A Rigorous Runtime Analysis of the 2-MMAS_{ib} on Jump Functions: Ant Colony Optimizers Can Cope Well With Local Optima

Riade Benbaki
École Polytechnique
Institut Polytechnique de Paris
Palaiseau, France

Ziyad Benomar
École Polytechnique
Institut Polytechnique de Paris
Palaiseau, France

Benjamin Doerr
Laboratoire d'Informatique (LIX)
CNRS, École Polytechnique
Institut Polytechnique de Paris
Palaiseau, France

ABSTRACT

Ant colony optimizers have been successfully used as general-purpose optimization heuristics. Due to the complicated nature of the random processes that describe the runs of ACO algorithms, the mathematical understanding of these algorithms is much less developed than that of other nature-inspired heuristics. In this first runtime analysis of a basic ACO algorithm on a classic multimodal benchmark, we analyze the runtime of the 2-MMAS_{ib} on jump functions. For moderate jump sizes $k \leq \alpha_0 \ln n$, $\alpha_0 > 0$ a constant, we prove a runtime of order $O(\sqrt{n}/\rho)$, when the evaporation factor ρ satisfies $\rho \leq Cn^{-1/2} \ln(n)^{-1}$ for a sufficiently small constant C . For $\rho = \Theta(n^{-1/2} \ln(n)^{-1})$, we thus obtain a runtime of $O(n \ln(n))$. This result shows that simple ACO algorithms can cope much better with local optima than many evolutionary algorithms, which need $\Omega(n^k)$ time.

CCS CONCEPTS

• Theory of computation → Random search heuristics.

KEYWORDS

Ant colony optimization, running time analysis, theory

ACM Reference Format:

Riade Benbaki, Ziyad Benomar, and Benjamin Doerr. 2021. A Rigorous Runtime Analysis of the 2-MMAS_{ib} on Jump Functions: Ant Colony Optimizers Can Cope Well With Local Optima. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21)*, July 10–14, 2021, Lille, France. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3449639.3459350>

1 INTRODUCTION

Ant colony optimization (ACO) algorithms are nature-inspired search heuristics. They construct new solutions for a given optimization problem by letting artificial ants conduct a pheromone-guided random walk on a so-called construction graph. For graph-based optimization problems, this often is just the graph of the problem.

This work was supported by a public grant as part of the Investissements d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

GECCO '21, July 10–14, 2021, Lille, France

© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-8350-9/21/07. <https://doi.org/10.1145/3449639.3459350>

For other problems, e.g., pseudo-Boolean optimization problems, artificial construction graphs have been successfully used.

ACO algorithms have achieved excellent results for various hard combinatorial optimization problems [22]. Since new solutions are not either accepted or rejected as in genetic algorithms, but instead the information derived from new solutions is cautiously integrated into the pheromone system, ACO algorithms have shown a good performance also in noisy, dynamic, and stochastic optimization [6, 27, 40].

In contrast to the numerous successful applications of ACO algorithms, the rigorous understanding of these methods is less developed. Compared to the extensive theoretical literature on classic evolutionary algorithms (EAs), see, e.g., [4, 18, 34, 52], only relatively few mathematical runtime analyses of ACO algorithms exist (see Section 2.1). These prove that ACO algorithms can be efficient, give suggestions on how to choose the parameters, and even manage to show that ACO algorithms can better deal with noise than many classic EAs. However, essentially all these results regard unimodal optimization problems only (problems without true local optima). Since a too strong focus of theoretical research on unimodal problems can lead to misconceptions (see [17] for an example from classic EAs), we conduct in this work a first mathematical runtime analysis focussed on understanding how a simple ACO algorithm performs on a classic multimodal benchmark.

As algorithm we regard the 2-MMAS_{ib}, a simplified variant of the MAX-MIN Ant System [59], that was previously analyzed with theoretical methods and was shown to be efficient even in the smallish case of only two ants [49]. As optimization problem, we regard the class of jump functions, which is the most intensively studied benchmark in the theory of evolutionary algorithms that is not unimodal. The jump function with jump size k is the pseudo-Boolean function that equals the easy ONEMAX function except that it has a very low fitness in a Hamming ball of radius $k - 1$ around the optimum. For that reason, many evolutionary algorithms easily reach the local optimum (all search points in Hamming distance k from the optimum), but then find it difficult to jump over or walk through the valley of low fitness, often leading to runtimes of order $\Theta(n^k)$, see Section 2.2.

Main Result: We conduct a mathematical runtime analysis of the 2-MMAS_{ib} on the jump functions benchmark. We prove that there are constants $\alpha_0, C > 0$ such that the 2-MMAS_{ib} with pheromone evaporation rate $\rho \leq Cn^{-1/2} \ln(n)^{-1}$ finds the optimum of any jump function with representation length n and jump size $k \leq \alpha_0 \ln(n)$ with high probability in time $O(\sqrt{n}/\rho)$. This is

the same runtime that was shown for the easy unimodal ONEMAX function [49]. Consequently, the 2-MMAS_{ib} is essentially unaffected by the moderate-size valley of low fitness of the jump function.

We note that a similar result was recently obtained for an estimation-of-distribution algorithm (EDA), namely the compact genetic algorithm (cGA). Building on previous work in [32], it was shown in [11] that the cGA with hypothetical population size $\mu = \Omega(\sqrt{n} \log n) \cap \text{poly}(n)$ with high probability optimizes any jump function with jump size $k < \frac{1}{20} \ln(n)$ in time $O(\mu\sqrt{n})$. Since both the cGA and the 2-MMAS_{ib} cautiously evolve a univariate distribution on the search space, this might be the common reason for these two results. We note, however, that the precise mechanisms of the EDA cGA and the ACO algorithm 2-MMAS_{ib} are quite different.

Because of these differences, we also need a significantly different proof method. The main technical difficulty we need to overcome is the following. Different from the cGA, when the 2-MMAS_{ib} samples two search points with distance from the optimum higher than the expectation, the pheromone model becomes worse. For this reason, unlike in [11], we cannot simply argue with waiting times for sufficiently strong model improvements. To estimate the joint progress from good and counterproductive pheromone updates, we thus have to resort to drift arguments. Drift arguments have been used before in the analysis of ACO algorithms. Closest to ours is the runtime analysis of the 2-MMAS_{ib} on the ONEMAX benchmark [49]. Unfortunately, the drift arguments used there, as often with drift analysis, only gave estimates for the expected runtime. Such estimates are not sufficient for our purposes since we aim at a runtime guarantee that holds with high probability (in fact, we are not convinced that our runtime estimate also holds in expectation – rather we expect that with very small probability $n^{-\Omega(1)}$ the pheromone model deteriorates so badly that an $n^{\Omega(k)}$ runtime results, which, if true, would imply an expected runtime of similar order of magnitude).

To overcome these difficulties, we not only estimate the drift in the expected distance D_t of a sample from the optimum, but we also compute an exponential moment of the absolute change $|D_{t+1} - D_t|$. From this, we argue that the supermartingale describing the positive deviation of D_t from a desired value admits Chernoff-type strong concentration estimates, and we conclude from this that, with high probability, D_t is indeed shrinking in the way the expected change of $D_{t+1} - D_t$ suggests. This argument seems novel in the theory of ACO. We are optimistic that it will find more applications in the future. We note that in [61], a tail bound was proven for the runtime of the EDA cGA via a variable drift theorem giving also tail bounds on the hitting time [43, 44]. This approach appears to be applicable to our problem as well. It also needs the computation of exponential moments, but does not need the separation of the process into a logarithmic number of phases.

The remainder of this work is organized as follows. We survey the state of the art in Section 2. We define precisely our problem in Section 3 and state our main result together with the main proof ideas. Our mathematical proofs can be found in Sections 4, 5, and 6. Finally, we present some experimental results in Section 7.

2 PREVIOUS WORK

In this section, we describe the most relevant previous works. We start by discussing the general state of the art of theoretical research in ACO and then present the main runtime analysis results for the jump functions benchmark.

2.1 Theory of ACO Algorithms

Since we will conduct a mathematical runtime analysis, we only review the state of the art of mathematical runtime analyses for ACO algorithms, and also here we only mention the most relevant works. For other theoretical results, in particular, a decent amount of convergence results from the early 2000s, we refer to the survey [30]. A recent general survey on ACO can be found, e.g., in [47].

The mathematical runtime analysis of ACO algorithms was started independently in [29, 50]. The main result in [50] is a proof that the 1-ANT algorithm with a suitable parameter setting optimizes ONEMAX in time $O(n^2)$ with high probability. The 1-ANT algorithm updates the pheromones only when a solution is found that is at least as good as the best-so-far solution. Quadratic runtimes were also shown for the LEADINGONES and BINVAL benchmarks [19]. In [51] the runtime of the 1-ANT on the minimum spanning tree problem was analyzed. Similar (and stronger for ONEMAX, where an $O(n \log n)$ runtime was shown) results have been obtained for the MMAS with best-so-far reinforcement [31]. None of these results could show an advantage of the ACO algorithm over comparably simple evolutionary algorithm, and many proofs suggest that the optimization behavior of these ACO algorithms is very similar to the one of the $(1 + 1)$ EA, possibly slowed down by the cautious pheromone update. These works, however, started a rigorous runtime analysis of ACO algorithms and detected important threshold behaviors with respect to the pheromone evaporation rate.

For graph-based problems, several interesting results have been shown for ACO algorithms with best-so-far reinforcement [37, 39, 63], some clearly showing a better performance than EAs. The reason for these good performances, most likely, is that the construction of new solutions in ACO algorithms can be better adjusted to the problem to be solved.

In [49], for the first time a runtime analysis of an ACO algorithm with iteration-best reinforcement was conducted. This did not lead to a runtime better than $O(n \log n)$ for ONEMAX, but showed a completely different optimization behavior, namely that the pheromone values in a uniform manner approach the desired upper end of the pheromone range. Surprisingly, this algorithm received not much attention since then. In the only other runtime analysis for this algorithm [61], a lower bound for the runtime on ONEMAX, matching the upper bound from [49] for large parameter ranges, was proven.

One clear advantage of ACO algorithms is their robustness to dynamic changes to the problem instance, to noise, and in stochastic optimization. This generally accepted view has been supported by several runtime analyses [15, 25, 27, 38, 46, 60] on shortest paths problems and linear functions.

In summary, we see that the vast majority of the runtime analyses of ACO algorithms regard unimodal problems such as ONEMAX, LEADINGONES, BINVAL, linear functions, or shortest paths problems. The few exceptions include the work [52] on the minimum spanning

tree problem (where one has to note that each non-optimal search point has a better search point in Hamming distance at most two, so flipping at most two bits suffices to leave a local optimum) and the works [37, 39, 63] on graph-based problems. These are, however, difficult to put into a general context as the construction of new solutions is problem-specific. The only other work we are aware of that analyzes the performance of ACO algorithms on multimodal problems is [48]. Here artificial problem instances are constructed to demonstrate advantages and disadvantages of combining ACO algorithms with local search. These instances are designed for this particular purpose, so we do not see that they yield more general insights on how ACO algorithms cope with local optima. In the light of this state of the art, it is safe to say that the theory of ACO algorithms is strongly biased towards unimodal problems and that no significant general insight exists on how ACO algorithms cope with local optima.

2.2 Runtime Analyses for Jump Functions

The observation that most runtime analyses of ACO algorithms regard unimodal problems is, to some extent, also valid for the larger class of all nature-inspired search heuristics (see, e.g., [4, 18, 34, 52]). Nevertheless, by now a decent number of results on problems with local optima exist. The overwhelming majority of these consider the jump functions benchmark, rendering it the only multimodal benchmark problem for which a real understanding over a broad set of algorithms exists in stark contrast to the isolated results on trap and valley functions or the DeceivingLeadingBlocks problem such as [16, 33, 42, 45, 53, 54].

Jump functions were introduced in the seminal work [23] as a multimodal benchmark of scalable difficulty. The jump function with jump parameter (jump size) k is the pseudo-Boolean function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ that is isomorphic to the ONEMAX function except that the fitness of all search points in Hamming distance 1 to $k - 1$ from the optimum is low and deceiving, that is, increasing with the distance from the optimum (see Section 3.2 for the precise definition). For that reason, simple elitist mutation-based EAs easily reach the local optimum (consisting of all search points in Hamming distance exactly k from the optimum), but then have no other way to make progress than by directly generating the global optimum. When using standard bit mutation with mutation rate $p = 1/n$, this jump over the fitness valley takes expected time $n^k (1 - 1/n)^{n-k} \geq \frac{1}{e} n^k$. Consequently, as proven formally in [23], the expected runtime of the $(1 + 1)$ EA is $\Theta(n^k)$ when $k \geq 2$ (for $k = 1$, the jump function is isomorphic to ONEMAX and thus is unimodal). For reasonable ranges of the parameters, this bound has been extended [12] to the $(\mu + \lambda)$ EA and, less obvious, the (μ, λ) EA, the latter result improving over the earlier $\exp(\Omega(k))$ lower bound of [41, Theorem 5].

Larger mutation rates can give some speed-ups [17]. For the $(1 + 1)$ EA, the asymptotically optimal mutation rate $p = k/n$ reduces the runtime on the jump function with jump size $k = o(\sqrt{n})$ by a factor of order $(1 - o(1))(k/e)^k$. A heavy-tailed mutation operator using a random mutation rate sampled from a power-law distribution with exponent $\beta > 1$ obtains a slightly smaller speed-up of $\Omega(k^{-\beta+0.5}(k/e)^k)$, but does so without parameters that need to be adapted to the jump size k (note that random mutation rated

have been used before in [13, 14]). In [55–57], several essentially parameter-less stagnation-detection mechanisms were developed that give the same asymptotic runtime as the best static mutation rate.

Jump functions were also analyzed in the context of crossover-based algorithms. The first such work [36] proved that a simple $(\mu + 1)$ genetic algorithm using uniform crossover with (an unusually small) rate $p_c = O(\frac{1}{kn})$ has an $O(\mu n^2 k^3 + 2^{2k} p_c^{-1})$ runtime when the population size is at least $\mu = \Omega(k \log n)$. For standard values of the crossover rate, a runtime of $O(n^{k-1} \log n)$ was achieved in [10, Theorem 2]. With additional diversity mechanisms, the runtime can be lowered to $O(n \log n + 4^k)$, see [9]. When choosing the three parameters suitably, the $(1 + (\lambda, \lambda))$ GA can optimize JUMP_k in time $O((n/k)^{(k+1)/2} e^{O(k)})$ [3], similar runtimes result from heavy-tailed parameter choices [1, 2]. With a three-parent majority vote crossover, among other results, a runtime of $O(n \log n)$ was obtained via a suitable island model for all $k = O(n^{1/2-\epsilon})$ [26]. A different voting algorithm also having an $O(n \log n)$ runtime can be found in [58]. Via a hybrid genetic algorithm using as variation operators only local search and a deterministic voting crossover, an $O(n)$ runtime was shown in [62].

Runtimes of $O(n \binom{n}{k})$ and $O(k \log(n) \binom{n}{k})$ were shown for the $(1 + 1)$ IA^{hyp} and the $(1 + 1)$ Fast-IA artificial immune systems, respectively [7, 8]. The runtime of a hyper-heuristic switching between elitist and non-elitist selection was studied in [45]. The lower bound of order $\Omega(n \log n) + \exp(\Omega(k))$ and the upper bound of order $O(n^{2k-1}/k)$, however, are too far apart to indicate an advantage or a disadvantage over most classic algorithms. In this work, it is also stated that the Metropolis algorithm has an $\exp(\Omega(n))$ runtime on jump functions. Extensions of jump functions were analyzed in [5, 35], a multi-objective variant in [21].

3 PROBLEM DESCRIPTION AND STATEMENT OF THE MAIN RESULT

In this section, we make precise the algorithm and the benchmark problem we regard. Also, we state our main result together with an overview of the proof.

3.1 Description of the Algorithm

In ACO, new solutions are constructed by letting artificial ants traverse a so-called construction graph, guided by pheromones they dispose on the graph and that evaporate slowly over time. The construction graph is a directed graph with a designated start vertex. It has the property that paths starting in this vertex, not visiting a vertex twice, and not being extendable, can be translated into solutions for the problem to be solved. Such paths are generated by random walks of the artificial ant as follows. The ant starts at the start vertex. From its current vertex it moves to a vertex chosen randomly from all unvisited out-neighbors with probability proportional to the pheromone value of the corresponding edge. These moves are repeated until a vertex is reached such that all out-neighbors have been visited already. The ant walks generated in one iteration are translated into solutions of the problem, their fitness is evaluated, and based on these fitnesses the pheromone values are updated. The pheromone values of all edges are subject to evaporation, that is, a multiplicative decrease by a factor of

$1 - \rho$, where the evaporation rate $\rho \in [0, 1]$ is a parameter of the ACO algorithm. Consequently, recently constructed solutions are reflected stronger in the pheromone system than older ones.

Figure 1: Construction graph for optimizing pseudo-Boolean functions (here for $n = 4$).

When optimizing pseudo-Boolean functions, that is, functions defined on the set $\{0, 1\}^n$ of bit strings of length n , this system can be simplified. In this case, the construction graph proposed in [28] and depicted in Figure 1 (for the case $n = 4$) is commonly used. In this graph, vertex 0 is the start vertex. The auxiliary vertices i^+ and i^- , $i = 1, \dots, n$ have only one outgoing edge, so no pheromones are needed for these. Since each of the more interesting vertices $i = 0, \dots, n - 1$ is visited exactly once, the pheromone values can be normalized in such a manner that the values on the two outgoing edges add up to one. In this case, the pheromone value directly describes the probability that the ant uses this edge. The possible ant walks are all paths starting in 0 and ending in n . Such a walk is translated into a bit string $x \in \{0, 1\}^n$ by setting $x_i = 1$ if and only if the ant was traversing the vertex i^+ , which happened with probability equal to the pheromone value on the edge $(i - 1, i^+)$. Denoting this pheromone value by τ_i , we see that an ant walk constructs a solution x such that $\Pr[x_i = 1] = \tau_i$ independently for all $i \in [1..n]$. We will write $x \sim \text{Sample}(\tau)$ to indicate this distribution in the remainder.

With these simplifications, the 2-MMAS_{ib} is the algorithm described in Algorithm 1. It starts with all pheromone values being $1/2$, which means that the ant walk generates a random bit string. In the 2-MMAS_{ib}, in each iteration two ant walks are performed simultaneously and then the update is performed with the ant walk leading to the better solution. The pheromone update consists first of a partial evaporation (with rate ρ , which is the only algorithm parameter of the 2-MMAS_{ib}) of the existing pheromone traces. The pheromone values of the edges used by the successful ant walk are then increased. To maintain the normalization of the pheromone values, this means an increase of τ_i by an additive term of ρ when the successful solution had a 1 in the i -th bit position.

Since in this theoretical work we regard the random variable that counts the number of iterations until the optimum is found (“runtime”), we did not specify a termination criterion. Note that in each iteration exactly two search points are evaluated. Hence the runtime is exactly half the number of fitness evaluations (which is an equally common performance measure).

We note that usually the 2-MMAS_{ib} is used with the pheromone boundaries $1/n$ and $1 - 1/n$, that is, when a pheromone value leaves

the interval $[1/n, 1 - 1/n]$, it is replaced by the corresponding boundary value. This prevents a premature convergence of the pheromone system. To ease the presentation, we omit such pheromone boundaries in this work. From our proofs it is clear that with high probability no pheromone value comes close to the wrong boundary value. In fact, we shall even show that with high probability, no pheromone value ever goes below $1/3$. For this reason, we do not need boundary values to show our main result. From inspecting the proof, it is also clear that our results would hold in an unchanged manner if we would use boundaries for the pheromone values (that is, we do not profit from not using boundaries). However, the technicality of the proofs would increase with such boundaries and this would obscure the main arguments. For this reason, we restrict our rigorous analysis to the setting without pheromone boundaries.

Algorithm 1: The 2-MMAS_{ib} (without pheromone boundaries) with evaporation rate ρ to maximize a function $f : \{0, 1\}^n \rightarrow \mathbb{R}$.

```

1  $t \leftarrow 0$ ;
2  $\tau_0 = (\frac{1}{2}, \dots, \frac{1}{2}) \in [0, 1]^n$ ;
3 repeat
4    $x_1 \leftarrow \text{Sample in } \{0, 1\}^n \text{ using } \tau_t$ ;
5    $x_2 \leftarrow \text{Sample in } \{0, 1\}^n \text{ using } \tau_t$ ;
6    $y \leftarrow \arg\max_{x \in \{x_1, x_2\}} f(x)$ ;
7    $\tau_{t+1} \leftarrow (1 - \rho)\tau_t + \rho y$ ;    /* pheromone update */
8    $t \leftarrow t + 1$ ;
9 until forever;
```

When formulating the 2-MMAS_{ib} as in Algorithm 1, we can easily compare this algorithm with the compact genetic algorithm (cGA). The cGA, here discussed again without frequency boundaries, can be obtained from the 2-MMAS_{ib} by replacing line 7 with the statement “Let $x \in \{0, 1\}^n$ such that $\{x, y\} = \{x_1, x_2\}$. Set $\tau_{t+1} \leftarrow \tau_t + \frac{1}{\mu}(y - x)$.” Here μ is the hypothetical population size, an algorithm parameter that regulates how strong the vector of sampling frequencies is modified in each iteration.

The main difference between the cGA and the 2-MMAS_{ib} is that the frequency update in the cGA uses the difference of the better and the worse solution, whereas the 2-MMAS_{ib} uses only the better search point (in a linear combination with the current pheromone vector). This difference has a crucial impact on the way the algorithm works, and consequently, on how it can be analyzed. When the cGA samples two solutions such that the one with higher fitness is also closer to the optimum x^* (e.g., when optimizing a function with good fitness-distance correlation), then the frequency update reduces the expected Hamming distance $D_t = \|\tau_t - x^*\|_1$ of a sample from the optimum. This is different for the 2-MMAS_{ib}: If both solutions sampled in one iteration are further away from the optimum than expected, that is, if $H(x_i, x^*) > D_t$ for $i = 1, 2$, then necessarily the expected distance of a sample increases (worsens), that is, we have $D_{t+1} > D_t$. Consequently, when optimizing an objective function with perfect fitness-distance correlation such as ONEMAX, the random process D_t is monotonically decreasing for the cGA, but not for the 2-MMAS_{ib}. This property was heavily exploited in the analyses of the cGA on jump functions [11, 32].

This difference is a main reason why our analysis is substantially different from the ones in these works.

A second difference between the cGA and the 2-MMAS_{ib} is the amount by which a frequency or pheromone value changes. For the cGA, this is always by an absolute value of $\frac{1}{\mu}$. For the 2-MMAS_{ib}, a pheromone value $\tau_i \in [0, 1]$ is changed to one of $\tau_i - \rho\tau_i$ and $\tau_i + \rho(1 - \tau_i)$. Consequently, the minimum number of iterations to move a frequency or pheromone value from 0.5 to, say, 0.25, is $\Theta(\mu)$ for the cGA and $\Theta(1/\rho)$ for the 2-MMAS_{ib}. Hence the scaling $\mu = 1/\rho$ could suggest a similar behavior. However, to bring a value from 0.5 to, say, $1/n$, a minimum number of $\Theta(\mu)$ iterations is needed for the cGA, but $\Theta((1/\rho) \log n)$ for the 2-MMAS_{ib}. Compared to the scaling suggested before, now the 2-MMAS_{ib} is slower by a logarithmic factor.

3.2 Jump Functions

The simplest pseudo-Boolean benchmark function in evolutionary computation is the ONEMAX function, counting the number of ones in a string of n bits. We recall that in [49], it was proven that the 2-MMAS_{ib} optimizes the ONEMAX function in an expected runtime of $O(\frac{\sqrt{n}}{\rho})$ when $\rho \leq \frac{c}{\sqrt{n} \log n}$ for a sufficiently small constant c .

For all representation lengths n and all $k \in [1..n]$, the *jump* function with gap size k is defined by

$$\text{JUMP}_{nk}(x) = \begin{cases} \|x\|_1 + k & \text{if } \|x\|_1 \in [0..n-k] \cup \{n\}, \\ n - \|x\|_1 & \text{if } \|x\|_1 \in [n-k+1..n-1], \end{cases}$$

for all $x \in \{0, 1\}^n$. Jump functions have a fitness landscape isomorphic to ONEMAX, except on the fitness valley or gap

$$G_{nk} := \{x \in \{0, 1\}^n \mid n-k < \|x\|_1 < n\}.$$

We refer to Section 2 for a discussion on how randomized search heuristics cope with this valley of low fitness. We note that there is no previous result showing how an ACO algorithm optimizes a jump function.

Since our main result does not depend on the fitness values in the gap region, we formulate and prove it for the more general class of all functions that agree with the jump function outside the valley and that have the original global optimum as one global optimum. To this aim, we define a *subjump* function with jump size k to be a function $\mathcal{F} : \{0, 1\}^n \rightarrow \mathbb{R}$ such that

- $\mathcal{F}(x) = \|x\|_1 + k$ if $\|x\|_1 \in [0..n-k] \cup \{n\}$,
- $\mathcal{F}(x) \leq n+k$ if $\|x\|_1 \in [n-k+1..n-1]$,

for all $x \in \{0, 1\}^n$.

3.3 Main Result and Proof Outline

Our main result is the following upper bound on the runtime of the 2-MMAS_{ib} on subjump functions.

THEOREM 3.1. *There exist constants $\alpha_0, C > 0$ such that the 2-MMAS_{ib} with any evaporation factor $\rho \leq C(\sqrt{n} \log n)^{-1}$ with probability $1 - o(1)$ finds the optimum of any subjump function with jump size $k \leq \alpha_0 \log n$ within $O(\sqrt{n}/\rho)$ iterations.*

In our proof, we analyze the optimization process in two parts. The first part lasts until $D_t := n - \|\tau_t\|_1 = O(\log n)$. In this part, the expected distance D_t of a constructed solution is large enough to rarely construct solutions in the gap area. Consequently, the

optimization process is similar to the one when optimizing ONEMAX and we therefore obtain a similar drift of D_t towards zero.

Once we have $D_t = O(\log n)$, and here is where the second part of our analysis starts, we have a fair chance that an ant walk constructs the optimum. Since the changes of the pheromones are slow, D_t stays small long enough that we indeed sample the optimum with high probability in relatively short time. This part of the analysis, which is the by far easier part, is essentially equal to the third part of the analysis of the cGA in [11].

The runtime we are aiming at is true with high probability, but most likely not in expectation. For this reason, we need a result that holds with high probability for the ONEMAX-style first part, which is difficult and uses other tools than those used in [49] to prove a bound on the expected runtime of the ONEMAX.

Both to have a sufficient drift in the first part and to have a sufficiently high chance of sampling the optimum in the second part of the analysis, we need that the pheromone values are bounded from below by a positive constant. To this aim, we invoke Theorem 2 from [20], an analysis of the so-called genetic drift in EDAs. This theorem, formulated for the EDAs cGA and PBIL, applies to our setting since the 2-MMAS_{ib} is the special case of PBIL with selection number $\mu = 1$. For the 2-MMAS_{ib}, we thus have that for all $\gamma > 0$, $T \in \mathbb{N}_{\geq 1}$, and $j \in [1..n]$

$$\Pr[\forall t \in [0..T] : |\tau_{j,t} - \frac{1}{2}| < \gamma] \geq 1 - 2 \exp\left(-\frac{\gamma^2}{2\rho^2 T}\right),$$

where $\tau_{j,t}$ is the pheromone value for the bit j at time t . Taking $\gamma = 1/6$ and using a union bound gives the following theorem.

THEOREM 3.2. *For all $T \in \mathbb{N}_{\geq 1}$, we have*

$$\Pr[\forall t \in [0..T], \forall j \in [1..n] : \tau_{j,t} > \frac{1}{3}] \geq 1 - 2n \exp\left(-\frac{1}{72\rho^2 T}\right).$$

Note that for $T = O(\sqrt{n}/\rho)$ and $\rho \leq C(\sqrt{n} \log n)^{-1}$ for a sufficiently small constant C , the claim of the theorem holds with the desired probability of $1 - o(1)$.

4 RUNTIME OF THE FIRST PART

Let us consider a subjump function $\mathcal{F}$ with jump size k . In this section, we will focus on the optimization part that begins with $\tau_0 = (\frac{1}{2}, \dots, \frac{1}{2})$ and lasts until $D_t := n - \|\tau_t\|_1 \leq 2k + O(1)$ or some pheromone value $\tau_{j,t}$ drops below $1/3$, where we denote the pheromone vector at time t by $\tau_t = (\tau_{1,t}, \dots, \tau_{n,t})$. We will prove that if $\rho = O(\frac{1}{\sqrt{n} \log n})$, then the duration of this part is $T = O(\sqrt{n}/\rho)$ with high probability.

4.1 Martingale Concentration Results

We will prove later (in Lemma 4.4) that the random variables $D_t := n - \|\tau_t\|_1$ satisfy $E[D_{t+1} - D_t \mid D_t] \leq -\rho\sqrt{D_t}/44$. This implies that $(D_t)_t$ is a supermartingale. Furthermore, if during some successive iterations we have $D_t \geq d$ for a given d , then $(D_t + \rho\sqrt{d}t/44)_t$ is a supermartingale during these iterations. We will exploit this property together with the following martingale concentration inequality, an adapted version of Theorem 2.1 from [24].

THEOREM 4.1. *If $(Y_t)_{t \geq 0}$ is a supermartingale such that for any $t \in \mathbb{N}^*$, $E[\exp(\lambda(Y_t - Y_{t-1})) \mid Y_0, \dots, Y_{t-1}] \leq 1$ for some $\lambda > 0$, then for all $t \in \mathbb{N}^*$ and $x > 0$, $\Pr[Y_t - Y_0 \geq x] \leq \exp(-\lambda x)$.*

4.2 Drift of D_t for ONEMAX

A fairly intuitive idea is that when we are far enough from the gap region, i.e., when $D_t := n - \|\tau_t\|_1$ is sufficiently larger than k , the probability of sampling in the gap region is low, and therefore the evolution of D_t when optimizing the subjump function would be similar to its evolution when optimizing ONEMAX. We will analyze in the following the drift of D_t when optimizing ONEMAX and we will show later that a similar drift occurs when optimizing the subjump function.

LEMMA 4.2. *Consider a run of 2-MMAS_{ib} on ONEMAX. Let t be some iteration such that all pheromone values are at least $1/3$ and $D_t \geq 3$. Then for any $j \in [1..n]$,*

$$E[\tau_{j,t+1} \mid \tau_t] \geq \tau_{j,t} + \frac{\rho}{11} \tau_{j,t} (1 - \tau_{j,t}) \left(\sum_{r \neq j} \tau_{r,t} (1 - \tau_{r,t}) \right)^{-1/2}.$$

PROOF. To prove this, we adapt the proof of Lemma 1 in [49]. From there we have, with $\sigma^2 = \sum_{i \neq j} \tau_{i,t} (1 - \tau_{i,t}) \geq \frac{1}{3} D_t \geq 1$, that

$$E[\tau_{j,t+1} \mid \tau_t] \geq \tau_{j,t} + \rho \tau_{j,t} (1 - \tau_{j,t}) \frac{4}{9(1 + 2\sqrt{3}\sigma)}.$$

With $\frac{9}{4}(1 + 2\sqrt{3}\sigma) \leq \frac{9}{4}(1 + 2\sqrt{3})\sigma \leq 11\sigma$, we obtain our claim. $\square$

From the definition of D_t and the linearity of the expectation, we immediately obtain the following estimate.

LEMMA 4.3. *Consider a run of 2-MMAS_{ib} on ONEMAX. Let t be some iteration such that all pheromone values are at least $1/3$ and $D_t \geq 3$. Then we have*

$$E[D_{t+1} - D_t \mid D_t] \leq -\frac{\rho}{33} \sqrt{D_t}.$$

4.3 Drift of D_t for Subjump Functions

Having established an estimate for the drift of D_t when optimizing ONEMAX, we now derive from this result a similar drift for subjump functions when we are far enough from the optimum, then prove a concentration result for $|D_{t+1} - D_t|$, and finally show in Lemma 4.6 an estimate for an exponentially scaled version of $|D_{t+1} - D_t|$, that will be the basis of an estimate for an exponentially scaled martingale difference (Lemma 4.9).

From estimating the negative effect of iterations with samples in the gap and Lemma 4.3, we obtain the following drift estimate.

LEMMA 4.4. *Let K' be a sufficiently large constant. If in some iteration t all pheromone values are at least $\frac{1}{3}$ and $D_t \geq \max\{K', 2k\}$, then*

$$E[D_{t+1} - D_t \mid D_t] \leq -\frac{\rho}{44} \sqrt{D_t}.$$

Applying a concentration result for sums of independent random variables, we prove the following concentration result on D_t .

LEMMA 4.5. *For any iteration t , and any $\delta \geq 0$, we have*

$$\Pr[|D_{t+1} - D_t| \geq \delta \mid D_t] \leq 4 \exp\left(-\frac{\delta^2}{3\rho^2 D_t}\right) + 2 \exp\left(-\frac{\delta}{3\rho}\right).$$

Lemma 4.5 yields the following estimate, which will be used later to prove that $E[\exp(\lambda(Y_t - Y_{t-1})) \mid Y_0, \dots, Y_{t-1}] \leq 1$ with a well chosen value of λ .

LEMMA 4.6. *Let K'' be a sufficiently large constant, c some positive constant and $\lambda \in (0, c(\rho\sqrt{n})^{-1}]$. For any iteration t such that $D_t \geq K''$, and for n sufficiently large, we have*

$$\lambda^2 E\left[|D_{t+1} - D_t|^2 e^{\lambda|D_{t+1} - D_t|} \mid D_t\right] \leq \varphi(c)(\lambda\rho\sqrt{D_t})^2,$$

with $\varphi(c) = 24e^{\frac{3c^2}{4}} \left(1 + \frac{3\sqrt{3}\pi}{2}c + \frac{3}{4}\sqrt{3}\pi c^3\right)$.

4.4 Proof of the Runtime in the First Part

Let us consider a run of 2-MMAS_{ib} on a subjump function with jump size k . Let $K \geq \max\{2k, K', K''\}$, where K' and K'' are the constants from Lemmas 4.4 and 4.6. Let T be the first time when $n - \|\tau_t\|_1$ is smaller than K or a pheromone value drops below $1/3$. We divide the process in this part into several smaller phases. For $i = 1, 2, \dots$, let $d_i = 2^{-i}n$. Without loss of generality, we may assume that $K = 2^{-\ell-1}n$ for some $\ell \in \mathbb{N}$. Note that $\ell \leq \log_2 n$. We say that the optimization process enters Phase i (and thus leaves its current phase) when for the first time $D_t \leq d_i$. Note that we stay in Phase i even when after entering this phase D_t increases beyond d_i . Note further that, by definition, the process starts in Phase 1. We also say that the current phase ends when a pheromone value reaches a value below $1/3$.

Let t_i be the time when phase i begins, T_i its duration, and $\varepsilon = \frac{\rho}{88} \sqrt{d_{i+1}}$. In this section, we will call D_t the process defined by $D_0 := n - \|\tau_t\|_1$, and for $t \geq 1$,

$$D_t := \begin{cases} n - \|\tau_{t_i+t}\|_1 & \text{if } D_{t-1} > d_{i+1}, \\ D_{t-1} - \varepsilon & \text{if } D_{t-1} \leq d_{i+1}. \end{cases}$$

Therefore we can write

$$\begin{aligned} T_i &= \min\{t \geq 0 : n - \|\tau_{t_i+t}\|_1 < d_{i+1} \text{ or } \exists j \in [1..n], \tau_{j,t_i+t} < 1/3\} \\ &= \min\{t \geq 0 : D_t < d_{i+1} \text{ or } \exists j \in [1..n], \tau_{j,t_i+t} < 1/3\}. \end{aligned}$$

We are only interested in the study of D_t until T_i (while it is larger than d_{i+1}). The definition we gave for $t > T_i$ has the sole purpose of guaranteeing the result of Corollary 4.8, which will allow us to conduct smoothly our further analysis.

LEMMA 4.7. *For any $t \in [0..T_i - 1]$, we have*

$$E[D_{t+1} - D_t \mid D_t] \leq -\varepsilon - \frac{\rho}{88} \sqrt{D_t}.$$

PROOF. If $t \in [0..T_i - 1]$, we have $D_t = n - \|\tau_{t_i+t}\|_1 \geq d_{i+1} \geq K$, and all pheromone values are at least $1/3$. Hence, from Lemma 4.4,

$$E[D_{t+1} - D_t \mid D_t] \leq -\frac{\rho}{44} \sqrt{D_t} \leq -\varepsilon - \frac{\rho}{88} \sqrt{D_t}. \quad \square$$

Now we define $Y_t := D_t + \varepsilon t$ for $t \geq 0$ to exploit this drift and the martingale concentration properties.

COROLLARY 4.8. *$(Y_t)_{t \geq 0}$ is a supermartingale.*

PROOF. For $t \geq T_i$, $E[D_{t+1} - D_t \mid D_t] = E[-\varepsilon \mid D_t] = -\varepsilon$. In the other case where $t < T_i$, the previous lemma gives $E[D_{t+1} - D_t \mid D_t] \leq -\varepsilon$. We deduce immediately that $(Y_t)_t$ is a supermartingale. $\square$

LEMMA 4.9. *Let c be the unique solution of $c\tilde{\varphi}(c) = 1$ with $\tilde{\varphi}(c) := 176e^{\frac{c}{88}}\varphi(c) + \frac{1}{44}e^{\frac{c}{44}}$ ($c \approx 2.36 \cdot 10^{-4}$). Then for n sufficiently large, $\lambda = c(\rho\sqrt{n})^{-1}$, and any $t \geq 0$, we have*

$$E[\exp(\lambda(Y_{t+1} - Y_t)) \mid Y_t] \leq 1.$$

PROOF. Assume that n is sufficiently large so that the inequality in Lemma 4.6 holds. Let $t \geq 0$. If $t \geq T_i$, then $Y_{t+1} = Y_t$, and $E[\exp(\lambda(Y_{t+1} - Y_t)) | Y_t] = 1$.

If $t < T_i$, using the well-known estimate $e^x \leq 1 + x + \frac{x^2}{2}e^{|x|}$ valid for all $x \in \mathbb{R}$, we obtain

$$E[\exp(\lambda(Y_{t+1} - Y_t)) | Y_t] \leq 1 + \lambda E[Y_{t+1} - Y_t | Y_t] + \frac{\lambda^2}{2} E[|Y_{t+1} - Y_t|^2 \exp(\lambda|Y_{t+1} - Y_t|) | Y_t].$$

From Lemma 4.7, we have

$$\lambda E[Y_{t+1} - Y_t | Y_t] = \lambda E[\varepsilon + D_{t+1} - D_t | D_t] \leq -\frac{1}{88} \lambda \rho \sqrt{D_t}.$$

We further compute

$$\begin{aligned} & \frac{\lambda^2}{2} E[|Y_{t+1} - Y_t|^2 \exp(\lambda|Y_{t+1} - Y_t|) | Y_t] \\ & \leq \frac{\lambda^2}{2} e^{\lambda \varepsilon} E[(|D_{t+1} - D_t| + \varepsilon)^2 \exp(\lambda|D_{t+1} - D_t|) | D_t] \\ & \leq 2\lambda^2 e^{\lambda \varepsilon} (E[|D_{t+1} - D_t|^2 \exp(\lambda|D_{t+1} - D_t|) | D_t] + e^{\lambda \varepsilon} \varepsilon^2) \\ & \leq 2e^{\lambda \varepsilon} \varphi(c) (\lambda \rho \sqrt{D_t})^2 + 2e^{2\lambda \varepsilon} \lambda^2 \varepsilon^2 \leq \frac{\tilde{\varphi}(c)}{88} (\lambda \rho \sqrt{D_t})^2. \end{aligned}$$

For the first inequality, we used $|Y_{t+1} - Y_t| \leq |D_{t+1} - D_t| + \varepsilon$. The second one results from the identity $(x + \varepsilon)^2 e^{\lambda x} \leq 4x^2 e^{\lambda x} + 4\varepsilon^2 e^{\lambda \varepsilon}$ valid for every $x \in \mathbb{R}$. Finally, we use Lemma 4.6, and then $\lambda \varepsilon \leq \frac{1}{88} \lambda \rho \sqrt{D_t} \leq \frac{c}{88}$. We deduce that

$$\begin{aligned} E[\exp(\lambda(Y_{t+1} - Y_t)) | Y_t] & \leq 1 - \frac{1}{88} \lambda \rho \sqrt{D_t} + \frac{1}{88} \tilde{\varphi}(c) (\lambda \rho \sqrt{D_t})^2 \\ & \leq 1 + \frac{1}{88} (\lambda \rho \sqrt{D_t}) (c\tilde{\varphi}(c) - 1) = 1. \end{aligned}$$

□

LEMMA 4.10. Let $\tilde{T}_i := \left\lceil 176 \frac{\sqrt{d_{i+1}}}{\rho} \right\rceil$. If n is sufficiently large, then

$$\Pr[T_i > \tilde{T}_i] \leq \exp\left(-\frac{cd_{i+1}}{\rho \sqrt{n}}\right).$$

PROOF. Given that $D_0 \leq d_i = \frac{\rho \sqrt{d_{i+1}}}{88} \cdot 176 \frac{\sqrt{d_{i+1}}}{\rho} \leq \varepsilon \tilde{T}_i$, we have the following implications.

$$T_i > \tilde{T}_i \Rightarrow D_{\tilde{T}_i} \geq d_{i+1} \Rightarrow D_{\tilde{T}_i} + \varepsilon \tilde{T}_i - D_0 \geq d_{i+1} \Rightarrow Y_{\tilde{T}_i} - Y_0 \geq d_{i+1}$$

For n sufficiently large, the previous lemma allows us to apply Lemma 4.1 on $(Y_t)_t$ with $\lambda = c(\rho \sqrt{n})^{-1}$, c being the constant defined in the previous lemma. We therefore have

$$\begin{aligned} \Pr[T_i > \tilde{T}_i] & \leq \Pr[Y_{\tilde{T}_i} - Y_0 \geq d_{i+1}] \\ & \leq \exp(-\lambda d_{i+1}) = \exp\left(-\frac{cd_{i+1}}{\rho \sqrt{n}}\right). \end{aligned} \quad \square$$

We now have a runtime for each phase $i \in [1..l]$ with high probability. We deduce from this an upper bound with high probability for $T := \sum_{i=1}^l T_i$, that is the first time when $n - \|\tau_t\|_1$ goes below K .

THEOREM 4.11. We consider a run of the 2-MMAS_{ib} on a subjump function with a jump size k . Let $K \geq \max\{2k, K', K''\}$, where K' and K'' are the constants from Lemmas 4.4 and 4.6, and such that $\frac{n}{k}$ is a power of two. Let c be the constant from Lemma 4.9. Let T be the

first time such that $D_t := n - \|\tau_t\|_1 \leq K$ or a pheromone value drops below $1/3$. If $\frac{K}{\rho \sqrt{n}} \geq (\ln 2)/c$, then

$$\Pr\left[T \leq 88(2 + \sqrt{2}) \frac{\sqrt{n}}{\rho}\right] = 1 - \exp\left(-\Omega\left(\frac{K}{\rho \sqrt{n}}\right)\right).$$

COROLLARY 4.12. If $K = \omega(1)$ and $\rho = O\left(\frac{1}{\sqrt{n \log n}}\right)$, then

$$\Pr\left[T \leq 88(2 + \sqrt{2}) \frac{\sqrt{n}}{\rho}\right] = 1 - \frac{1}{n^{\Omega(k)}} = 1 - \frac{1}{n^{\omega(1)}}.$$

5 RUNTIME OF THE SECOND PART

In this second part, we consider that D_t has reached in some iteration t_0 a value that is $O(\log n)$. We will first argue that once such a value has been reached, D_t stays that small long enough after t_0 with high probability. Then we will show that the probability of not sampling the optimum $(1, \dots, 1)$ during these iterations is $o(1)$.

LEMMA 5.1. Let t_0 be the first iteration such that $D_{t_0} \leq \alpha \ln n$ for some positive constant $\alpha < 1$. If $\rho \leq 3/4$, then for every $T \leq \frac{\alpha(1-\rho)}{8\rho \ln n}$, and with probability at least $1 - 2T \exp\left(-\frac{\ln^2 n}{3}\right)$, we have for any $t \in [t_0..t_0 + T]$: $D_t \leq 2D_{t_0}$.

LEMMA 5.2 (SAMPLING THE OPTIMUM WHEN CLOSE ENOUGH). Let t_0 be the first iteration such that $D_{t_0} \leq \alpha \ln n$ for some $\alpha < 1/(6 \ln 3)$. If all pheromone values stay larger than $1/3$ for the next $\sqrt{n}$ iterations and $\rho = O\left(\frac{1}{\sqrt{n \log n}}\right)$, then with probability $1 - n^{-\omega(1)}$ the optimum is sampled within the next $\sqrt{n}$ iterations.

From Lemma 7 in [11], we know that if all pheromone values are larger than some constant β , then $\Pr[\text{Sample}(\tau) = (1, \dots, 1)] \geq \beta^{\frac{n - \|\tau\|_1}{1-\beta}}$. This, combined with Lemma 5.1, gives the result.

6 FINAL RESULT

THEOREM 6.1. Let $\alpha < 1/(6 \ln 3)$. We consider a run of the 2-MMAS_{ib} on a subjump function with jump size $k \leq (\alpha/2) \ln n$ and $\rho = O((\sqrt{n \log n})^{-1})$. Let T be the first time such that the optimum $(1, \dots, 1)$ is sampled or a pheromone value $\tau_{j,t}$ drops below $1/3$. Then with probability $1 - n^{-\omega(1)}$ we have

$$T \leq \left(88(2 + \sqrt{2}) + 1\right) \frac{\sqrt{n}}{\rho}.$$

THEOREM 6.2. Let $\alpha < 1/(6 \ln 3)$ and $C \leq 1/(72(88(2 + \sqrt{2}) + 1))$ two positive constants. We consider a run of the 2-MMAS_{ib} on a subjump function with jump size $k \leq (\alpha/2) \ln n$ and $\rho \leq C(\sqrt{n \log n})^{-1}$. Then with probability $1 - o(1)$, the optimum $(1, \dots, 1)$ is sampled within less than $\left(88(2 + \sqrt{2}) + 1\right) \sqrt{n}/\rho$ iterations.

PROOF. Let $A := \left(88(2 + \sqrt{2}) + 1\right)$ and $T = A \frac{\sqrt{n}}{\rho}$. Using Theorem 3.2, we have $\Pr[\forall t \in [0..T], \forall j \in [1..n] : \tau_{j,t} > \frac{1}{3}] = 1 - o(1)$. Hence, using the previous theorem, we have our desired result. □

Figure 2: Mean runtime (along with the first and third quartiles) over 10 runs to find the optimum of JUMP_{n4} for 2-MMAS_{ib} ($\rho = \frac{1}{8\sqrt{n} \ln n}$) and the fast (1 + 1) EA (log scale).

Figure 3: Mean runtime (along with the first and third quartiles) over 10 runs to find the optimum of JUMP_{n4} for 2-MMAS_{ib} ($\rho = \frac{1}{2\sqrt{n} \ln n}$) (log scale).

7 EXPERIMENTS

We implemented Algorithm 1 on jump functions. We first compare its runtime with the runtime of the (1 + 1) EA with a fast mutation operator for $n \in [20..80]$ and a jump size $k = 4$. This maximum value of n is the biggest that we could reach while keeping a reasonable runtime for the EA. As shown in Figure 2, 2-MMAS_{ib} is faster than the (1 + 1) EA with a fast mutation, which in turn is faster than the classical (1 + 1) EA.

We were able to run the 2-MMAS_{ib} on n values that are inaccessible to the (1 + 1) EA, as show in Figure 3.

In Figures 4 and 5, we plot for $n \in \{100, 1000\}$ and $k = 1, \dots, 12$ the smallest integer ℓ for which $\rho = 2^{-\ell}$ is small enough to sample the optimum, and the mean runtime of the algorithm corresponding for this value of ρ .

For both values of n , we observe that for rather small k values, the fitness valley seems to have no effect on the efficiency of the algorithm. However, starting from a threshold value for k , the runtime increases steeply, and the integer ℓ moves to the next values. This threshold is 6 for $n = 100$ and 8 for $n = 1000$. Its small

Figure 4: The smallest integer ℓ needed to sample the optimum for $n = 100$ and $\rho = 2^{-\ell}$, and the corresponding mean runtime.

Figure 5: The smallest integer ℓ needed to sample the optimum for $n = 1000$ and $\rho = 2^{-\ell}$, and the corresponding mean runtime.

variance with respect to n suggests that it is a logarithmic function of n , which confirms the hypothesis $k = O(\log n)$ made in our analysis.

8 CONCLUSION

This is, to the best of our knowledge, the first rigorous analysis of an ACO algorithm on a classic multi-modal benchmark function, and it shows that the 2-MMAS_{ib} can optimize jump functions with small jump sizes with high probability in asymptotically the same time as the simple ONEMAX function. The 2-MMAS_{ib} is thus little affected by the valley of low fitness.

Being a mathematical runtime analysis, this work relies on the precise definition of the ACO algorithm and the jump problem, and thus does not immediately extend to other algorithms or other problems. Our mathematical proofs suggest that one reason for our positive result is that ACO algorithms do not react too drastically on single fitness evaluations. This gives some hope that other ACO algorithms share the advantage in traversing valleys of low fitness. Supporting this believe with rigorous results is an interesting direction for future research.

REFERENCES

- [1] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. 2021. Lazy parameter tuning and control: choosing all parameters randomly from a power-law distribution. In *Genetic and Evolutionary Computation Conference, GECCO 2021*. ACM.
- [2] Denis Antipov and Benjamin Doerr. 2020. Runtime analysis of a heavy-tailed $(1 + (\lambda, \lambda))$ genetic algorithm on jump functions. In *Parallel Problem Solving From Nature, PPSN 2020, Part II*. Springer, 545–559.
- [3] Denis Antipov, Benjamin Doerr, and Vitalii Karavaev. 2020. The $(1 + (\lambda, \lambda))$ GA is even faster on multimodal problems. In *Genetic and Evolutionary Computation Conference, GECCO 2020*. ACM, 1259–1267.
- [4] Anne Auger and Benjamin Doerr (Eds.). 2011. *Theory of Randomized Search Heuristics*. World Scientific Publishing.
- [5] Henry Bambury, Antoine Bultel, and Benjamin Doerr. 2021. Generalized jump functions. In *Genetic and Evolutionary Computation Conference, GECCO 2021*. ACM.
- [6] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. 2009. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing* 8 (2009), 239–287.
- [7] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2017. On the runtime analysis of the Opt-IA artificial immune system. In *Genetic and Evolutionary Computation Conference, GECCO 2017*. ACM, 83–90.
- [8] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2018. Fast artificial immune systems. In *Parallel Problem Solving from Nature, PPSN 2018, Part II*. Springer, 67–78.
- [9] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2016. Escaping local optima with diversity mechanisms and crossover. In *Genetic and Evolutionary Computation Conference, GECCO 2016*. ACM, 645–652.
- [10] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2018. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation* 22 (2018), 484–497.
- [11] Benjamin Doerr. 2019. A tight runtime analysis for the cGA on jump functions: EDAs can cross fitness valleys at no extra cost. In *Genetic and Evolutionary Computation Conference, GECCO 2019*. ACM, 1488–1496.
- [12] Benjamin Doerr. 2020. Does comma selection help to cope with local optima?. In *Genetic and Evolutionary Computation Conference, GECCO 2020*. ACM, 1304–1313.
- [13] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2018. Static and self-adjusting mutation strengths for multi-valued decision variables. *Algorithmica* 80 (2018), 1732–1768.
- [14] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2019. Solving problems with unknown solution length at almost no extra cost. *Algorithmica* 81 (2019), 703–748.
- [15] Benjamin Doerr, Edda Happ, and Christian Klein. 2012. Crossover can provably be useful in evolutionary computation. *Theoretical Computer Science* 425 (2012), 17–33.
- [16] Benjamin Doerr and Martin S. Krejca. 2020. The univariate marginal distribution algorithm copes well with deception and epistasis. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2020*. Springer, 51–66.
- [17] Benjamin Doerr, Huu Phuoc Le, Régis Makhlara, and Ta Duy Nguyen. 2017. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*. ACM, 777–784.
- [18] Benjamin Doerr and Frank Neumann (Eds.). 2020. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer. Also available at <https://cs.adelaide.edu.au/~frank/papers/TheoryBook2019-selfarchived.pdf>.
- [19] Benjamin Doerr, Frank Neumann, Dirk Sudholt, and Carsten Witt. 2011. Runtime analysis of the 1-ANT ant colony optimizer. *Theoretical Computer Science* 412 (2011), 1629–1644.
- [20] Benjamin Doerr and Weijie Zheng. 2020. Sharp bounds for genetic drift in estimation-of-distribution algorithms. *IEEE Transactions on Evolutionary Computation* 24 (2020), 1140–1149.
- [21] Benjamin Doerr and Weijie Zheng. 2021. Theoretical analyses of multi-objective evolutionary algorithms on multi-modal objectives. In *Conference on Artificial Intelligence, AAAI 2021*. AAAI Press. To appear.
- [22] Marco Dorigo and Thomas Stützle. 2004. *Ant colony optimization*. MIT Press.
- [23] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the $(1+1)$ evolutionary algorithm. *Theoretical Computer Science* 276 (2002), 51–81.
- [24] Xiequan Fan, Ion Grama, and Quansheng Liu. 2015. Exponential inequalities for martingales with applications. *Electronic Journal of Probability* 20 (2015), 22 pp.
- [25] Matthias Feldmann and Timo Kötzing. 2013. Optimizing expected path lengths with ant colony optimization using fitness proportional update. In *Foundations of Genetic Algorithms, FOGA 2013*. ACM, 65–74.
- [26] Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Samadhi Nallaperuma, Frank Neumann, and Martin Schirneck. 2016. Fast building block assembly by majority vote crossover. In *Genetic and Evolutionary Computation Conference, GECCO 2016*. ACM, 661–668.
- [27] Tobias Friedrich, Timo Kötzing, Martin S. Krejca, and Andrew M. Sutton. 2016. Robustness of ant colony optimization to noise. *Evolutionary Computation* 24 (2016), 237–254.
- [28] Walter J. Gutjahr. 2006. On the finite-time dynamics of ant colony optimization. *Methodology and Computing in Applied Probability* 8 (2006), 105–133.
- [29] Walter J. Gutjahr. 2008. First steps to the runtime complexity analysis of ant colony optimization. *Computers & Operations Research* 35 (2008), 2711–2727.
- [30] Walter J. Gutjahr. 2011. Ant colony optimization: recent developments in theoretical analysis. In *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, Anne Auger and Benjamin Doerr (Eds.). Vol. 1. World Scientific, 225–254.
- [31] Walter J. Gutjahr and Giovanni Sebastiani. 2008. Runtime analysis of ant colony optimization with best-so-far reinforcement. *Methodology and Computing in Applied Probability* 10 (2008), 409–433.
- [32] Václav Hasenöhrl and Andrew M. Sutton. 2018. On the runtime dynamics of the compact genetic algorithm on jump functions. In *Genetic and Evolutionary Computation Conference, GECCO 2018*. ACM, 967–974.
- [33] Jens Jägersküpper and Tobias Storch. 2007. When the plus strategy outperforms the comma strategy and when not. In *Foundations of Computational Intelligence, FOCI 2007*. IEEE, 25–32.
- [34] Thomas Jansen. 2013. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer.
- [35] Thomas Jansen. 2015. On the black-box complexity of example functions: the real jump function. In *Foundations of Genetic Algorithms XIII, FOGA 2015*. ACM, 16–24.
- [36] Thomas Jansen and Ingo Wegener. 2002. The analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica* 34 (2002), 47–66.
- [37] Timo Kötzing, Per Kristian Lehre, Frank Neumann, and Pietro S. Oliveto. 2010. Ant colony optimization and the minimum cut problem. In *Genetic and Evolutionary Computation Conference, GECCO 2010*. ACM, 1393–1400.
- [38] Timo Kötzing and Hendrik Molter. 2012. ACO beats EA on a dynamic pseudo-Boolean function. In *Parallel Problem Solving from Nature, PPSN 2012, Part I*. Springer, 113–122.
- [39] Timo Kötzing, Frank Neumann, Heiko Röglin, and Carsten Witt. 2010. Theoretical properties of two ACO approaches for the traveling salesman problem. In *Swarm Intelligence, ANTS 2010*. Springer, 324–335.
- [40] Guillermo Leguizamón and Enrique Alba. 2013. Ant colony based algorithms for dynamic optimization problems. In *Metaheuristics for Dynamic Optimization*. Springer, 189–210.
- [41] Per Kristian Lehre. 2010. Negative drift in populations. In *Parallel Problem Solving from Nature, PPSN 2010*. Springer, 244–253.
- [42] Per Kristian Lehre and Phan Trung Hai Nguyen. 2019. On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In *Foundations of Genetic Algorithms, FOGA 2019*. ACM, 154–168.
- [43] Per Kristian Lehre and Carsten Witt. 2013. General drift analysis with tail bounds. *CoRR abs/1307.2559* (2013).
- [44] Per Kristian Lehre and Carsten Witt. 2014. Concentrated hitting times of randomized search heuristics with variable drift. In *International Symposium on Algorithms and Computation, ISAAC 2014*. Springer, 686–697.
- [45] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. 2019. On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation. In *Conference on Artificial Intelligence, AAAI 2019*. AAAI Press, 2322–2329.
- [46] Andrei Lissovoi and Carsten Witt. 2016. MMAS versus population-based EA on a family of dynamic fitness functions. *Algorithmica* 75 (2016), 554–576.
- [47] Manuel López-Ibáñez, Thomas Stützle, and Marco Dorigo. 2018. Ant colony optimization: a component-wise overview. In *Handbook of Heuristics*, Rafael Martí, Panos M. Pardalos, and Mauricio G. C. Resende (Eds.). Springer International Publishing, Cham, 371–407.
- [48] Frank Neumann, Dirk Sudholt, and Carsten Witt. 2008. Rigorous analyses for the combination of ant colony optimization and local search. In *Ant Colony Optimization and Swarm Intelligence, ANTS 2008*. Springer, 132–143.
- [49] Frank Neumann, Dirk Sudholt, and Carsten Witt. 2010. A few ants are enough: ACO with iteration-best update. In *Genetic and Evolutionary Computation Conference, GECCO 2010*. ACM, 63–70.
- [50] Frank Neumann and Carsten Witt. 2009. Runtime analysis of a simple ant colony optimization algorithm. *Algorithmica* 54 (2009), 243–255.
- [51] Frank Neumann and Carsten Witt. 2010. Ant colony optimization and the minimum spanning tree problem. *Theoretical Computer Science* 411 (2010), 2406–2413.
- [52] Frank Neumann and Carsten Witt. 2010. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer.
- [53] Pietro S. Oliveto, Tiago Paixão, Jorge Pérez Heredia, Dirk Sudholt, and Barbora Trubenová. 2018. How to escape local optima in black box optimisation: when non-elitism outperforms elitism. *Algorithmica* 80 (2018), 1604–1633.
- [54] Tiago Paixão, Jorge Pérez Heredia, Dirk Sudholt, and Barbora Trubenová. 2017. Towards a runtime comparison of natural and artificial evolution. *Algorithmica* 78 (2017), 681–713.

- [55] Amirhossein Rajabi and Carsten Witt. 2020. Self-adjusting evolutionary algorithms for multimodal optimization. In *Genetic and Evolutionary Computation Conference, GECCO 2020*. ACM, 1314–1322.
- [56] Amirhossein Rajabi and Carsten Witt. 2021. Stagnation detection in highly multimodal fitness landscapes. In *Genetic and Evolutionary Computation Conference, GECCO 2021*. ACM.
- [57] Amirhossein Rajabi and Carsten Witt. 2021. Stagnation detection with randomized local search. In *Evolutionary Computation in Combinatorial Optimization, EvoCOP 2021*. Springer, 152–168.
- [58] Jonathan E. Rowe and Aishwaryaprajna. 2019. The benefits and limitations of voting mechanisms in evolutionary optimisation. In *Foundations of Genetic Algorithms, FOGA 2019*. ACM, 34–42.
- [59] Thomas Stützle and Holger H. Hoos. 2000. MAX-MIN ant system. *Future Generation Computer Systems* 16, 8 (2000), 889–914.
- [60] Dirk Sudholt and Christian Thyssen. 2012. A simple ant colony optimizer for stochastic shortest path problems. *Algorithmica* 64 (2012), 643–672.
- [61] Dirk Sudholt and Carsten Witt. 2019. On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization. *Algorithmica* 81 (2019), 1450–1489.
- [62] Darrell Whitley, Swetha Varadarajan, Rachel Hirsch, and Anirban Mukhopadhyay. 2018. Exploration and exploitation without mutation: solving the jump function in $\Theta(n)$ time. In *Parallel Problem Solving from Nature, PPSN 2018, Part II*. Springer, 55–66.
- [63] Yuren Zhou. 2009. Runtime analysis of an ant colony optimization algorithm for TSP instances. *IEEE Transactions on Evolutionary Computation* 13 (2009), 1083–1092.