\tau_{\min}}^2 \geq \ell/\tau_{\min}$ iterations, and $\Omega(\ell/\tau_{\min})$ iterations are required to rediscover the shortest paths with overwhelming probability if only subpaths with a single non-$\tau_{\max}$ arc are discovered. Therefore, the expected number of iterations before the shortest paths are rediscovered by λ-MMAS (with $\lambda = 1$ and $\rho = 1$) after switching from w_1 to w_2 is $\Omega(\ell/\tau_{\max})$.

Conversely, there also exist combinations of weight functions for which the shortest paths may be rediscovered relatively quickly – for example, switching from w_2 to w_1 on the graph of Figure 2.1 is easier than the alternative, as a low $\ell = 2$ value reduces the interdependency between shortest paths, allowing them to be discovered in parallel without requiring ants to select more than one non-$\tau_{\max}$ arc at a time. Additionally, changing the weight of the (r, t) arc to values between the extremes considered in w_1 and w_2 would change only some of the shortest paths, reducing the amount of pheromone values that need to be updated.

These analyses suggest that λ-MMAS spends most of the iterations *waiting* to discover the next subpath. The expected number of iterations spent waiting can be reduced by increasing the number of ants simulated in each iteration, as expanded upon in Theorem 2.5; this is similar in effect to increasing the offspring population size in a $(1 + \lambda)$ EA, as considered by [JJW05].

Theorem 2.5 *Using $\lambda = 2e/\tau_{min}$ ants allows λ-MMAS to discover new shortest paths in expected constant time, allowing all shortest paths to be rediscovered in $O(\ell + \ell \ln(\tau_{max}/\tau_{min})/\rho)$ iterations.*

Proof. The probability that a single ant will construct a specific path with only one non-$\tau_{\max}$ arc is at least $\tau_{\min}/2e$, so the probability that at least one of λ ants constructs it is at least p_c:

$$p_c \geq 1 - (1 - \tau_{\min}/2e)^{2e/\tau_{\min}} \geq 1 - 1/e$$

Shortest paths up to ℓ arcs long may therefore be discovered using the same mechanism as in Theorem 2.3, with each subpath taking an expected $e/(e - 1) = O(1)$ iterations to discover, and $\ln(\tau_{\max}/\tau_{\min})/\rho$ additional iterations to freeze (as the freezing rate is unaffected by the increased number of ants simulated in each iteration). The total number of iterations required to rediscover and freeze all shortest paths is, per linearity of expectation, at most $O(\ell + \ell \ln(\tau_{\max}/\tau_{\min})/\rho)$.

While increasing the number of ants simulated does not reduce the amount of work performed by the algorithm, it does allow the work to be parallelized to a greater extent, as each ant in an iteration can be simulated independently of others, only needing to compare the fitness values of the constructed paths at the end of the iteration.

The theorems presented in this section used ℓ, the maximum number of arcs in any shortest path to t, to bound the complexity of rediscovering the shortest paths after a weight function change. In the examples considered so far, this has been accurate; though there are graphs where it is an overestimation (see e.g. Figure 2.2). More precise bounds could instead consider, for each shortest path, the number of vertices along the path at which an ant would have opportunity to deviate from the shortest path, and for which the shortest path was altered by the weight function change.

2.4 Periodic local changes

If the weight functions change more frequently than the expected number of iterations required to rediscover and re-freeze the shortest paths after each change, the process can no longer be treated as a series of recoveries from one-time changes. Taken to the extreme, the weight functions could change in *every* iteration of λ-MMAS. In this section, we consider such rapid oscillation between two weight functions the shortest paths of which are extremely similar, differing only in the choice of outgoing edge from a single vertex in the graph. Somewhat surprisingly, λ-MMAS, with only a constant number of ants, maintains at least a constant probability of constructing the optimum shortest path solution in each iteration in this setting. We will then show that if the oscillation between these two weight functions is slower (though not as slow as to allow analysis as a series of one-time changes), 1-MMAS is also able to track the optimum.

Throughout this section, λ-MMAS is considered able to track the optimum solution if it is able to maintain a constant probability of constructing the shortest paths in any given iteration, i.e. the best possible probability of success when a starting a constant number of ants at each vertex. In the local changes setting, only a single choice of outgoing edge changes between the shortest paths of the two different weight functions. λ-MMAS must therefore keep the pheromone values on the oscillating edges within a constant distance of $1/2$, as approaching the pheromone bounds would reduce the probability that the correct shortest paths is constructed in every iteration to a sub-constant value. We note that adjusting the pheromone bounds directly, to keep the pheromones close to $1/2$, would be counterproductive: such bounds would reduce the probability of ants following the non-oscillating portion of the shortest paths, which would've been reinforced, to a sub-constant value when alternative paths are available. Additionally, a low evaporation rate is required in order for λ-MMAS to be able to keep the pheromone values on the oscillating edges close to $1/2$ for significant amounts of time, as, intuitively, a too-high evaporation rate causes λ-MMAS to forget that an arc was part of a shortest path sooner rather than later.

This section considers oscillation between two similar weight functions, shown below, on the graph in Figure 2.2.

$$w_1(\alpha) = \begin{cases} 1 & \text{if } \alpha = c \\ 0 & \text{otherwise} \end{cases} \qquad w_2(\alpha) = \begin{cases} -1 & \text{if } \alpha = c \\ 0 & \text{otherwise} \end{cases}$$

The graph and weight functions are chosen such that the longest shortest path has at least $n - 2$ arcs, the shortest paths are only changed at a single vertex, which isolates the simplest possible dynamic shortest path problem, and the

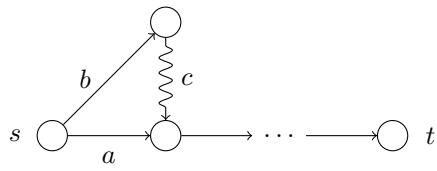

Figure 2.2: Changing the weight of arc c controls which of the a and b arcs is on the shortest path from s to t.

simulated ants only have the opportunity to deviate from the shortest path at a single vertex. The latter limit on the opportunities to deviate from the shortest paths is used only to simplify the analysis, and could be relaxed at the cost of increasing the required number of ants by a constant factor, as discussed at the end of this section; thus, similar results could be obtained for less artificial graphs, as long as the oscillation could be isolated to a single triangle in the graph.

In this situation, it is possible for λ-MMAS to use relatively few ants and still maintain the ability to correctly identify the shortest path in each iteration with at least constant probability for a super-polynomial number of iterations.

Lemma 2.6 *Consider switching between the weight functions w_1 and w_2 on the graph of Figure 2.2 during every iteration. Starting $\lambda = 4$ ants at each vertex is sufficient to ensure that λ-MMAS, with $\rho \in o(1/\log n)$, will keep the pheromone values on outgoing arcs from s in the $[0.3, 0.7]$ range for a super-polynomial number of iterations with respect to n with a probability super-polynomially close to 1.*

Proof. Let $\tau'_t = \min(\tau_t(a), \tau_t(b))$ be the minimum of the pheromone values on the arcs a and b after iteration t. Consider the effect of two sequential pheromone updates on τ'_t:

$$-\rho < \tau'_{t+2} - \tau'_t < 2\rho \tag{2.1}$$

as evaporation may at most reduce the smaller value by $\rho/2$ (as $\tau_t(a)+\tau_t(b) = 1$, and hence $\tau'_t \leq 1/2$), and reinforcement may at most increase the smaller value by ρ.

If the arc with the smaller pheromone value τ'_t is only reinforced in the second of two pheromone updates, the effect is:

$$\tau'_{t+2} - \tau'_t \geq \tau'_t(1-\rho)^2 + \rho - \tau'_t \geq \rho - 2\rho\tau'_t \tag{2.2}$$

It is convenient to scale the minimum pheromone values, such that a constant number of pheromone updates alter the scaled values by at most a constant:

$$g(\tau_t) = \min(\tau_t(a), \tau_t(b))/\rho$$

Consider the expected change $\Delta_t(\tau'_t) = g(\tau_{t+2}) - g(\tau_t)$ of these scaled values two pheromone updates after an update that favored reinforcing the arc with the τ'_t pheromone value. For a lower bound, assume that the τ'_t arc is never reinforced during the next iteration (when it is not on the shortest path), and let p_f be the probability that it is also not reinforced during the iteration after the next (when it is on the shortest path), which yields:

$$\begin{aligned} E(\Delta_t(\tau'_t)) &> (1 - p_\mathrm{f}) \cdot (1 - 2\tau'_t) + p_\mathrm{f} \cdot (-1) \\ &= 1 - 2p_\mathrm{f}(1 - \tau'_t) - 2\tau'_t \end{aligned}$$

by applying g to the bounds in (2.1) and (2.2). For the weight functions w_1 and w_2, $p_\mathrm{f} \leq (1 - \tau'_t(1 - \rho))^\lambda$. The drift is at least 0.02 when $0.3 \leq \tau'_t \leq 0.4$, $\lambda = 4$, and $\rho \leq 0.05$.

The simplified drift theorem [OW11, OW12] can then be applied to the scaled values when the pheromones are within this region, treating two iterations of λ-MMAS as a single step in a Markov process over the scaled values, ensuring that the arc with the minimum pheromone value was always favored by the last iteration in the step. This requirement simplifies the following analysis, but may mean that the pheromone values enter the considered drift region during a "wrong" iteration; to accommodate this, we will adjust the size of the drift region by the equivalent of one iteration, pessimistically assuming that the iteration resulted in the pheromone values being brought closer to the bounds.

The drift is greater than 0.02 within a region specified by $a = (3/10)/\rho$ and $b = (4/10)/\rho$. As mentioned perviously, the length of the drift region may need to be reduced by the equivalent of a single pheromone update to ensure that the last pheromone update favored reinforcing the τ'_t arc, so the length of the drift region is $\ell \geq b - a - 1 = \omega(\log n)$. This region then satisfies the first requirement of the simplified drift theorem.

As a consequence of (2.1), the process cannot make large changes to the scaled values in a single iteration. The second requirement of the simplified drift theorem is satisfied by setting $r(\ell) = 4$ and $\delta = 1$:

$$P(|\Delta_t| \geq j) \leq \frac{r(\ell)}{(1+\delta)^j} = 2^{2-j}$$

as $P(|\Delta_t| > 2) = 0$, and the right side of the inequality is at least 1 for $j \leq 2$.

Then, per the simplified drift theorem, there exists a constant c^* such that with probability $1 - 2^{-\omega(\log n)}$, the Markov process first reaches a state $X_t \leq a$ (and hence first encounters a $\tau'_t \leq 3/10$) after at least $2^{c^*\omega(\log n)}$ iterations.

Given that the pheromone values remain within this range for a super-polynomial number of iterations, the probability that the correct shortest path x^*_s is constructed also remains bounded for a super-polynomial number of iterations.

A more detailed analysis than the approximations used in Lemma 2.6 is needed to show whether $\lambda = 3$ ants are also sufficient. This setting is revisited as part of an experiment in Section 2.7.

Theorem 2.7 *Consider switching between the weight functions w_1 and w_2 on the graph of Figure 2.2 during every iteration. 4-MMAS with evaporation rate $\rho \in o(1/\log n)$ will with probability super-polynomially close to 1 be able to find the correct s-t shortest path with at least a constant probability in each iteration for any polynomial number of iterations.*

Proof. Let $\tau'_t = \min(\tau_t(a), \tau_t(b))$ after iteration t; then, the probability that x^*_s is correct is at least the probability of any ant constructing a path through the arc with pheromone value τ'. As long as τ'_t is at least a constant, e.g. 0.3, this probability is also at least a constant:

$$P(x^*_s \text{ is correct} \mid \tau'_t \geq 0.3) \geq 1 - (1 - 0.3)^4 = 0.7599$$

Per Lemma 2.6, $\tau'_t \geq 0.3$ for a super-polynomial number of iterations with probability super-polynomially close to 1, which proves the theorem.

The proof relies on a special property of the considered weight functions: only the choice made at s affects whether the ant constructs a correct shortest path to t. If the weight function or graph were changed to require the ants to follow a pheromone-reinforced path after leaving s, $\lambda = 4$ ants may be insufficient to ensure a positive drift. Using the usual pheromone bounds, the probability of an ant successfully following any pheromone-reinforced path is at least $1/e$, so the failure probability can be adjusted accordingly:

$$p_{\mathrm{f}} \leq (1 - \tau'_t(1 - \rho) \cdot 1/e)^{\lambda}$$

With this adjustment, $\lambda = 12$ ants are sufficient to ensure that the drift is greater than 0.003 for the same constraints on τ'_t and ρ as were used in Lemma 2.6.

2.5 Slower local changes

The preceding section has shown that an ant colony can be useful when tracking rapid oscillations, as increasing the number of ants increases the amount of exploration the λ-MMAS algorithm performs before altering pheromone values. If the evaporation rate is also sufficiently low, during rapid oscillation between two similar shortest paths, λ-MMAS is able to keep the pheromone values on the affected arcs close to 0.5, essentially being able to remember that both outgoing arcs have been part of the shortest path recently.

A colony with only a single ant cannot track *rapid* oscillation in this fashion, but as the following theorem shows, it can keep the pheromone values within a constant range if the oscillation is sufficiently slow (i.e. each shortest path remains the optimum for at least $T \in \omega(\log n)$ iterations). With only a single ant, the pheromones are instead kept within a constant range by the difference in magnitude of the pheromone updates which change the values toward the middle, and those toward the extremes.

Theorem 2.8 *Consider switching between the weight functions w_1 and w_2 on the graph of Figure 2.2 every $T = 2/\rho$ iterations. 1-MMAS with an evaporation rate $\rho \in o(1/\log n)$ will keep the pheromone values within the $[0.01, 0.99]$ range for an expected super-polynomial number of iterations.*

Proof. Consider the optimization process as a series of phases, with alternating phases using w_1 and w_2 as the weight functions. Then, at the beginning of an optimization phase, at iteration t, let $\tau_t(a)$ be the pheromone value on the arc favored by (i.e. on the shortest path of) the weight function used in the phase that has just began. We shall show that if $\tau_t(a) \in [0.03, 0.5]$, then with super-polynomially high probability, the pheromone value on the non-favored arc will be in this range at the end of this optimization phase, i.e. $\tau_{t+T}(b) \in [0.03, 0.5]$.

If the new shortest path is discovered immediately, the pheromone value $\tau_t(b)$ will not be reinforced in the following T iterations. Recall that $\tau_t(a) + \tau_t(b) = 1$, so $\tau_t(b) \geq 0.5$, which after T evaporating pheromone updates will be reduced to no less than 0.03, given that $\rho \leq 0.1$, i.e. n is large enough:

$$\begin{aligned}\tau_{t+T}(b) &\geq \tau_t(b)(1-\rho)^T \\ &\geq 0.5(1-\rho)^{2/\rho} \geq 0.03\end{aligned}$$

So regardless of when the new shortest path is discovered, the pheromone value $\tau_{t+T}(b)$ cannot be reduced below the range acceptable for the next optimization phase.

Suppose that the new shortest path is not discovered immediately, so the pheromone value on arc a is evaporated until the shortest path is discovered. In the first $T/2$ iterations, the pheromone value on arc a cannot be reduced below 0.01, given that $\rho \leq 0.1$ as before:

$$\begin{aligned}\tau_{t+T/2}(a) &\geq \tau_t(a)(1-\rho)^{T/2} \\ &\geq 0.03(1-\rho)^{1/\rho} \geq 0.01\end{aligned}$$

Thus the shortest path is discovered in the first $T/2$ iterations with super-polynomially high probability:

$$1-(1-0.01)^{T/2} \geq 1-0.99^{1/\rho} \geq 1-0.99^{\omega(\log n)} \geq 1-n^{-\omega(1)}$$

If the shortest path is discovered in the first $T/2$ iterations, the pheromone value on arc b will be evaporated for at least $T/2$ iterations:

$$\begin{aligned}\tau_{t+T}(b) &\leq \tau_{t+T/2}(b)(1-\rho)^{T/2} \\ &\leq 0.99(1-\rho)^{1/\rho} \leq 0.99/e < 0.5\end{aligned}$$

and will therefore be within the acceptable range for the next optimization phase.

Therefore, if at the beginning of the phase, the pheromone value on arc favored by the phase is within $[0.03, 0.5]$, the pheromone value on the arc favored by the next phase will also be within this interval at the start of the next phase with super-polynomially high probability.

The first phase of the optimization process satisfies this precondition (as pheromone values are initialized s.t. $\tau_1(a) = \tau_1(b) = 0.5$); and as each phase has a super-polynomially small probability of failing, a failure occurs (potentially causing pheromone values to exit the $[0.01, 0.99]$ range) in an expected super-polynomial number of iterations.

This illustrates that if the oscillation is sufficiently slow, and the evaporation rate is sufficiently low, the pheromone update mechanism in MMAS can still prevent the pheromones from freezing for an expected super-polynomial number of iterations, even in a single-ant colony. More specifically, the magnitude of the change in the pheromone values of an update bringing pheromone values closer to $1/2$ is greater than that in an update bringing pheromone values closer to their bounds.

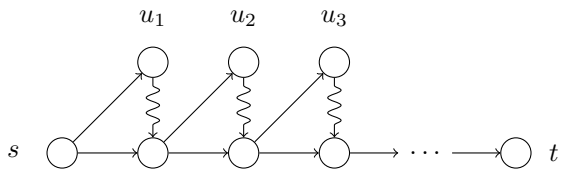

Figure 2.3: Multiple oscillating triangles on the path from s to t.

2.6 Periodic global changes

Using pheromones as an implicit memory, λ-MMAS is able to reliably construct shortest paths even with a constant number of ants when the oscillating weight functions only make slight alterations to the shortest paths in the graph. In this section, we consider an oscillation between two weight functions where the shortest paths at many vertices change, and there are many opportunities for an ant to deviate from the shortest path.

Suppose that there are k oscillating triangles on the path from s to t, as illustrated in Figure 2.3. The weight functions w_1 and w_2 can be extended such that the shortest path using w_1 avoids all of the vertices $u_1, \ldots, u_k$, while w_2 favors visiting all of those vertices. Let U be the set of arcs leaving the u_i vertices; the extended weight functions are then:

$$w_1(\alpha) = \begin{cases} 1 & \text{if } \alpha \in U \\ 0 & \text{otherwise} \end{cases} \qquad w_2(\alpha) = \begin{cases} -1 & \text{if } \alpha \in U \\ 0 & \text{otherwise} \end{cases}$$

When k is large, i.e. $k \in \omega(\log n)$, ensuring that the correct shortest path from s to t is constructed with at least constant probability in each iteration requires starting a super-polynomial number of ants at each vertex. While tracking the optimum during an oscillation between two dramatically different weight functions is difficult, this setting illustrates a limitation of the pheromone memory used by MMAS: it is unable to store distinct solutions (unlike e.g. a $(\mu + \lambda)$ EA, which might be able to store both shortest path trees in the population, and clone both in any given iteration if μ and λ are large enough).

Theorem 2.9 *Rapid oscillation between two weight functions may require as many as $2^{\Omega(n)}$ ants to be started at each vertex in order for λ-MMAS with $\rho \leq 0.5 - \Omega(1)$ to maintain at least a constant probability of constructing the shortest paths in each iteration.*

PROOF. The theorem is proved by example: we will show that on the graph of Figure 2.3 with $k = \Omega(n)$ triangles, with weight functions swapping between w_1 and w_2 during every iteration, λ-MMAS requires $\lambda = 2^{\Omega(k)} = 2^{\Omega(n)}$ ants to be started at each vertex to maintain a constant probability of constructing the shortest path from s in each iteration. Essentially, λ-MMAS is no better in this setting than exhaustive search during every iteration; this is somewhat expected as the shortest paths for w_1 and w_2 are substantially different, and λ-MMAS is not able to store distinct solutions in the pheromone memory.

Consider the pheromone state in an iteration of λ-MMAS. Classify triangle i as *favorable* with respect to the current weight function if the pheromone value on the correct shortest path arc from s_i is at least 0.5. If less than $k/2$ triangles are favorable, constructing the shortest path from s requires a number of ants that is exponential with respect to k, as at least some ant has to make the correct choice of outgoing arc in all of at least $k/2$ unfavorable triangles. Let p_s be the probability that one of the λ ants started at s constructs the correct shortest path, p_k be the probability that a particular ant correctly navigates the $k/2$ unfavorable triangles, and $p_1 \leq 0.5$ be the probability that a particular ant correctly navigates a specific unfavorable triangle, then

$$p_s \leq 1 - (1 - p_k)^\lambda \leq 1 - (1 - (0.5)^{k/2})^\lambda \leq \lambda \cdot 2^{-k/2}$$

To ensure that p_s is at least a constant, at least $2^{\Omega(k)}$ ants must be started from s.

If more than half the triangles are favorable in the current iteration, an exponential number of ants will be required to construct the shortest paths from s in the next iteration. Let $\tau \geq 0.5$ be the pheromone value on the correct shortest path arc in a triangle in the current iteration; then $1 - \tau$ is the pheromone value on the other arc in the same triangle. In the next iteration, that other arc will be on the shortest path, and its pheromone value will be at most $0.5 + \rho$. As long as ρ is upper-bounded by a constant less than 0.5, the derivation of p_s can be repeated using $p_1 \leq 0.5 + \rho$ to show that an exponential number of ants must be started from s in the next iteration.

Thus, at least every second iteration requires λ to be exponential with respect to the number of triangles in order for λ-MMAS to have a constant probability of at least one ant constructing the shortest path from s to t.

A closer analysis may be able to show that in expectation, no more than some constant fraction of the triangles will favor a specific weight function. This would mean that an exponential number of ants is required in every iteration (rather than at least every second iteration as concluded in the proof of Theorem 2.9) to ensure a constant probability of the correct s-t shortest path being constructed in an arbitrary iteration.

In general, the smaller the evaporation rate, the less effect the first iteration has on the pheromone values, reducing the amount of bias towards w_1 introduced at the start of the optimization process. A smaller λ has a similar effect: ants started further away from t may not be able to find the full shortest path, which reduces the probability that the w_1-favored arc is reinforced. Subsequent iterations will, due to drift, reduce the bias introduced by the asymmetry at the start of the process; i. e. if more triangles favor w_1 than w_2, in expectation more w_1-favoring triangles will be reinforced towards w_2 during a w_2 iteration than w_2-favoring triangles towards w_1 during a w_1 iteration.

In any case, the example illustrates the nature of the difficulty: the pheromone memory is not capable of separating the two weight functions, and thus does not scale well if the differences between the weight functions extend over multiple vertices.

2.7 Experiments

To illustrate the behavior of λ-MMAS in settings introduced in previous sections, a number of simulations were performed; all averages and medians presented in the figures in this section are computed based on data from 1000 simulations for each combination of MMAS parameters ρ and λ, and, where applicable, oscillation frequency.

In the local changes setting with a single triangle, λ-MMAS with $2 \leq \lambda \leq 4$ ants started at each vertex was simulated with different evaporation rates, recording the first iteration at which the pheromone values on the triangle exited the $[1/10, 9/10]$ range (i. e. were about to freeze) for the first time; the averages are shown in Figure 2.4. While it seems the $\lambda = 2$ curve is concave, which suggests that $\lambda = 2$ ants are not sufficient to prevent pheromones from freezing for a super-polynomial number of iterations, it seems that $\lambda = 3$ ants may be sufficient to achieve this.

For 1-MMAS in the local changes setting, the single-ant colony was simulated with different evaporation rates and phase lengths T, recording the first iteration at which pheromone values on the triangle arcs exited specific ranges; the results are shown in Figure 2.5. There appears to be a more significant difference between $T = 4$ and $T = 3$ than $T = 3$ and $T = 2$. Additionally, it seems that the pheromone values also remain within more restrictive ranges than $[0.01, 0.99]$ for significant numbers of iterations.

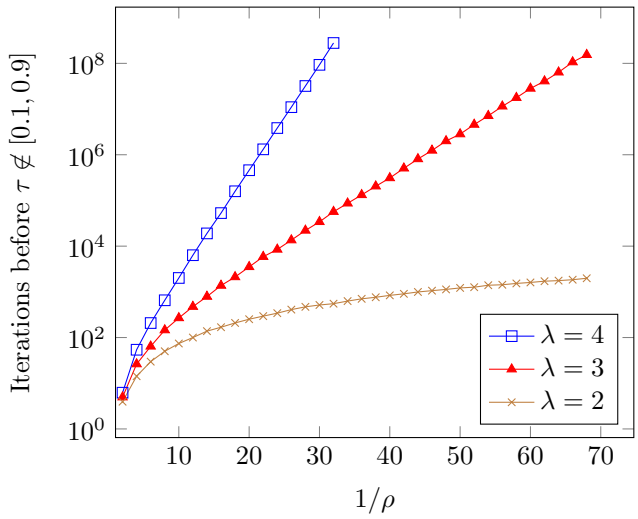

Figure 2.4: Number of iterations before pheromone values on the triangle leave $[0.1, 0.9]$ for the first time for varying evaporation rates for λ-MMAS, with $\lambda = 2, 3, 4$ ants started at each vertex during a single iteration.

In the global changes setting with $k = 200$ triangles in series, λ-MMAS was simulated with $\lambda = 6$ and $\rho = 1/50$. The number of triangles with pheromone values outside the $[1/4, 3/4]$ range was recorded; Figure 2.6 displays the average number of triangles favoring with pheromone values outside this range, collated by which weight function the pheromone values were favoring. Notably, the number of triangles favoring either weight function increases at approximately equal rates, and eventually stabilizes, keeping approximately a third of the triangles in the graph within the $[1/4, 3/4]$ pheromone range.

Figure 2.7 presents the same metric in the global changes setting with the evaporation rate set to $\rho = 1$ to maximize the impact of the first iteration. Somewhat surprisingly, even with this extreme evaporation rate, the ratio between the number triangles frozen to favor w_1 and w_2 is initially only 3:2. As suggested in Section 2.6, the ratio decreases gradually over time, illustrating that the impact of the initial bias is counteracted, rather than compounded, by subsequent iterations.

The number of triangles favoring w_1 in Figure 2.6 is consistently greater than the number of triangles favoring w_2; this is an artifact of always recording the number of triangles after an iteration using the w_1 weight function. In 2.7, the

data points alternate between iterations using w_1 and w_2, causing the lines to appear jagged.

2.8 Conclusions

We have studied a simple ACO algorithm called λ-MMAS for dynamic variants of the single-destination shortest paths problem. Building upon previous results for the special case of 1-MMAS, it was studied to what extent an enlarged colony using λ ants per vertex helps in tracking an oscillating optimum. We showed that λ-MMAS, even with constant number of ants per vertex, can deal with dynamic shortest paths problems where the shortest paths are changed infrequently (by rediscovering the shortest paths before the next change occurs), or changed rapidly between a small set of possible similar solutions (by keeping the pheromone values close to 1/2 for the affected arcs). It has also been shown that even a single-ant colony can prevent the pheromone values from freezing for a sufficiently slow oscillation. However, we also identified an example where a fast oscillation between two weight functions that are sufficiently different is so hard to track that a super-polynomial number of ants is needed. Furthermore, we have discussed properties of the dynamics that make the problem hard. Experiments show that the theoretical results are also valid for small problem dimensions and illustrate effects that are not yet visible in theorems.

2.9 Future Work

In the future, the performance of λ-MMAS in settings with more complex weight function changes could be analyzed – for instance, generalizing the results to oscillations with constant phase lengths, oscillations between more than two weight functions, or using a less regular schedule than was considered here. Having different components of the graph oscillate at different frequencies could also pose an interesting challenge to ACO algorithms, one that could potentially be addressed by maintaining several sets of pheromones updated using different evaporation rates.

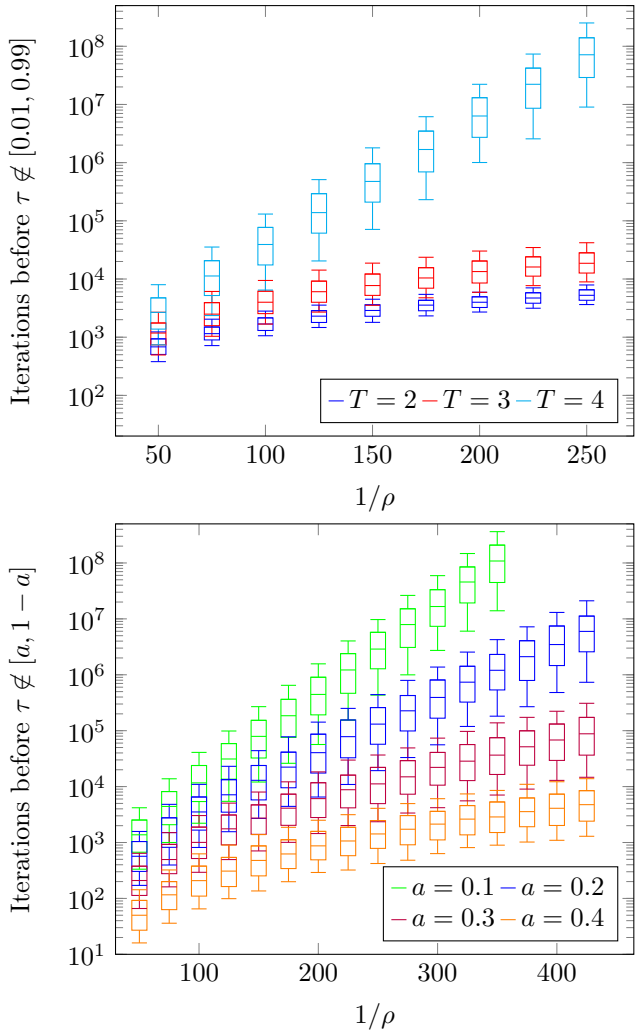

Figure 2.5: Local changes with slower oscillation periods; showing the median, quartiles, and 9th and 91st percentiles of the number of observations required to leave specific pheromone ranges. The left subfigure illustrates the effect of phase length, while the right subfigure illustrates, for $T = 4$, the number of iterations during which 1-MMAS pheromone values remain within $[a, 1 - a]$ for various constants a.

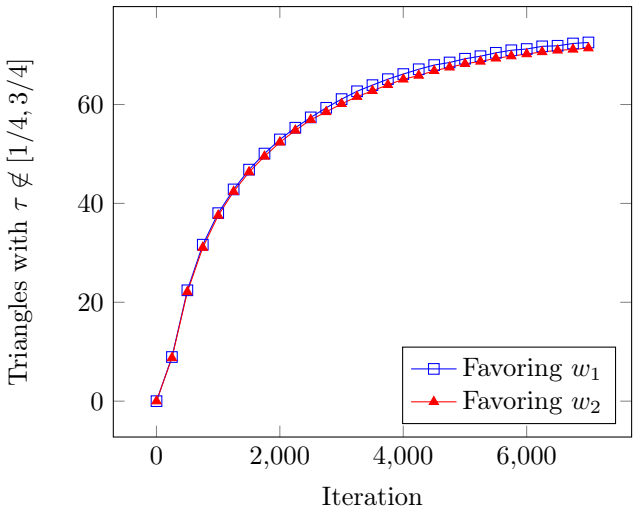

Figure 2.6: 200 triangles in series; average across 1000 simulations; $\lambda = 6, \rho = 1/50$.

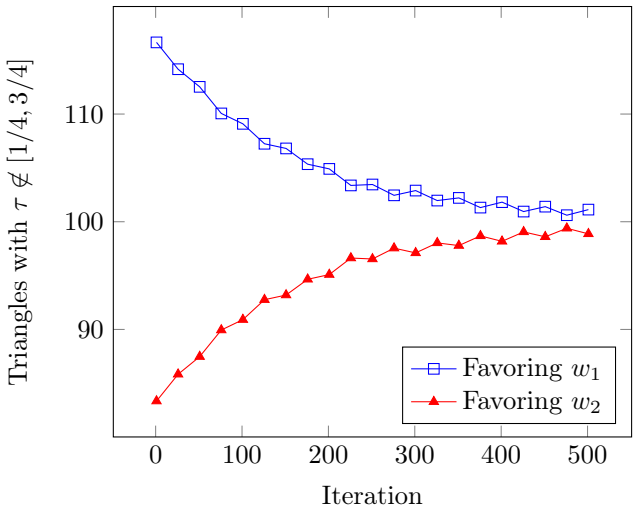

Figure 2.7: 200 triangles in series; average across 1000 simulations; $\lambda = 6, \rho = 1$.

Chapter 3

MMAS versus Population-Based EA on a Family of Dynamic Fitness Functions

Andrei Lissovoi Carsten Witt

DTU Compute, Technical University of Denmark

We study the behavior of a population-based EA and the Max-Min Ant System (MMAS) on a family of deterministically-changing fitness functions, where, in order to find the global optimum, the algorithms have to find specific local optima within each of a series of phases. In particular, we prove that a (2+1) EA with genotype diversity is able to find the global optimum of the Maze function, previously considered by Kötzing and Molter (PPSN 2012, 113–122, [KM12]), in polynomial time. This is then generalized to a hierarchy result stating that for every μ, a (μ+1) EA with genotype diversity is able to track a Maze function extended over a finite alphabet of μ symbols, whereas population size $\mu - 1$ is not sufficient. Furthermore, we show that MMAS does not require additional modifications to track the optimum of the finite-alphabet Maze functions, and, using a novel drift statement to simplify the analysis, reduce the required phase length of the Maze function.

3.1 Introduction

Evolutionary algorithms (EAs) are a class of nature-inspired algorithms that can be applied to solve a wide variety of optimization problems. Runtime analysis of nature-inspired algorithms has advanced considerably in recent years [AD11, NW10b], though most focus on static optimization problems, where the objective is simply to find the global optimum within the least amount of steps. Many real-world optimization problems are dynamic in nature, meaning that the optimal solution to a given problem may change as the problem conditions change over time, and the algorithms therefore need to be able to not only find the optimum at some point of time, but also to *track* the optimal solution over time as the problem changes.

Application of EAs to Dynamic Optimization Problems is the subject of study in the Evolutionary Dynamic Optimization field, which in recent years has attracted much activity. Many applications of evolutionary algorithms on dynamic problems are considered in literature [NYB12, ANS13], and there are runtime analyses building on theoretical studies of evolutionary algorithms for dynamic problems [RLY09, JS05, Dro03]. The utility of a population for tracking problems was studied in evolutionary computation by Jansen and Schellbach [JS05], while different mechanisms for ensuring population diversity have been considered by Oliveto and Zarges [OZ13]. In particular, a mechanism called *genotype diversity* was proved to be inefficient on a particular dynamic problem. Recently, Jansen and Zarges [JZ14] compared evolutionary algorithms and artificial immune systems on a bi-stable dynamic optimization problem.

In [KM12], Kötzing and Molter introduced a dynamic pseudo-boolean function called Maze, the optimum of which slowly moves from the all-ones to the all-zeroes bit string in n phases, in each of which the optimum oscillates between two solutions that differ in the value of a single bit. The paper shows that while the Max-Min Ant System is able to track the changes occurring in this fitness function and find the optimum all-zeroes string within polynomial time, a (1+1) EA loses track of the optimum and requires an exponential amount of time to find the all-zeroes optimum with high probability.

In this paper, we consider the impact of introducing a population and a simple diversity mechanism to the (1+1) EA, showing that a (2+1) EA with genotype diversity is able to track the optimum of the Maze function. We then generalize the Maze to a function over a finite alphabet, and prove a hierarchy result with respect to the population size. More precisely, for any μ and any $c > 0$, there is a variant of the Maze such that a (μ+1) EA with genotype diversity will with probability $1 - O(n^{-c})$ succeed in tracking the optimum, whereas an EA with population size $\mu - 1$ will with probability $1 - O(n^{-c})$ lose track of the

optimum. Finally, we return to consider the performance of the MMAS* Max-Min Ant System Ant Colony Optimization (ACO) algorithm, and conclude that MMAS*, due to its pheromone memory model, is able to successfully track the optimum of the finite-alphabet version of MAZE, even with shorter phase lengths than considered in [KM12] (if the alphabet is not too big). Our proofs are based on mixing time arguments, fitness levels with tail bounds, and a new variant of a variable drift theorem, which allows for a tail bound on the probability distribution of the pheromone value in MMAS*.

The paper is structured as follows. Section 3.2 defines the dynamic fitness function MAZE, and the (μ+1) EA with genotype diversity and MMAS* algorithms generalized to larger alphabets. Section 3.3 proves the positive result for the simple (2+1) EA w. r. t. the classical MAZE function on bit strings. The hierarchy result for larger alphabets is proven in Section 3.4. Finally, Section 3.5 shows that MMAS* is efficient in tracking the optimum for every polynomial alphabet size. We finish with some conclusions.

3.2 Preliminaries

The MAZE dynamic fitness function defined in [KM12] consists of $n+1$ phases of $t_0 = kn^3 \log n$ iterations each, where k is a sufficiently large constant. During the first phase, which we will for convenience refer to as phase 0, MAZE is equivalent to ONEMAX, a fitness function that simply counts the number of 1-bits in an n-bit string. In each subsequent phase i, the function assigns fitness values $n+2$ and $n+1$ to bit strings $0^{i-1}01^{n-i}$ and $0^{i-1}11^{n-i}$, oscillating between assigning the higher fitness value to these individuals in a 0-0-1 pattern, with 0^i1^{n-i} being favored, i.e. having the higher fitness value, every two iterations out of three.

The version shown below has been extended to assign fitness values to n-character strings over a finite alphabet; for $r = 1$, i. e., bit strings, it is exactly equivalent to the original MAZE. In this context, ONEMAX counts the number of literal 1 characters in the string, and the sets OPT_p and ALT_p generalize the

Algorithm 3.1 (μ+1) EA with genotype diversity for a finite alphabet $\Sigma = \{0, 1, \ldots, r\}$.

Initialize $x^* = \{x_1, \ldots, x_\mu\}$, st. $x_i \neq x_j$ for all $i \neq j$.
for $t \leftarrow 1, 2, \ldots$ **do**
  Select x_a from x^* uniformly at random.
  Let $x_t = \mathrm{mut}_r(x_a)$.
  if $x_t \notin x^*$ **then**
    $G \leftarrow x^* \cup \{x_t\}$
    $x_{\min} \leftarrow \arg\min_{x \in G} f(x, t)$, chosen uniformly at random
    $x^* \leftarrow G \setminus \{x_{\min}\}$

0-0-1 oscillation pattern.

$$\textsc{Maze}(x,t) = \begin{cases} n+2 & \text{if } t > t_0 \wedge x \in \mathrm{OPT}(t) \\ n+1 & \text{if } t > t_0 \wedge x \in \mathrm{ALT}(t) \\ \textsc{OneMax}(x) & \text{otherwise} \end{cases}$$

$$\mathrm{OPT}(t) = \begin{cases} \mathrm{OPT}_{\lfloor t/t_0 \rfloor} & \text{if } t \neq 0 \mod 3 \\ \mathrm{ALT}_{\lfloor t/t_0 \rfloor} & \text{otherwise} \end{cases}$$

$$\mathrm{ALT}(t) = \begin{cases} \mathrm{ALT}_{\lfloor t/t_0 \rfloor} & \text{if } t \neq 0 \mod 3 \\ \mathrm{OPT}_{\lfloor t/t_0 \rfloor} & \text{otherwise} \end{cases}$$

$$\mathrm{ALL}_p = \left\{0^{p-1}x1^{n-p} \mid x \in \{0, 1, \ldots, r\}\right\}$$

$$\mathrm{OPT}_p = \left\{0^p 1^{n-p}\right\}$$

$$\mathrm{ALT}_p = \mathrm{ALL}_p \setminus \mathrm{OPT}_p$$

In this paper, we will examine how a (2 + 1) Evolutionary Algorithm (EA) performs on the original Maze function, and how the (μ+1) EA and MMAS algorithms perform on our finite-alphabet version. Rather than only focusing on expected optimization times (i.e. the expected iteration during which OPT_n is first constructed following the start of the final phase of the Maze), we consider the probability that OPT_n will be constructed by the algorithms by the end of phase n of the Maze, as well as what is likely to happen if it is not.

The (μ+1) EA with genotype diversity [Sto08, OZ13] is shown as Algorithm 3.1. Definition 3.1 extends the mutation operator mut_r to support a finite alphabet as in [Gun05, DP12]; for $r = 1$, it is equivalent to the standard mutation operator of the (1+1) EA.

Several lemmas throughout this paper state that "a specific event occurs with high probability." Definition 3.2 provides a more formal definition of this concept.

Algorithm 3.2 The MMAS* algorithm on a directed multigraph $G = (V, E)$, with pheromone bounds $\tau_{\min}$ and $\tau_{\max}$, and evaporation rate ρ, where $v_0, v_n \in V$ are the start and destination vertices respectively, and $\deg^+(v)$ is the outdegree of a vertex.

Initialize $\tau_a \leftarrow 1/\deg^+(v)$ for all $a = (v, v') \in E$
for $t \leftarrow 1, 2, \ldots$ **do**
  Let x_t be an empty path.
  $p \leftarrow v_0,\ S \leftarrow \{(p, v') \in E \mid v' \notin x_t\}$
  while $|S| > 0$ and $p \neq v_n$ **do**
    Select edge $e = (p, h')$ from S with probability:
      $p_e = \tau_e / \sum_{s \in S} \tau_s$
    Append e to x_t
    $p \leftarrow h',\ S \leftarrow \{(p, v') \in E \mid v' \notin x_t\}$
  if $t = 1$ **or** $f(x_t, t) > f(x^*, t)$ **then**
    $x^* \leftarrow x_t$
  for each $e \in E$ **do**
    $$\tau_e \leftarrow \begin{cases} \min(\tau_{\max}, (1-\rho)\tau_e + \rho) & \text{if } e \in x^* \\ \max(\tau_{\min}, (1-\rho)\tau_e) & \text{otherwise} \end{cases}$$

Definition 3.1 Let $\Sigma = \{0, 1, \ldots, r\}$ be a finite alphabet.

The mutation operator mut_r creates an image $y \in \Sigma^n$ from $x \in \Sigma^n$ by independently replacing each character x_i of x $(1 \leq i \leq n)$ with probability $1/n$ with a symbol drawn uniformly at random from $\Sigma \setminus \{x_i\}$.

Definition 3.2 An event E is said to occur *with high probability* if, for every constant $c > 0$, $\text{Prob}(E) = 1 - O(n^{-c})$. An event E is said to occur within $O(f(n))$ iterations *with high probability* if for every constant $c > 0$ there exists a $c' > 0$, possibly depending on c but not on n, such that E occurs within $c' f(n)$ iterations with probability $1 - O(n^{-c})$.

In Section 3.5, we will consider how the Max-Min Ant System [SH00] algorithm MMAS*, shown as Algorithm 3.2, is able to track the optimum of the finite-alphabet Maze function.

To use a path-constructing algorithm with a fitness function that assigns values to n-character strings, we use the construction graph shown in Figure 3.1: every n-character string $x \in \Sigma^n$ corresponds to the path from v_0 to v_n consisting of the edges $e_{i,c}$ for which $x_i = c$, and every v_0-v_n path corresponds to some $x \in \Sigma^n$ in this fashion. We use the standard choice for $\tau_{\max} = 1 - 1/n$, and set $\tau_{\min} = 1/(rn)$ and $\rho = \Omega(1/(rn))$ to accommodate an $(r+1)$-character

alphabet of the extended MAZE function. Notably, for $r > 1$, the sum of the pheromone values on edges leaving a vertex is no longer always equal to 1, and, when considering probabilities of selecting a particular edge, we have to use the bounds presented in Lemma 3.3, similar to [ST12a, Lemma 1] and [Sud11, Lemma 15].

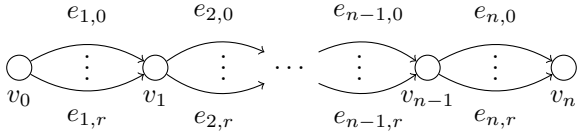

Figure 3.1: Construction graph used for MMAS* on the finite-alphabet MAZE function. There are $r + 1$ edges between each pair of vertices (v_{i-1}, v_i).

LEMMA 3.3 *The sum of the pheromone values on edges leaving any specific vertex v, τ_{sum}, can be bounded as:*

$$1 \leq \tau_{sum} \leq 1 + (deg^+(v) - 1)\tau_{min} = 1 + 1/n,$$

where $deg^+(v)$ is the out-degree of vertex v.

PROOF. Recall that $\deg^+(v) = r+1$ and $\tau_{\min} = 1/(rn)$. We prove these bounds by induction, noting that both hold at initialization, where $\tau_{\text{sum}} = 1$.

If, prior to a pheromone update, $\tau_{\text{sum}} \geq 1$, and no pheromone values are affected by the $\tau_{\max}$ border, $\tau_{\text{sum}}(1 - \rho) + \rho \geq 1$ after the update; while if there are pheromone values capped at $\tau_{\max}$ after the update, we note that even if all the other pheromone values are $\tau_{\min}$, $\tau_{\max} + r \cdot \tau_{\min} = 1$, proving the first inequality.

If, prior to a pheromone update, $\tau_{\text{sum}} \leq 1 + 1/n$, $\tau_{\text{sum}}(1 - \rho) + \rho \geq \tau_{\text{sum}}$ can only occur as a consequence of pheromone values being affected by the lower pheromone border (as $\tau_{\text{sum}} \geq 1$), i. e., those for which $\tau(1-\rho) \leq \tau_{\min}$, increasing τ_{sum} by at most $\rho\tau_{\min}$ for each such value. We note that there can be at most r such values, as the reinforced pheromone value cannot drop below $\tau_{\min}$, thus the sum of pheromone values after the update is at most $\tau_{\text{sum}}(1-\rho)+\rho+r\cdot\rho\tau_{\min} = \tau_{\text{sum}}(1 - \rho) + \rho + \rho/n \leq 1 + 1/n$, proving the second inequality. □

Note that both algorithms re-evaluate the fitness function when updating the population or the best-so-far solution. Similarly to [JZ14], the considered clock t is external to the MAZE function, making it possible to evaluate many solutions in one clock tick of the MAZE; this corresponds to having hardware available to

evaluate many solutions in parallel, while the problem changes occur at fixed intervals regardless of the number of parallel evaluations.

3.3 (2+1) EA on Maze

The (1+1) EA will with high probability require an exponential amount of time to find the 0^n optimum on the MAZE function, because there is at least a constant probability of ending each phase $p > 0$ with $x^* \neq 0^p1^{n-p}$, which lets the (1+1) EA revert to optimizing ONEMAX, destroying the 0-prefix constructed so far, and eventually requiring a large mutation to recover [KM12]. In this section, we will show that a (2+1) EA with genotype diversity avoids this problem by being able to store both of the oscillating individuals in each phase, thereby ensuring that 0^p1^{n-p} is in the population at the start of the next phase. This ensures that at the end of the last phase, the 0^n individual is in the population with high probability.

THEOREM 3.4 *The* (2+1) *EA with genotype diversity will with high probability have the* 0^n *optimum in the population at the end of the last phase of the* MAZE.

To show this, we will prove that the $1^n = \mathrm{OPT}_0$ individual is found with high probability during the initial ONEMAX phase, and the EA is then able to follow the oscillation process: if OPT_{p-1} is in the population at the beginning of phase p, OPT_p will be in the population at the end of that phase – meaning that at the end of final phase, $\mathrm{OPT}_n = 0^n$ will be in the population.

LEMMA 3.5 *The* $(2+1)$ *EA will discover the* 1^n *individual within* $O(n \log n)$ *iterations with high probability.*

PROOF. Recall that each phase of the MAZE lasts $kn^3 \log n$ iterations for some constant $k > 0$. The fitness level method can be applied: partition the possible populations into levels based on the maximum fitness value of any individual within the population. The probability of an iteration leaving the level where the highest fitness value is $n - i$ is at least $p_i \geq (1/2) \cdot (1/n) \cdot (1 - 1/n)^{n-1} \geq i/(2ne)$: i. e., that of selecting the best-valued individual in the population as the ancestor, and flipping a single zero bit. Even if the EA starts at the lowest of $n+1$ fitness levels, it has to leave at most n levels in order to reach the level where 1^n is in the population; in expectation, this takes $\mathrm{E}(T) \leq \sum_{i=1}^{n} \frac{1}{p_i} = 2en \log n$ iterations.

The high probability result is obtained by applying the tail bounds on fitness levels derived in [Wit14, Theorem 2]: using $s = 50n^2 > 4\pi^2 e^2 n^2/6 \geq \sum_{i=1}^{n} \frac{1}{{p_i}^2}$, and $h = 1/(2en) \leq \min p_i$, the probability that 1^n is found within $\mathrm{E}(T) + \delta$ iterations is at least $1 - e^{-\frac{\delta}{4} \cdot \min\{\delta/s, h\}}$. Setting $\delta = 50cn \log n$, where $c > 0$, and observing that $h < \delta/s$ for sufficiently large n, yields a probability of finding 1^n within $O(n \log n) = o(kn^3 \log n)$ iterations of at least

$$1 - e^{-\frac{50cn \log n}{4} \cdot \min\left\{\frac{50cn \log n}{50n^2}, \frac{1}{2en}\right\}} \geq 1 - e^{-\frac{50c \log n}{8e}} > 1 - n^{-2c}. \square$$

Lemma 3.6 *If OPT_{p-1} is in the population at the beginning of phase p, OPT_p will with high probability be in the population when phase p ends.*

Proof. As $\mathrm{OPT}_{p-1} \in \mathrm{ALL}_p$, it has a fitness value of at least $n + 1$ during phase p, and therefore cannot be removed from the population. The OPT_p individual can be constructed by selecting OPT_{p-1} as the ancestor and flipping a single bit, which occurs in each iteration with probability at least $1/(2en)$. The probability of this occurring within $n^3 \log n$ iterations is at least:

$$1 - (1 - 1/(2en))^{n^3 \log n} \geq 1 - e^{-\frac{1}{2e} n^2 \log n} = 1 - n^{-\Omega(n^2)}.$$

As OPT_p has a fitness value of at least $n+1$ during phase p, when constructed, it will replace a OneMax-valued individual in the population, and cannot be removed from the population during phase p. $\square$

Combined, Lemma 3.5 and Lemma 3.6 prove Theorem 3.4.

Proof of Theorem 3.4. By applying a union bound on the probabilities of failure during each of Maze's $n+1$ phases, we can conclude that the (2+1) EA finishes phase n with 0^n in the population with high probability. $\square$

3.4 (μ+1) EA and the finite-alphabet Maze

While a (2+1) EA with genotype diversity is able to track the optimum of the original Maze, it is interesting to consider whether there exist Maze-like functions for which a larger population is required. In this section, we use the Maze function extended over a finite alphabet $\Sigma = \{0, 1, \ldots, r\}$, and consider a ($\mu$+1) EA with genotype diversity, where $r \in O(n)$ and $\mu < n/2$. The phase

length is still $kn^3 \log n$ for some sufficiently large constant $k > 0$. We will build toward two results: a population of $\mu \leq r$ is insufficient to track the optimum, while $\mu > r$ is sufficient and enables the (μ+1) EA to find 0^n in polynomial time. These results are formalized in Theorems 3.7 and 3.8.

Theorem 3.7 *If $r \geq \mu$, $r \in O(n)$, and $\mu \leq n/(2+\varepsilon)$, where $\varepsilon > 0$ is an arbitrarily small constant, (μ+1) EA with genotype diversity will with high probability not find the 0^n optimum on* Maze *within a polynomial number of iterations.*

Theorem 3.8 *If $\mu, r \in O(n)$, and $r < \mu$, (μ+1) EA with genotype diversity will with high probability finish the last phase of the* Maze *with the 0^n optimum in the population given that an appropriately large constant k is chosen for the phase length t_0.*

As before, we need to verify that 1^n is found during the initial OneMax phase; this is done in Lemma 3.9; notably, the constant k in the phase length t_0 may need to be adjusted based on the high-probability constant c: i. e., in order to make sure that 1^n is located initially, the initial phase needs to be sufficiently long. Then, Lemma 3.10 shows that if an iteration begins with one of ALL_p individuals (i. e., those with a non-OneMax value during that phase) in the population, the population will with high probability be saturated with ALL_p individuals before the phase is over. These two lemmas are used in the proofs of both Theorems 3.7 and 3.8.

Lemmas 3.11 and 3.12 are used for Theorem 3.7. The former shows that once a population of $\mu \leq r$ individuals is filled with ALL_p individuals, the probability of OPT_p being in the population is at most a constant after a small number of additional iterations; while the latter states that if a phase p begins with no ALL_p individuals in the population (i. e., OPT_{p-1} was not in the population when the phase $p-1$ ended), the (μ+1) EA loses track of the optimum and reverts to optimizing OneMax with at least constant probability. This proof strategy is inspired by the lower bound for the (1+1) EA on the original Maze [KM12].

Lemma 3.9 *The (μ+1) EA will discover the 1^n individual within $O(\mu rn \log n) = O(n^3 \log n)$ iterations with high probability.*

The method used to prove Lemma 3.5 can be applied to prove Lemma 3.9 as well: we require that the best-fitness individual in the population is chosen as the ancestor, and the mutation operator changes a single non-1 character into a 1, resulting in an individual with a higher OneMax-value than any previously in the population.

Lemma 3.10 *During phase p, once an individual from ALL_p is in the population, the population will contain $\min(\mu, r+1)$ individuals from ALL_p within $O(\mu r n \log n)$ iterations with high probability.*

Proof. The general form of the fitness level method can be applied by partitioning the μ-individual populations into levels by the number of ALL_p individuals they contain, from 0 to $\min(\mu, r+1)$, observing that the number of ALL_p individuals in the population cannot be reduced during phase p. As the phase starts with a population containing at least one ALL_p individual, there are at most $\min(\mu, r+1)-1$ "population levels" that the process may need to pass through before the population is saturated with ALL_p individuals; the time for this to happen can be bounded as a sum of geometrically distributed waiting times to leave each "level".

If $i < \min(\mu, r+1)$ is the number of ALL_p individuals in the population, the probability of a single iteration creating a new ALL_p individual (and hence moving to a higher level) by selecting one of the i as the ancestor and performing a one-character mutation is $p_i \geq \frac{i}{\mu}\frac{r+1-i}{r}\frac{1}{en}$. Let T be the number of iterations before the population is saturated with ALL_p individuals, and $m = \min(\mu, r+1)-1$ be the number of ALL_p individuals that need to be added to the population to achieve saturation, thus the expectation $\mathrm{E}(T)$ is at most:

$$\sum_{i=1}^{m} \frac{\mu}{i}\frac{r}{r+1-i}en = r\mu en \sum_{i=1}^{m} \frac{1}{i(r+1-i)} = O(r\mu n).$$

Applying the tail bounds from [Wit14, Theorem 2]: using $s = 13\mu^2 r^2 n^2 > \sum_{i=1}^{m} \frac{1}{p_i{}^2}$, and $h = 1/(\mu r e n) < \min p_i$, the probability that m ALL_p individuals are added to the population within $\mathrm{E}(T)+\delta$ iterations is at least $1 - e^{-\frac{\delta}{4}\cdot\min\{\delta/s, h\}}$. Setting $\delta = 13c\mu r n \log n$, where $c > 0$, and observing that $h < \delta/s$ for sufficiently large n, yields a probability of reaching the final level within $O(\mu r n \log n)$ iterations of at least

$$1 - e^{-\frac{13cr\mu n \log n}{4}\cdot\min\left\{\frac{13cr\mu n\log n}{13\mu^2 r^2 n^2}, \frac{1}{er\mu n}\right\}} \geq 1 - e^{-\frac{13c\log n}{4e}} > 1 - n^{-c}.$$

Thus, if an ALL_p individual exists in the population during phase p, the population will be saturated with ALL_p individuals in $O(\mu r n \log n)$ iterations with high probability. □

The following lemmas consider the situation for $\mu \leq r$, i. e., a population size too small to contain every individual in ALL_p. In this case, OPT_p can be removed

from the population after being discovered. Interestingly, this happens with constant probability regardless of μ.

Lemma 3.11 *If, during phase p, the population consists of μ individuals from ALL_p, and $r \geq \mu$, then after $\Omega(rn)$ iterations, the probability that OPT_p is in the population is at most a constant smaller than 1.*

Proof. OPT_p can be replaced by an ALT_p individual that was not in the population during an iteration that favors ALT_p individuals over OPT_p. The probability p_L of OPT_p being replaced by one of $r+1-\mu$ ALT_p individuals not yet in the population can then be bounded:

$$p_L = \frac{r+1-\mu}{r}\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1},$$
$$\frac{1}{rne} \leq p_L \leq \frac{1}{n}.$$

The OPT_p individual can be added to the population during any iteration favoring it over the ALT_p individuals by a single-character mutation of any individual in ALT_p. The probability p_W of OPT_p being rediscovered during an iteration favoring it can be bounded:

$$p_W = \frac{1}{r}\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1},$$
$$\frac{1}{rne} \leq p_W \leq \frac{1}{rn}.$$

Consider the probability of the OPT_p individual being in the population after three additional iterations: it is greatest when the first of these three iterations favors ALT_p individuals, and the subsequent two favor OPT_p. Let p_{W3} be the probability that OPT_p is constructed during the two iterations favoring it (being added to the population if it was not already there):

$$p_{W3} = p_W + (1-p_W)\cdot p_W,$$
$$\frac{1}{rne} \leq p_W \leq p_{W3} \leq 2p_W \leq \frac{2}{rn},$$

and p_{L3} be the probability that OPT_p is replaced by an ALT_p individual in the first iteration, and then not constructed in the two following iterations:

$$p_{L3} = p_L \cdot (1-p_{W3}),$$
$$\frac{1}{2rne} \leq \frac{1}{rne}\frac{rn-2}{rn} \leq p_{L3} \leq \frac{1}{n}\left(1-\frac{1}{rne}\right) < \frac{1}{n},$$

where the lower bound holds for $n \geq 4$.

This behavior can be modeled in a two-state Markov chain, where the states represent having or not having OPT_p in the population, with transition probabilities p_{W3} and p_{L3}; each step of this Markov chain thus corresponds to three iterations of the (μ+1) EA. Let π_H be the steady-state probability of the OPT_p individual being in the population:

$$\begin{aligned} \pi_H \cdot p_{L3} &= (1 - \pi_H) \cdot p_{W3} \\ \frac{1 - \pi_H}{\pi_H} &= \frac{p_{L3}}{p_{W3}} \geq \frac{1}{4e} \\ \pi_H &\leq \frac{1}{1 + \frac{1}{4e}} < 0.916. \end{aligned}$$

Assume, as the worst case, that the OPT_p individual is in the population when it becomes saturated with ALL_p individuals. Markov chain mixing time, $t(\varepsilon)$, can then be used to find the number of steps required to reduce the total variation distance to the steady state π_H to at most some ε (and hence the number of iterations required to reduce the probability of OPT_p being in the population to at most some constant smaller than 1), and can be upper-bounded by coupling time as in [Sud11, Corollary 4]:

$$t(\varepsilon) \leq \min \left\{ t : \max_{x,y \in \Omega} \Pr(T_{xy} > t) \leq \varepsilon \right\},$$

where $T_{xy} = \min\{t : X_t = Y_t \mid X_0 = x, Y_0 = y\}$ is the coupling time, i. e., the earliest time at which the two equivalent Markov chains X_t and Y_t, initialized in different states x, y, are in the same state.

With only two states (and symmetry), the coupling time T_{xy} is greatest when the chains begin in different states. The probability that the chains remain in different states for at least t steps is then:

$$\max_{x,y \in \Omega} \Pr(T_{xy} > t) = \left(p_{L3} p_{W3} + (1 - p_{L3})(1 - p_{W3})\right)^t.$$

To get an upper bound on $t(\varepsilon)$, an upper bound on the expression in parentheses is needed. Inserting the appropriate bounds on p_{W3} and p_{L3} yields:

$$\begin{aligned} 2p_{W3}p_{L3} + 1 - p_{L3} - p_{W3} &\leq \frac{4}{rn^2} + 1 - \frac{3}{2rne} \\ \leq 1 - \frac{1}{rne}\left(1.5 - \frac{4e}{n}\right) &< 1 - \frac{1}{rne} \text{ (for } n \geq 22), \\ t(\varepsilon) &\leq \min \left\{ t : \left(1 - \frac{1}{rne}\right)^t \leq \varepsilon \right\}. \end{aligned}$$

After, e. g., $t(0.01) < 4.61rne$ steps of the Markov chain, the probability that coupling has not occurred is at most 0.01. As coupling time is an upper bound for mixing time, and each Markov chain step corresponds to three EA iterations, this means that after at most $37.60rn$ iterations, the probability that OPT_p is in the population is at most $\pi_H + 0.01$, and so the probability that OPT_p is not in the population is at least 0.083.

Therefore, $\Omega(rn)$ iterations after the population is saturated with ALL_p individuals, the probability that OPT_p is in the population is at most a constant. □

Lemma 3.11 therefore implies that there is at least a constant probability of phase $p+1$ beginning with only ALT_p individuals in the population when $r \geq \mu$. The following lemma considers the consequences of this – as OPT_p is the only individual in both ALL_p and ALL_{p+1}, this means that phase $p+1$ begins without any ALL_{p+1} individuals in the population, leaving the EA with only OneMax-valued individuals at the beginning of phase $p+1$.

Lemma 3.12 *If at the beginning of phase $p \geq n/2+4$, the population contains no individuals from ALL_p, and only individuals with fitness values between $n-p$ and n, with at least constant probability, the population at the end of phase p will consist only of 1^n and one-character mutations of 1^n.*

Proof. Observe that ALL_p contains only individuals with fitness values $n-p$ and $n-p-1$: thus, in order to construct an ALL_p individual from the individuals in the population, a mutation must change at least one character to a specific value. If an ALL_p individual is constructed, it will be accepted, and the optimization process will resume according to Lemma 3.10. If no such individual is constructed, individuals with higher OneMax-values will continue being accepted into the population, eventually leading to the discovery of 1^n as in Lemma 3.9.

Until an ALL_p individual has been constructed, the probability of constructing one can be upper-bounded by the probability of a mutation changing a character at a specific position to a specific value, i. e., $1/(rn)$.

Let a *value-improving mutation* be a mutation that produces an individual with a strictly higher OneMax-value than the worst individual currently in the population. Note that even if a value-improving mutation occurs, the resulting individual might already be one of the $\mu - 1$ other individuals in the population, in which case the population is not modified. We will now show that the probability p_I of such a mutation occurring while the minimum OneMax-value

of any individual in the population is below $n-p+4$ is at least:

$$p_I \geq \frac{n}{2}\frac{1}{n}\frac{1}{r}\left(1-\frac{1}{n}\right)^{n-1} \geq \frac{1}{2er}.$$

Let $n-p \leq v \leq n-p+3$ be the minimum OneMax-value of an individual in the population during an iteration. If the ancestor selected during an iteration has a OneMax-value of at least $v+2$, *any* one-character mutation will produce an individual of sufficient value to be accepted; there are $n \geq n/2$ such mutations. If the selected ancestor has a value of at most $v+1$, it contains at least $n-(n-p+4) \geq n/2$ non-1 characters, and hence at least $n/2$ one-character mutations can produce an individual with a strictly higher OneMax-value than such an ancestor. Thus, regardless of the ancestor selection, there are at least $n/2$ possible one-character mutations leading to an acceptable fitness value.

No more than a constant fraction of those value-improving mutations can fail to be accepted due to already being in the population (as $\mu \leq n/(2+\varepsilon)$). This means that the probability of a value-improving mutation occurring and being accepted during a single iteration is at least $\Omega(p_I) = \Omega(1/r)$.

Consider the probability of the minimum OneMax-value of the population rising to at least $n-p+4$ without any ALL_p individual being constructed; this requires that at most 4μ value-improving mutations to be accepted (as a value-improving mutation can only introduce an individual with the same fitness value into the population at most μ times). This occurs with at least constant probability: let A be the event that an ALL_p individual is constructed, and V be the event that a value-improving mutation is accepted, then:

$$\mathrm{Prob}(A \mid A \vee V) \leq \frac{1/(rn)}{1/(rn)+\Omega(1/r)} = O(n^{-1}),$$

$$\mathrm{Prob}(4\mu\ V\text{s without } A) \geq (1-\mathrm{Prob}(A \mid A \vee V))^{4\mu} = (1-O(n^{-1}))^{O(n)} = \Omega(1).$$

Once the minimum OneMax-value of the population is raised to $n-p+4$, constructing an ALL_p individual requires at least three characters to be changed into a 0 simultaneously, which occurs with probability at most $1/(rn)^3$.

Thus, with at least constant probability, if phase p begins without an ALL_p individual in the population, no ALL_p individual is constructed within the $O(\mu rn \log n)$ iterations required to find the 1^n individual with high probability (per Lemma 3.9). Furthermore, once the 1^n individual is in the population, one-character mutations of that individual will fill the population in $O(\mu \log n)$ iterations with high probability, making construction of an ALL_p individual require a $p-2 = \Omega(n)$ character mutation. □

By combining these lemmas, it is now possible to prove Theorems 3.7 and 3.8.

Proof of Theorem 3.7. Per Lemma 3.9, the EA is able to construct the 1^n individual with high probability during phase 0, and therefore also fill the population with one-character mutations of 1^n. At the start of each phase p, the fitness value of individuals in the population is thus with high probability not below $n - p$, as any individual would have to have been accepted during the previous phase, so it was either an individual in ALL_{p-1}, which have a fitness value of at least $n - p$, or has a OneMax fitness value at least that of an individual already in the population.

Pessimistically assume that at the end of the phase $n/2 + 3$, the population does not yet consist of 1^n and one-character mutations of it, i.e. the EA has not fully reverted to optimizing OneMax during the first $n/2+3$ phases. $\Omega(n)$ of the remaining phases, per Lemma 3.11, have at least a constant probability of beginning without an ALL_p individual in the population, and with all individuals in the population having fitness values between $n - p$ and n; per Lemma 3.12, a phase beginning under such circumstances will with at least constant probability end with a population consisting of 1^n and one-character mutations thereof. Once the population is in this configuration, constructing an ALL individual (for this or any future phase) will require at least $\Omega(n)$ characters to be mutated correctly in a single iteration, which occurs with probability $(rn)^{-\Omega(n)}$.

Therefore, with high probability, the EA will find the initial 1^n optimum, but will at some point fail to track the oscillation, revert to optimizing OneMax, and require an exponential number of iterations to find the final 0^n optimum. □

Proof of Theorem 3.8. Per Lemma 3.9, the EA is able to find the 1^n individual with high probability during phase 0. For the subsequent n oscillation phases, apply Lemma 3.10: if the phase begins with OPT_{p-1} (i. e., an individual from ALL_p) in the population, then the remaining individuals from ALL_p, including OPT_p will be added to the population before phase p ends with high probability. Add up the failure probabilities in these $n+1$ phases using a union bound; with high probability, none of the phases fail, and phase n ends with $\mathrm{OPT}_n = 0^n$ in the population – so the (μ+1) EA is able to find the 0^n optimum within a polynomial number of iterations with high probability when $\mu > r$. □

3.5 ACO on larger alphabets

Kötzing and Molter [KM12] study the case where $r = 1$, i. e., the case of bit strings, and show that MMAS* with overwhelming probability (i. e., probability $1 - 2^{-\Omega(n)}$) will track the optimum of Maze with oscillation phase length $t_0 = kn^3 \log n$ in polynomial time. Their proof can be summarized as follows, wherein all references to lemmas and theorems refer to the paper:

(i) While a bit is oscillating, taking the total over three steps of a so-called OPT-OPT-ALT oscillation (where OPT is favored in the first two steps and ALT in the last step) there is a drift of the pheromone value on the 0-edge of the bit (i. e., the edge associated with OPT) towards its maximum. With overwhelming probability, the pheromone value will reach its upper border in $O(n^3 \log n)$ steps (Lemma 2 in their paper) provided the bit keeps oscillating so long. To obtain the bound on the probability, a multiplicative drift theorem is used, which, according to [KM12], "wastes a lot" since the bound on the drift obtained is additive.

(ii) Despite the fact that the pheromone value reaches its upper border within the oscillation phase of a bit, the multiplicative drift theorem does not imply that the pheromone value stays close to the border by the end of the oscillation phase. To prove that this is unlikely, a negative drift theorem is applied (Lemma 3), implying that it takes a large polynomial amount of time until the pheromone value drops below $1 - O((\log^2 n)/n)$.

(iii) The transition from "bit i oscillating" to "bit $i + 1$ oscillating" is analyzed. It is shown that a string outside ALL will be best-so-far string only temporarily for at most $O(\log n)$ steps after the transition, with high probability (Lemma 3). Drift arguments towards the 0-value are applied afterwards.

(iv) Using an $O((n \log n)/\rho)$ bound from the literature on the time from initialization until the all-ones string with is found high probability (Lemma 5) and basically applying union bounds leads to the result (Theorem 6).

It turns out that the above analysis to a very large extent carries over to larger alphabets. Basically, one can group together all edges belonging to the non-0 entries of a character and identify the sum of their pheromone values with the pheromone on the 1-edge in the binary case. The only thing to pay attention to is the new lower bound on the pheromone values, which may increase the time to reach the upper border by a factor of r.

However, we are going to present a stronger and, as we think, more elegant analysis here. In particular, we contribute a technique that supplements the

drift result from Lemma 3 in [KM12] with a statement on probability distributions (also called occupation probabilities), which makes the application of the negative drift theorem (Lemma 4 in [KM12]) unnecessary. In addition, the stronger analysis allows us to work with shorter oscillation phase lengths, as detailed below.

We now present the tool by which the statement on occupation probabilities is obtained. The following lemma is a variable drift theorem that is not concerned with the expected first hitting time of a set of target states but with the probability of being in this set at any time t (however, results on the hitting time can be easily obtained from this probability). It is a spin-off of the variable drift theorem with tail bounds from [LW13a] and goes back to a statement from Hajek's paper [Haj82] that, to the best of our knowledge, has not been applied in the running time analysis of evolutionary algorithms yet. The lemma requires a bound on the moment-generating function of a potential function g that is usually derived from the one-step drift (via the function h).

Lemma 3.13 *Let $(X_t)_{t\geq 0}$, be a stochastic process, adapted to a filtration $(\mathcal{F}_t)_{t\geq 0}$, over some state space $S \subseteq \{0\} \cup [x_{\min}, x_{\max}]$, where $x_{\min} \geq 0$. Let $a, b \in \{0\} \cup [x_{\min}, x_{\max}]$, $b > a$. Let $h\colon [x_{\min}, x_{\max}] \to \mathbb{R}^+$ be such that $1/h$ is integrable on $[x_{\min}, x_{\max}]$ and define $g\colon \{0\} \cup [x_{\min}, x_{\max}] \to \mathbb{R}^{\geq 0}$ by $g(x) := \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{x} \frac{1}{h(y)} \,\mathrm{d}y$ for $x \geq x_{\min}$ and $g(0) := 0$.*

If there exist $\lambda > 0$, $\beta < 1$ and $D > 0$ such that for all $t \geq 0$

$$\mathrm{E}(e^{-\lambda(g(X_t)-g(X_{t+1}))} \cdot \mathbb{1}\{X_t > a\} \mid \mathcal{F}_t) \leq \beta$$
$$\text{and } \mathrm{E}(e^{-\lambda(g(a)-g(X_{t+1}))} \cdot \mathbb{1}\{X_t \leq a\} \mid \mathcal{F}_t) \leq D,$$

then

$$\mathrm{Prob}(X_t \geq b \mid X_0) < \beta^t \cdot e^{\lambda(g(X_0)-g(b))} + \frac{1-\beta^t}{1-\beta} D e^{\lambda(g(a)-g(b))}$$

for all $t > 0$.

Proof.

We use ideas implicit in the proof of Inequality 2.6 of [Haj82], which uses the exponential method (a generalized Chernoff bound), and argue

$$\begin{aligned}\mathrm{Prob}(X_t \geq b \mid X_0) = \mathrm{Prob}(g(X_t) \geq g(b) \mid X_0) &= \mathrm{Prob}(e^{\lambda g(X_t)} \geq e^{\lambda g(b)} \mid X_0) \\ &\leq \mathrm{E}(e^{\lambda g(X_t)-\lambda g(b)} \mid X_0),\end{aligned}$$

where the first inequality uses that $g(x)$ is non-decreasing, the equality that $x \mapsto e^x$ is a bijection, and the last inequality is Markov's inequality. Now,

$$\begin{aligned}
\mathrm{E}(e^{\lambda g(X_t)} \mid X_0) &= \mathrm{E}\left(e^{\lambda g(X_t)} \cdot \mathbb{1}\{X_{t-1} > a\} \mid X_0\right) + \mathrm{E}\left(e^{\lambda g(X_t)} \cdot \mathbb{1}\{X_{t-1} \le a\} \mid X_0\right) \\
&= \mathrm{E}\left(e^{\lambda g(X_{t-1})} \cdot \mathrm{E}(e^{-\lambda(g(X_{t-1})-g(X_t))} \cdot \mathbb{1}\{X_{t-1} > a\} \mid \mathcal{F}_t) \mid X_0\right) \\
&\quad + \mathrm{E}\left(e^{\lambda g(a)} \cdot \mathrm{E}(e^{-\lambda(g(a)-g(X_t))} \cdot \mathbb{1}\{X_{t-1} \le a\} \mid \mathcal{F}_t) \mid X_0\right) \\
&\le \beta \mathrm{E}\left(e^{\lambda g(X_{t-1})} \mid X_0\right) + De^{\lambda g(a)}
\end{aligned}$$

by our prerequisites; we omitted the condition on X_0 for space reasons in lines 2–4. Inductively (note that this does not assume independence of the $g(X_{t-1}) - g(X_t)$), we get

$$\mathrm{E}\left(e^{\lambda g(X_t)} \mid X_0\right) \le \beta^t e^{\lambda g(X_0)} + \sum_{r=0}^{t-1} \beta^r D e^{\lambda g(a)},$$

altogether

$$\mathrm{Prob}(X_t \ge b \mid X_0) \le e^{\lambda(g(X_0)-g(b))}\beta^t + \frac{1-\beta^t}{1-\beta} D e^{\lambda(g(a)-g(b))}$$

as suggested. □

The preceding lemma will be applied to show Lemma 3.15, whose purpose is to combine Lemmas 2 and 3 from [KM12] and which shows that the pheromone value after a certain point of time will come close to its border and stay close with high probability. In fact, due to the strength of the lemma, we can work with a smaller oscillation length per character if r is not too big. More precisely, we redefine $t_0 := kr^2n^2 \ln(rn)$ in the MAZE function, for some constant k, which is less than the original $kn^3 \log n$ if $r = o(\sqrt{n})$. By contrast, [KM12] needs $\Omega(n^3 \log n)$ oscillations per bit due to the application of the multiplicative drift theorem along with negative drift.

To prepare the proof of Lemma 3.15, we set up an appropriate potential function to define a non-negative stochastic processs $(X_t)_{t\ge0}$, which is partially based on the distance of the pheromone value to its border. However, as in [KM12], it also includes an indicator random variable based on the best-so-far entry. The following lemma summarizes some crucial properties of the process. Hereinafter, we call the pheromone values of a character i, $1 \le i \le n$, *saturated* according to the best-so-far solution x^* if the edge corresponding to x_i^* has pheromone $\tau_{\max}$ and all other edges for character i have value $\tau_{\min}$.

Lemma 3.14 *Assume $\rho \leq \frac{1}{7rn}$ and $\rho = \Omega(\frac{1}{rn})$. Consider an OPT-OPT-ALT oscillation of a single character, with the best-so-far solution from ALL, and pheromone values of the non-oscillating characters saturated according to the best-so-far solution. Let τ_t be the pheromone value on the edge corresponding to the 0-entry for the oscillating character after t OPT-OPT-ALT oscillations, C_t be an indicator that the best-so-far solution at the start of the oscillation is OPT, and $X_t := 1 - 1/n - \tau_t + \frac{7}{2}\rho(1 - C_t)$ be a potential function. Then, for $X_t \geq 1/n$, the following observations hold:*

(i) $\text{Prob}(X_{t+1} > X_t \mid C_t = 1) = O(X_t)$ *and* $\text{Prob}(X_{t+1} < X_t \mid C_t = 0) = O(1 - X_t)$,

(ii) $(X_t - X_{t+1} \mid C_t = 1) = O(X_t\rho)$ *and* $(X_{t+1} - X_t \mid C_t = 0) = O((1 - X_t)\rho)$,

(iii) $\text{E}(X_t - X_{t+1} \mid C_t = 1) = \Omega(X_t\rho)$ *and* $\text{E}(X_t - X_{t+1} \mid C_t = 0) = \Omega((1 - X_t)\rho)$.

Proof. Consider the statements conditioned on $C_t = 1$ first: i. e., those that consider oscillations that begin with OPT as the best-so-far solution. In such oscillations, the first two iterations will always reinforce τ_t, as the best-so-far solution OPT has the highest possible fitness value; while the last iteration may replace OPT with a solution from ALT if one is constructed.

We note that the potential function X_t decreases unless an ALT solution is constructed in the third iteration, and upper-bound this probability using τ_t, noting that iterations reinforcing τ_t do not increase the probability that ALT solutions are constructed:

$$\text{Prob}(X_{t+1} > X_t \mid C_t = 1) < 1 - \tau_t = X_t + \tfrac{1}{n} \leq 2X_t,$$

as $X_t \geq 1/n$.

If an ALT solution is not constructed, τ_t is reinforced thrice during the oscillation, decreasing X_t by at most:

$$\begin{aligned}(X_t - X_{t+1} \mid C_t = 1) &\leq \tau_t(1 - \rho)^3 + \rho\left(1 + (1 - \rho) + (1 - \rho)^2\right) - \tau_t \\ &= (1 - \tau_t)\rho(3 - 3\rho + \rho^2) < 3(X_t + \tfrac{1}{n}) = 6X_t\rho,\end{aligned}$$

as $X_t \geq 1/n$.

Finally, for the expected decrease in X_t, bound the probability of constructing ALT in the third iteration using $p_a \leq (1 - \tau_t/\tau_{\text{sum}})\tau_{\max}{}^{n-1} \leq (1 - \tau_t/\tau_{\text{sum}})/2$, where τ_{sum} denotes the sum of pheromone values on the edges belonging to the character. The upper bound equals the probability of selecting an oscillating

edge corresponding to an ALT solution, and $n-1$ other edges with pheromone values $\tau_{\max}$ corresponding to the remaining characters in the ALT solution, using that ${\tau_{\max}}^{n-1} \leq 1/2$ for $n > 1$. Hence,

$$\begin{aligned}
\mathrm{E}(\tau_{t+1} \mid C_t = 1) &= \tau_t(1-\rho)^3 + \rho(1-\rho)^2 + \rho(1-\rho) + (1-p_a)\rho, \\
\mathrm{E}(X_t - X_{t+1} \mid C_t = 1) &= \mathrm{E}(\tau_{t+1} \mid C_t) - \tau_t - p_a \cdot \left(\tfrac{7}{2}\rho\right) \\
&= \rho\left((1-\tau_t)\left(3 - 3\rho + \rho^2\right) - \tfrac{9}{2}p_a\right) \\
&\geq \rho\left((1-\tau_t)\left(\tfrac{3}{4} - 3\rho + \rho^2\right) - \tfrac{9}{4}\tau_t/n\right) = \Omega(\rho X_t),
\end{aligned}$$

recalling that $\tau_{\mathrm{sum}} \leq 1 + 1/n$, yielding $(1 - \tau_t/\tau_{\mathrm{sum}}) \leq 1 - \tau_t + \tau_t/n$.

In the statements conditioned on $C_t = 0$, the best-so-far solution is in ALT. The first two iterations will evaporate τ_t unless an OPT solution is constructed; while the final iteration will evaporate τ_t if an ALT solution is constructed, or if OPT was not constructed during the first two.

We note that X_t increases unless an OPT solution is constructed during at least one of the first two iterations, and upper-bound this probability using τ_t, ${\tau_{\max}}^{n-1} \leq 1/2$ for $n > 1$, and a union bound:

$$\mathrm{Prob}(X_t > X_{t+1} \mid C_t = 0) < 2(\tau_t/2) = \tau_t = 1 - X_t - \tfrac{1}{n} + \tfrac{7}{2}\rho < 1 - X_t.$$

If an OPT solution is not constructed, τ_t is evaporated thrice during the oscillation, increasing X_t by at most:

$$\begin{aligned}
(X_{t+1} - X_t \mid C_t = 0) &= \tau_t - \tau_t(1-\rho)^3 = \tau_t\rho(3 - 3\rho + \rho^2) \\
&< 3(1 - X_t - \tfrac{1}{n} + \tfrac{7}{2}\rho)\rho < 3(1 - X_t)\rho.
\end{aligned}$$

Finally, to compute the expected decrease in X_t, we need to derive bounds for three probabilities: p_o, the probability that OPT is constructed at least once in the first two iterations; p_f, the probability that given OPT is constructed in the first two iterations, it was constructed in the first iteration; and p_a, the probability that given that OPT was constructed in the first two iterations, an ALT solution is constructed in the third iteration.

Using ${\tau_{\max}}^{n-1} \geq 1/e$, we upper-bound the probability that OPT is *not* constructed by $1 - p_o \leq (1 - \tau_t/(e\tau_{\mathrm{sum}}))(1 - \tau_t(1-\rho)/(e\tau_{\mathrm{sum}}))$, which provides a lower bound on p_o.

In most cases, OPT is more likely to be constructed in the first iteration compared to the second, as its corresponding pheromone value would decrease due to pheromone evaporation. However, if $\tau_t(1-\rho)$ drops below $\tau_{\min}$, and $\tau_{\mathrm{sum}} > 1$

is reduced, the second iteration may be more likely to construct OPT than the first. This effect is greatest when $\tau_t = \tau_{\min}$ and τ_{sum} is as large as possible, i. e., $1 + 1/n$, i.e. the probability that OPT is constructed during the first iteration can be bounded as $p_1 \geq \tau_{\min}/(1 + 1/n)$, while the probability that OPT is constructed in the second given that it was not constructed in the first can be bounded as $p_2 \leq \tau_{\min}/(1 + 1/n - \rho/n)$, i. e., the first iteration is at least $\frac{p_1}{p_2} = \frac{n+1-\rho}{n+1} \geq \frac{2-1/7}{2} = \frac{13}{14}$ times as likely to construct OPT as the second. The probability that OPT is then constructed in the first iteration given that it is constructed at all during the first two iterations is then $p_f \geq \frac{p_1}{p_1+(1-p_1)p_2}$, and thus $p_f > 1/3$ can be used as a very coarse lower-bound by substituting $p_2 \geq p_1 \geq \frac{13}{14}p_2$ and computing the limit as p_2 approaches 0 from above.

Pessimistically assuming that OPT is not constructed in the first iteration, so the pheromone value on the OPT edge evaporates during the first iteration, and lower-bounding the effect of the pheromone evaporation and reinforcement in the second iteration as 0, the pheromone value on the OPT edge is at least $\tau(1 - \rho)$ during the third iteration, and hence the probability of constructing an ALT solution can be upper-bounded as $p_a \leq (1 - \tau_t(1 - \rho)/\tau_{\text{sum}})\tau_{\max}{}^{n-1} \leq (1 - \tau_t(1 - \rho)/\tau_{\text{sum}})(38/100)$ for $n > 15$.

Combining the three probability bounds, we consider the expected pheromone value τ_{t+1} on the edge corresponding to OPT, and then the distance X_{t+1}, which additionally decreases by $\frac{7}{2}\rho$ if OPT is constructed in either of the first two iterations, and ALT is not constructed in the third iteration:

$$
\begin{aligned}
\mathrm{E}(\tau_{t+1} \mid C_t = 0) &\geq \tau_t(1-\rho)^3 + p_o\rho(p_f(1-\rho)^2 + (1-\rho) + (1-p_a)),\\
\mathrm{E}(X_t - X_{t+1} \mid C_t = 0) &\geq \mathrm{E}(\tau_{t+1} \mid C_t = 0) - \tau_t + p_o(1-p_a)\tfrac{7}{2}\rho\\
&\geq \rho\left(p_o\left(\tfrac{35}{6} - \tfrac{9}{2}p_a\right) - 3\tau_t\right) + \rho^2\left(3\tau_t - \tfrac{5}{3}p_o\right) - \rho^3\left(\tau_t + \tfrac{1}{3}p_o\right)\\
&= \rho\tau_t\left(\left(\tfrac{35}{6} - \tfrac{9}{2}p_a\right)b - 3 + \rho\left(3 - \tfrac{5}{3}b\right) - \rho^2\left(1 - \tfrac{1}{3}b\right)\right)\\
&= \rho\tau_t\left(\left(\tfrac{1237}{300} + \tfrac{171}{100}\tfrac{\tau_t}{\tau_{\text{sum}}}\right)b - 3 + \rho\left(3 - \tfrac{5}{3}b - \tfrac{171}{100}\tfrac{\tau_t b}{\tau_{\text{sum}}}\right) - \rho^2\left(1 - \tfrac{1}{3}b\right)\right),
\end{aligned}
$$

where $b = \frac{2-\rho}{e\tau_{\text{sum}}} - \tau_t\frac{1-\rho}{e^2{\tau_{\text{sum}}}^2} \leq p_o/\tau_t$. By bounding $\frac{2}{e} > b > \frac{2}{e} - \frac{\tau_t}{e^2} - o(1)$,

$$
\begin{aligned}
\mathrm{E}(X_t - X_{t+1} \mid C_t = 0) &\geq \rho\tau_t\left(\left(\tfrac{1237}{300} + 1.71\tau_t/\tau_{\text{sum}}\right)b - 3 - O(\rho)\right)\\
&> \rho\tau_t\left(\left(\tfrac{1237}{300} + 1.71\tau_t/\tau_{\text{sum}}\right)\left(2/e - \tau_t/e^2 - o(1)\right) - 3 - o(1)\right)\\
&= \rho\tau_t\left(\tfrac{1237}{150e} - 3 + \tau_t\left(\tfrac{3.42}{e\tau_{\text{sum}}} - \tfrac{1237}{300e^2} - \tfrac{1.71\tau_t}{e^2\tau_{\text{sum}}}\right) - o(1)\right)\\
&> \rho\tau_t\left(0.033 - o(1)\right) = \Omega(\rho(1 - X_t)),
\end{aligned}
$$

by observing that for a sufficiently large n (and hence a τ_{sum} sufficiently close to 1), the expression multiplied with the inner τ_t is positive. □

Lemma 3.15 *Let the assumptions from Lemma 3.14 hold; recall that τ_t denotes the pheromone value on the edge corresponding to the 0-entry for the oscillating character after t oscillations. For $t \geq C(rn^2 + \alpha n)$, where $C > 0$ is some sufficiently large constant, and all $\alpha \geq 0$ it holds*

$$\mathrm{Prob}\left(\tau_t \leq 1 - O\left(\frac{1}{n} + \frac{\log^2 n}{rn}\right)\right) \leq e^{-\alpha} + e^{-\Omega(\log^2 n)}.$$

Proof. We will apply Lemma 3.13, using the bounds from Lemma 3.14. Note that $0 \leq X_t \leq 1-\frac{1}{n}-\frac{1}{rn}+\frac{7}{2}\rho \leq 1-\frac{1}{2n}$ since $\frac{1}{rn} \leq \tau_t \leq 1-\frac{1}{n}$ and $\rho \leq \frac{1}{7rn}$. We will consider the cases $C_t = 0$ and $C_t = 1$ separately. First, let $C_t = 1$. If $X_t \geq 1/n$, we get from the third item of Lemma 3.14 and the choice $\rho = \Theta(1/(rn))$ a drift of the X_t-process according to $\mathrm{E}(X_t - X_{t+1} \mid X_t; X_t \geq 1/n) = \Omega(X_t\rho) \geq \frac{c_1 X_t}{rn}$ for some sufficiently small constant $c_1 > 0$. We therefore set $x_{\min} := 0$, $a := \frac{1}{n}$, $b := a + \frac{\log^2 n}{rn}$, $x_{\max} = 1 - 1/(2n)$ and $h(x) := \frac{c_1}{rn^2}$ in Lemma 3.13. The choice of $h(x)$ results in $\mathrm{E}(X_t - X_{t+1} \mid X_t; X_t > a) \geq h(X_t)$. Moreover, by definition, $g(x) := \frac{xrn^2}{c_1}$ since $x_{\min} = 0$. Hence, we have the following simple bound on the drift of the $g(X_t)$-process:

$$\begin{aligned}\mathrm{E}(g(X_t) - g(X_{t+1}) \mid X_t; X_t > a) &\geq \frac{\mathrm{E}(X_t - X_{t+1} \mid X_t; X_t > a)}{c_1/(rn^2)} \\ &\geq \frac{c_1 X_t/(rn)}{c_1/(rn^2)} = X_t n.\end{aligned}$$

Note that $X_t - X_{t+1} \leq \frac{13}{2}\rho$ since three iterations can change τ_t by at most 3ρ and $C_t \leq 1$ holds. This implies $g(X_t) - g(X_{t+1}) \leq c_2 rn^2\rho$ for some constant $c_2 > 0$. Hence, as $\rho = \Theta(1/(rn))$, we get $g(X_t) - g(X_{t+1}) \leq c_3 n$ for another constant $c_3 \geq 1$. An expansion of the exponential function will show for $\lambda := \frac{c_4}{n}$, where $c_4 > 0$ is a sufficiently small constant, that

$$\begin{aligned}\mathrm{E}(e^{-\lambda(g(X_t)-g(X_{t+1}))} \cdot \mathbb{1}\{X_t > a\} \mid X_t) &\leq \mathrm{E}(e^{-\lambda(g(X_t)-g(X_{t+1}))} \mid X_t; X_t > a) \\ &\leq 1 - \frac{\lambda}{2} \leq e^{-\lambda/2},\end{aligned}$$

which then can be used to prove the lemma.

We supply the details for the expansion now. By setting $c_4 \leq \frac{1}{c_3}$, we get $\lambda(g(X_t)-g(X_{t+1})) \leq 1$. Using $e^{-x} \leq 1-x+x^2$ for $x \leq 1$, we get for the moment-generating function (mostly omitting the conditions $X_t; X_t > a$ in expectations

for readability)

$$\begin{aligned}
&\mathrm{E}\left(e^{-\lambda(g(X_t)-g(X_{t+1}))} \mid X_t; X_t > a\right) \\
&\quad \leq 1 - \lambda \mathrm{E}\left(g(X_t) - g(X_{t+1})\right) + \lambda^2 \mathrm{E}\left(\left(g(X_t) - g(X_{t+1})\right)^2\right) \\
&\quad \leq 1 - \lambda \mathrm{E}\left(g(X_t) - g(X_{t+1})\right) + \lambda^2 \mathrm{E}\left(|g(X_t) - g(X_{t+1})|\right)(c_3 n)
\end{aligned}$$

We alrady know that $\mathrm{E}(g(X_t) - g(X_{t+1})) \geq X_t n$. We are left with an estimate for $\mathrm{E}(|\Delta|)$, where $\Delta := g(X_t) - g(X_{t+1})$. By the law of total probability (and again using $|\Delta| \leq c_3 n$, where $c_3 \geq 1$),

$$\mathrm{E}(|\Delta|) \leq \mathrm{E}(\Delta \mid \Delta > 0) + c_3 n \,\mathrm{Prob}(\Delta < 0) \leq c_5 X_t n + c_3 n c_6 X_t \leq (c_5 + c_6) c_3 n X_t,$$

where we used the first and second item from Lemma 3.14 and introduced appropriate constants $c_5, c_6 > 0$ to cover the implicit constants from O-notation and the factor $1/c_1$ from $g(x)$.

Hence,

$$\begin{aligned}
\mathrm{E}(e^{-\lambda(g(X_t)-g(X_{t+1}))}) &\leq 1 - \lambda X_t n + \lambda^2 (c_5 + c_6) X_t (c_3 n)^2 \\
&\leq 1 - \lambda X_t n + \lambda \frac{c_4}{n} (c_5 + c_6)(c_3^2 n) X_t n.
\end{aligned}$$

Choosing $c_4 \leq 1/(2c_3^2(c_5 + c_6))$, we get from last bound that

$$\mathrm{E}(e^{-\lambda(g(X_t)-g(X_{t+1}))}) \leq 1 - \lambda X_t n + \frac{\lambda}{2} X_t n = 1 - \frac{\lambda}{2} X_t n \leq 1 - \frac{\lambda}{2},$$

which completes the analysis for $C_t = 0$ if $X_t > a = 1/n$.

If $C_t = 0$, we can redo the above calculations analogously with $\mathrm{E}(g(X_t) - g(X_{t+1})) \geq (1 - X_t)n$ and replace X_t with $1 - X_t$. We note that $1 - X_t \geq 1/(2n)$, hence still $\mathrm{E}(X_t - X_{t+1} \mid X_t; X_t \geq 1/n) = \Omega(1/(rn^2))$. In the estimation of $\mathrm{E}(|\Delta|)$, the events $\Delta < 0$ and $\Delta > 0$ are swapped. The constants may take different values, but remain constants. Choosing c_4 small enough to cover both cases, we get

$$\mathrm{E}(e^{-\lambda(g(X_t)-g(X_{t+1}))} \mid X_t; X_t > a) \leq 1 - \frac{\lambda}{2} \leq e^{-\lambda/2}$$

regardless of whether $C_t = 0$ or $C_t = 1$.

We are left with the case $X_t \leq a$ in Lemma 3.13. Pessimistically assuming the maximum change $(13/2)\rho$ of the X_t-values, we can bound

$$\mathrm{E}(e^{-\lambda(g(a)-g(X_{t+1}))} \cdot \mathbb{1}\{X_t \leq a\} \mid X_t) \leq e^{c_4 c_2 \rho r n} \leq D$$

for some constant $D > 0$. Applying Lemma 3.13 with $\beta := 1 - \lambda/2 \leq e^{-\lambda/2}$,

$$\text{Prob}(X_t \geq b) \leq e^{-\frac{tc_4}{2n} + \frac{c_4}{n}\frac{rn^2}{c_1}} + \frac{D}{\lambda/2} \cdot e^{\lambda(g(a)-g(b))}$$

since $X_0 \leq 1$. Now, if $t = \frac{2rn^2}{c_1} + \frac{2\alpha n}{c_4}$ then $\text{Prob}(X_t \geq b) \leq e^{-\alpha} + \frac{2D}{\lambda} e^{\lambda(g(a)-g(b))}$. The second term is $O(n) \cdot e^{-\Omega(\log^2(n))} = e^{-\Omega(\log^2(n))}$. Setting $C := \max\{\frac{2}{c_1}, \frac{2}{c_4}\}$ and noting that $X_t \geq b$ corresponds to $\tau_t \leq 1 - O(\frac{1}{n} + \frac{\log^2 n}{rn})$, the lemma follows. □

We remark here that the statement of Lemma 3.15 can most likely be strengthened to hold already for $t \geq C(rn \ln n + \alpha n)$ by using a different $h(x)$. However, since the bottleneck will be in the analysis of the time needed for phase 0, we are content with the present statement.

The following lemma takes the role of Lemma 4 in [KM12], which analyzes the transition from character i oscillating to character $i+1$ oscillating. It applies to the case that the best-so-far solution at the end of phase i is not OPT_i but ALT_i despite the pheromone values favoring OPT_i. Then, when the new phase starts and the best-so-far is reevaluated, the fitness function will equal OneMax. However, it is very likely that MMAS* recovers quickly from this; more precisely, it will sample the solution OPT_i, which is in ALL_{i+1}, again before the pheromones have changed significantly. The lemma can be proven in the very same way as in [KM12]. In fact, the probability of setting a character being 0 in the best-so-far solution to 1, assuming saturated pheromones, will even be $1/(rn)$. This is less than the bound $1/n$ used in the proof from [KM12].

Lemma 3.16 *Let $\rho = \Theta(\frac{1}{rn})$. Assume for $i \in \{1, \dots, n\}$ that the current-best solution is $0^{i-1}1^{n-i+1}$ and that the pheromones of the first $i-1$ edges belonging to 0-entries as well as the last $n-i$ edges belonging to 1-entries all are $\tau_{max} = 1 - 1/n$. Finally, assume that the pheromone belonging to the i-th 0-entry is $1 - O((\log^2 n)/n)$. Then for all $c > 0$, MMAS* will sample 0^i1^{n-i} within $O(\log n)$ iterations with probability $1 - O(n^{-c})$.*

Finally, Theorem 5 in [KM12] states a tail bound on the optimization time of classical MMAS* on OneMax, which is used in phase 0 of Maze, where the all-ones string is the first target. This theorem carries mostly over to our enlarged search space, see Theorem 3.18 below, except for that the modified lower pheromone border introduces a factor of r at two places. The following lemma is used for the analysis.

Lemma 3.17 *Assume there is a character whose value remains fixed in all best-so-far solutions of MMAS*. Then the pheromone values of the character*

will be saturated according to the best-so-far solution after at most $\ln(rn)/\rho$ *steps.*

PROOF. The edges belonging to the r entries different from the best-so-far value each have a pheromone value of at most

$$\tau_{\max} \cdot (1-\rho)^t$$

or are already capped by their lower border after t steps, all of which by assumption reinforce the edge belonging to the best-so-far entry. Setting $t := \ln(rn)/\rho$, the expression becomes at most $1/(rn) = \tau_{\min}$. Since by Lemma 3.3 always $\tau_{\text{sum}} \geq 1$, the value for the best-so far entry must be at least $1 - r\tau_{\min} = \tau_{\max}$. □

THEOREM 3.18 *For all* $c > 0$ *with probability* $1 - O(n^{-c})$*, MMAS* for the search space* $\{0, \dots, r\}^n$ *optimizes* ONEMAX *in* $O(nr \log(rn)/\rho)$ *iterations and then saturates the all-ones string in pheromone.*

PROOF. We will use a fitness-level argument combined with an analysis of "freezing time" as commonly used in the analysis of ACO algorithms [NW10b, page 125]. The number of characters being 1 in the best-so-far solution is non-decreasing over time. By Lemma 3.17, pessimistically assuming no update of the best-so-far, the pheromone values of every 1-entry must be saturated after at most $\ln(rn)/\rho$ steps. This applies to all characters simultaneously, also to the entries different from 1. Hence, pheromone values are saturated according to the best-so-far solution after a so-called freezing time of $O(\ln(rn)/\rho)$ steps (or an improvement is found before).

Given a current ONEMAX-value of $i \leq n-1$, the probability of finding an improvement in such a situation is at least

$$\binom{n-i}{1} {\tau_{\max}}^i \tau_{\min} \geq \frac{n-i}{ern} =: p_i$$

Hence, in the notation of Theorem 2 in [Wit14], we have $n+1$ fitness levels $A_0, \dots, A_n$ corresponding to the ONEMAX-values and corresponding probabilities of improvement (assuming saturated pheromones) given by p_i for $0 \leq i \leq n-1$. Now, $s = \sum_{i=0}^{n-1} 1/p_i^2 = O(r^2n^2)$, $h = \Omega(1/(rn))$ and $\sum_{i=0}^{n-1} 1/p_i = O(rn \log n)$. Hence, by setting $\delta = Ccrn \ln n$ for some constant $C > 0$ in the theorem, the time to reach the last level (without the freezing time) is $O(rn \log n)$ with probability $1 - n^{-c}$

On at most n levels, the pheromone values need to be saturated according to the best-so-far solution in order for the fitness-level argument to apply; one more saturation may be required for the final all-ones string. This accounts for a deterministic term of $O(n \ln(rn)/\rho)$ that has to be added to the time bound given by the fitness-level argument. Taking the two bounds together, the lemma follows. □

We note that the proof of Theorem 3.18 is the only place in our analysis where the strict $>$-selection for the update of the best-so-far solution by MMAS* is used. Otherwise, the arguments would hold for the algorithm using non-strict $\geq$-selection, which is often simply called MMAS in the literature.

Putting everything together, we obtain the following theorem, taking the role of Theorem 6 in [KM12]. Recall that t_0, the length of the so-called oscillation phase, is the number of iterations that a character is oscillating as OPT-OPT-ALT; however, the very first phase of length t_0, called phase 0, has objective function OneMax.

Theorem 3.19 *Given any $r > 0$, choose $\rho \leq \frac{1}{7rn}$ and $\rho = \Omega(\frac{1}{rn})$. We say that MMAS* tracks the optimum of the* Maze *for the $(r+1)$-character alphabet if the best-so-far solution at the end of every oscillation phase has Hamming distance at most 1 to the optimum. Then for all $c > 0$ there is a constant k such that choosing $t_0 \geq kr^2n^2 \ln(rn)$ makes MMAS* track the optimum with probability $1 - O(n^{-c})$.*

Proof. We follow the argumentation in [KM12]. Let k' be the largest implicit constant in the bounds of Lemma 3.15, Lemma 3.16 and Theorem 3.18 for obtaining a failure probability of $O(n^{-c-1})$. Let $k = 3k'$. In the following, we assume that the events proved to hold with probability $1 - O(n^{-c-1})$ actually happen and call it a failure otherwise.

Since Maze equals OneMax within phase 0, we know from Theorem 3.18 that MMAS* finds the all-ones string and saturates all pheromones in the phase with probability $1 - O(n^{-c-1})$. The conditions of Lemma 3.15 hold for the start of phase 1, where the first character is oscillating. Setting $\alpha = n$, we get that the pheromone value on the corresponding 0-edge will be at least $1 - O(\log^2 n/(rn))$ by the end of the phase with probability $1 - o(n^{-c-1})$. The best-so-far solution by the end of phase 1 is guaranteed to be in ALL_1; however, it might be ALT_1, which does not belong to ALL_2.

We now analyze the transition to phase 2. According to Lemma 3.16, a solution from ALL_2 will be created in the first third of the phase with probability

$1 - O(n^{-c-1})$. Within at most $O(rn \ln(rn))$ steps of the second third of the phase, MMAS* by Lemma 3.17 saturates all pheromones except for the oscillating character corresponding to the solutions from ALL_2. Now the conditions of Lemma 3.15 are satisfied. By the end of the final third of the phase, the pheromone value on the 0-edge for character 2 will be $1 - O(\log^2 n/(rn))$ with probability $1 - o(n^{-c-1})$. The subsequent phases are analyzed in the very same way as the second one.

By a union bound, the failure probability over all $n + 1$ phases is $O(n^{-c})$. □

3.6 Conclusions

We have revisited the analysis of evolutionary algorithms and ACO on the dynamic fitness function Maze [KM12]. First, we have shown that a (2+1) EA with a simple population diversity mechanism is able to track the optimum, which the (1+1) EA cannot. Subsequently, we have generalized this to a hierarchy result on strings over finite alphabets, where for given μ, there exists a Maze variant for which a population size of at least μ in a (μ+1) EA with genotype diversity is sufficient to track the optimum, whereas population size $\mu - 1$ causes the EA lose track of the optimum. Surprisingly, it turns out that a generalization of MMAS* to the larger state space does not require a population and is sufficient on all functions from the hierarchy. Along the way, we have introduced a variable drift theorem dealing with occupation probabilities, which allows for a more precise and simpler analysis of the pheromone values in MMAS* compared to [KM12]. As a subject for future research, it is interesting to study the benefits and the limitations of ACO on other dynamic problems.

Chapter 4

A Runtime Analysis of Parallel Evolutionary Algorithms in Dynamic Optimization

Andrei Lissovoi Carsten Witt

DTU Compute, Technical University of Denmark

A simple island model with λ islands and migration occurring after every τ iterations is studied on the dynamic fitness function Maze. This model is equivalent to a (1+λ) EA if $\tau = 1$, i. e., migration occurs during every iteration. It is proved that even for an increased offspring population size up to $\lambda = O(n^{1-\varepsilon})$, the (1+$\lambda$) EA is still not able to track the optimum of Maze. If the migration interval is increased, the algorithm is able to track the optimum even for logarithmic λ. The relationship of τ, λ, and the ability of the island model to track the optimum is then investigated more closely. Finally, experiments are performed to supplement the asymptotic results, and investigate the impact of the migration topology.

4.1 Introduction

Evolutionary algorithms (EAs) are a class of nature-inspired algorithms which can be applied to solve a wide variety of optimization problems. Rigorous runtime analysis of nature-inspired algorithms, building on mathematical methods from the analysis of classical algorithms, has advanced considerably in recent

years [AD11, NW10b]. While most of these studies focus on so-called static optimization problems, whose set of optima is fixed, there has been increasing interest in the analysis of evolutionary and other nature-inspired algorithms on so-called dynamic problems. Many real-world optimization problems are subject to dynamics in that the optimal solution may change as the problem conditions change over time, and the algorithms therefore need to be able to not only find or approximate the optimum at some point of time, but also to *track* the optimal solution over time as the problem changes.

Application of EAs to dynamic optimization problems is the subject of study in the Evolutionary Dynamic Optimization field, which in recent years has attracted much activity. Many applications of evolutionary algorithms on dynamic problems are considered in the literature [NYB12, ANS13], and there are already a number of runtime analyses of evolutionary algorithms for dynamic problems [Dro03, JS05, RLY09, JZ14, KLW15, DJL15].

Despite the increasing interest in the area, it has not been well understood what mechanisms allow EAs or related nature-inspired algorithms to efficiently track the optimum of a dynamic problem. In [KM12], Kötzing and Molter introduced a dynamic pseudo-boolean function called Maze that separates simple evolutionary algorithms and ant colony optimization. More precisely, the paper shows that while a Max-Min Ant System (MMAS) is able to track the changes occurring in the Maze fitness function and finds the optimum within polynomial time, a (1+1) EA loses track of the optimum and requires with high probability an exponential amount of time to find the optimum. Very recently, we have built upon this study [LW15a] and shown that introducing a parent population makes the evolutionary algorithm efficient in tracking the Maze again, presenting a generalization of the Maze function that allows for a hierarchy result, where a family of functions parameterized by μ is defined such that a parent population size of μ allows efficient tracking of the optimum, while population size $\mu - 1$ makes the algorithm lose track of the optimum with high probability. Additionally, a simple MMAS is proved insensitive with respect to the parameter as it is able to track the optimum without any modifications for a wide range of values for μ.

In this work, we consider a different mechanism and analyze its benefit in tracking the optimum of the Maze benchmark function. We focus on parallel nature-inspired algorithms, which are heavily employed in practice due the rapid development of parallel computer architectures. The survey by Alba, Luque and Nesmachnow [ALN13] describes important applications and theoretical studies in this area. In particular, it refers to experiments with parallel nature-inspired algorithms in dynamic optimization, including a study of a parallel swarm algorithm for dynamic vehicle routing problems [KSA+11]. It is therefore interesting to determine the theoretical properties of parallel nature-inspired algorithms

that allow them to track the optimum of a dynamic problem. Both the number of so-called islands (independent subpopulations) and the communication between them seem influential. From a more general perspective, [NYB12] emphasizes the usefulness of memory and diversity-maintaining operators in EAs for dynamic optimization.

Our contribution is represented by a runtime analysis of a parallel EA for the dynamic Maze problem. We define a simple parallel EA using an island model with communication occurring within regular intervals, the so-called migration intervals, in the style of Lässig and Sudholt [LS14], who pioneered the runtime analysis of parallel EAs. The impact of two parameters is studied, namely the number of islands λ and length of the migration intervals τ. In the extreme case that $\tau = 1$, i. e., migration occurs in every generation, the model boils down to a $(1+\lambda)$ EA. It is shown that offspring population sizes, i. e., number of islands, of up to $\lambda = O(n^{1-\varepsilon})$, where n is the problem size and ε an arbitrarily small positive constant, do not allow this algorithm to track the Maze efficiently. In contrast, if τ is chosen appropriately, ensuring that migration does not occur too frequently, already $\lambda = \Omega(\log n)$ islands allow efficient tracking of the optimum of the Maze. Moreover, more general conditions on the choice of τ are worked out, resulting in either efficient tracking or losing track of the optimum. To the best of our knowledge, our contribution represents the first runtime analysis of parallel EAs in dynamic optimization. The results indicate that carefully choosing the migration policy and thereby the communication strategy of an island model can be more advantageous than a mere increase of offspring population size.

This paper is structured as follows. In Section 4.2, we introduce the parallel EA and the dynamic optimization problem Maze studied throughout the paper, and define important tools used in the analysis. Section 4.3 is concerned with the negative result for the parallel EA with $\tau = 1$, i. e., the $(1+\lambda)$ EA. The case of appropriately chosen τ, leading to efficient tracking with a small number of islands, is analyzed in Section 4.4. Moreover, the section elaborates on the impact of the choice of τ on efficient tracking in a more general sense. Section 4.5 validates the theoretical results of previous sections by presenting experimental results. We finish with some conclusions.

4.2 Preliminaries

The Maze fitness function, proposed in [KM12], and defined formally below, consists of $n+1$ phases of $t_0 = kn^3 \log n$ iterations each. This phase length was used in [KM12] to allow the Max-Min Ant System (MMAS) algorithm time to

adjust the pheromone values during each phase, and is preserved here for mostly historical reasons. For convenience, we will assume that k is chosen such that t_0 is a multiple of 3.

During the first phase of the MAZE, which we will for convenience refer to as phase 0, the function is equivalent to ONEMAX: the fitness of an n-bit string is equal to the number of 1-bits in the string. In the next n phases, higher fitness values $n+1$ and $n+2$ are assigned to two individuals determined by the phase in an oscillating pattern: every two iterations out of three, the OPT_p individual is assigned the fitness value $n+2$ while the ALT_p individual is assigned the fitness value $n+1$, and during every third iteration, these assignments are reversed; all other individuals retain their ONEMAX values. Past the last oscillating phase ("phase n"), MAZE behaves in a fashion similar to TRAP: all individuals except 0^n are assigned ONEMAX values, while 0^n is the global optimum, being assigned the highest fitness value. [KM12] proves that a (1+1) EA loses track of the optimum of this MAZE function, reverting to optimizing ONEMAX, and is therefore not able to construct the final $\mathrm{OPT}_n = 0^n$ optimum in a polynomial number of iterations.

$$
\begin{aligned}
\textsc{Maze}(x,t) &= \begin{cases} n+2 & \text{if } t > (n+1)\cdot t_0 \ \wedge\ x = 0^n \\ n+2 & \text{if } t > t_0 \ \wedge\ x = \mathrm{OPT}(t) \\ n+1 & \text{if } t > t_0 \ \wedge\ x = \mathrm{ALT}(t) \\ \textsc{OneMax}(x) & \text{otherwise} \end{cases} \\
\mathrm{OPT}(t) &= \begin{cases} \mathrm{OPT}_{\lfloor t/t_0 \rfloor} & \text{if } t \neq 0 \mod 3 \\ \mathrm{ALT}_{\lfloor t/t_0 \rfloor} & \text{otherwise} \end{cases} \\
\mathrm{ALT}(t) &= \begin{cases} \mathrm{ALT}_{\lfloor t/t_0 \rfloor} & \text{if } t \neq 0 \mod 3 \\ \mathrm{OPT}_{\lfloor t/t_0 \rfloor} & \text{otherwise} \end{cases} \\
\mathrm{OPT}_p &= 0^p 1^{n-p} \qquad \text{for } p \leq n \\
\mathrm{ALT}_p &= 0^{p-1} 1^{n-p+1} \qquad \text{for } p \leq n \\
\mathrm{ALL}_p &= \{\mathrm{ALT}_p, \mathrm{OPT}_p\}
\end{aligned}
$$

We note that the clock t is considered external to the MAZE function, allowing the fitness value of multiple solutions to be evaluated in each clock value t. For (1+λ) EA, and the λ island model, this corresponds to having hardware available to evaluate many solutions in parallel, or having the problem changes occur at fixed intervals regardless of the number of parallel evaluations.

We consider the behavior of the (1+λ) EA, shown as Algorithm 4.1, and that of a simple island model running λ (1+1) EAs in parallel with various choices for the

frequency of migration, shown as Algorithm 4.2, on the MAZE function. Both algorithms use the standard bit mutation operator, formalized in Definition 4.1.

For our purposes, Algorithm 4.2, if migration is set to occur in every iteration, behaves equivalently to the (1+λ) EA on MAZE: the algorithms differ only in their initialization of x^* and the first offspring population, which is less significant as both are able to find the ONEMAX optimum within the initial phase with high probability. The order of mutation and migration in Algorithm 4.2 has been selected to allow for this similarity, essentially allowing an ALT individual constructed during an ALT-favoring iteration of the MAZE to migrate to all islands, similar to how it would be assigned to x^* in a (1+λ) EA.

Algorithm 4.1 (1+λ) EA

Select x^* uniformly at random from $\{0,1\}^n$.
for $t \leftarrow 1, 2, \ldots$ **do**
  for $i \leftarrow 1, \ldots, \lambda$ **do**
    $x_i \leftarrow \text{mutate}(x^*)$
  $x_m \leftarrow \arg\max_{x_i} f(x_i, t)$
  if $f(x_m, t) \geq f(x^*, t)$ **then**
    $x^* \leftarrow x_m$

Algorithm 4.2 Island model with λ islands running (1+1) EAs in parallel.

for $i \leftarrow 1, \ldots, \lambda$ **do**
  Select x_i^* uniformly at random from $\{0,1\}^n$.
for $t \leftarrow 1, 2, \ldots$ **do**
  for $i \leftarrow 1, \ldots, \lambda$ **in parallel do**
    $x_i \leftarrow \text{mutate}(x_i^*)$
    if $f(x_i, t) \geq f(x_i^*, t)$ **then**
      $x_i^* \leftarrow x_i$
  if migration occurs during iteration t **then**
    Let $m = \arg\max_i f(x_i^*, t)$
    $x_i^* \leftarrow x_m$ for all $i \in \{1, \ldots, \lambda\}$

DEFINITION 4.1 (STANDARD BIT MUTATION) The mutation operator mutate(x) creates an image $y \in \{0,1\}^n$ from $x \in \{0,1\}^n$ by independently replacing each bit x_i of x ($1 \leq i \leq n$) with $1 - x_i$ with probability $1/n$.

In the analysis of the $(1 + \lambda)$ EA and the simple island model, we make use of Markov chain mixing times to bound the probability that the algorithm is in a particular state (i. e. has a particular best-so-far individual) after a certain number of iterations. This has been applied to ant colony optimization in

[Sud11] and to analyze the $(\mu+1)$ EA on MAZE in [LW15a]; for completeness, we repeat the definitions of mixing and coupling times below.

DEFINITION 4.2 (MIXING TIME) Consider an ergodic Markov chain over a state space Ω with stationary distribution π. Let $p_x^{(t)}$ denote the distribution of the Markov chain t iterations after starting in state x, and let

$$t(\varepsilon) := \max_{x\in\Omega} \min \left\{ t : \frac{1}{2} \sum_{y\in\Omega} \left| p_x^{(t)}(y) - \pi(y) \right| \leq \varepsilon \right\}.$$

The mixing time t_{mix} of the Markov chain is then defined as $t_{\text{mix}} = t(1/(2e))$.

DEFINITION 4.3 (COUPLING TIME) Consider a pair process $(X^{(t)}, Y^{(t)})$, where both $X^{(t)}$ and $Y^{(t)}$, viewed in isolation, are instances of the same Markov chain. Coupling time T_{xy} is the random time until the two processes, initialized in different states x and y, are in the same state for the first time:

$$T_{xy} = \min\{t : X^{(t)} = Y^{(t)} \mid X^{(0)} = x,\, Y^{(0)} = y\}.$$

The worst-case coupling time is an upper bound on the mixing time:

$$t(\varepsilon) \leq \min \left\{ t : \max_{x,y\in\Omega} P(T_{xy} > t) \leq \varepsilon \right\}.$$

Additionally, the following drift theorem, adapted from [DG13, DJW12], is useful when considering longer migration intervals.

THEOREM 4.4 (MULTIPLICATIVE DRIFT) *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum $s_{\min} > 0$. Let $\{X^{(t)}\}_{t\geq 0}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random first point in time $t \geq 0$ for which $X^{(t)} = 0$.*

Suppose there exists a $\delta > 0$ such that

$$E(X^{(t)} - X^{(t+1)} \mid X^{(t)} = s) \geq \delta s$$

for all $s \in S$ with $P(X^{(t)} = s) > 0$. Then for all $s_0 \in S$ with $P(X^{(0)} = s_0) > 0$,

$$E(T \mid X^{(0)} = s_0) \leq \frac{\ln(s_0/s_{\min}) + 1}{\delta}$$

Moreover, it holds that $P(T > (\ln(s_0/s_{\min}) + r)/\delta) \leq e^{-r}$ for any $r > 0$.

Several lemmas throughout this paper state that "a specific event occurs with high probability." Definition 4.5 provides a more formal definition of this concept.

Definition 4.5 An event E is said to occur *with high probability* if, for every constant $c > 0$, $P(E) = 1 - O(n^{-c})$.

In general, we say that an algorithm is able to track the optimum of the Maze when it is able to construct the OPT_n individual in polynomial time (with high probability). Typically, this would correspond to maintaining at most a constant Hamming distance to the OPT_p intermediate optima during the oscillating phases.

4.3 (1+λ) EA on Maze

In this section, we analyze the behavior of the (1+λ) EA on Maze. As long as λ is not too large (trivializing the problem by exploring the entire neighborhood of hamming distance 1 during every iteration), the EA is not able to track the optimum of the Maze, and reverts to optimizing OneMax. This is formalized in the following theorem, whose proof is inspired by the strategy taken in [LW15a].

Theorem 4.6 *(1+λ) EA with $\lambda \in O(n^{1-\varepsilon})$, for any constant $\varepsilon > 0$, will with high probability lose track of the optimum of* Maze*, i. e. with high probability it will require an exponential number of iterations to construct OPT_n.*

We first note that the EA is able to find $\mathrm{OPT}_0 = 1^n$ successfully, but then has at least a constant probability of ending each of the following n oscillating phases with $x^* \neq \mathrm{OPT}_p$, and at least a constant probability of ending the next phase after at least a constant fraction of such phases with $x^* = 1^n$; if this occurs sufficiently late in the optimization process, constructing an ALL_p individual from 1^n requires a large simultaneous mutation, the waiting time for which is exponential with respect to n.

Lemma 4.7 *$(1 + \lambda)$ EA constructs $OPT_0 = 1^n$ during the initial phase with high probability.*

Proof. The initial phase consists of optimizing OneMax, which a (1+1) EA accomplishes in $O(n \log n)$ iterations with high probability. Increasing the size of the offspring population λ can only decrease the time required; and we note that t_0 is asymptotically greater than $O(n \log n)$.

To prove that the (1+1) EA finds 1^n on OneMax with high probability in $O(n \log n)$ time, apply Theorem 4.4, defining $X^{(t)}$ as the number of zero-bits in the bit string x^* during iteration t, and pessimistically assuming $X^{(0)} \leq n$, which yields:

$$E(X^{(t)} - X^{(t+1)} \mid X^{(t)} = x) \geq \frac{x(1-1/n)^{n-1}}{n} \geq \frac{1}{ne}x$$
$$E(T \mid X^{(0)} \leq n) \leq \frac{\ln(n)+1}{1/(ne)} = ne\ln(n) + ne$$

Applying the tail-bound with $r = c_1 \ln n$ yields an upper bound on the probability of exceeding the expected number of iterations to reach $X^t = 0$ (i. e. to find the 1^n optimum) by more than $c_2 n \ln n$ additional iterations, where both c_1, c_2 are constants:

$$e^{-c_1 \ln n} = n^{-c_1}$$

i. e. 1^n is constructed by the (1+1) EA in $ne\ln(n) + ne + c_2 n \ln n = O(n \log n)$ iterations with probability $1 - n^{-c_1}$. As previously stated, increasing the offspring population size λ can only reduce the number of iterations required by the (1+λ) EA. □

Lemma 4.8 *Given that $x^* \in ALL_p$ at least $c\lambda/n$ iterations before the end of phase p, the probability that phase p ends with $x^* = OPT_p$ is in $\Theta(1)$, and the probability that phase p ends with $x^* = ALT_p$ is also in $\Theta(1)$.*

Proof. Once an ALL_p individual is constructed, only OPT_p and ALT_p can be accepted as the best-so-far individuals during the remainder of the phase, with single-bit mutations at specific iterations of the oscillation allowing the EA to switch between the two.

Let p_1 be the probability of a specific single-bit mutation occurring in a single iteration of $(1 + \lambda)$ EA, i. e. the probability that OPT_p is constructed from ALT_p or vice versa:

$$p_1 = 1 - \left(1 - (1-1/n)^{n-1} \cdot 1/n\right)^{\lambda}$$
$$1 - (1-1/(en))^{\lambda} \leq p_1 \leq 1 - (1-1/(2n))^{\lambda}$$
$$\lambda/(2en) \leq 1 - e^{-\lambda/(en)} \leq p_1 \leq \lambda/(2n)$$

assuming $n \geq 2$ for the upper bound, and using $(1-x) \leq e^{-x}$, and $e^{-x} \leq 1 - x/2$ (for $x \leq 1$) for the lower bound.

Consider the phase as a series of oscillations, each oscillation consisting of three iterations. During the first two iterations of an oscillation, OPT_n has a higher fitness value than ALT_n (and vice versa during the third and final iteration). We then consider the probabilities p_O and p_A of the best-so-far individual switching from OPT_p to ALT_p, and from ALT_p to OPT_p respectively during a single oscillation:

$$\begin{aligned} p_O &= (p_1 + (1-p_1)p_1)(1-p_1) \\ &= 2p_1 - 3{p_1}^2 + {p_1}^3 = p_1(2 - 3p_1 + {p_1}^2) \\ p_A &= p_1 \end{aligned}$$

The identity of the best-so-far individual x^* of the $(1+\lambda)$ EA, observed at the end of each OPT-OPT-ALT oscillation can be modeled using a two-state Markov chain, with one state corresponding to $x^* = \mathrm{OPT}_p$, the other to $x^* = \mathrm{ALT}_p$, and transition probabilities between the states as above.

Let π_O and $\pi_A = 1 - \pi_O$ be the steady-state probabilities of $x^* = \mathrm{OPT}_p$ and $x^* = \mathrm{ALT}_p$ respectively; per the definition of a steady-state probability:

$$\begin{aligned} \pi_O p_A &= \pi_A p_O \\ \pi_O p_1 &= (1-\pi_O)p_1(2 - 3p_1 + {p_1}^2) \\ \pi_O &= \frac{2 - 3p_1 + {p_1}^2}{3 - 3p_1 + {p_1}^2} \end{aligned}$$

i. e. π_O approaches a constant; we note that $\pi_O \leq 2/3$, and as $\lambda \in o(n)$ and hence $p_1 \leq \lambda/(2n) \leq 0.5$, $\pi_O \geq 3/7$.

Over time, the probability of OPT_p being the best-so-far individual at the end of an oscillation will approach the steady-state probability π_O. Markov chain mixing time can be used to bound the number of oscillations required until this probability is within an ε of π_O. Markov chain mixing time can be upper-bounded by coupling time, i. e. the maximum number of steps T_{xy} until the probability that two independent instances of the chain initialized in different

states are in the same state becomes sufficiently small:

$$\begin{aligned} t(\varepsilon) &\leq \min\left\{t : \max_{x,y\in\Omega} P(T_{xy} > t) \leq \varepsilon\right\} \\ \max P(T_{xy} > t) &= (p_A p_O + (1-p_A)(1-p_O))^t \\ &= (1 + 2p_A p_O - p_A - p_O)^t \\ &= (1 - p_1(3 - 7p_1 + 7{p_1}^2 - 2{p_1}^3))^t \\ &\leq \left(1 - \frac{\lambda}{2en}\left(3 - \frac{7\lambda}{2n} + \frac{7\lambda^2}{4e^2n^2} - \frac{2\lambda^3}{8n^3}\right)\right)^t \\ &< \left(1 - \frac{\lambda}{2en}\right)^t \end{aligned}$$

by recalling that $\lambda = o(n)$, and observing that the expression in the inner parentheses is greater than 1 when $\lambda/n \leq 0.5$. Then, an upper bound on the coupling time is:

$$t(\varepsilon) \leq \min\left\{t : (1 - \lambda/(2en))^t \leq \varepsilon\right\}$$

After at most $t(0.01) < 9.22en/\lambda$ steps of the Markov chain, i. e. at most $76n/\lambda$ iterations of the $(1+\lambda)$ EA, the probability that $x^* = \mathrm{OPT}_p$ is therefore within $[\pi_O - 0.01, \pi_O + 0.01]$, and, as $3/7 \leq \pi_O \leq 2/3$, in $\Theta(1)$. Similarly, the probability that $x^* = \mathrm{ALT}_p$ is within $[1 - \pi_O - 0.01, 1 - \pi_O + 0.01]$, and therefore in $\Theta(1)$. □

Lemma 4.9 *If a phase $p > n/2 + 3$ begins with $x^* \notin ALL_p$ satisfying $f(x^*) > n - p + 1$, the $(1+\lambda)$ EA with offspring population size $\lambda = O(n^{1-\varepsilon})$, for any constant $\varepsilon > 0$, ends the phase with $x^* = 1^n$ with at least constant probability.*

Proof. At the start of phase p, x^* contains strictly more 1-bits than any individual in ALL_p, and the Hamming distance between x^* and the closest ALL_p individual is at least 1. Let $p_R \leq p_1$ be the probability that an ALL_p individual is constructed during an iteration.

We want to consider the probability that the number of 1-bits in x^* exceeds that in any ALL_p individual by at least 3 before an ALL_p individual is constructed. An individual with a greater OneMax value is constructed via a single-bit mutation with probability at least p_L:

$$\begin{aligned} p_L &\geq 1 - (1 - n/2 \cdot (1-1/n)^{n-1}/n)^\lambda \\ &\geq 1 - 0.75^\lambda \geq 1/4 \end{aligned}$$

as there are at least $n/2$ 0-bits that can be flipped to increase OneMax value. We note that after at most 2 OneMax-improvements, constructing the closest ALL_p individual requires at least 3 1-bits to be flipped simultaneously.

Consider the probability that two OneMax improvements occur before an ALL_p individual is constructed. Let V be the event that a OneMax-improving single-bit mutation occurs, and A be the event that an ALL_p individual is constructed:

$$\begin{aligned} P(A \mid A \vee V) &\leq \frac{p_R}{p_R + p_L} \leq \frac{\lambda}{2n\left(\frac{\lambda}{2en} + 1/4\right)} \\ &= \frac{2e\lambda}{en + 2\lambda} \in O(\lambda/n) \\ P(2\ V\text{s without } A) &\geq (1 - P(A \mid A \vee V))^2 \in \Omega(1) \end{aligned}$$

Once this occurs, constructing an ALL_p individual requires at least 3 specific bits to mutate simultaneously, which with high probability does not happen within the time required to find 1^n per Lemma 4.7. Thus, the $(1 + \lambda)$ EA has at least a constant probability of ending the phase with $x^* = 1^n$. □

These lemmas can then be combined to prove Theorem 4.6.

Proof of Theorem 4.6. With high probability, $\mathrm{OPT}_0 = 1^n$ is found during phase 0 per Lemma 4.7. At the start of each subsequent phase p, $f(x^*) > n - p$, as only individuals in the ALL sets of the preceding phases can be accepted while decreasing the number of 1-bits in x^*, and the minimum OneMax value of any individual in sets $\mathrm{ALL}_0, \ldots, \mathrm{ALL}_{p-1}$ is $n - p + 1$. Furthermore, if $x^* \notin \mathrm{ALL}_p$, $f(x^*) > n - p + 1$, as this excludes $x^* = \mathrm{OPT}_{p-1}$, which had the lowest fitness value of all individuals in the union of the previous ALL sets.

If $x^* \neq 1^n$ at the start of phase $p \geq n/2 + 3$, the phase has at least a constant probability of ending with $x^* \neq \mathrm{OPT}_p$ per Lemma 4.8, and hence $x^* \notin \mathrm{ALL}_{p+1}$.

If phase $p + 1$ begins with $x^* \notin \mathrm{ALL}_{p+1}$, it has at least a constant probability of ending with $x^* = 1^n$ per Lemma 4.9.

Thus, at least a constant fraction of $\Omega(n)$ phases beyond $n/2 + 3$ have at least a constant probability of ending with $x^* = 1^n$; i. e. with high probability, at least one of those phases will end with $x^* = 1^n$. Constructing an ALL_p individual from 1^n in future phases requires at least $\Omega(n)$ bits to be flipped simultaneously, which with high probability does not occur in polynomial time. □

We note that the proof of Theorem 4.6 relies on $\lambda = o(n)$ primarily in the bounds on p_1 in Lemma 4.8, although, if λ is increased a little further to $\Omega(n \log n)$, the behavior described by Lemma 4.9 would also no longer occur, allowing (1+λ) EA to recover from any phase which ends with an ALT_p with high probability.

Lemma 4.10 *(1+λ) EA with offspring population size $\lambda \geq c_1 n \log n$, where $c_1 > 0$ is a sufficiently large constant, is able to track the optimum of the* Maze *function, constructing OPT_n at the end of the* Maze *with high probability.*

Proof. We pessimistically assume that each of the n oscillating Maze phases ends with $x_* = \mathrm{ALT}_p$. Consider the probability p_r of constructing an ALL_{p+1} individual in the first iteration of the next phase, which can be lower-bounded by the probability that a specific one-bit mutation occurs (flipping the previously oscillating bit to a 0),

$$\begin{aligned} p_r &= 1 - (1 - (1-1/n)^{n-1}/n)^{c_1 n \log n} \\ &\geq 1 - (1 - 1/(ne))^{c_1 n \log n} \\ &\geq 1 - e^{-c_2 \log n} = 1 - n^{-c_2} \end{aligned}$$

We can then use a union bound to lower-bound the probability p_R that all n phases construct an ALL_p individual in their first iteration:

$$p_R \geq {p_r}^n \geq 1 - n^{1-c_2}$$

By picking a sufficiently large constant c_1 in $\lambda = c_1 n \log n$, we can ensure that the (1+λ) EA constructs an ALL_p individual in the first iteration following every phase transition with high probability. □

4.4 A simple island model

Splitting the λ offspring onto λ islands, which only compare the fitness values of their x^* individuals periodically (for instance, every τ iterations, where $\tau > 0$ is the migration interval), allows the resulting island model to track Maze even with a modest λ. In this section, we consider the effect of various migration schedules on how the island model is able to track the Maze.

To begin with, consider an island model where migration occurs on the first iteration of every phase, i. e. every $\tau = t_0$ iterations of the Maze. This ensures

that an ALL_p individual migrates to all islands if any of the islands end the preceding phase with $x^* = \mathrm{OPT}_{p-1}$.

Theorem 4.11 *An island model with $\lambda = c \log n$ islands, where c is a sufficiently large constant, each island running a (1+1) EA, and migration occurring during the first iteration of every phase (i. e. with migration interval $\tau = t_0$) is able to find the OPT_n optimum of the* Maze *with phase length $t_0 = kn^3 \log n$ in polynomial time with high probability.*

Proof. We note that individually, the islands behave exactly like a (1+1) EA on Maze, and the effects of migration are limited to selecting the best-so-far individual at the start of each phase, and propagating it to all islands. Thus, as long as any island ends phase p with $x^* = \mathrm{OPT}_p$, all islands will receive an ALL_{p+1} individual during the first iteration of phase $p + 1$.

The initial OneMax optimum, OPT_0, is found during phase 0 on each island with high probability. Lemma 4.8, applied with $\lambda = 1$, states that the probability that an island that begins phase p with $x^* \in \mathrm{ALL}_p$ ends the phase with $x^* = \mathrm{OPT}_p$ with at least constant probability; let $p_s = \Omega(1)$ be a lower bound on this probability, and p_f an upper bound on the probability that all λ islands end the phase with $x^* \neq \mathrm{OPT}_p$. As long as the latter event does not occur, all islands will receive an ALL_{p+1} individual at the start of the next phase, allowing the argument to be repeated inductively. A union bound can then be used to upper-bound the probability of failing in any of the n phases:

$$p_f \leq (1 - p_s)^\lambda$$
$$np_f \leq n(1 - p_s)^\lambda \leq nc_1^{c_2 \log n} \leq n^{1 + c_2 \log c_1}$$

noting that for any constant $c > 0$, choosing $c_2 \geq -(1 + c)/\log c_1$ (recall that $c_1 \leq p_f < 1$, so $\log(c_1)$ is negative) results in $p_f \leq n^{-c}$.

Thus, with $\lambda = c_2 \log n$ islands, where c_2 is a sufficiently large constant, at least one island ends each phase with $x^* = \mathrm{OPT}_p$ with high probability; this individual is propagated to all other islands at the start of the next phase, allowing OPT_n to be constructed and propagated to all islands at the end of the last phase. □

4.4.1 Shorter migration intervals

It is also possible to track Maze with shorter migration intervals $\tau < t_0$. We first consider the case where migration may occur multiple times during a phase,

as long as the final migration occurs at least $\Omega(n)$ iterations before the end of the phase, allowing the islands time to reconstruct OPT_p if migration propagated ALT_p to all islands.

Theorem 4.12 *With the migration interval $\tau \leq t_0$, $\lambda = c_1 \log n$ islands are sufficient to track the optimum of the* Maze *as long as no migration occurs during $c_2 n$ iterations preceding any phase transition, where c_1 and c_2 are sufficiently large constants, and migration occurs at least once during each phase.*

Proof. We follow the proof of Theorem 4.11, and pessimistically assume that the last migration during each phase occurs $c_2 n$ iterations before the end of the phase, where $c_2 > 0$ is a sufficiently large constant, during an ALT-favoring iteration, and propagates the ALT individual to all islands.

Lemma 4.8 can then be applied: on each island, the probability of ending phase p with $x^* = \text{OPT}_p$ is then at least a constant greater than 0, as $x^* = \text{ALT}_p \in \text{ALL}_p$ has been propagated to the island at least $c_2 n$ iterations before the end of the phase. Thus, the situation at the end of the phase returns to that considered in Theorem 4.11: each of $\lambda = \Omega(\log n)$ islands has at least a constant probability of ending phase p with $x^* = \text{OPT}_p$.

With a large-enough λ, we can with high probability conclude that during each phase p, migration propagates ensures that all islands have an ALL_p individual as the best-so-far solution at least $c_2 n$ iterations before the end of the phase, and therefore each phase will with at least one island having $x^* = \text{OPT}_p$. That island will then have an ALL_{p+1} individual during the next phase, which will migrate to the any islands that lose track of the Maze during the next phase, allowing the argument to be repeated inductively for each of n phases. □

Additionally, if migration is only allowed to occur during iterations assigning the OPT individual the greatest fitness value, it can never replace an OPT individual on an island with an ALT individual; in this case, migration can occur close to the end of the phase without negative consequences.

Theorem 4.13 *When migration occurs only during iterations when $f(ALT_p) < f(OPT_p)$, and occurs for the first time at least $c_1 n$ iterations before the end of each phase, $\lambda = c_2 \log n$ islands, where c_1 and c_2 are sufficiently large constants, are sufficient to track the optimum of the* Maze.

Proof. The initial OneMax phase still succeeds with high probability, as migration cannot decrease the fitness of x^* on each island, and will instead only increase the fitness of the islands lagging behind the current global best-so-far.

During the subsequent MAZE phases, migration only occurs when OPT individuals have the highest fitness value, and therefore cannot cause an island to replace an OPT best-so-far individual with an ALT individual; thus, it can only increase the probability that $x^* = \mathrm{OPT}_p$ on an island at any given time in comparison with the situation considered in Theorem 4.11.

Thus, the argument from the proof of Theorem 4.11 applies: by selecting a sufficiently-large constant for $\lambda = c \log n$, the probability that all islands end phase p with $x^* \neq \mathrm{OPT}_p$ can be made small enough to ensure that this event does not occur with high probability during the n phases of MAZE: then, as long as at least one island ends each phase with $x^* = \mathrm{OPT}_p$, that island's $x^* \in \mathrm{ALL}_{p+1}$ will be propagated to the other islands, and per Lemma 4.8, $c_1 n$ iterations later, every island will once again have at least a constant probability of having $x^* = \mathrm{OPT}_{p+1}$. □

Thus, we have shown that $\lambda = c_2 \log n$ islands running a (1+1) EA are sufficient to track the optimum of the MAZE with varying migration intervals, as long as migration occurs at least $c_1 n$ iterations before a phase transition, and any migration occurring within $c_1 n$ iterations prior to a phase transition occurs only while the OPT individual has a higher fitness value than the ALT individual. Both theorems rely on a mixing time argument to ensure that at least a constant fraction of the $\Omega(\log n)$ islands end the phase with $x^* = \mathrm{OPT}_p$, and migration is used to repopulate any islands that lose track of the MAZE prior to the next phase transition.

4.4.2 Longer migration intervals

Migration intervals longer than the MAZE phase length are also viable: essentially, there is no need to repopulate all islands after *every* phase transition. When $\tau > t_0$, we merely need to ensure that migration occurs frequently enough to repopulate the islands that lose track of the MAZE before *all* islands do so. We will show that if $\tau \geq c\, t_0 \log \lambda$, $\lambda = O(\log n)$ islands are no longer sufficient to track the MAZE.

THEOREM 4.14 *For $\tau = c_1\, t_0 \log \lambda$, where $c_1 > 0$ is a sufficiently large constant, $\lambda = O(\log n)$ islands are not sufficient to track the optimum of the* MAZE.

PROOF. Consider an interval of $c\, t_0 \log \lambda$ iterations during which no migration will occur, where $c_1 \geq c > 0$ is a sufficiently large constant, starting at the beginning of some phase p, such that $n/2 + 3 < p < n - \log \lambda$. Pessimistically

assume that before the start of phase p, none of the $\lambda = c \log n$ islands lost track of the MAZE, and thus all islands begin phase p at least a constant probability of having an ALL_p individual per Lemma 4.8.

Considering each island individually, each of the $\log \lambda$ phases in the interval has at least a constant probability of ending with $x^* \neq \text{OPT}_p$, causing the next phase to have at least a constant probability of ending with $x^* = 1^n$; let $p_L > 0$ be a constant lower bound on the probability of each phase ending with $x^* = 1^n$. Let $X^{(t)}$ be the number of islands with $x^* \neq 1^n$ at the start of phase $p + t$ (and $X^0 = \lambda$); it then holds that:

$$E(X^{(t)} - X^{(t+1)} \mid X^{(t)}) \geq p_L X^{(t)}$$
$$E(X^{(t)} \mid X^{(0)}) = (1 - p_L)^t X^{(0)}$$

Theorem 4.4 can then be applied: the expected number of phase transitions $T = \min_t\{X^{(t)} = 0\}$ until all islands have lost track of the MAZE (i. e. have $x^* = 1^n$) is:

$$E(T \mid X^{(0)}) \leq \frac{\ln(X^{(0)}) + 1}{1 - p_L} = O(\ln \lambda)$$

By applying Markov's inequality, we can bound $P(T \geq 2E(T \mid X^0)) \leq 1/2$, and hence by setting $\tau = 2E(T \mid X^0) = c \log \lambda$, where c is a sufficiently large constant, lower-bound the probability of all islands ending the interval with $x^* = 1^n$ by at least $1/2$. As there are $\Omega(n/(\log \lambda)) = \Omega(n/(\log \log n))$ such intervals following phase $p > n/2 + 3$, the probability that the island model does loses track of the MAZE in at least one of these intervals is at least $1 - 2^{-\Omega(n/\log \log n)}$. □

4.5 Experiments

To supplement the asymptotic results expressed in Theorems 4.6 and 4.11, we have also performed simulations of the island model (Algorithm 4.2) with $\tau = 1$ (migration occurring in every iteration) and $\tau = t_0$ (migration occurring at the beginning of each phase) for $n = 75$, $t_0 = n^3$, and various choices of λ.

The results of the simulations are summarized in Figure 4.1: $\lambda = 10$ is sufficient for none of the 250 simulations to lose track of the MAZE over the $n + 1$ phases

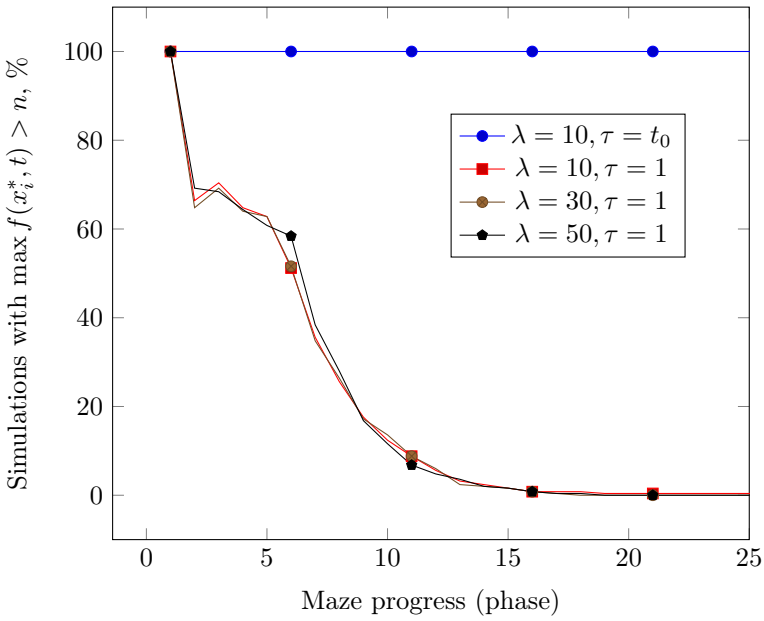

Figure 4.1: Number of simulations of Algorithm 4.2 with $n = 75, t_0 = n^3$, and various choices of λ and τ, having an individual with a better-than-OneMax value at the start of each Maze phase; 250 simulations for each choice of λ. Only the first 25 phases are shown here, as there were no further changes observed in the subsequent phases.

when $\tau = t_0$, while when $\tau = 1$, all of the simulations performed lose track of the Maze even when $\lambda = 50$. During the first few phases of Maze, it is possible for the islands to reconstruct an ALL_p individual by a modest simultaneous-bit mutation within a single phase, slowing the failure rate of the first few phases; this becomes exponentially less likely as Maze progresses. Notably, increasing λ does not have a strong positive effect in the $\tau = 1$ setting for the considered values of λ, perhaps because each phase ends on an iteration where the ALT individual is assigned a higher fitness value.

In Algorithm 4.2, the migration topology is a complete graph: the best individual among all islands in a given iteration is chosen to migrate to all islands. This does not maintain any diversity at migration points and is potentially dangerous when migration occurs frequently. We therefore wondered whether a sparser migration topology might be more beneficial with frequent migration. In an experiment, we have replaced the complete topology with a directed ring: during migration, each island selects a new best-so-far individual among its current best-so-far individual, and the best-so-far individual of its predecessor in the

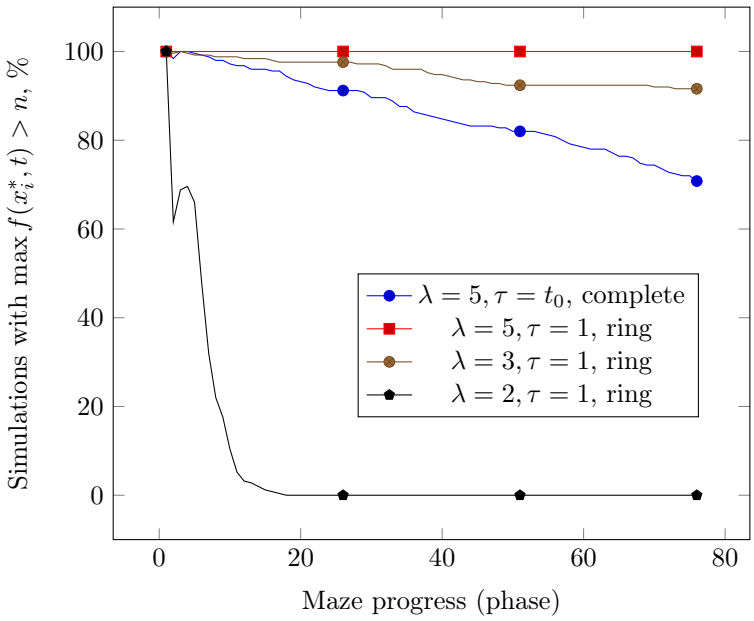

Figure 4.2: With $n = 75, t_0 = n^3$, with various choices of λ, τ, and the migration topology (either a complete graph or a directed ring); 250 simulations in each setting. For $\lambda = 2$, the ring and complete topologies are equivalent.

ring. Figure 4.2 displays the results in the setting with $n = 75, t_0 = n^3$ and the directed ring topology used with $\tau = 1$, compared to the complete topology and $\tau = t_0$. Experimentally, choosing $\tau = 1, \lambda > 2$ and the ring migration topology appears to yield better probability of tracking Maze than $\tau = t_0, \lambda = 5$ and the complete migration topology.

4.6 Conclusions

We have presented a first runtime analysis of parallel EAs in dynamic optimization. A simple island model with λ islands and length of the migration interval τ was studied on the dynamic benchmark function Maze. In the case of extreme communication, i. e., $\tau = 1$, even a large number of islands does not allow efficient tracking of the optimum. However, with a carefully chosen value for τ, already a logarithmic number of islands was proven to be sufficient for efficient tracking. Finally, the relationship of τ, λ, and the ability of the island model to

track the optimum was investigated more closely. Our results indicate that the careful choice of the migration policy, and more generally, the communication in parallel EAs, can be significantly more advantageous than a large population.

In future work, we would like to study parallel EAs on different dynamic optimization problems in order to understand the interplay of migration intervals and number of islands more thoroughly. As our positive results are crucially dependent on a proper choice of τ, it may also be worth studying adaptive or even self-adaptive choices of the migration interval in order to automatically determine a good value for τ. Here the adaptive model suggested in [MS14] could be interesting. Additionally, the impact of the migration topology, something that we have only investigated experimentally, could be considered in more detail and formalized.

Chapter 5

(1+1) EA on Generalized Dynamic OneMax

Timo Kötzing[+] Andrei Lissovoi[*] Carsten Witt[*]

[+] Friedrich-Schiller-Universität, Jena, Germany
[*] DTU Compute, Technical University of Denmark

Evolutionary algorithms (EAs) perform well in settings involving *uncertainty*, including settings with stochastic or dynamic fitness functions. In this paper, we analyze the (1+1) EA on dynamically changing OneMax, as introduced by [Dro03]. We re-prove the known results on first hitting times using the modern tool of drift analysis. We extend these results to search spaces which allow for more than two values per dimension.

Furthermore, we make an *anytime analysis* as suggested by [JZ14], analyzing how closely the (1+1) EA can track the dynamically moving optimum over time. We get tight bounds both for the case of bit strings, as well as for the case of more than two values per position. Surprisingly, in the latter setting, the expected quality of the search point maintained by the (1+1) EA does not depend on the number of values per dimension.

5.1 Introduction

Randomized search heuristics, such as evolutionary algorithms (EAs), are general purpose optimization algorithms applicable to virtually any (formal) op-

timization task. In particular, EAs (and other randomized search heuristics) have been applied very successfully in domains featuring *uncertainty*; for example, the objective functions (so-called fitness functions) can be *randomized* or *dynamically changing* (see JB05 for an excellent survey on evolutionary algorithms in settings featuring uncertainty). In this paper we focus on dynamically changing fitness functions.

We call a fitness function *dynamic* if the fitness values of search points depend on the iteration number (but might be *deterministic* for each iteration). For example, the shortest path between two cities might depend on whether it is rush hour or not. The classical task of an optimization algorithm is to find the best solution it can (in terms of fitness); for dynamic optimization, there need not be a single solution which is good at all times: solutions that are good now might be bad later. Thus, algorithms in this domain need to be able to find *and track* the optimal solution (or at least a good solution) over time as the problem changes.

With this paper we contribute to the *theoretical foundations* of randomized search heuristics, for the domain of dynamic fitness functions. While there has been a lot of work on the theory of randomized search heuristics in static settings (see AD11, NW10b, Jan13), there are only a few works on dynamically changing fitness functions. The utility of a population for tracking problems was studied in evolutionary computation by [JS05], while different mechanisms for ensuring population diversity have been considered by [OZ13]. In particular, a mechanism called genotype diversity was proved to be inefficient on a particular dynamic problem. The papers by [KM12] and [LW15a] consider dynamic pseudo-Boolean functions where the optimum moves slowly from the all-ones to the all-zeros bit string; the papers show that, while the Max-Min Ant System is able to track the changes occurring in this fitness function, an evolutionary algorithm (in LW15a using a population) loses track of the optimum. [JZ14] analyzed the performance of a standard evolutionary algorithm on a dynamically changing fitness function, introducing "anytime analysis", the expected distance to the optimum at any given point in time.

The oldest theoretical running time analyses of evolutionary algorithms for dynamic fitness functions are probably due to [Dro02, Dro03]. Here the fitness function is the (Hamming-) distance to a (dynamically changing) point in the hypercube (so-called *dynamic* OneMax). In each iteration, the current optimum is changed by flipping each bit with some fixed probability p; from [Dro03] we know that the standard (1+1) EA is able to *find* the optimum in polynomial time if and only if $p = O(\log n/n^2)$.

In this paper we build on the setting of dynamic OneMax from [Dro03]. We re-prove the classic results using the modern tools of drift theory and extend

them as follows. First, we generalize the domain by allowing not only bit strings, but each position can take any of the values in $\{0, \dots, r-1\}$. Fitness is again distance from the current optimum; we measure distances as the sum of the distances of each component, where in each component we measure distance "with wrap around" (giving each component the metric space of a ring, see Section 5.2 for a detailed definition). We extend the (1+1) EA by letting mutation change any position independently with probability $1/n$; any changed position is randomly increased or decreased by one (with probability 1/2 each). Note that similar extensions of the OneMax function (without dynamic changes) have been studied by [DJS11] and [DP12]; they considered arbitrary linear functions over $\{0, \dots, r\}$, and a mutation where changing a position means selecting a new value at this position uniformly at random (excluding the old value). We chose the ring topology, as we consider it more natural for a dynamically moving optimum, which can now never run into a boundary.

The second extension to [Dro03] is that we do not only consider the first hitting time of the optimum but, as suggested by [JZ14], we give an "anytime analysis", an analysis of the distance to the optimum at any time of the search process.

We state our setting more formally in Section 5.2. In Section 5.3 we give our anytime results, considering cases with $p = o(1/n)$. The first part is about the case of bit strings (i. e. $r = 2$), where we show that the distance to the optimum is (in the limit) strongly concentrated at $\Theta(pn^2)$. This gives an anytime result as suggested by [JZ14]. The second part shows that, for large r, the distance to the optimum in each dimension is strongly concentrated at $O(1)$, leading to an expected distance of $O(n)$ from the optimum (again an anytime result). Note that this shows that the distance is independent of r.

In Section 5.4 we consider the expected hitting times of the (randomly moving) optimum. Here we re-prove the result of [Dro03] (who considered the case of $r = 2$) that the first hitting time is polynomial if $p = O(\log n/n^2)$. We use modern drift theory, leading to a much shorter and more elegant proof, resulting in a better bound. We extend this result to arbitrary r.

[Dro03] also gave a lower bound, which shows that for $p = \omega(\log n/n^2)$ we do not get polynomial hitting times; in Section 5.5 we re-prove this result (again with modern drift theory) and extend them to arbitrary values of r.

As mentioned, we will use modern drift theory to derive our results. In Section 5.2 we restate known drift theorems, partly in more general form than before, and also present new variants. A new theorem regards variable drift, which allows for *negative* drift close to the optimum and shows how stochastic processes can bridge such an area of headwind. For our anytime analysis, the crucial tool is a lemma by [LW15a], which we restate as Lemma 5.6 below, effec-

tively turning expected drift into probabilities about deviating from the target of the drift after having reached that target.

5.2 Preliminaries

In this section we first make our setting formal (see Section 5.2.1) and then give a number of helpful theorems, both from the literature and new theorems (see Section 5.2.2).

5.2.1 Setting

[Dro03] proposes a dynamic version of OneMax and analyses the performance of (1+1) EA on this dynamic fitness function in terms of first hitting times of the optimum. We extend this dynamic version of OneMax as follows.

For all $r \in \mathbb{N}$, let $[r] = \{0, \dots, r-1\}$; for two elements $x, y \in [r]$, we let $d(x,y) = \min((y-x) \bmod r, (x-y) \bmod r)$ (intuitively, d is the metric of $[r]$ with wrap-around). We consider the search space $[r]^n$ (note that $r = 2$ gives the standard setting of bit strings).

Given a current optimum a, we let

$$\textsc{OneMax}_a : [r]^n \to \mathbb{R}, x \mapsto \sum_{i=1}^{n} d(a_i, x_i).$$

The goal of the (1+1) EA is to evolve and *maintain* bit strings with as small as possible $\textsc{OneMax}_a$-value. In particular, in this setting optimization means *minimizing*.

We consider the following mutation operator on $[r]^n$, parametrized by $p \in [0,1]$. Given $x \in [r]^n$, create mutant x' by choosing, for each component $i \leq n$ independently,

$$x'_i = \begin{cases} x_i + 1 \mod r, & \text{with probability } p/2; \\ x_i - 1 \mod r, & \text{with probability } p/2; \\ x_i, & \text{with probability } 1-p. \end{cases}$$

We use this operator with $p = 1/n$ for the (1+1) EA (see Algorithm 5.1).

Algorithm 5.1 (1+1)-EA

Choose $x \in [r]^n$ uniformly at random
for ever do
  $x' \leftarrow \text{mutate}(x)$
  if $\text{fitness}(x') \leq \text{fitness}(x)$ **then**
    $x \leftarrow x'$

In each iteration, we change the optimum by applying the mutation operator with some fixed p. This extends the setting of [Dro03], where only the case of $r = 2$ was addressed.

5.2.2 Drift Theorems

In this section we first discuss drift theorems regarding first hitting times, and afterwards discuss how one can turn statements about the drift into statements about occupation probabilities (of a random process).

5.2.2.1 First Hitting Times

As mentioned in the introduction, almost all of our proofs use state-of-the-art drift statements, many of which were not available to [Dro03]. The simplest case is the one of additive drift, as described in the following theorem. It goes back to [HY01]; however, is presented in a more general form here, which is proved in [LW14a].

Theorem 5.1 (Additive drift, expected time)
Let $(X^t)_{t\geq 0}$, be a stochastic process, adapted to some filtration $\mathcal{F}_t$, over a bounded state space $S \subseteq \mathbb{R}_0^+$. Let $T_0 := \min\{t \geq 0 \colon X^t = 0\}$ denote the first hitting time of 0 and assume that both $\mathrm{E}(X^0)$ and $\mathrm{E}(T_0 \mid X^0)$ are finite. Then:

(i) If $\mathrm{E}(X^t - X^{t+1} \mid \mathcal{F}_t; X^t > 0) \geq \varepsilon$ then $\mathrm{E}(T_0 \mid X^0) \leq \frac{X^0}{\varepsilon}$.

(ii) If $\mathrm{E}(X^t - X^{t+1} \mid \mathcal{F}_t; X^t > 0) \leq \varepsilon$ then $\mathrm{E}(T_0 \mid X^0) \geq \frac{X^0}{\varepsilon}$.

Intuitively, the filtration $\mathcal{F}_t$ describes the history of the process up to time t. For Markov processes, it simplifies to the state at time t; for instance, the first drift condition would read $\mathrm{E}(X^t - X^{t+1} \mid X^t; X^t > 0) \geq \delta$ instead.

Often, the state space Z of the underlying stochastic process and the support S of the random variables X^t are not identical. Obviously, this is the case if the state space is not a subset of $\mathbb{R}$, e. g., if we are dealing with bit strings, where we have $Z := \{0,1\}^n$. In particular, even if the state space is the real numbers, it might be convenient to introduce a so-called potential function (also called Lyapunov function) $g\colon Z \mapsto S$, which leads to a new stochastic process $Y^t := g(X^t)$ on the new state space S. One reason might be that the drift of Y^t is easier to compute. We abstract away from this mapping by allowing the random variables X^t from Theorem 5.1 to represent a process obtained after any transformation of the original process using a potential function. Such a transformation might turn Markovian processes into non-Markovian ones.

Theorem 5.1 is only concerned with bounds on the expected value of the first hitting time of the target state 0. Recently, it has been shown in [Köt14] that the first hitting time is sharply concentrated (exhibits so-called tail bounds) if additional assumptions are made on the step size. We restate this in the following theorem.

Theorem 5.2 (Additive Drift, Tail Bounds)
Let the prerequisites of Theorem 5.1 hold and assume additionally that $|X^t - X^{t+1}| < c$ *for some* $c > 0$ *and all* $t \geq 0$. *Then:*

(i) If $\mathrm{E}\big(X^t - X^{t+1} \mid \mathcal{F}_t; X^t > 0\big) \geq \varepsilon$ *then* $\Pr(T_0 > s) \leq \exp(-s\varepsilon^2/(16c^2))$ *for all* $s \geq 2X^0/\varepsilon$.

(ii) If $\mathrm{E}\big(X^t - X^{t+1} \mid \mathcal{F}_t; X^t > 0\big) \leq \varepsilon$ *then* $\Pr(T_0 < s) \leq \exp(-(X^0)^2/(16c^2 s))$ *for all* $s \leq X^0/(2\varepsilon)$.

The previous two theorems dealt with a drift towards the target state 0. If the drift is directed away from the target, lower bounds on the hitting time can be proved. This is the realm of *negative drift theorems*, several variants of which exist (see OW11, OW12 for the original version). In this work we use the following version, adapted from [RS14, Theorem 4], which takes into account the probabilities of staying at a state. For technical reasons, it is restricted to Markov processes and the use of a possible potential function is made explicit.

Theorem 5.3 (Negative drift with self-loops)
Let $(X^t)_{t\geq 0}$, *be a Markov process over a state space* S. *Suppose there exist an interval* $[a,b] \subseteq \mathbb{R}_0^+$, *two constants* $\delta, \varepsilon > 0$, *a function* $r(\ell)$ *satisfying* $1 \leq r(\ell) = o(\ell/\log(\ell))$, *and a potential function* $g\colon S \to \mathbb{R}_0^+$, *such that for all* $t \geq 0$, *the following two conditions hold:*

(i) $\mathrm{E}(\Delta^t \mid X^t; a < g(X^t) < b) \geq \varepsilon(1 - p_{t,0})$,

(ii) $\Pr(|\Delta^t| \geq j \mid X^t; a < g(X^t)) \leq \frac{r(\ell)(1-p_{t,0})}{(1+\delta)^j}$ *for* $j \in \mathbb{N}_0$,

where $\Delta^t = g(X^{t+1}) - g(X^t)$ *and* $p_{t,0} := \Pr(\Delta^t = 0 \mid X^t)$.

Then there is a constant $c^* > 0$ *such that for* $T^* := \min\{t \geq 0 : g(X^t) \leq a \mid g(X^0) \geq b\}$ *it holds*

$$\Pr\left(T^* \leq 2^{c^* \ell / r(\ell)}\right) = 2^{-\Omega(\ell/r(\ell))}.$$

Intuitively, drift away from the target makes it difficult to reach the target. Nevertheless, if the drift is negative only for a few states and directed towards the target at the remaining states, the expected first hitting time of the target might still be small. Such a scenario of "headwind drift" on the way towards the target will appear in our analyses if the probability of flipping a bit of the optimum is small, e. g., $p = O((\log n)/n^2)$), resulting in only very few states close to the target having negative drift.

The following novel theorem proves upper bounds in the presence of possibly negative drift. The bounds $\delta(i)$ are lower bounds on the drift at state i, pessimistically assuming that all steps towards the target improve only by 1. The $p^-(i)$ and $p^+(i)$ are bounds on the probability of improving by at least 1 and worsening by at least 1, respectively. The theorem is general enough to analyze different scenarios, e. g., blind random walks on the hypercube. However, we will mostly apply Corollary 5.5, which is easier to use.

For notational convenience, we state the theorem only for Markov processes, however, it can easily be generalized to non-Markovian ones. Extensions to continuous search spaces seem also possible; however, these are not straightforward. Therefore, the state space is restricted to be non-negative integers.

Theorem 5.4 (Headwind drift, upper bound)
Let $(X^t)_{t\geq 0}$ *be a Markov process on* $\{0, \dots, N\}$. *Let bounds*

$$p^-(i) \leq \Pr\big(X^{t+1} \leq i - 1 \mid X^t = i\big)$$

and

$$p^+(i) \geq \Pr\big(X^{t+1} \geq i + 1 \mid X^t = i\big),$$

where $0 \leq i \leq N$, *be given, and define*

$$\delta(i) := p^-(i) - \mathrm{E}\big((X^{t+1} - i) \cdot \mathbb{1}\{X^{t+1} > i\} \mid X^t = i\big).$$

Assume that $\delta(i)$ *is monotone increasing w. r. t.* i *and let* $\kappa \geq \max\{i \geq 0 \mid \delta(i) \leq 0\}$ *(noting that* $\delta(0) \leq 0$*). The function* $g\colon \{0, \dots, N+1\} \to \mathbb{R}^+$ *is defined by*

$$g(i) := \sum_{k=i+1}^{N} \frac{1}{\delta(k)}$$

for $i \geq \kappa$ (in particular, $g(N) = g(N+1) = 0$), and inductively by

$$g(i) := \frac{1 + (p^+(i+1) + p^-(i+1))g(i+1)}{p^-(i+1)}$$

for $i < \kappa$.

Then it holds for the first hitting time $T := \min\{t \geq 0 \mid X^t = 0\}$ of state 0 that

$$\mathrm{E}\big(T \mid X^0\big) \leq g(0) - g(X^0).$$

Remark. $\delta(i)$ respects the following simple lower bound:

$$\delta(i) \geq \mathrm{E}\big((i - X^{t+1}) \cdot \mathbb{1}\{X^{t+1} \geq i-1\} \mid X^t = i\big).$$

Proof. We will prove that $g(i)$ is a monotone decreasing function and can be used as a potential function to satisfy the drift condition

$$\mathrm{E}\big(g(X^{t+1}) - g(i) \mid X^t = i \wedge i > 0\big) \geq 1.$$

Due to the monotonicity of $g(i)$, the first hitting time where $g(X^t) = g(0)$ equals the first hitting time where $X^t = 0$ for the original X^t-process. Then the theorem follows by the additive drift theorem (Theorem 5.1).

To prove the monotonicity of $g(i)$, we observe that $g(i) - g(i+1) \geq 0$ for $i \geq \kappa$ immediately by definition. For $i < \kappa$, we get

$$\begin{aligned}\frac{g(i)}{g(i+1)} &= \frac{\frac{1}{g(i+1)} + (p^+(i+1) + p^-(i+1))}{p^-(i+1)} \\ &\geq \frac{p^+(i+1) + p^-(i+1)}{p^-(i+1)} \geq 1,\end{aligned}$$

where we used $g(i+1) \geq 0$. This completes the proof of the monotonicity.

To prove the drift condition, we distinguish between two cases. Suppose $X^t = i > \kappa$. The monotonicity of the $\delta(i)$ implies the "concavity" condition $g(i-1) -$

$g(i) \geq g(i) - g(i+1)$ for $i > \kappa$. We obtain

$$\begin{aligned}
&\mathrm{E}\big(g(X^{t+1}) - g(i) \mid X^t = i\big) \\
&\geq p^-(i)(g(i-1) - g(i)) \\
&\qquad - \left(\sum_{k=1}^{N} (g(i) - g(i+k))\Pr\big(X^{t+1} = i+k\big)\right) \\
&\geq p^-(i)(g(i-1) - g(i)) \\
&\qquad - \left(\sum_{k=1}^{N} k(g(i-1) - g(i))\Pr\big(X^{t+1} = i+k\big)\right) \\
&= p^-(i)(g(i-1) - g(i)) \\
&\qquad - (g(i-1) - g(i))\mathrm{E}\big((X^{t+1} - i)\mathbb{1}\{X^{t+1} > i\} \mid X^t = i\big) \\
&= (g(i-1) - g(i))\delta(i) = \frac{1}{\delta(i)} \cdot \delta(i),
\end{aligned}$$

where the second inequality used the concavity repeatedly. If $X^t = i \leq \kappa$, we pessimistically assume all steps towards the target to reach $i-1$ and all away from it to reach N (resulting in zero g-value). Hence, using the definition of $g(i-1)$,

$$\begin{aligned}
&\mathrm{E}\big(g(X^{t+1}) - g(i) \mid X^t = i\big) \\
&\geq \; p^-(i)(g(i-1) - g(i)) - p^+(i)g(i) \\
&= \; p^-(i)\left(\frac{1 + (p^+(i) + p^-(i))g(i)}{p^-(i)} - g(i)\right) - p^+(i)g(i) \\
&= \; 1,
\end{aligned}$$

which proves the bound on the drift and, therefore, the theorem.

We now state the announced corollary, which gives us a closed expression for the expected first hitting time $\mathrm{E}\big(T \mid X^0\big)$. This expression involves the factor $\sum_{k=\kappa+1}^{N} \frac{1}{\delta_k}$ that is reminiscent of the formula for the expected first hitting time of state κ under variable drift towards the target (see, e. g., RS14 for a formulation of the variable drift theorem). For the states less than κ, where drift away from the target holds, the product $\prod_{k=1}^{\kappa} \frac{p^+(k)+p^-(k)}{p^-(k)}$ comes into play. Intuitively, it represents the waiting time for the event of taking κ consecutive steps against the drift. Since the product involves probabilities conditioned on leaving the states, which effectively removes self-loops, another sum of products must be added. This sum, represented by the second line of the expression for $\mathrm{E}\big(T \mid X^0\big)$, intuitively accounts for the self-loops.

Corollary 5.5 *Let the assumptions of Theorem 5.4 hold. Then*

$$\mathrm{E}(T \mid X^0) \leq \left(\left(\sum_{k=\kappa+1}^{N} \frac{1}{\delta_k}\right)\left(\prod_{k=1}^{\kappa} \frac{p^+(k)+p^-(k)}{p^-(k)}\right)\right) + \left(\sum_{k=1}^{\kappa} \frac{1}{p^-(k)} \prod_{j=1}^{k-1} \frac{p^+(j)+p^-(j)}{p^-(j)}\right).$$

Proof. It is sufficient to prove that the right hand side is an upper bound on the $g(0)$ defined in Theorem 5.4. We note that $\sum_{k=\kappa+1}^{N} \frac{1}{\delta_k} = g(\kappa)$. The inductive expression for $g(i)$ yields $g(i) \leq \frac{1+(p^+(i+1)+p^-(i+1))g(i+1)}{p^-(i+1)}$ for $i \leq \kappa-1$. Inductively

$$g(0) \leq \left(\prod_{j=0}^{\kappa-1} \frac{p^+(j+1)+p^-(j+1)}{p^-(j+1)}\right) g(\kappa) + \left(\sum_{k=0}^{\kappa-1} \frac{1}{p^-(1)} \prod_{j=1}^{k} \frac{p^+(j)+p^-(j)}{p^-(j+1)}\right),$$

and the corollary follows by index transformations and regrouping terms.

5.2.2.2 Occupation Probabilities

In this section we move away from analyses of the first hitting time of a state and direct our attention to so-called occupation probabilities. For the anytime analysis in Section 5.3, we want to make statements about how far from the optimum the (1+1) EA will stray, and with what probability. In particular, we want to know the probability that the current search point is more than j away from the optimum in iteration t, for large t. The idea is that a stochastic process $(X^t)_{t\geq 0}$ on $\mathbb{R}$ which has a drift towards 0 will, after hitting 0 for the first time, likely stay in the proximity of 0 and stray off only with a low probability. This is what is meant by "occupation probabilities".

[Haj82], in his third section, already gives some general bounds on these probabilities. This is also the idea of another lemma regarding occupation probabilities given in [LW15a, Lemma 13] (restated below as Lemma 5.6), from which we will here derive a simple version (Theorem 5.7), tailored to the case of additive drift and Markov processes with self-loops.

Lemma 5.6 (LW15a, Lemma 13)
Let $(X^t)_{t\geq 0}$, be a stochastic process, adapted to a filtration $(\mathcal{F}_t)_{t\geq 0}$, over some state space $S \subseteq \{0\} \cup [x_{\min}, x_{\max}]$, where $x_{\min} \geq 0$. Let $a, b \in \{0\} \cup [x_{\min}, x_{\max}]$, $b > a$. Let $h\colon [x_{\min}, x_{\max}] \to \mathbb{R}^+$ be such that $1/h$ is integrable on $[x_{\min}, x_{\max}]$ and define $g\colon \{0\} \cup [x_{\min}, x_{\max}] \to \mathbb{R}^{\geq 0}$ by $g(x) := \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{x} \frac{1}{h(y)}\,\mathrm{d}y$ for $x \geq x_{\min}$ and $g(0) := 0$.

If there exist $\lambda > 0$, $\beta < 1$ and $D > 0$ such that

$$\mathrm{E}\Big(e^{-\lambda(g(X^t)-g(X^{t+1}))} \cdot \mathbb{1}\left\{X^t > a\right\} \mid \mathcal{F}_t\Big) \leq \beta$$
$$\text{and } \mathrm{E}\Big(e^{-\lambda(g(a)-g(X^{t+1}))} \cdot \mathbb{1}\left\{X^t \leq a\right\} \mid \mathcal{F}_t\Big) \leq D$$

then

$$\Pr(X^t \geq b \mid X^0) < \beta^t \cdot e^{\lambda(g(X^0)-g(b))} + \frac{1-\beta^t}{1-\beta} D e^{\lambda(g(a)-g(b))}$$

for $t > 0$.

We give two definitions regarding Markov processes before giving our theorem regarding occupation probabilities. Let a Markov process $(X^t)_{t\geq 0}$ on $\mathbb{R}_0^+$ be given. We say that $(X^t)_{t\geq 0}$ *has step size at most* $c \in \mathbb{R}$ if, for all t, $|X^t - X^{t+1}| \leq c$. We say that $(X^t)_{t\geq 0}$ *has self-loop probability at least* $p_0 \in \mathbb{R}$ iff, for all t such that $X^t > 0$ we have $\Pr\big(X^t = X^{t+1} \mid X^t\big) \geq p_0$. From Lemma 5.6 we derive the following statement on occupation probabilities for the case of bounded step sizes.

Theorem 5.7 (Occupation probabilities)
Let a Markov process $(X^t)_{t\geq 0}$ on $\mathbb{R}_0^+$ with additive drift of at least d towards 0 be given, starting at 0 (i.e. $X_0 = 0$), with step size at most c and self-loop probability at least p_0. Then we have, for all $t \in \mathbb{N}$ and $b \in \mathbb{R}_0^+$,

$$\Pr(X^t \geq b) \leq 2e^{\frac{2d}{3c(1-p_0)}(1-b/c)}.$$

Proof. First, we define a new Markov process $(Y^t)_{t\geq 0}$ obtained from X^t by omitting all steps that do not change the current state; formally, since we consider Markov chains, we have $Y^t - Y^{t+1} = X^t - X^{t+1}$ in the conditional space where $X^{t+1} \neq X^t$. By definition of conditional probability and expectation, we obtain

$$\begin{aligned}\mathrm{E}\big(Y^t - Y^{t+1} \mid X^t\big) &= \frac{\mathrm{E}\big(X^t - X^{t+1} \mid X^t\big)}{\Pr(X^{t+1} \neq X^t \mid X^t)} \\ &\geq \frac{\mathrm{E}\big(X^t - X^{t+1} \mid X^t\big)}{1-p_0}\end{aligned}$$

since the probability of changing the state is at most $1 - p_0$.

The theorem makes a statement for all t, however, to prove it, it is enough to consider steps that actually change state. Hence, in the following, the aim is to analyze the Y-process using Lemma 5.6 with $a := x_{\min} := 0$ and the constant function $h(x) := 1$. From this we obtain the trivial potential function $g(x) = x$. From our prerequisites and the previous paragraph, we get

$$\begin{aligned}
&\mathrm{E}\big(g(Y^t) - g(Y^{t+1}) \mid Y_t; Y^t > 0\big) \\
&\quad = \mathrm{E}\big(Y^t - Y^{t+1} \mid Y_t; Y^t > 0\big) \\
&\quad \geq \frac{\mathrm{E}\big(X^t - X^{t+1} \mid X^t; X^t > 0\big)}{1 - p_0} \\
&\quad \geq \frac{d}{1 - p_0}.
\end{aligned}$$

Let $d^* := d/(1 - p_0)$. To bound the moment-generating function of the drift, we abbreviate $\Delta^t := Y^t - Y^{t+1}$. We already know that $\mathrm{E}(\Delta^t \mid Y^t; Y^t > 0) \geq d^*$ and argue

$$\mathrm{E}\Big(e^{-\lambda\Delta^t} \cdot \mathbb{1}\{Y^t > 0\} \mid Y^t\Big) \leq \mathrm{E}\Big(e^{-\lambda\Delta^t} \mid Y^t; Y^t > 0\Big)$$

In the following, we condition on $Y^t; Y^t > 0$ everywhere but omit this from the formulas for the sake of readability. Using the Taylor expansion of the exponential function, we get

$$\mathrm{E}\Big(e^{-\lambda\Delta^t}\Big) \leq 1 - \lambda\mathrm{E}\big(\Delta^t\big) + \sum_{k=2}^{\infty} \frac{\lambda^k \mathrm{E}\big(|\Delta^t|^k\big)}{k!},$$

which for any $\eta \geq \lambda$ is at most

$$1 - \lambda\mathrm{E}\big(\Delta^t\big) + \frac{\lambda^2}{\eta^2} \sum_{k=2}^{\infty} \frac{\eta^k \mathrm{E}\big(|\Delta^t|^k\big)}{k!}.$$

Now, by setting $\eta := 1/c$, $\lambda := 2d^*/(3c^2)$ and noting that $\Delta^t \leq c$ (also for the Y-process), we get the bound

$$1 - \lambda\mathrm{E}\big(\Delta^t\big) + \lambda \frac{2d^*}{3c^2 \cdot (1/c^2)} \cdot \sum_{k=2}^{\infty} \frac{1}{k!} \leq 1 - \lambda\mathrm{E}\big(\Delta^t\big) + \lambda\frac{d^*}{2}$$

where the last inequality used that $\sum_{k=2}^{\infty} \frac{1}{k!} = e - 2 \leq 3/4$. Altogether, using $\mathrm{E}(\Delta^t) \geq d^*$, we get

$$\begin{aligned}
\mathrm{E}\Big(e^{-\lambda\Delta^t} \cdot \mathbb{1}\left\{Y^t > 0\right\}\Big) &\leq 1 - \frac{\lambda d^*}{2} \leq e^{-\lambda d^*/2} \\
&= e^{-\frac{(d^*)^2}{3c^2}} =: \beta < 1.
\end{aligned}$$

Moreover, in order to apply Lemma 5.6, we need to bound

$$\mathrm{E}\Big(e^{-\lambda(a-Y^{t+1})} \cdot \mathbb{1}\left\{Y^t = 0\right\} \mid Y_t\Big) \leq e^{c\lambda} = e^{\frac{2d^*}{3c}} =: D > 1$$

using $a := 0$ and the bounded step size. Altogether, from the lemma we get

$$\begin{aligned}\Pr(Y^t \geq b) &\leq \left(\beta^t + \frac{1-\beta^t}{1-\beta} D\right) e^{-\lambda b} \\ &\leq (1+D)e^{-\lambda b} \leq 2e^{\frac{2d^*}{3c}} e^{-\frac{2bd^*}{3c^2}} \\ &= 2e^{\frac{2d}{3c(1-p_0)} - \frac{2bd}{3c^2(1-p_0)}},\end{aligned}$$

and the last expression is also a bound on $\Pr(X^t \geq b)$ as it does not depend on t.

5.3 An Anytime Analysis

In this section we give our anytime analysis, separately for the cases of $r = 2$ and for large r. We will start in Section 5.3.1 with the classical case of $r = 2$, i. e., bit strings. In Section 5.3.2 we consider large r. We restrict ourselves to $p = o(1/n)$, i. e., in expectation less than one bit of the optimum is changed.

5.3.1 The Case of $r = 2$

We fix $r = 2$ and start by computing the expected change (drift) in the search point. We expect the (1+1) EA, starting from a random string, to make some progress towards the optimum until the number of incorrect bits is lower than the drift caused by the dynamically changing optimum.

More precisely, we consider the process X^t given by the current OneMax-value (i. e., the number of incorrect bits) and assume a current OneMax-value of $X^t = i < n/2$. We identify a forward drift

$$\Delta^-(i) := (i - X^{t+1}) \cdot \mathbb{1}\{X^{t+1} < i\}$$

caused by the selection mechanism of the (1+1) EA and a backward drift

$$\Delta^+(i) := (X^{t+1} - i) \cdot \mathbb{1}\{X^{t+1} > i\}$$

caused by the random movement of the optimum. The total drift $\Delta^t = (X^t - X^{t+1})$ under $X_t = i$ satisfies $\Delta^t = \Delta^-(i) - \Delta^+(i)$, and also $\mathrm{E}(\Delta^t) = \mathrm{E}(\Delta^-(i)) -$

$\mathrm{E}(\Delta^+(i))$. We bound the forward and backward drift. Progress is made when one of the incorrect bits flips and neither the rest of the bits nor the optimum flips, and can only be made by flipping incorrect bits of the current string or the optimum. The total expected number of flipping bits among i bits is $i(p+1/n)$. We obtain

$$(1-p)^n \left(1-\frac{1}{n}\right)^{n-i} \frac{i}{n} \;\leq\; \mathrm{E}\big(\Delta^-(i)\big) \;\leq\; \frac{i}{n} + ip,$$

and, since $p = o(1/n)$,

$$\frac{i}{e^2 n} \;\leq\; \mathrm{E}\big(\Delta^-(i)\big) \;\leq\; \frac{2i}{n}.$$

Similarly, since the OneMax-value can only increase (move away from the optimum) by flipping bits of the optimum, we get

$$(1-1/n)^n (n-i)p(1-p)^i \;\leq\; \mathrm{E}\big(\Delta^+(i)\big) \;\leq\; np,$$

implying, since $i < n/2$

$$\frac{np}{4e^2} \leq \mathrm{E}\big(\Delta^+(i)\big) \leq np.$$

We solve $\mathrm{E}(\Delta^t) = 0$ to find an i^* where we have a drift of zero, and get from the inequalities above that

$$\frac{1}{8e^2} n^2 p \;\leq\; i^* \;\leq\; e^2 n^2 p.$$

If $i > e^2 n^2 p$, there is certainly a drift towards the optimum; and if $i < \frac{1}{8e} n^2 p$, there is certainly a drift away from the optimum. In the region of $i^* = \Theta(n^2 p)$ there is an equilibrium with zero drift, and we would expect the (1+1) EA to approach this region and not to move significantly away from it afterwards. This is made precise in the following theorem.

Theorem 5.8 *Let $r = 2$, $p = o(1/n)$ and $1/p = n^{O(1)}$. Let $(x^t, a^t)_{t\in\mathbb{N}}$ be the sequence of random variables denoting the pair of current search point and current optimum as given by running the (1+1) EA on dynamic* OneMax*. Then, for any $t \geq 0$ and any $\alpha = \omega(\ln n)$ there is $b_t := n - \frac{tpn}{7}$ such that*

$$\Pr\big(d(x^t, a^t) \geq \max\{\alpha b_t,\, 2e^2 n^2 p + \alpha\}\big) \leq e^{-\Omega(\alpha)}.$$

Moreover, for all $t \geq \alpha/p$,

$$\Pr\big(d(x^t, a^t) \geq 2e^2 n^2 p + \alpha\big) \leq e^{-\Omega(\alpha)}.$$

Proof. Still, $X^t := \mathrm{OneMax}_{a^t}(x^t) = d(x^t, a^t)$. We recall that there is a drift towards the target if $X^t > i^*$. More precisely, from the estimations presented

before this theorem we obtain

$$\begin{aligned}\mathrm{E}\big(i - X^{t+1} \mid X^t = i\big) &= \mathrm{E}\big(\Delta^-(i)\big) - \mathrm{E}\big(\Delta^+(i)\big) \\ &\geq \frac{i}{e^2 n} - np \geq np\end{aligned}$$

for $i \geq 2e^2n^2p$. In the following, we analyze the X^t-process using Lemma 5.6 with $a := 2e^2n^2p$, $b = \max\{\alpha b_t, a + \alpha\}$, $x_{\min} := 0$ and the constant function $h(x) := 1$. From this we obtain the trivial potential function $g(x) = x$. To bound the moment-generating function of the drift if $X^t > a$, we use $\Delta := g(X^t) - g(X^{t+1}) = X^t - X^{t+1}$ and argue

$$\mathrm{E}\big(e^{-\lambda\Delta} \cdot \mathbb{1}\{X^t > a\} \mid X^t\big) \leq \mathrm{E}\big(e^{-\lambda\Delta} \mid X^t; X^t > a\big).$$

In the following, we condition on $X^t; X^t > a$ in all expectations unless stated otherwise but omit this for the sake of readability. Using the Taylor expansion of the exponential function, we get

$$\mathrm{E}\big(e^{-\lambda\Delta}\big) \leq 1 - \lambda\mathrm{E}(\Delta) + \sum_{k=2}^{\infty} \frac{\lambda^k \mathrm{E}\big(|\Delta|^k\big)}{k!},$$

which for any $\eta \geq \lambda$ is at most

$$\begin{aligned}&1 - \lambda\mathrm{E}(\Delta) + \frac{\lambda^2}{\eta^2} \sum_{k=2}^{\infty} \frac{\eta^k \mathrm{E}\big(|\Delta|^k\big)}{k!} \\ &\quad = 1 - \lambda\mathrm{E}(\Delta) + \frac{\lambda^2}{\eta^2} \underbrace{\Big(\mathrm{E}\Big(e^{\eta|\Delta|}\Big) - \eta\mathrm{E}(|\Delta|) - 1\Big)}_{\Psi}.\end{aligned}$$

The aim now is to bound the term in parentheses such that $\Psi = O(\eta^2(pn + i/n))$. To this end, note that $|\Delta|$ is stochastically dominated by a sum of two independent random variables

$$Z \sim \mathrm{Bin}(n, p) + \mathrm{Bin}(i, 1/n)$$

since it is necessary to flip a bit of the optimum or a wrong bit of the current state to change the state. The sum of the two random variables overestimates the change of distance since the two types of flips might cancel each other.

The moment-generating function of the binomial distribution is well known and, for our parameters, given by

$$\begin{aligned}
\mathrm{E}\big(e^{\eta Z}\big) &= \mathrm{E}\Big(e^{\eta \mathrm{Bin}(n,p)}\Big) \cdot \mathrm{E}\Big(e^{\eta \mathrm{Bin}(i,1/n)}\Big) \\
&= (pe^{\eta} + 1 - p)^n \cdot \left(\frac{1}{n} e^{\eta} + \left(1 - \frac{1}{n}\right)\right)^i \\
&\leq \left(p\left(1 + \eta + \eta^2\right) + 1 - p\right)^n \cdot \left(\frac{1}{n}\left(1 + \eta + \eta^2\right) + 1 - \frac{1}{n}\right)^i \\
&= \left((\eta + \eta^2)p + 1\right)^n \left(\frac{\eta + \eta^2}{n} + 1\right)^i
\end{aligned}$$

for $\eta \leq 1$ as $e^x \leq 1 + x + x^2$ for $x \leq 1$. Using $1 + x \leq e^x$, we obtain from this

$$\begin{aligned}
\mathrm{E}\big(e^{\eta Z}\big) &\leq \left(e^{(\eta+\eta^2)p}\right)^n \left(e^{\eta/n + \eta^2/n}\right)^i \\
&= e^{(pn+i/n)\eta + (pn+i/n)\eta^2}
\end{aligned}$$

Introducing $q := pn + i/n \leq 2$, we have

$$\begin{aligned}
\mathrm{E}\big(e^{\eta Z}\big) &\leq 1 + (q\eta + q\eta^2) + (q\eta + q\eta^2)^2 \\
&= 1 + q\eta + q\eta^2 + q^2\eta^2 + 2q^2\eta^3 + q^2\eta^4 \\
&\leq 1 + q\eta + q\eta^2 + 2q\eta^2 + 2q\eta^3 + 2q\eta^4 \\
&\leq 1 + q\eta + 7q\eta^2,
\end{aligned}$$

where we assumed $q\eta + q\eta^2 \leq 1$, which holds for $\eta \leq 1/4$. Since $\mathrm{E}(Z) = q$, we have established

$$\begin{aligned}
\Psi &= \sum_{k=2}^{\infty} \frac{\eta^k \mathrm{E}\big(|\Delta|^k\big)}{k!} \leq \sum_{k=2}^{\infty} \frac{\eta^k \mathrm{E}\big(Z^k\big)}{k!} = \mathrm{E}\big(e^{\eta Z}\big) - \eta \mathrm{E}(Z) - 1 \\
&\leq (1 + q\eta + 7q\eta^2) - q\eta - 1 = 7q\eta^2
\end{aligned}$$

since $\eta \leq 1$.

Plugging this into the above bound on $\mathrm{E}\big(e^{-\lambda\Delta}\big)$, we get

$$\mathrm{E}\big(e^{-\lambda\Delta}\big) \leq 1 - \lambda \mathrm{E}(\Delta) + 7\lambda^2 q$$

for $\lambda \le \eta \le 1/4$. Since $\mathrm{E}(\Delta) \ge \frac{i}{e^2 n} - np$, we choose $\lambda = \frac{1}{21e^2}$ (and $\eta = 1/4$) to get

$$\begin{aligned}
\mathrm{E}\big(e^{-\lambda\Delta}\big) &\le 1 - \lambda\left(\frac{i}{e^2 n} - np\right) + 7\lambda\frac{1}{21e^2}q \\
&= 1 - \lambda\left(\frac{i}{e^2 n} - np\right) + \frac{\lambda}{3e^2}\left(\frac{i}{n} + pn\right) \\
&\le 1 - \lambda\left(\frac{2i}{3e^2 n} - \frac{22pn}{21}\right) =: \beta \le 1 - \frac{6pn}{441e^2}
\end{aligned}$$

where the final inequality holds since $i \ge a = 2e^2 n^2 p$. In addition, we have then

$$\beta \le e^{-6pn/(441e^2)}.$$

So far, we have bounded the moment-generating function of the drift by less than 1. We are left with a bound on

$$\mathrm{E}\Big(e^{-\lambda(a - X^{t+1})} \cdot \mathbb{1}\left\{X^t \le a\right\}\Big).$$

Noting that the exponent is positive for $X^{t+1} \ge a \ge X^t$, we bound the expression by

$$\begin{aligned}
\mathrm{E}\Big(e^{\lambda|X^t - X^{t+1}|} \mid X^t \le a\Big) &\le \mathrm{E}\big(e^{\lambda Z} \mid X_t \le a\big) \\
&\le 1 + 8\lambda q \le 1 + 16\lambda =: D
\end{aligned}$$

using the estimations that bounded $\mathrm{E}\big(e^{\eta Z}\big)$ further above. Applying Lemma 5.6, we get

$$\begin{aligned}
\Pr\big(X^t \ge b\big) &\le \beta^t \cdot e^{\lambda(X^0 - b)} + \frac{1}{1-\beta} D e^{-\lambda\alpha} \\
&\le e^{-\frac{6tpn}{441e^2}} e^{\frac{n-b}{21e^2}} + \frac{441e^2}{6pn}\left(1 + \frac{16}{21e^2}\right) e^{-\frac{\alpha}{21e^2}}.
\end{aligned}$$

As $b \ge b_t = n - \frac{tpn}{7}$, the first term is $e^{-b/(882e^2)}$. Hence, if $b \ge \alpha(n - \frac{tpn}{7})$, it is $e^{-\Omega(\alpha)}$. Assuming $\alpha = \omega(\ln n)$, the second term is $e^{-\Omega(\alpha)} = n^{-\omega(1)}$, which makes the polynomial $\frac{441e^2}{6pn}$ negligible. This proves the first statement from the theorem. The second one follows for $t \ge n\alpha/(pn) = \alpha/p$ since then the first term is clearly $e^{-\Omega(\alpha)}$.

Altogether, for t large enough, we have

$$\Pr\big(X^t \ge b\big) \le e^{-\Omega(\alpha)}.\square$$

Theorem 5.8 shows that after a polynomial amount of time, the distance is very likely to be not by much above the equilibrium state. We can also show a somewhat symmetrical statement, showing that it is very likely to be not by much below the equilibrium state. This is proven in the following theorem.

Theorem 5.9 *Let $r = 2$, $p = o(1/n)$ and $1/p = n^{O(1)}$. Let $(x^t, a^t)_{t\in\mathbb{N}}$ be the sequence of random variables denoting the pair of current search point and current optimum as given by running the (1+1) EA on dynamic* OneMax*. Then, for any $\alpha = \omega(\ln n)$ and all $t \geq 0$*

$$\Pr\big(d(x^t, a^t) \leq n^2 p/(16e^2) - \alpha\big) \leq e^{-\Omega(n)} + e^{-\Omega(\alpha)}.$$

Proof. We essentially follow the analysis from the proof of Theorem 5.8, but focus on a region close to the target where the negative drift is stronger than the positive one. To match the drift theorem, we flip the orientation of the space and let $X^t = n - \text{OneMax}_{a^t}(x^t)$. We recall that there is a drift away from the optimum if $n - X^t < i^*$. More precisely, from the estimations presented at the beginning of this subsection we obtain

$$\begin{aligned}\mathrm{E}\big((n-i) - X^{t+1} \mid X^t = n-i\big) &\\ &= \mathrm{E}\big(\Delta^+(i)\big) - \mathrm{E}\big(\Delta^-(i)\big)\\ &\geq \frac{np}{4e^2} - \frac{2i}{n} \geq \frac{np}{8e^2}\end{aligned}$$

for $i \leq n^2 \frac{p}{16e^2}$. This corresponds to $X^t \geq n - n^2 \frac{p}{16e^2}$. In the following, the aim is to analyze the X^t-process using Lemma 5.6 with $a := n - n^2p/(16e^2)$, $b = a + \alpha$, $x_{\min} := 0$ and the constant function $h(x) := 1$. From this we obtain the trivial potential function $g(x) = x$. We define Δ and bound the moment-generating with the same procedure as in the proof of Theorem 5.8. Then (on $X^t; X^t > a$, which means $i < n^2p/(16e^2)$)

$$\mathrm{E}\big(e^{-\lambda\Delta}\big) \leq 1 - \lambda\mathrm{E}(\Delta) + 7\lambda^2 q.$$

Since $\mathrm{E}(\Delta) \geq \frac{np}{8e^2}$, we choose $\lambda = \frac{1}{112e^2}$ to get

$$\mathrm{E}\big(e^{-\lambda\Delta}\big) \leq 1 - \lambda\left(\frac{np}{17e^2}\right) = 1 - \frac{np}{1904e^4} =: \beta < 1.$$

The bound

$$\mathrm{E}\Big(e^{-\lambda(a-X^{t+1})} \cdot \mathbb{1}\left\{X^t \leq a\right\}\Big) \leq 1 + 16\lambda =: D$$

is the same as in the proof of Theorem 5.8 since it only takes into account the worst-case distribution of Δ. Altogether,

$$\Pr\bigl(X^t \geq b\bigr) \leq e^{-\frac{tpn}{1904e^4}} e^{\frac{X^0-b}{112e^2}} + \frac{1904e^4}{np}\left(1+\frac{16}{112e^2}\right) e^{-\frac{\alpha}{112e^2}}.$$

If $X^0 \leq 2n/3$, which happens with probability $1-2^{-\Omega(n)}$ according to Chernoff bounds, the first term is $e^{-\Omega(n)}$ since $b = n - O(n^2p) = n - o(n)$. The second term is $e^{-\Omega(\alpha)}$ by the same arguments as in the proof of Theorem 5.8. Hence, turning back to the original state space,

$$\Pr\bigl(d(x^t,a^t) \leq n^2p/(16e^2) - \alpha\bigr) \leq e^{-\Omega(n)} + e^{-\Omega(\alpha)}$$

as suggested.

5.3.2 The Case of Large r

In this section we consider large values of r. With the next theorem we show that, even for exponentially large r, the (1+1) EA maintains search points which differ from the optimum only by a constant in each dimension (in expectation)! This holds after an initial mixing phase, the length of which depends linearly on r.

Theorem 5.10 *Let $p = o(1/n)$ and let $(x^t,a^t)_{t\in\mathbb{N}}$ be the sequence of random variables denoting the pair of current search point and current optimum as given by running the (1+1) EA on dynamic* OneMax. *Then there are $k_0, k_1 > 1$ such that, for all $t \geq k_0\, rn^2$,*

$$\forall b \geq 4 : \Pr\bigl(\textsc{OneMax}_{a^t}(x^t) \geq bn\bigr) \leq n2^{-k_1\, b} \tag{5.1}$$

and

$$\mathrm{E}\bigl(\textsc{OneMax}_{a^t}(x^t)\bigr) = O(n). \tag{5.2}$$

In particular, this bound is independent of r. In addition, for all $i \leq n$ and all $t \geq k_0\, rn^2$,

$$\forall b \geq 4 : \Pr\bigl(d(a_i^t,x_i^t) \geq b\bigr) \leq 2^{-k_1\, b}. \tag{5.3}$$

Proof. We start by showing Equation (5.3). Fix a bit position $i \leq n$. We reason with drift on $d^t = d(a_i^t,x_i^t)$ and show that it leads towards 0. Note that this value can change by at most two per iteration (one movement step of the algorithm, one of the optimum). Let some time t be given and suppose $d^t \neq 0$.

Let E be the event that the (1+1) EA keeps all bits other than i unchanged. We bound the expectation of moving in the *wrong* direction conditional on $\overline{E}$ as

$$\mathrm{E}\big(d^{t+1} - d^t \mid \overline{E}\big) = o(1/n),$$

as the optimum might move away with probability $o(1/n)$ but, if the new solution is accepted at all, it is more likely to be accepted if the i was changed in the right direction than when it was changed in the wrong direction. We further bound expectation of moving in the *right* direction conditional on E as

$$\mathrm{E}\big(d^t - d^{t+1} \mid E\big) = \Omega(1/n),$$

as we will not accept a worsening, but do accept an improvement, which will happen with probability $\Omega(1/n)$. Using that $P(E)$ approaches $1/e$ as n approaches infinity, we get a drift of $\Omega(1/n)$ towards the optimum.

With the use of the additive drift theorem (Theorem 5.1), this shows that the first time we have $d^t = 0$ is expected to be at most $k_1\, rn$ iterations, for some k_1 large enough. Let T be the random variable denoting the smallest t with $d^t = 0$. Using concentration bounds for additive drift (Theorem 5.2), we get

$$\forall s \geq 2k_1\, rn : P(T \geq s) \leq \exp\left(-\frac{s}{64n^2}\right).$$

Let $t_0 = 2k_1\, rn^2$. Thus, we do not have $d^{t_0} = 0$ for the first time within the first t steps with probability $2^{-\Omega(r)}$.

We now set up to use Theorem 5.7 to derive bound for straying from the optimum after reaching it for the first time. In the notation of that lemma, our process has a drift of $d = O(1/n)$, a self-loop probability of $p_0 = 1 - O(1/n)$ and a step size of at most $c = 2$. Thus, Lemma 5.7 gives some k such that, for all t,

$$\forall b \geq 4 : \Pr\big(d(a_i^t, x_i^t) \geq b \mid t > T\big) \leq 2^{-k\,b}.$$

We have, for all $t \geq t_0$,

$$\begin{aligned}\Pr\big(X^t \geq b\big) &\leq \Pr\big(d(a_i^t, x_i^t) \geq b \mid t > T\big) + \Pr(t \leq T) \\ &\leq 2^{-k\,b} + 2^{-\Omega(r)}.\end{aligned}$$

This gives the existence of a k_1 as desired, which shows Equation (5.3). Equation (5.1) now follows from the union bound, while Equation (5.2) follows from linearity of expectation (and the trivial bound on the expectation of exponentially decaying random variables).

5.4 Upper Bound on Hitting Time of Target

In this section, we re-prove the upper bound given by [Dro03] in Theorem 5.12 and then extend it to the case of arbitrary r in Theorem 5.14. We start with the case of $r = 2$. Let X^t, $t \geq 0$, be the Hamming distance of the current optimum string and the current search point of the (1+1) EA at time t. Hence, we get a process on $\{0, \ldots, n\}$ with target state 0. We lower bound the parameter $p^-(i)$ and the "drift" $\delta(i)$ in the sense of Theorem 5.4.

Lemma 5.11 *For $i > 0$, $\Pr(X^{t+1} = i - 1 \mid X^t = i) \geq (1-p)^n \frac{i}{en}$, $\Pr(X^{t+1} \geq i + 1 \mid X^t = i) \leq pn$ and $\delta(i) \geq (1-p)^n \frac{i}{en} - pn$.*

Proof. The distance to the optimum decreases if the optimum does not move (probability $(1-p)^n$) and exactly one wrong bit flips (probability $(1-1/n)^{n-1} \frac{i}{n} \geq \frac{i}{en}$), which proves the bound on $\Pr(X^{t+1} = i - 1 \mid X^t = i)$. The distance to the optimum can only increase if the dynamic component flips a bit. By a union bound, the probability is at most pn, which proves the bound on $\Pr(X^{t+1} \geq i + 1 \mid X^t = i)$.

To bound $\mathrm{E}\big((X^{t+1} - i) \cdot \mathbb{1}\{X^{t+1} > i\} \mid X^t = i\big)$, which appears in the definition of $\delta(i)$, we pessimistically assume that each change of the optimum string increases the distance to the current search point. The expected number of bits changed by the dynamic component equals pn, which altogether leads to the bound on $\delta(i)$.

Hereinafter, we work with $p^-(i) = (1-p)^n \frac{i}{en}$ and $p^+(i) = pn$. We get the following polynomial upper bound in the case of bit strings.

Theorem 5.12 *Let $r = 2$ and $p \leq c\frac{\ln n}{n^2}$ for some constant c. Then the expected optimization time of the (1+1) EA on the dynamic* OneMax *is $O(n^{4.8c+2} \ln^2 n)$.*

Proof. By solving the equation $\delta(i) = 0$ with the bounds from Lemma 5.11, we are allowed to set $\kappa := \frac{pen^2}{(1-p)^n}$. Using the assumption on p, $\kappa = (1+o(1))ec \ln n \leq 3c \ln n$ for n large enough. Moreover, we have $p^+(i) + p^-(i) \leq \frac{c \ln n}{n} + \frac{c \ln n}{n} \leq$

$\frac{2c\ln n}{n}$ for $i \le \kappa$. Then

$$\begin{aligned}
&\mathrm{E}\big(T \mid X^0\big) \\
&\le \left(\left(\sum_{k=3c\ln n+1}^{n} \frac{1}{\frac{k}{en} - (c\ln n)/n}\right)\prod_{k=1}^{3c\ln n} \frac{\frac{2c\ln n}{n}}{(1-p)^n\frac{k}{en}}\right) \\
&\qquad + \left(\sum_{k=1}^{3c\ln n} \frac{1}{(1-p)^n\frac{k}{en}}\prod_{j=1}^{k-1} \frac{\frac{2c\ln n}{n}}{(1-p)^n\frac{j}{en}}\right) \\
&\le \sum_{k=3c\ln n+1}^{n} \frac{1}{k/(10n\ln n)} \cdot P + 3c\ln n \cdot P,
\end{aligned}$$

where $P := en(1-p)^{-n}\prod_{k=2}^{3c\ln n}\frac{2ce\ln n}{k}$. If n is not too small, we have

$$\mathrm{E}\big(T \mid X^0\big) \le (10n\ln^2 n) + 3c\ln n)P \le (11n\ln^2 n)P.$$

Now,

$$\begin{aligned}
P &\le en\left(1 - \frac{c\ln n}{n^2}\right)^{-n}\frac{(2ce\ln n)^{3c\ln n}}{(3c\ln n)!} \\
&\le en(1+o(1))\left(\frac{2ce^2}{3c}\right)^{3c\ln n} \\
&\le 2ene^{3c\ln(2e^2/3)\ln n} \le 2en^{4.8c+1}
\end{aligned}$$

using $k! \ge (k/e)^k$. Altogether,

$$\mathrm{E}\big(T \mid X^0\big) \le 22en^{4.8c+2}\ln^2 n$$

for n large enough.

For comparison, [Dro03] proves the upper bound $\mathrm{E}(T) = O(n^{4ce/\ln(2)+1}\ln n)$, i. e., the exponent is almost $12c$. Hence, state-of-the-art drift analysis yields more precise results, is more versatile and leads to cleaner and shorter proofs than the previous analysis by [Dro03].

It is not to difficult to generalize Theorem 5.12 to arbitrary r if we replace the prerequisite on p by $p \le c\frac{\ln n}{rn^2}$. Basically, a factor of r is lost if we work under the worst case assumption that wrong positions have distance r from the optimum, resulting in only $\textsc{OneMax}_{a^t}(x^t)/r$ wrong positions. To increase the regime of polynomial hitting times, we have to prove this worst case to be unlikely. Fortunately, the anytime analysis from Theorem 5.10 can be used here to show that we lose a factor of at most $O(\log n)$ regardless of r. To this end, we will use the following lemma, which immediately follows from Theorem 5.10.

Lemma 5.13 *Let $p = o(1/n)$. Then there are constants $a > 0, b > 1$ such that for any $i \in [n]$, $j \in \mathbb{N}$ and all $t \geq rn^2 \ln n$ it holds* $\Pr(d(a_i^t, x_i^t) \geq j) \leq ab^{-j}$.

We now state the theorem concerned with polynomial hitting times for large r. To ease the statement, we only consider polynomial-sized r.

Theorem 5.14 *Let $r \leq n^k$ for some constant k and $p \leq \frac{c \ln n}{\min\{r, \ln n\} n^2}$ for some sufficiently small constant c (possibly depending on k). Then the expected optimization time of the (1+1) EA on the dynamic* OneMax *is polynomial in n.*

Proof. Let $t^* = rn^2 \ln n$. Pessimistically ignoring the case that the optimum is hit in less than t^* steps, we apply Lemma 5.13. Choosing $j^* = \ln(an^2 t^*)/(\ln b) = O(\log n)$, we obtain that $\Pr(d(a_i^t, x_i^t) \geq j^*) \leq 1/(t^* n^2)$ for any j and $t \geq t^*$. By a union bound, the probability that for all $i \in [n]$ we have $d(a_i^t, x_i^t) \leq j^*$ is $1 - O(1/(t^* n))$. From now on, we assume this to hold in a phase of length t^*, starting from time t^* up to time $2t^* - 1$. Again by a union bound, the probability that within t^* steps all positions have distance at most j^* from the target is still $1 - O(1/n) - n^{-\omega(1)} = 1 - o(1)$. If $j^* \geq r$, the assumption holds trivially, i. e., with probability 1.

Under our assumption, we conduct a drift analysis with respect to $X^t := \text{OneMax}_{a_t}(x_t)$. Similarly to Lemma 5.11, $\Pr(X^{t+1} \geq i + 1 \mid X^t = i) \leq pn$ since each changing position of the target increases the distance by at most 1 and also

$$\mathrm{E}\big((X^t - X^{t+1}) \cdot \mathbb{1}\left\{X^{t+1} > X^t\right\} \mid X^t = i\big) \leq pn.$$

If $X^t = i$, then there are at least $\frac{i}{\min\{r, j^*\}}$ wrong positions, hence $p^-(i) = \Pr(X^{t+1} = i - 1 \mid X^t = i) \geq (1-p)^n \frac{i}{\min\{r, j^*\} en}$ and

$$\begin{aligned}\delta(i) &\geq (1-p)^n \frac{i}{\min\{r, j^*\} en} - pn \\ &\geq (1-p)^n \frac{i}{\min\{r, j^*\} en} - \frac{c \ln n}{\min\{r, \ln n\} n}.\end{aligned}$$

By our assumptions, $j^* \leq kc_1 \ln n$ for some constant c_1 (depending on a and b) for large enough n. Using our assumption on p, we get that $\delta(i) \geq 0$ for $i \geq c_3 \ln n$, where c_3 is a constant such that $c_3 = cc_1 k + c_2$ for another constant c_2. Hence, we work with $\kappa := c_3 \ln n$. If c_2 is chosen appropriately, then we also have $p^+(i) + p^-(i) \leq \frac{c_3 \ln n}{2n}$ for $i \leq \kappa$. Similarly as in the proof of Theorem 5.12, we get for small enough c that

$$\mathrm{E}\big(T \mid X^0\big) \leq \sum_{\ell = c_3 \ln n}^{j^* n} \frac{1}{\ell/(2ekc_1 n \ln n)} \cdot P + c_3 \ln n \cdot P,$$

where

$$P := en \min\{r, j^*\}(1-p)^{-n} \prod_{\ell=2}^{c_3 \ln n} \frac{(c_3/2)\ln n}{\ell}.$$

If n is not too small, $\mathrm{E}\big(T \mid X^0\big) \leq 3ekc_1c_3n(\ln^2 n)P$, so we are left with an estimate for P. We get

$$P \leq O(n \ln^2 n) \left(\frac{c_3 e \ln n}{2c_3 \ln n}\right)^{c_3 \ln n} \leq n^{c_3}$$

for n large enough. Altogether,

$$\mathrm{E}\big(T \mid X^0\big) \leq n^{cc_1k+c_2}.$$

If n is large enough and c is sufficiently small but still constant, then $n^{cc_1k+c_2} \leq t^*/2$. Hence, by Markov's inequality, the probability that a phase of t^* steps is successful, i. e., the optimum is hit, is at least $1/2$; still conditioning on maximum distance j^* for all positions. By the considerations from above, the unconditional probability of a successful phase is at least $1/2 - o(1)$. In case of a failure, we consider the subsequent phase of t^* steps. The expected number of phases is at most $2 + o(1)$, hence the overall expected first hitting time of the target is at most $t^* + (2 + o(1))t^* = (3 + o(1))t^*$, i. e., polynomial.

We conjecture that the assumption on p in Theorem 5.14 can be replaced by $p \leq \frac{c \ln n}{n^2}$, i. e., that the same regime for polynomial first hitting time holds regardless of r. However, we cannot prove this at the moment since the processes describing the distance from the target for different positions are not independent.

5.5 Lower Bound on Hitting Time of Target

When the mutation probability applied to the optimum is asymptotically larger than $\log n/n^2$, [Dro03] shows that the first hitting time of (1+1) EA on OneMax for $r = 2$ is polynomial only with super-polynomially small probability. We reprove this result for any $r \geq 2$ and $p \leq 1/n$ using drift analysis.

Theorem 5.15 *With $p \in \omega(\log n/n^2)$ and $p \leq 1/n$, the first hitting time of the (1+1) EA on the dynamic* OneMax *for any $r \geq 2$ is polynomial only with super-polynomially small probability.*

Proof. To prove the result, we let X^t be the current solution of the EA and apply Theorem 5.3 using potential function $g(X^t) = \sum_{i=1}^{n} [a_i^t \neq x_i^t]$, i. e., the number of characters the individual and the optimum differ by at time t.

Consider the effects of mutating the optimum and the mutation/selection step separately. When $g(X^t) \leq n/2$, the effect of optimum mutation on $g(X^t)$ non-matching characters is countered by the effect of mutation on $g(X^t)$ matching characters, leaving $n - 2g(X^t)$ matching characters which cause a drift away from the optimum:

$$\mathrm{E}\big(g(X^{t+1}) - g(X^t) \mid S^t\big) \geq (n - 2g(X^t)) \cdot p,$$

where S^t is the event that no mutation occurs during the mutation/selection step of iteration t.

For the mutation/selection step, the expected increase in the number of matching characters is at most the number of mutated non-matching characters. We can consider the mutation/selection occurring after the optimum is mutated, so the number of non-matching characters is in expectation increased by at most 1 for $p \leq 1/n$, leading to a combined drift of:

$$\mathrm{E}\big(g(X^{t+1}) - g(X^t)\big) \geq (n - 2g(X^t)) \cdot p - (g(X^t) + 1) \cdot 1/n.$$

Let $p = \alpha(n) \cdot \log n/n^2 \leq 1/n$, where $\alpha(n) \in \omega(1) \leq n/\log n$; limiting $g(X^t) < b = \alpha(n)^c \log n$, where $c < 1$ is a constant, reveals a drift away from the optimum:

$$\begin{aligned}
&\mathrm{E}\big(g(X^{t+1}) - g(X^t) \mid g(X^t) < b\big) \\
&\quad \geq (n - 2g(X^t)) \cdot p - (g(X^t) + 1) \cdot 1/n \\
&\quad \geq \frac{\alpha(n) \log n}{n} - \frac{\alpha(n)^c \log n}{n} \left(\frac{2\alpha(n) \log n}{n} + 1 \right) - \frac{1}{n} \\
&\quad \in \Omega\left(\frac{\alpha(n) \log n}{n} \right).
\end{aligned}$$

With low p and $g(X^t)$, a large number of iterations might not alter the value of $g(X^t)$ (causing a "self-loop"). The probability p_0 of an iteration resulting in a self-loop can be bounded by considering the probability that none of the characters of the optimum mutate, and none of the non-matching characters in the current individual mutate:

$$\begin{aligned}
p_0 &= \Pr\big(g(X^{t+1}) = g(X^t) \mid X^t\big) \\
&\geq (1-p)^n \cdot (1 - 1/n)^{g(X^t)} \\
&\geq 1 - \frac{n\alpha(n) \log n}{n^2} - \frac{\alpha(n)^c \log n}{n} \\
(1 - p_0) &\in O\left(\frac{\alpha(n) \log n}{n} \right).
\end{aligned}$$

Thus, there exists a constant $\varepsilon > 0$ satisfying the first requirement of Theorem 5.3. We then need to bound the probabilities of $g(X^t)$ changing significantly in a single iteration. Throughout the following, let M be the event that a self-loop does not occur, i. e., $g(X^{t+1}) \neq g(X^t)$.

Let $c_1 = n - g(X^t) \leq n$ be the number of matching characters in the optimum and the current individual (for which $d(a_i, x_i) = 0$). We note that mutating such a character in the optimum would increase $g(X^{t+1})$ unless it is also mutated in the current individual; let C_1 be the number of such mutations that occur:

$$\begin{aligned}
\Pr(M) &\geq c_1 \cdot p(1-p)^{n-1} \cdot (1-1/n)^n \\
&\geq c_1 p/4e \\
\Pr(C_1 \geq j) &\leq \binom{c_1}{j} p^j \\
\Pr(C_1 \geq j \mid M) &\leq \frac{(c_1 p)^{j-1} 4e}{2^{j-1}} \leq \frac{8e}{2^j}
\end{aligned}$$

for $n \geq 2$.

Let $c_2 \leq g(X^t) \leq n$ be the number of characters for which $d(a_i, x_i) \geq 2$, i. e., those that would not transform into matching characters even if improved by mutation; notably, $c_2 = 0$ if $r = 2$. When both, a matching character mutation, and a mutation improving such a character occurs in the current individual, $g(X^{t+1})$ increases without reducing fitness, allowing the mutated individual to be accepted; let C_2 be the number of such mutations that occur:

$$\begin{aligned}
\Pr(M) &\geq c_1 c_2/(8en^2) \\
\Pr(C_2 \geq j) &\leq \binom{c_1}{j}\binom{c_2}{j} n^{-2j} \\
\Pr(C_2 \geq j \mid M) &\leq \frac{(c_1 c_2)^j}{2^{2(j-1)} n^{2j}} \frac{8en^2}{c_1 c_2} \\
&\leq \frac{(c_1 c_2)^{j-1}}{n^{2(j-1)}} \frac{32e}{4^j} < \frac{32e}{4^j}
\end{aligned}$$

as $c_1 c_2 \leq n^2$.

The increase in potential value is at most the sum of these two effects, and hence:

$$\Pr\big(g(X^{t+1}) - g(X^t) \geq j \mid M\big) \leq \Pr(C_1 + C_2 \geq j \mid M).$$

By Lemma 5.16, there exists a choice of $r(\ell)$ and δ that satisfies the second condition of Theorem 5.3 for jumps away from the optimum.

For jumps toward the optimum, let k_1 be the number of characters for which $d(a_i, x_i) = 1$, i. e., those that can be corrected by a mutation in either the optimum or the current individual; let J_1 be the number of such characters corrected in a given iteration. Proceeding as before,

$$\begin{aligned}\Pr(J_1 \geq j \mid M) &\leq \binom{2k_1}{j} \frac{1}{n^j} \frac{8en}{k_1} \\ &\leq \frac{(2k_1)^{j-1} 8e}{n^{j-1} 3^{j-2}} \leq \frac{36e}{1.5^j},\end{aligned}$$

noting that if $r > 2$, a mutation in a specific direction is required while considering $P(M)$, while a mutation either direction is acceptable to upper-bound $P(J_1 \geq j)$.

Furthermore, let k_2 be the number of characters for which $d(a_i, x_i) = 2$, i. e., those that can match if they are mutated appropriately in both the optimum and the current individual; let J_2 be the number of such characters corrected in a given iteration. Similarly,

$$\begin{aligned}\Pr(J_2 \geq j \mid M) &\leq \binom{k_2}{j} \left(\frac{p}{4n}\right)^j \frac{16en}{k_2 p} \\ &\leq \frac{{k_2}^{j-1} 4e}{8^{j-1}} \left(\frac{p}{n}\right)^{j-1} \leq \frac{32e}{8^j}\end{aligned}$$

as $k_2 \leq n \leq 1/p$.

The reduction in potential value is at most the sum of these two effects, and so:

$$\Pr\big(g(X^{t+1}) - g(X^t) \leq -j \mid M\big) \leq \Pr(J_1 + J_2 \geq j \mid M).$$

Per Lemma 5.16, there exists a choice of $r(\ell)$ and δ that satisfies the second condition of Theorem 5.3 for jumps toward the optimum.

Thus, there exists a choice of $r(\ell)$ and δ that satisfies the second requirement of Theorem 5.3 both for jumps away from and jumps toward the optimum.

Finally, we note that the probability of a randomly initialized character matching the optimum is $1/r \leq 1/2$. Using Chernoff's inequality, the probability that more than $3n/4$ characters are initialized correctly is at most $e^{-n/12}$, and therefore $g(X^0) \geq n/4 > b$ with high probability.

By applying Theorem 5.3 with $g(X^t)$, $b = \alpha(n)^c \log n$ and $a = 0$, and hence $\ell = \omega(\log n)$, we can conclude that if $p \in \omega(\log n/n^2) \leq 1/n$, the (1+1) EA finds the optimum in polynomial time with only super-polynomially small probability.

While proving that large jumps are exponentially unlikely even after removing self-loops from the process, we used the following lemma to combine upper bounds for different kinds of jumps.

Lemma 5.16 *Let $J = J_1 + J_2$; if there exist constants $r_1, r_2 \geq 1$, and $d_1, d_2 > 1$, s.t. for some event E,*

$$\Pr(J_1 \geq j \mid E) \leq r_1/{d_1}^j$$
$$\Pr(J_2 \geq j \mid E) \leq r_2/{d_2}^j$$

and it holds that $\Pr(J_2 \geq j_2 \mid E, J_1 \geq j_1) \leq \Pr(J_2 \geq j_2 \mid E)$, then there also exist constants r, $d > 1$, s.t.

$$\Pr(J \geq j \mid E) \leq r/d^j.$$

Proof. Let $r_* = \max(r_1, r_2)$ and $d_* = \min(d_1, d_2)$; given the conditions, it is the case that:

$$\begin{aligned}
&\Pr(J \geq j \mid E) \\
&\quad \leq \sum_{i=0}^{j} \Pr(J_1 \geq i \mid E)\Pr(J_2 \geq j - i \mid E, J_1 \geq j_1) \\
&\quad \leq \sum_{i=0}^{j} \Pr(J_1 \geq i \mid E)\Pr(J_2 \geq j - i \mid E) \\
&\quad \leq r_*^2 d_*^{-j}(j+1).
\end{aligned}$$

We note that $(j+1)d_*^{-j} \leq \sqrt{d_*}^{-j}$ for $j \geq 16/(\ln^2 d_*)$. It is possible to pick a constant $c = \sqrt{d_*}^{16/(\ln^2 d_*)}$, ensuring that $c/\sqrt{d_*}^j \geq 1$ for $j \leq 16/(\ln^2 d_*)$, which proves the lemma with $r = r_* c$ and $d = \sqrt{d_*}$.

5.6 Conclusion

In this paper we revisited the setting of dynamic OneMax as introduced by [Dro03], where the optimum moves by flipping the bit of each position with some fixed probability p.

We showed that his results, both the upper and the lower bound, extend to versions of dynamic OneMax where each dimension has r different possible values. By using modern drift analysis, the proof is shorter and more elegant.

Furthermore, we made an analysis of how far from the optimum the (1+1) EA strays after getting close for the first time. For the case of bit strings, this value is concentrated around $\Theta(pn^2)$ (for $p = o(1/n)$), which shows that the optimum is very elusive unless p is small. On the other hand, we showed that only the *dimension*, and not the *size*, of the search space has an impact on the ability of the (1+1) EA to track good solutions. We did this by considering search spaces with r possible values in each dimension, and saw that r does not influence the resulting bounds, i. e., the distance is bounded by a constant in expectation in each dimension if r is large (see Theorem 5.10).

We believe that the methods we used, especially the statements about the occupation probabilities as given in Lemma 13 of [LW15a] or in our Theorem 5.7, will be beneficial in many more settings, especially those aiming at an anytime analysis for dynamic problems.

Acknowledgments

The authors would like to thank Benjamin and Carola Doerr for interesting discussions on the topic of this paper. The anonymous reviewers of the preconference reviews made many valuable suggestions which improved the paper. Furthermore, Golnaz Badkobeh, Per Kristian Lehre and Dirk Sudholt gave many useful suggestions as part of the FOGA post-conference reviews. Financial support from the Danish Council for Independent Research (grant no. DFF–4002-00542) is gratefully acknowledged.

Chapter 6

The Impact of Migration Topology on the Runtime of Island Models in Dynamic Optimization

Andrei Lissovoi Carsten Witt

DTU Compute, Technical University of Denmark

We introduce a simplified island model with behaviour similar to the λ (1+1) islands optimizing the Maze fitness function considered in our previous work [LW15b], and investigate the effects of the migration topology on the ability of the simplified island model to track the optimum of a dynamic fitness function. More specifically, we prove that there exist choices of model parameters for which using a unidirectional ring as the migration topology allows the model to track the oscillating optimum through n Maze-like phases with high probability, while a using a complete graph as the migration topology results in the island model losing track of the optimum with overwhelming probability. This serves to illustrate that a less-dense migration topology may be useful when optimizing dynamic functions with oscillating behaviour, and requires less problem-specific knowledge to determine when migration may be allowed to occur.

6.1 Introduction

Optimization problems are often dynamic in nature, as the environment in which they have to be solved often changes with the passing of time. Nature-inspired algorithms are based on approaches to solving optimization problems observed in nature, and we might therefore hope that they would also provide a reasonable solution to copying with dynamic changes in optimization problems.

In a dynamic optimization problem, the optimum is allowed to move in the search space over time, as conditions of the problem change. The goal of the evolutionary algorithm is then not only to locate the optimum once, as in the case of static optimization problems, but also be able to track the optimum as it moves, maintaining good solutions over time.

The Maze fitness function, introduced in [KM12], is an artificial fitness function consisting of $n+1$ long oscillating phases, over the course of which the optimum slowly shifts from the all-ones bit string to the all-zeroes bit string, while oscillating between two specific solutions during each phase. In [KM12], it is shown that a simple (1+1) EA is not able to track the oscillating optimum through all n phases. In subsequent work [LW15a, LW15b], we have considered how various diversity mechanisms impact the ability of evolutionary algorithms to track the optimum of this function, observing that an island model can provide the necessary diversity as long as migration on a complete migration topology is carefully managed, and does not occur too close to a Maze phase transition – conditions which require somewhat specific knowledge of the fitness function, which may not be available for other problems.

In this paper, we investigate whether using a less-dense migration topology, such as a unidirectional ring, can be beneficial, and allow some of the requirements on when migration is allowed to occur to be relaxed. Intuitively, changing the migration topology weakens the negative effect of migration on population diversity, and may allow good solutions to survive migration occurring at inopportune times.

We base our analysis on a simplified version of the island model studied in [LW15b], which incorporates the major elements of the original setting: an oscillating fitness function, islands performing independent mutation/selection steps, and the effect of Maze phase transitions on the islands' ability to track the optimum based on their current-best individuals at the time of the transition. The simplified model incorporates more randomization, as both the oscillating pattern and migration are made non-deterministic, which both simplifies the analysis, and disallows some of the more artificial solutions possible in the origi-

nal model, such as only performing migrations on iterations that assign a higher fitness value to the desirable solution.

Using this simplified model, we prove that the unidirectional ring migration topology allows the island model to track the optimum of the dynamic fitness functions in some settings where the complete migration topology does not.

This paper is structured as follows. In the next section, we introduce the simplified island model, and highlight its key differences by comparing to the setting of [LW15b]. Section 6.3 introduces the drift theorems used in subsequent proofs. Sections 6.4.1 and 6.4.2 consider the case of migration occurring in every iteration, the former proving that a complete migration topology leads to a failure to track the optimum, while the latter proves that switching to the unidirectional ring topology allows tracking the optimum with high probability. We finish with some conclusions, as well as a discussion of further possibilities for analysis.

6.2 The Simplified Island Model

In order to analyze the impact of migration topology on the island model behaviour, and remove some of the artifacts arising from the Maze fitness function, we will construct a somewhat simplified model of the optimization algorithm, while maintaining similarities to λ islands using (1+1) EAs to optimize Maze. The simplified model is shown as Algorithm 6.1 below; this section discusses the key differences.

Some changes have been made to the model of the Maze fitness function: instead of a fully deterministic oscillation defined on bit strings, Algorithm 6.1 distinguishes between three states, OPT, ALT, and DEAD, and non-deterministically selects which of OPT and ALT has a higher fitness value, independently favouring OPT over ALT in each iteration with probability p_{OPT}. When a Maze phase transition occurs, all islands holding an OPT individual transition to holding an ALT individual, while all other islands transition to holding a DEAD individual. The OPT, ALT and DEAD individuals thus correspond to having OPT, ALT, and OneMax-valued individuals in the original Maze, where the OPT individual in each phase becomes the ALT individual of the next phase, while the ALT individual becomes a OneMax-valued individual following a phase transition.

Each island behaves like a simplified (1+1) EA, maintaining a current-best solution $x_i^*(t)$ by applying mutation and selection. The mutation operator is simplified to allow construction of OPT from ALT (or vice versa) with probability p_{mut}, while preventing construction of either OPT or ALT from DEAD. With

an appropriate value of p_{mut} based on a probability of a specific single-bit mutation occurring, this is a pessimistic model of (1+1) EA's behaviour on MAZE, where for $\Omega(n)$ phases, beginning a phase with a OneMax valued individual (i.e. a DEAD individual in the simplified model) would cause the (1+1) EA to revert to optimizing OneMax with at least constant probability, leaving it with an overwhelmingly small probability of finding the oscillating optimum again.

Finally, migration is made non-deterministic by allowing it to occur in each iteration independently at random with probability p_{mut}. This essentially prevents the algorithm from being able to ensure that it performs migration only on OPT-favouring iterations.

Algorithm 6.1 Simplified island model for Maze on $G = (V, \mathcal{A})$.

Set $x_i^*(1) = \text{OPT}$ for all $i \in V$.
for $t \leftarrow 1, 2, \ldots$ **do**
  With probability p_{OPT}, $y^+ \leftarrow$ OPT, $y^- \leftarrow$ ALT; otherwise, vice versa.
  $M \leftarrow \text{bernoulli}(p_{\text{mig}})$
  for all $i \in V$ **in parallel do**
    Let $N = \{x_i^*(t)\} \cup \{x_j^*(t) \mid M = 1 \wedge (j, i) \in \mathcal{A}\}$

$$x_i' \leftarrow \begin{cases} y^+ & \text{if } y^+ \in N \\ y^- & \text{if } y^- \in N \\ \text{DEAD} & \text{otherwise} \end{cases}$$

$$x_i^*(t+1) \leftarrow \begin{cases} y^+ & \text{with probability } p_{\text{mut}} \text{ if } x_i' = y^- \\ x_i' & \text{otherwise} \end{cases}$$

  if $t \bmod t_0 = 0$ **then** ▷ A phase transition occurs
    for all $i \in V$ **do**

$$x_i^*(t+1) \leftarrow \begin{cases} ALT & \text{if } x_i^*(t+1) = OPT \\ DEAD & \text{otherwise} \end{cases}$$

Thus, the parameters of the simplified model are:

- n, number of phases being considered,
- t_0, number of iterations in each phase,
- p_{OPT}, probability of OPT having a higher fitness value than ALT,
- λ, number of islands,
- p_{mut}, probability of constructing OPT/ALT from ALT/OPT,
- p_{mig}, probability of migration occurring,
- $G = (V, \mathcal{A})$, a directed graph specifying migration topology,

where the impact of G on the algorithm's ability to track the oscillating OPT/ALT optimum (i.e. have at least one island remain in the OPT/ALT state after n phases) is of interest to us.

We note that the following choice of parameters yields a setting similar to the original MAZE considered in [KM12]: $t_0 = n^3$, $p_{\text{OPT}} = 2/3$, $\lambda = \Omega(\log n)$, $p_{\text{mut}} = \Theta\left(1/(ne)\right)$, $p_{\text{mig}} = 1/\tau$ (where τ is the original interval between migrations), and $G = K_\lambda$.

6.3 Drift theorems

To derive our theoretical results, we use the following two drift theorems, describing the expectation of first-hitting time in the presence of additive drift, and a tail bounds on the first-hitting time in the presence of negative drift.

THEOREM 6.1 (ADDITIVE DRIFT, EXPECTED TIME, [LW14A, HY01])

Let $(X^t)_{t\geq 0}$, be a stochastic process over a bounded state space $S \subseteq \mathbb{R}_0^+$, and let $T_0 := \min\{t \geq 0 \colon X^t = 0\}$ denote the first hitting time of 0 and assume that both $\mathrm{E}\big(X^0\big)$ and $\mathrm{E}\big(T_0 \mid X^0\big)$ are finite. Then, if

$$\mathrm{E}\big(X^t - X^{t+1} \mid \mathcal{F}_t; X^t > 0\big) \geq \varepsilon,$$

it holds that $\mathrm{E}\big(T_0 \mid X^0\big) \leq \frac{X^0}{\varepsilon}$.

THEOREM 6.2 (NEGATIVE DRIFT, [OW11, OW12]) *Let $(X^t)_{t\geq 0}$, be a Markov process over a state space S. Suppose there exist an interval $[a, b] \subseteq \mathbb{R}_0^+$, two constants $\delta, \varepsilon > 0$, a function $r(\ell)$ satisfying $1 \leq r(\ell) = o(\ell/\log(\ell))$, and a potential function $g \colon S \to \mathbb{R}_0^+$, such that for all $t \geq 0$, the following two conditions hold:*

(i) $\mathrm{E}(\Delta^t \mid X^t; a < g(X^t) < b) \geq \varepsilon$,

(ii) $\Pr(|\Delta^t| \geq j \mid X^t; a < g(X^t)) \leq \frac{r(\ell)}{(1+\delta)^j}$ *for* $j \in \mathbb{N}_0$,

where $\Delta^t = g(X^{t+1}) - g(X^t)$.

Then there is a constant $c^* > 0$ such that for $T^* := \min\left\{t \geq 0 : g(X^t) \leq a \mid g(X^0) \geq b\right\}$ it holds

$$\Pr\left(T^* \leq 2^{c^* \ell / r(\ell)}\right) = 2^{-\Omega(\ell/r(\ell))}.$$

6.4 Continuous migration

As a simple case, consider setting $p_{\text{mig}} = 1$, i. e. requiring migration to occur in every iteration. We consider the impact of two extreme choices of G in this setting: a complete graph K_λ, and a λ-vertex unidirectional ring.

6.4.1 Complete migration topology

We first prove that using the complete migration topology with continuous migration results in the simplified model being unable to track the optimum of the MAZE through all n phases.

THEOREM 6.3 *When $p_{\text{mig}} = 1$, and $G = K_\lambda$, i. e. the case of using the complete migration topology and migration occurring in every iteration, and $t_0 = \Omega(n)$, $p_{\text{mut}} = 1/(en)$, $\lambda = O(n)$, and a constant $0 < p_{\text{OPT}} < 1$, the probability that all islands are in the DEAD state after $n \cdot t_0$ iterations is overwhelmingly large.*

PROOF. We note that the probability at least one mutation occurs during a phase with at least a constant probability:

$$1 - (1 - p_{\text{mut}})^{\lambda t_0} \geq 1 - e^{-c}$$

while the probability that mutation occurs in a single iteration is at most a constant:

$$(1 - p_{\text{mut}})^{\lambda} \leq e^{-c'}$$

where $c > 0$ and $c' > 0$ are constants.

Thus, with at least a constant probability, the last mutation in a phase occurs at least one iteration before the phase transition. With probability $(1 - p_{\text{OPT}})^2$, i. e. at least a constant probability, both the iteration when the last mutation occurs, and the iteration immediately following it favour ALT over OPT; thus, if all islands were in the OPT state, the mutation would produce an ALT individual which would migrate to all islands, while if at least one island was in the ALT state, its original individual would migrate to all other islands. As no further mutation occurs before the phase transition, we conclude that each phase has at least a constant probability of ending with all islands having the ALT individual, and thus losing track of the oscillating optimum following the next phase transition.

Thus, if each of n phases has at least a constant probability of failing $p_f > 0$, the probability that all n phases complete successfully is at most $1-(1-p_f)^n = 1-2^{-\Omega(n)}$, i. e. overwhelmingly small. □

It is worth noting that this proof approach is flexible enough to be adapted to settings where migration occurs less often – as long as with at least constant probability, the number of iterations in which mutations occur following the final migration in a phase can be bounded by a constant.

6.4.2 Unidirectional ring topology

For the other extreme, suppose that G is minimally connected, i. e. a unidirectional ring of λ vertices and λ arcs. This reduces the effect that continuous migration has on the island memory, making it impossible for a single migration to propagate an undesirable individual to all islands. In this section, we will prove that the simplified island model is able to track the oscillating optimum for the full n phases.

Theorem 6.4 *When G is an λ-vertex unidirectional ring, $p_{\text{mig}} = 1$, and $t_0 = \Omega\left(n^2\right)$, $p_{\text{mut}} = 1/(en)$, $\lambda = \Omega\left(\sqrt{n}\right)$, and $p_{\text{OPT}} = 2/3$, the simplified island model is able to track the oscillating optimum for at least n phases with high probability.*

We will prove this by showing that as long as each phase begins with at least one island still tracking the optimum, the phase will end with at least one island having $x_i^*(t) = \text{OPT}$. Notably, any constant $p_{\text{OPT}} > 1/2$ is sufficient, and $p_{\text{OPT}} = 2/3$ was chosen to correspond to the oscillation pattern of the original Maze.

Lemma 6.5 *Under the preconditions of Theorem 6.4, if a phase begins with at least one island having $x_i^*(t) \neq \text{DEAD}$, with high probability, there will exist an iteration t' before the phase ends such that all islands will have $x_i^*(t') = \text{OPT}$ with high probability.*

Proof. We note that after at most λ iterations, all islands will have a non-DEAD individual as their current-best solution – in the worst case, λ iterations are enough to migrate the non-DEAD individual from the single surviving island to all other islands, with fewer iterations being required if there is more than one surviving island.

Let t' be the iteration during which no islands have a DEAD individual as their current-best solution, and consider the drift in X_t, the number of islands having $x_i^*(t+t') = \text{ALT}$. Let $S_t = |\{(u,v) \in \mathcal{A} \mid x_u^*(t+t') = \text{OPT} \wedge x_v^*(t+t') = \text{ALT}\}|$, i. e. the number of segments in the unidirectional ring of the migration topology which are composed of islands having OPT as their current-best solution.

$$\begin{aligned} E(X_t - X_{t+1} \mid X_t < \lambda, S_t) &\geq p_{\text{OPT}} \left(p_{\text{mig}} S_t + p_{\text{mut}}(\lambda - X_t)\right) - \\ &\quad (1 - p_{\text{OPT}}) \left(p_{\text{mut}} X_t + p_{\text{mig}} S_t\right) \\ &= p_{\text{OPT}} \left(2p_{\text{mig}} S_t + p_{\text{mut}} \lambda\right) - p_{\text{mut}} X_t - p_{\text{mig}} S_t \\ &> 2p_{\text{OPT}} S_t - S_t - p_{\text{mut}} X_t \\ &> \Omega(1) - p_{\text{mut}} \lambda = \Omega(1) \end{aligned}$$

and

$$E(X_t - X_{t+1} \mid X_t = \lambda) = p_{\text{OPT}} p_{\text{mut}} \lambda = \frac{2}{3e\sqrt{n}}.$$

Applying the additive drift theorem, the expected first hitting time $T = \min\{t : X_t = 0\} = O(\sqrt{n}\lambda) = O(n)$. As this is much shorter than the phase length $t_0 = \Omega(n^2)$, we can conclude that $X_t = 0$ is hit during the phase with high probability, and hence at least at some point during the phase, all islands have OPT as their current-best solution. □

We then need to show that it is not likely that the island model will manage to replace OPT with ALT on all islands during the remainder of the current phase.

Lemma 6.6 *Under the preconditions of Theorem 6.4, if there occurs an iteration where $x_i^*(t) = \text{OPT}$ for all islands i, with high probability, an iteration where for all islands i, $x_i^*(t) \neq \text{OPT}$ does not occur before the next phase transition.*

Proof. We note that it is difficult to apply the Negative Drift Theorem directly in this setting, as the drift depends on S_t: if there are many OPT/ALT boundaries in the migration topology, migration may cause drastic changes in the number of islands having OPT as their current-best individual. Instead, our strategy is to bound the number of islands having ALT as their current-best individual by considering the effects of each OPT-to-ALT mutation that occurs in isolation, i. e. as if it was the only ALT segment around at any specific time. Then, by bounding the maximum length such a segment may reach, the number of iterations such a segment survives, and the rate at which such mutations are accepted, we arrive at an upper-bound on the total number of islands having ALT as their current-best solution at any one time.

When considered in isolation, an OPT-to-ALT mutation creates an ALT segment with initial length 1 in the migration topology. We only consider its length to be modified by migration: increased by 1 if migration occurs during an ALT-favouring iteration, and decreased by 1 if migration occurs during an OPT-favouring iteration; any further OPT-to-ALT mutations would be treated as separate isolated segments, and pessimistically, no ALT-to-OPT mutations occur within the ALT segment. It is straightforward to apply the Negative Drift Theorem to bound the maximum length of such an isolated segment: it decreases by a constant in expectation, as $p_{\mathrm{OPT}} = 2/3$, and the maximal possible change is by 1 in either direction. Thus, per the negative drift theorem (Theorem 6.2), the probability that the length of an ALT segment exceeds $\ell = \sqrt{\lambda}$ within $t_0 = O(n^2) = O(\ell^8)$ iterations is no more than $2^{-\Omega(n^{1/4})}$.

To bound the maximum number of iterations before a freshly-created ALT segment is reduced to length 0 by migration, use a tail bound on the binomial distribution: the segment is guaranteed to be reduced to length 0 if in $2k$ iterations, more than k favour OPT. Let X_{2k} be the number of iterations that favour OPT of $2k$ iterations:

$$P(X_{2k} \leq k) \leq \exp\left(-2\frac{(2kp_{\mathrm{OPT}} - k)^2}{k}\right) = e^{-2k/9}$$

using Hoeffding's inequality and recalling $p_{\mathrm{OPT}} = 2/3$. Setting $k = n^{2/3}$, we conclude that with probability $1 - e^{-\Omega(n^{2/3})}$, an OPT-to-ALT mutation disappears after $n^{2/3}$ iterations. We note that in total, the expected number of OPT-to-ALT mutations within a phase is at most $(1-p_{\mathrm{OPT}})p_{\mathrm{mut}}\lambda t_0 = O(n^{1.5})$, so by a straightforward union bound on the probabilities of an ALT iteration surviving more than $n^{2/3}$ iterations, none of the OPT-to-ALT mutations that occur in the considered interval survive for more than the desired number of iterations with high probability.

Finally, we need to show that the rate at which OPT-to-ALT mutations are accepted is low enough to allow any accepted mutations to dissolve through migration without overrunning the island model. To that end, we can bound Y_k, the number of OPT-to-ALT mutations that are accepted within $k = n^{2/3}$ iterations using a Chernoff bound:

$$\begin{aligned} E(Y_k) &< k\lambda p_{\mathrm{mut}} = O(n^{1/6}) = \mu \\ P(Y_k \geq 3\mu) &\leq e^{-\mu} = e^{-\Omega n^{1/6}} \end{aligned}$$

recalling that $\lambda = \sqrt{n}$, $p_{\mathrm{mut}} = 1/(ne)$, and ignoring the possibility that some of these mutations occur during iterations which assign a higher fitness value to OPT, and therefore would not be accepted.

Thus, no more than $O(n^{1/6})$ OPT-to-ALT mutations are accepted during a $n^{2/3}$ iteration period with high probability, and all accepted mutations disappear after $n^{2/3}$ iterations with high probability. By dividing the MAZE phase into blocks of $n^{2/3}$ iterations each, we can conclude that with high probability, at most $2 \cdot O(n^{1/6}) = O(n^{1/6})$ OPT-to-ALT segments can be active at the same time: with high probability, no more than $O(n^{1/6})$ appear at the exact end of an $n^{2/3}$ iteration block, and no more than $O(n^{1/6})$ appear during the next block, with the former group being all being reduced to length 0 before the next-next block begins.

This yields a bound on the total number of islands that can have ALT as their best-so-far individual at the same time: with high probability (as all of the individual bounds apply with high probability to a single ALT segment, and only a polynomial number of ALT segments can appear within a phase), this can be no more than

$$O(\sqrt{\lambda}) \cdot O(n^{1/6}) = O(n^{5/12}) = o(\lambda)$$

islands during a phase: thus, for a sufficiently large n, there will with high probability still be an island with $x_i^*(t) = \mathrm{OPT}$ at the end of the phase. □

We note that the bounds used in Lemma 6.6 take a very dim view of the situation, and could probably be improved significantly. In practical simulations, we observe that the simplified island model converges to a larger-than-p_{OPT} majority of islands having OPT as their current-best solution, and any OPT-to-ALT mutations disappear quickly.

Applying Lemmas 6.5 and 6.6 inductively over n phases yields a proof of Theorem 6.4.

PROOF OF THEOREM 6.4. For the first iteration, Lemma 6.6 may be applied immediately, as all islands are initialized with the OPT individual. Per the lemma, at least one island ends the phase with $x_i^*(t) = \mathrm{OPT}$ with high probability, allowing Lemma 6.5 to be applied at the beginning of the next phase. Per that Lemma, there is with high probability an iteration within the phase when OPT is the current-best individual on all islands, allowing Lemma 6.6 to be applied again.

As both of these Lemmas succeed with high probability, and we only require n successes of each Lemma, we can use a simple union bound on the failure probabilities to conclude that with high probability, at least one island is still tracking the oscillating optimum after the n phases are over. □

Thus, we have proven that using a unidirectional ring as the migration topology can allow the simplified island model to track the oscillating optimum of the MAZE in settings where this is not possible for the complete migration topology. Intuitively, this is achieved by removing the ability of a single ill-timed migration to propagate an undesirable individual to all islands.

6.5 Conclusion

We have demonstrated there exist choices of parameters for our simplified island model for which a complete migration topology with high probability results in a failure to track the oscillating optimum through all n phases, while a unidirectional ring migration topology allows the optimum to be tracked through all n phases with high probability. This example illustrates that a less-dense migration topology can mitigate the effects of migration occurring during unfavourable iterations of an oscillating fitness function, reducing the need to rely on problem-specific knowledge as in [LW15b].

In future work, it would be useful to extend these results beyond the extreme case of $p_{\text{mig}} = 1$, i. e. for migration occurring less often than during every iteration. We note that while our theoretical analysis here does not prove this directly, $p_{\text{mig}} = 1$ combined with a low λp_{mut} actually leads to a reduction in population diversity, with the majority of the islands settling on OPT as their current-best solution, rather than achieving a p_{OPT}-like balance between OPT and ALT islands. We conjecture that with slower migration, such a balance could be achieved.

Bibliography

[AD11] Anne Auger and Benjamin Doerr, editors. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing, 2011.

[AF08] Nattapat Attiratanasunthron and Jittat Fakcharoenphol. A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs. *Information Processing Letters*, 105(3):88–92, 2008.

[ALN13] Enrique Alba, Gabriel Luque, and Sergio Nesmachnow. Parallel metaheuristics: recent advances and new trends. *International Transactions in Operational Research*, 20(1):1–48, 2013.

[ANS13] Enrique Alba, Amir Nakib, and Patrick Siarry. *Metaheuristics for Dynamic Optimization*. Studies in Computational Intelligence. Springer, 2013.

[DG13] Benjamin Doerr and Leslie Ann Goldberg. Adaptive drift analysis. *Algorithmica*, 65(1):224–250, 2013.

[DHK12] Benjamin Doerr, Ashish Hota, and Timo Kötzing. Ants easily solve stochastic shortest path problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '12)*, pages 17–24. ACM Press, 2012.

[DJL15] Duc-Cuong Dang, Thomas Jansen, and Per Kristian Lehre. Populations can be essential in dynamic optimisation. In Laredo et al. [LSE15], pages 1407–1414.

[DJS11] Benjamin Doerr, Daniel Johannsen, and Martin Schmidt. Runtime analysis of the (1+1) evolutionary algorithm on strings over finite alphabets. In *Proceedings of Foundations of Genetic Algorithms Workshop (FOGA '11)*, pages 119–126. ACM Press, 2011.

[DJW12] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[DP12] Benjamin Doerr and Sebastian Pohl. Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '12)*, pages 1317–1324. ACM Press, 2012.

[Dro02] Stefan Droste. Analysis of the (1+1) EA for a dynamically changing OneMax-variant. In *Proc. of CEC'02*, pages 55–60. IEEE Press, 2002.

[Dro03] Stefan Droste. Analysis of the (1+1) EA for a dynamically bitwise changing OneMax. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '03)*, pages 909–921. Springer, 2003.

[FK13] Matthias Feldmann and Timo Kötzing. Optimizing expected path lengths with ant colony optimization using fitness proportional update. In *Proceedings of Foundations of Genetic Algorithms (FOGA '13)*, pages 65–74. ACM Press, 2013.

[GB02] Ryan M. Garlick and Richard S. Barr. Dynamic wavelength routing in WDM networks via ant colony optimization. In Marco Dorigo, Gianni Di Caro, and Michael Sampels, editors, *Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 250–255. Springer, 2002.

[Gun05] Christian Gunia. On the analysis of the approximation capability of simple evolutionary algorithms for scheduling problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pages 571–578. ACM Press, 2005.

[Haj82] Bruce Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14:502–525, 1982.

[HY01] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:57–85, 2001. Erratum in Artif. Intell. 140(1/2): 245-248 (2002).

[Jan13] Thomas Jansen. *Analyzing Evolutionary Algorithms: The Computer Science Perspective*. Natural Computing Series. Springer, 2013.

[JB05] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments—A survey. *IEEE Transactions on Evolutionary Computation*, 9:303–317, 2005.

[JJW05] Thomas Jansen, Kenneth A. De Jong, and Ingo Wegener. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13(4):413–440, 2005.

[JS05] Thomas Jansen and Ulf Schellbach. Theoretical analysis of a mutation-based evolutionary algorithm for a tracking problem in the lattice. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*, pages 841–848. ACM Press, 2005.

[JZ14] Thomas Jansen and Christine Zarges. Evolutionary algorithms and artificial immune systems on a bi-stable dynamic optimisation problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '14)*, pages 975–982. ACM Press, 2014.

[KLNO10] Timo Kötzing, Per Kristian Lehre, Frank Neumann, and Pietro S. Oliveto. Ant colony optimization and the minimum cut problem. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2010)*, pages 1393–1400. ACM Press, 2010.

[KLW15] Timo Kötzing, Andrei Lissovoi, and Carsten Witt. (1+1) EA on generalized dynamic OneMax. In *Proceedings of Foundations of Genetic Algorithms Workshop (FOGA '15)*, pages 40–51. ACM Press, 2015.

[KM12] Timo Kötzing and Hendrik Molter. ACO beats EA on a dynamic pseudo-boolean function. In *Procedings of Parallel Problem Solving from Nature (PPSN XII)*, pages 113–122. Springer, 2012.

[KNRW12] Timo Kötzing, Frank Neumann, Heiko Röglin, and Carsten Witt. Theoretical analysis of two ACO approaches for the traveling salesman problem. *Swarm Intelligence*, 6(1):1–21, 2012.

[Köt14] Timo Kötzing. Concentration of first hitting times under additive drift. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '14)*, pages 1391–1398, 2014.

[KSA+11] Mostepha Redouane Khouadjia, Briseida Sarasola, Enrique Alba, Laetitia Jourdan, and El-Ghazali Talbi. Multi-environmental cooperative parallel metaheuristics for solving dynamic optimization problems. In *Proceedings of 2011 IEEE International Parallel & Distributed Processing Symposium*, pages 395–403. IEEE Press, 2011.

[Lis12] Andrei Lissovoi. Analysis of ant colony optimization for dynamic shortest path problems. Master's thesis, Department of Informatics and Mathematical Modelling, Technical University of Denmark, 2012. http://findit.dtu.dk/en/catalog/230255971.

[LS14] Jörg Lässig and Dirk Sudholt. General upper bounds on the runtime of parallel evolutionary algorithms. *Evolutionary Computation*, 22(3):405–437, 2014.

[LSE15] Juan Luis Jiménez Laredo, Sara Silva, and Anna Isabel Esparcia-Alcázar, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '15)*. ACM, 2015.

[LW13a] Per Kristian Lehre and Carsten Witt. General drift analysis with tail bounds. arXiv:1307.2559, http://arxiv.org/abs/1307.2559, 2013.

[LW13b] Andrei Lissovoi and Carsten Witt. Runtime analysis of ant colony optimization on dynamic shortest path problems. In Christian Blum and Enrique Alba, editors, *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*, pages 1605–1612. ACM, 2013.

[LW14a] Per Kristian Lehre and Carsten Witt. Concentrated hitting times of randomized search heuristics with variable drift. In *Proceedings of the 25th International Symposium on Algorithms and Computation (ISAAC'14)*, volume 8889 of *Lecture Notes in Computer Science*, pages 686–697. Springer, 2014. extended version at http://arxiv.org/abs/1307.2559.

[LW14b] Andrei Lissovoi and Carsten Witt. MMAS vs. population-based EA on a family of dynamic fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '14)*, pages 1399–1406. ACM Press, 2014.

[LW15a] Andrei Lissovoi and Carsten Witt. MMAS versus population-based EA on a family of dynamic fitness functions. *Algorithmica*, 2015. in press, final version online at http://dx.doi.org/10.1007/s00453-015-9975-z.

[LW15b] Andrei Lissovoi and Carsten Witt. On the utility of island models in dynamic optimization. In Laredo et al. [LSE15], pages 1447–1454.

[LW15c] Andrei Lissovoi and Carsten Witt. Runtime analysis of ant colony optimization on dynamic shortest path problems. *Theoretical Computer Science*, 561:73–85, 2015.

[MS14] Andrea Mambrini and Dirk Sudholt. Design and analysis of adaptive migration intervals in parallel evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '14)*, pages 1047–1054. ACM Press, 2014.

[NSW10] Frank Neumann, Dirk Sudholt, and Carsten Witt. A few ants are enough: ACO with iteration-best update. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2010)*, pages 63–70, 2010.

[NW10a] Frank Neumann and Carsten Witt. Ant colony optimization and the minimum spanning tree problem. *Theoretical Computer Science*, 411(25):2406–2413, 2010.

[NW10b] Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Natural Computing Series. Springer, 2010.

[NYB12] Trung Thanh Nguyen, Shengxiang Yang, and Jürgen Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012.

[OW11] Pietro S. Oliveto and Carsten Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59(3):369–386, 2011.

[OW12] Pietro S. Oliveto and Carsten Witt. Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation. *arXiv:1211.7184*, 2012. `http://arxiv.org/abs/1211.7184`.

[OZ13] Pietro S. Oliveto and Christine Zarges. Analysis of diversity mechanisms for optimisation in dynamic environments with low frequencies of change. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '13)*, pages 837–844. ACM Press, 2013.

[RLY09] Philipp Rohlfshagen, Per Kristian Lehre, and Xin Yao. Dynamic evolutionary optimisation: An analysis of frequency and magnitude of change. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '09)*, pages 1713–1720. ACM Press, 2009.

[RS14] Jonathan E. Rowe and Dirk Sudholt. The choice of the offspring population size in the (1,λ) evolutionary algorithm. *Theoretical Computer Science*, 545:20–38, 2014.

[SH00] Thomas Stützle and Holger H. Hoos. MAX-MIN ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[ST12a] Dirk Sudholt and Christian Thyssen. Running time analysis of ant colony optimization for shortest path problems. *Journal of Discrete Algorithms*, 10:165–180, 2012.

[ST12b] Dirk Sudholt and Christian Thyssen. A simple ant colony optimizer for stochastic shortest path problems. *Algorithmica*, 64(4):643–672, 2012.

[Sto08] Tobias Storch. On the choice of the parent population size. *Evolutionary Computation*, 16(4):557–578, 2008.

[Sud11] Dirk Sudholt. Using markov-chain mixing time estimates for the analysis of ant colony optimization. In *Proceedings of the 11th Workshop on Foundations of Genetic Algorithms (FOGA '11)*, pages 139–150. ACM Press, 2011.

[Wit14] Carsten Witt. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Information Processing Letters*, 114(1):38–41, 2014.

[XL08] Wei Xiang and Heow Pueh Lee. Ant colony intelligence in multi-agent dynamic manufacturing scheduling. *Engineering Applications of Artificial Intelligence*, 21(1):73 – 85, 2008.

[Zho09] Yuren Zhou. Runtime analysis of an ant colony optimization algorithm for TSP instances. *IEEE Transactions on Evolutionary Computation*, 13(5):1083–1092, 2009.