Ring Migration Topology Helps Bypassing Local Optima

Clemens Frahnow(✉) and Timo Kötzing

Hasso Plattner Institute, Prof.-Dr.-Helmert-Straße 2-3, Potsdam 14482, Germany
hpi-info@hpi.de
https://hpi.de/friedrich

Abstract. Running several evolutionary algorithms in parallel and occasionally exchanging good solutions is referred to as island models. The idea is that the independence of the different islands leads to diversity, thus possibly exploring the search space better. Many theoretical analyses so far have found a complete (or sufficiently quickly expanding) topology as underlying migration graph most efficient for optimization, even though a quick dissemination of individuals leads to a loss of diversity.

We suggest a simple fitness function Fork with two local optima parametrized by $r \geq 2$ and a scheme for composite fitness functions. We show that, while the $(1+1)$ EA gets stuck in a bad local optimum and incurs a run time of $\Theta(n^{2r})$ fitness evaluations on Fork, island models with a complete topology can achieve a run time of $\Theta(n^{1.5r})$ by making use of rare migrations in order to explore the search space more effectively. Finally, the ring topology, making use of rare migrations and a large diameter, can achieve a run time of $\tilde{\Theta}(n^r)$, the black box complexity of Fork. This shows that the ring topology can be preferable over the complete topology in order to maintain diversity.

Keywords: Evolutionary computation · Island models
Ring topology · Run time analysis

1 Introduction

In heuristic optimization, evolutionary algorithms are a technique that is capable of finding good solutions by employing strategies inspired by evolution [2]. One way to understand why some optimization algorithms are more successful than others is to prove rigorous run time bounds on test functions which embody a typical challenge occurring in realistic optimization problems. For example, the famous OneMax function, which assigns the number of 1s of a bit string x as the fitness of x, embodies the challenge of solving independent problems concurrently. The LeadingOnes function, counting the number of leading 1s of a bit string, embodies the problem of solving otherwise independent problems sequentially (where the order is typically unknown). Thus, OneMax and

A. Auger et al. (Eds.): PPSN 2018, LNCS 11102, pp. 129–140, 2018.
https://doi.org/10.1007/978-3-319-99259-4_11

LeadingOnes are fruitful test functions to analyze search heuristics on, they simulate important properties of realistic search spaces in an analyzable way.

We introduce a new representative fitness function Fork which poses a choice of two possible directions, the fork. One of the directions is a dead end, a local and not global optimum; we call this the *valley*. The other is the global optimum. We use a parameter $r > 1$ and formalize Fork by using OneMax, but assigning two elements with (disjoint) sets of r 0s to be the global optimum and the valley. Thus, once trapped in the valley, it is hard to find the global optimum. Since the probability of finding the global optimum before the valley is exactly $1/2$ due to the symmetry of the search space (see also Lemma 4), making random restarts with the well-known (1+1) EA (or about $\log n$ independent runs) can efficiently find the global optimum.

However, for realistic optimization problems, forks can happen not just as the last step of the search, but over and over again. The probability that a run will succeed and choose the right path each time decreases exponentially with the number of fork decisions to be made. We formalize this with *composite fitness functions*. We give a general scheme for building fitness functions out of base functions by dividing the bit string into blocks. As an example for solving k successive Fork functions, we can divide the bit string of length n into k equal parts. Each part contributes to the total fitness with its Fork-value, but only if all previous blocks are already optimized; this scheme was already used in essence by [11]. Intuitively, the resulting composite fitness function is like LeadingOnes, where each bit is a Fork function on bit strings of length n/k. Clearly, the (1+1) EA as well as independent runs on such a succession of Fork functions are unlikely to succeed.

Exactly to deal with such fitness landscapes, different techniques have been introduced. One possible way in evolutionary computation is to employ *island models*, meaning multiple computing agents (so called islands) that run the same algorithm in parallel and which can share information. Various analyses of island models have been made that show its usefulness: Alba used parallel evolutionary algorithms to achieve super-linear speedups [1]. Lässig and Sudholt gave a formal analysis of island models for many different migration topologies in [12] showing how one can gain a speedup from parallelism; Badkobeh, Lehre and Sudholt could even show where the cut-off points are from which on linear speedup is no longer possible and discovered some bounds on different topologies [3]. In 2017, Lissovoi and Witt explored the performance of a parallel approach on dynamic optimization problems [13]. In all these works, the idea is to exploit the computing power that comes along with multiple islands, gaining a speedup from parallelism. Intuitively, a set up where each islands sends its best solution to all islands (called a *complete migration topology*) as often as possible leads to the smallest run times, since all islands can share the progress of all others. Doerr et al. showed that if one considers the communication between islands also as time consuming, Rumor Spreading or Binary Trees perform even better on OneMax and LeadingOnes [4], but still the emphasis is on informing all islands as efficiently as possible about every improvement found.

An essentially different work was given by Lässig and Sudholt in [11]. They used a composite fitness function where each component tries to trick the algorithm to walk up a path leading to a local optimum, which is hard to escape (similar to Fork introduced above, but here we have paths that lead to the local optima). They give an island setting that can efficiently optimize this composite function, while simple hill climbers get stuck with high probability. Note that the complete topology also performs well in this setting, even though such high connectivity typically implies the loss of diversity, which was found important in many areas of heuristic optimization. Here the diversity was maintained by focusing on rare migration and making sure that migration only occurs at opportune times.

In this paper we want to show that a high connectivity in a topology, for any frequency of migration, can lead to a loss of diversity and therefore to worse run times on Fork. In contrast to this we will show that the ring topology allows to maintain diversity. We choose the ring on λ vertices, since it is the unique graph with maximal diameter among all vertex transitive graphs with λ vertices, in contrast to the complete graph, which has minimal diameter, thus highlighting the role of a large diameter (which implies a slow spread of migrants).

First, in Sect. 2, we introduce the algorithms we deal with. Section 3 introduces the fitness functions more formally, especially Fork and our scheme for composite fitness functions; here we also give a general result for the (1+1) EA applied to such composite fitness functions which is of independent interest.

In Sect. 4 we show that the (1+1) EA fails to optimize Fork efficiently, with an expected run time of $\Theta(n^{2r})$ (see Theorem 5). Independent runs of the (1+1) EA similarly fail for compositions of Fork-functions.

Regarding island models, while it is typical to consider as optimization time the time until just one island has found the optimum, we consider the time until *all* islands have found the optimum: consider the case of optimizing k successive Fork functions as introduced above. In order to be able to continue optimization after the first Fork function has been optimized, we need a sufficient number of islands which have passed this first phase; if we were to lose a constant fraction for each Fork function, then quickly all islands would be used up and the algorithm will get stuck in a local optimum. If *all* islands make it to the next stage, then the optimization can proceed as in the first stage. We leave the rigorous argument to future work and contend ourselves with finding the time until all islands find the optimum of Fork, showing in what way the ring topology can be beneficial and, in fact, preferable to the complete topology.

In Sect. 5 we consider an island model with the complete topology, that is, the different islands run a (1+1) EA, but they occasionally share their best individual with *all* other islands. In particular, we use a parameter τ such that each round with probability $1/\tau$ each island sends its best (and only current) individual to all other islands, continuing the search with the best individual among all incoming and own individuals.[1] For optimal choice of τ and the number of islands λ, the

[1] Note that in some papers migration is considered to happen deterministically every τ rounds.

time until all islands have found the optimum here is $\Theta(n^{1.5r})$ fitness evaluations (see Corollary 17).

Next, in Sect. 6, we show that the ring topology requires only $O(n^r(\log n)^2)$ fitness evaluations until all islands have found the optimum (see Corollary 24), which equals, up to polylogarithmic factors, the black-box complexity of FORK, which is $\Theta(n^r)$ (see Proposition 1). In this sense the ring topology achieves the best possible optimization time over all black-box algorithms (up to the factor of $(\log n)^2$).

Finally, in Sect. 7, we conclude the paper with some final remarks. Almost all proofs are omitted due to space constraints, but they can be found in "Ring Migration Topology Helps Bypassing Local Optima" on https://arxiv.org.

2 Algorithms

The island model makes use of the (1+1) Evolutionary Algorithm ((1+1) EA for brevity). The goal of that algorithm is to maximize a given fitness function by trying different search points and remembering the one that gave the best result so far. The fitness function is defined on bit strings of a specific length n. The algorithm starts with a bit string chosen uniformly at random. A new individual for the input is generated each step by taking the best known input and flipping every bit independently with a probability of $\frac{1}{n}$ (*standard bit mutation*). If the fitness function yields a value that is not smaller than the best known so far, it becomes the new best individual. Algorithm 1 makes this more formal.

Algorithm 1. (1+1) EA optimizing f.

```
1 t ← 0;
2 x ← solution drawn u.a.r. from {0,1}^n;
3 while termination criterion not met do
4 |   t ← t + 1;
5 |   y ← flip each bit of x independently w/ prob. 1/n;
6 |   if f(y) ≥ f(x) then x ← y ;
```

Usually the termination criterion is met when the optimum is found. In later chapters we let this algorithm run in parallel multiple times until the optimum is found everywhere. In this case we change the termination criterion accordingly. The run time of the (1+1) EA is determined by the value of t after the algorithm terminates. For our research we use the *island model* as an approach on parallel evolutionary computation, cf. [12,14,15]. The topology is defined by an undirected graph $G = (V, E)$, where $\lambda = |V|$. Every vertex, called *island*, represents an independent agent running the (1+1) EA using standard bit mutation. Like in the (1+1) EA, the initial bit string is chosen uniformly at random. This happens independently on every island. All islands run in lockstep, meaning

they all make the same amount of fitness evaluations in the same time. Copies of the best found individual so far are shared along the edges of G whenever a migration step happens. An island overwrites its best solution when a received individual has a fitness that is not smaller than the resident best individual. Ties among incoming migrants (with maximum fitness) are broken uniformly at random. With a probability of $\frac{1}{\tau}$, every island sends its best individual to all of its neighbors. Algorithm 2 makes this more formal, where $\mathbf{x}^{(j)}$ denotes the best individual on island j. Again, t determines the run time (*optimization time*) we are mainly interested in, as it counts the number of fitness evaluations of a single island. If multiplied by λ, one gains the total number of fitness evaluations.

Algorithm 2. Island model with migration topology $G = (V, E)$ on λ islands and migration probability $1/\tau$.

$t \leftarrow 0$;
for $1 \leq j \leq \lambda$ **in parallel do**
  $\mathbf{x}^{(j)} \leftarrow$ solution drawn u.a.r. from $\{0,1\}^n$;
while *termination criterion not met* **do**
  $t \leftarrow t + 1$;
  $m \leftarrow$ **true** with probability $1/\tau$ else **false**;
  for $1 \leq j \leq \lambda$ **in parallel do**
    $\mathbf{y}^{(j)} \leftarrow$ flip each bit of $\mathbf{x}^{(j)}$ independently w/ prob. $1/n$;
    if $f(\mathbf{y}^{(j)}) \geq f(\mathbf{x}^{(j)})$ **then** $\mathbf{x}^{(j)} \leftarrow \mathbf{y}^{(j)}$;
    if m **then**
      Send $\mathbf{x}^{(j)}$ to all islands k with $\{j, k\} \in E$;
      $N = \{\mathbf{x}^{(i)} \mid \{i, j\} \in E\}$;
      $M = \{\mathbf{x}^{(i)} \in N \mid f(\mathbf{x}^{(i)}) = \max_{\mathbf{x} \in N} f(\mathbf{x})\}$;
      $\mathbf{y}^{(j)} \leftarrow$ solution drawn u.a.r. from M;
      if $f(\mathbf{y}^{(j)}) \geq f(\mathbf{x}^{(j)})$ **then** $\mathbf{x}^{(j)} \leftarrow \mathbf{y}^{(j)}$;

Observe that the final value of t is a random variable; in this paper whenever we use T we refer to this random variable.

All bounds in this work will be in terms of n, λ, τ and r simultaneously. We consider $\lambda = \lambda(n)$ and $\tau = \tau(n)$ as positive, non-decreasing, integer-valued functions whereas r is a fixed but arbitrary constant that has to be at least 2. The bounds we give describe the univariate asymptotics of the expected optimization times with respect to n for any choices of λ and τ within the given boundaries.

3 Fitness Functions

In this paper we investigate the maximization of pseudo-Boolean functions $f\colon \{0,1\}^n \to \mathbb{R}_{\geq 0}$ on bit strings $\mathbf{x} = \mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_{n-1}$ of length n. When we talk about the *fitness* of a bit string $\mathbf{x}$ it refers to $f(\mathbf{x})$.

We want to create composites of fitness functions by nesting them into each other. To start, we will examine the run times of the two functions below. Let $|\mathbf{x}|_1$ denote the number of bits set to 1 within $\mathbf{x}$. For $r \geq 2$ and $n \geq 2r$ we define

$$\textsc{LeadingOnes}(\mathbf{x}) = \sum_{i=0}^{n-1} \prod_{j=0}^{i} \mathbf{x}_j \quad \textsc{Fork}_r^n(\mathbf{x}) = \begin{cases} n+1, & \text{if } \mathbf{x} = \{0\}^r\{1\}^{n-r}; \\ n+2, & \text{if } \mathbf{x} = \{1\}^{n-r}\{0\}^r; \\ |\mathbf{x}|_1, & \text{otherwise.} \end{cases}$$

In our work when we talk about the $\textsc{Fork}_r^n$ fitness function we will call the bit strings of fitness $n+1$ *valley*, as getting to the optimum from there is harder than it is from any other fitness. This definition of Fork is not related to the one in the work of Gießen [9].

We start by considering the general difficulty of optimizing $\textsc{Fork}_r^n$ as formalized by the *unrestricted black-box complexity* [7], for which we consider the class of $\textsc{Fork}_r^n$ composed with any automorphism of the hypercube.

Proposition 1. *For constant r, the unrestricted black-box complexity of the $\textsc{Fork}_r^n$ function class is $\Theta(n^r)$.*

Proof. The lower bound comes from having to find at least one out of two search points among all search points with Hamming distance r to the third-best individual, of which there are $\Theta(n^r)$ many. The upper bound comes from being able to find the third-best individual efficiently, for example with a (1+1) EA, and then testing in a fixed order all search points with Hamming distance r to this point. □

3.1 Composite Fitness Function

Here we define composite fitness functions formally. Let $f = (f^n)_{n\in\mathbb{N}}$ be a family of fitness functions such that, for all $n \in \mathbb{N}$, $f^n : \{0,1\}^n \to \mathbb{R}_{\geq 0}$. For this construction, we suppose that 1^n is the unique optimum (if a different bit string is the unique optimum, we apply a corresponding bit mask to make 1^n the unique optimum, but will not mention it any further). With *LeadingOnes with k-block f* we denote the fitness function which divides the input $\mathbf{x}$ into bit strings of length k $((\mathbf{x}_0\mathbf{x}_1\dots\mathbf{x}_{k-1}), (\mathbf{x}_k\mathbf{x}_{k+1}\dots\mathbf{x}_{2k-1}), \dots)$. The blocks contribute to the total fitness using fitness function f^k, but only if all previous blocks have reached the unique optimum 1^k. More formally,

$$LO_k^f(\mathbf{x}) = \sum_{i=0}^{\frac{n}{k}-1} f^k(\mathbf{x}_{ik}\mathbf{x}_{ik+1}\dots\mathbf{x}_{ik+k-1}) \cdot \prod_{j=0}^{ik-1} \mathbf{x}_j.$$

Similarly, OneMax with k-block f is defined as

$$OM_k^f(\mathbf{x}) = \sum_{i=0}^{\frac{n}{k}-1} f^k(\mathbf{x}_{ik}\mathbf{x}_{ik+1}\dots\mathbf{x}_{ik+k-1}).$$

In this paper we only analyze the run times around the LeadingOnes version. Note that it equals the LOB_b function in the work of Jansen and Wiegand [10]. In general, also other fitness functions can be used as the outer function of that nesting method, as exemplified in our definition of OneMax with k-block.

3.2 Run Time of LeadingOnes with k-block f

In the following we develop a general approach for proving run times on LeadingOnes with k-block f. We make use of the observation that there is always exactly one block that contributes to the overall fitness except all previous blocks that are already optimal.

Theorem 2. *Let T be the run time of a* (1+1) EA *Algorithm on LO_k^f to optimize a bit string of length n. We have*

$$E(T) = E\left(T_k^n\right) \frac{\left(\frac{n}{n-1}\right)^n - 1}{\left(\frac{n}{n-1}\right)^k - 1} \in \Theta\left(E\left(T_k^n\right) \frac{n}{k}\right)$$

where T_k^n is the run time to optimize f^k with bit flip probability $\frac{1}{n}$.

With this Theorem as a tool we can now derive exact bounds on functions like LeadingOnes. The following equation was already shown by Sudholt [16, Corollary 1]. Here we use the approach of composite functions, which generalizes to many other fitness functions, but note that the underlying proof idea is the same.

Corollary 3. *The expected run time of a* (1+1) EA *on* LeadingOnes *is exactly*

$$E(T_{LO}(n)) = \frac{\left(\frac{n}{n-1}\right)^{n-1} + \frac{1}{n} - 1}{2} n^2.$$

4 No Migration

In the next sections we will frequently use that m islands have to make a jump from the same fitness level to another one, where for each bit string in the current level the probability to do the jump is the same. When we call E_j the expected run time for a single island to do the jump, we can bound the expected run time $E(T)$ until one of them succeeds by

$$\frac{E_j}{2m} \leq E(T) \leq \frac{E_j}{m} + 1. \tag{1}$$

This holds due to the fact that $E(T)$ is distributed geometrically with success probability $p = \frac{1}{E(T)}$ and because $\frac{pm}{1+pm} \leq 1 - (1-p)^m \leq \frac{2pm}{1+pm}$ as shown by Badkobeh et al. [3]. We already gave an intuition of the following lemma that will be used in several proofs.

Lemma 4. *For any run of the* (1+1) EA *on* Fork_r^n *let V denote the event that the valley string occurs as a best solution before the optimum. Then,* $\Pr(V) = \frac{1}{2}$.

4.1 The (1+1) EA

For the (1+1) EA we can use Lemma 4 to see that it will be trapped with probability 1/2, which leads to the following two theorems.

Theorem 5. *The expected optimization time of the* (1+1) EA *on* $\textsc{Fork}_r^k$ *with bit flip probability* $\frac{1}{n}$ *and* $n \geq k$ *is* $E(T(k)) \in \Theta\left(n^{2r}\right)$.

Theorem 6. *The expected optimization time of the* (1+1) EA *on* LeadingOnes *with k-block* $\textsc{Fork}_r^n$ *is* $E(T) \in \Theta\left(\frac{1}{k} n^{2r+1}\right)$.

4.2 Independent Runs

In this section we examine the performance of λ islands in isolation, all running the (1+1) EA on $\textsc{Fork}_r^n$ or LeadingOnes with k-block $\textsc{Fork}_r^n$. Isolation of the islands means there is no migration between them at all. The next lemma will help us to get to the final bounds.

Lemma 7. *Let* λ *be the number of islands running the* (1+1) EA *optimizing* $\textsc{Fork}_r^n$. *Let* T_{all}^{λ} *denote the run time for all islands to get to the optimum, valley or* 1^n *as their best solution respectively, regardless of the migration topology and policy. If* λ *is polynomial in* n, $E\left(T_{all}^{\lambda}\right) \in \mathrm{O}(n \log n)$.

Theorem 8. *For* $\lambda \leq n^r$ *isolated islands the expected time to optimize* $\textsc{Fork}_r^n$ *is* $E(T) \in \mathrm{O}\left(n \log(n) + \frac{n^{2r}}{\lambda 2^{\lambda}} + \frac{n^r}{\lambda}\right)$.

Corollary 9. *If we use* $r \log n \leq \lambda \leq n^r$ *islands,* $E(T) \in \mathrm{O}\left(n \log(n) + \frac{n^r}{\lambda}\right)$.

Corollary 10. *The expected number of evaluations until all* $\lambda \leq n^r$ *islands get the optimum is in* $\Omega\left(\lambda n^{2r} \log \lambda\right)$.

Proof. To achieve this, all islands have to find the optimum. Observe that we expect half of the islands to get trapped. The probability that there are more than half as much is - by using Chernoff bounds - asymptotically more than a constant. Therefore we expect to need at least $\Omega\left(n^{2r} \log \lambda\right)$ rounds, which results in the given number of evaluations. □

Theorem 11. *For* $k \leq \frac{n}{\log \lambda}$, *the expected run time of* λ *islands optimizing* LeadingOnes *with k-block* $\textsc{Fork}_r^n$ *by running the* (1+1) EA *can be bound by* $E(T) \in \Omega\left(\frac{n^{2r}}{\lambda}\right)$.

Proof. Let V denote the event that every island gets trapped in a valley at least once during a run. From Lemma 4 one can derive that $\Pr(V) = \left(1 - \frac{1}{2^{\frac{n}{k}}}\right)^{\lambda}$.

Again we apply the law of total expectation and use the lower bound $\frac{E}{2\lambda}$ on the expected run time of λ islands until one of them makes a jump where E is the expected number of steps for a single island (Eq. (1)).

$$E(T) = E(T \mid V) \Pr(V) + E\left(T \mid \overline{V}\right) \Pr\left(\overline{V}\right) \geq \frac{n^{2r}}{2\lambda} \Pr(V) = \frac{n^{2r}}{2\lambda} \left(1 - \frac{1}{2^{\frac{n}{k}}}\right)^{\lambda}$$

Using that $k \leq \frac{n}{\log(\lambda)}$ finally gives $E(T) \geq \frac{n^{2r}}{2\lambda} \left(1 - \frac{1}{\lambda}\right)^{\lambda} \in \Omega\left(\frac{n^{2r}}{\lambda}\right)$. □

5 Complete Topology

The disadvantage of the complete topology when optimizing $\textsc{Fork}_r^n$ is that if at least one island gets the chance to migrate the valley, all islands get trapped. To get to the bounds, we first investigate the worst run time that could appear. Second, we calculate a bound on the probability for one island to find the optimum or the valley during the time it takes all islands to get to 1^n, the valley or the optimum.

In this and the following sections we frequently use the worst case run time that can occur if we do not find the optimum early enough.

Lemma 12. *For $\lambda \in \mathrm{O}\left(\frac{n^{2r-1}}{\log n}\right)$ islands being polynomial in n and any migration topology and/or policy, the expected run time to optimize $\textsc{Fork}_r^n$ can be bounded by $E(T) \in \mathrm{O}\left(\frac{n^{2r}}{\lambda}\right)$.*

To get a lower bound on the run time we want to concentrate on the case that all islands come to a state where every island has 1^n as its solution. That is why in the next lemma we show how likely it is that the optimum or the valley is generated before that state is reached.

Lemma 13. *The probability p_{ov} for a single island to generate the optimum or the valley during its way to 1^n while optimizing $\textsc{Fork}_r^k$ with (1+1) EA $_{1/n}$ is $p_{ov} \in \mathrm{O}\left(\frac{1}{k^{r-1}}\right)$ for $k \leq n$.*

Next we give a lemma that we use for the upper and the lower bound, which considers the event that and islands finds the valley and broadcasts it, thus drowning diversity.

Lemma 14. *Under the condition of at least one of $\lambda \in \mathrm{O}\left(n^{r-1}\right)$ island having 1^n and all others the valley as solution, the probability for the event Q that one island will find the valley and a migration will be made before the optimum is found by any island is $\frac{c}{2}\frac{n^r}{n^r+\tau\lambda} \leq \Pr(Q) \leq \frac{n^r+\frac{\lambda}{2}}{2n^r+\tau\lambda}$ for a constant $0 < c < 1$.*

We continue with the lower bound for the complete topology, followed by the upper bound. The restriction for λ to be in $\mathrm{O}\left(\frac{n^{r-1}}{\log n}\right)$ in the next theorems is useful since the expected number of derived optima during the first $\mathrm{O}(n \log n)$ steps is constant if we chose $\lambda \in \Theta\left(n^{r-1}\right)$. This follows from Lemmas 7 and 13.

Theorem 15. *The expected run time of $2r \log n \leq \lambda \in \mathrm{O}\left(\frac{n^{r-1}}{\log n}\right)$ islands optimizing $\textsc{Fork}_r^n$ on a complete graph is $E(T) \in \Omega\left(\frac{n^{3r}+\tau\lambda n^r}{\lambda n^r+\tau\lambda^2}\right)$.*

Theorem 16. *The expected run time of $2 \log n \leq \lambda \in \mathrm{O}\left(\frac{n^{r-1}}{\log n}\right)$ islands optimizing $\textsc{Fork}_r^n$ on a complete graph is in $E(T) \in \mathrm{O}\left(\frac{n^{3r}+\tau\lambda n^r}{\lambda n^r+\tau\lambda^2} + \frac{n^{2r+1}\log n}{\tau\lambda}\right)$.*

Corollary 17. *If we choose $\tau \in \Omega(n \log n)$ and $2r \log n \leq \lambda \in \mathrm{O}(n^{r-1-\varepsilon})$ for $\varepsilon > 0$ constant, then the optimization time for $\textsc{Fork}_r^n$ on a complete graph is $E(T) \in \Theta\left(\frac{n^{3r}+\tau\lambda n^r}{\lambda n^r+\tau\lambda^2}\right)$.*

Proof. We already have shown the lower bound in Theorem 15. The second term of the upper bound in Theorem 16 is dominated by the rest if $\tau \in \Omega(n \log n)$. Therefore it matches the lower bound. □

Corollary 18. *The number of fitness evaluations to spread the optimum to all islands is in $\Theta\left(n^{1.5r}\right)$ for the best choice of parameters λ and τ.*

Proof. This can be shown by recalling that the number of evaluations to get there is in $\Omega\left(\frac{n^{3r}+\tau\lambda n^r}{n^r+\tau\lambda} + \tau\lambda\right)$ and in $\mathrm{O}\left(\frac{n^{3r}+\tau\lambda n^r}{n^r+\tau\lambda} + \frac{n^{2r+1}\log n}{\tau} + \tau\lambda\right)$ (Theorems 15 and 16). There is no way to choose $\tau\lambda$ to get below $\Theta\left(n^{1.5r}\right)$. □

6 Ring Topology

We expect the ring topology to perform better than the complete graph due to the fact that even if one island finds the valley, there is enough time for the others to come up with the optimum before they would get informed of the valley by a neighbor. In this section we want to prove this assumption.

First we show that we expect just a small number of islands to find the valley before all other islands get to 1^n. After that we prove that we can choose a migration probability so that valleys are unlikely to be shared too often. As final step we show that all other islands have enough time to find the optimum so that we get an upper bound of the expected run time.

For those steps we define two events that can occur.

Definition 19. *Let V_b be the event that the valley was generated on at most b islands during the time until all other islands have 1^n, the valley or the optimum as their solution.*

Definition 20. *Let B_c be the event that after the time until all islands have 1^n, the valley or the optimum as their solution, there is a maximum of $c \log n$ valleys.*

Lemma 21. *If $b \geq 1 + \frac{r}{\varepsilon}$ is constant and $\lambda \in \mathrm{O}\left(n^{r-1-\varepsilon}\right)$, where $\varepsilon > 0$ is a constant, it holds that the expected run time of to optimize $\textsc{Fork}_r^n$ on any island is $E(T) \leq E(T \mid V_b) + \mathrm{O}(n)$.*

Another event that could possibly lead to a high run time is when one of the found valleys is shared too fast to all other islands. Like before we will show that this case is unlikely enough to not dominate the run time.

Lemma 22. *Let $\varepsilon > 0$, $b \geq 1 + \frac{r}{\varepsilon}$ and $c \geq 7rb$ be constants and $\lambda \in \mathrm{O}\left(n^{r-1-\varepsilon}\right)$. When $\frac{1}{\tau}$ denotes the migration probability and $\tau \in \Omega(n \log n)$, the expected run time to optimize $\textsc{Fork}_r^n$ on any island is $E(T) \leq E(T \mid V_b \cap B_c) + \mathrm{O}\left(\frac{n^r}{\lambda}\right)$.*

From the previous lemmas we can derive that if we choose τ large enough and λ small enough, we get an upper bound on the expected run time by just looking at the case of $B_c \cap V_b$ and adding $\mathrm{O}\left(\frac{n^r}{\lambda}\right)$.

Theorem 23. *For* $\tau \in \Omega(n \log n)$, $12r \log n \leq \lambda \in \mathrm{O}\left(n^{r-1-\varepsilon}\right)$ *and* $\lambda^2 \tau \geq 17r^2 n^r \log^2 n$, *the expected optimization time for* $\textsc{Fork}_r^n$ *on a Ring topology is* $E(T) \in \mathrm{O}\left(\frac{n^r}{\lambda}\right)$.

The next corollary sums up our findings and shows performance for the optimal choice of parameters.

Corollary 24. *The expected number of fitness evaluations to have all islands at the optimum is in* $\mathrm{O}\left(n^r \log^2 n\right)$ *if* τ *and* λ *are set appropriately.*

Proof. If we use the results from Theorem 23, we see that there are $\mathrm{O}\left(n^r + \tau\lambda^2\right)$ evaluations until all islands are informed. If we consider that $\tau\lambda^2 \in \mathrm{O}\left(n^r \log^2 n\right)$ we get the bound. □

7 Conclusion

The results we obtained regarding the expected number of fitness evaluations of Fork by different algorithms (until all islands have the optimum, in case of an island model) and on optimal parameter settings are

- (1+1) EA– $\Theta\left(n^{2r}\right)$;
- Independent runs – $\Omega\left(\lambda n^{2r} \log \lambda\right)$;
- Complete – $\Theta\left(n^{1.5r}\right)$;
- Ring – $\mathrm{O}\left(n^r \log^2 n\right)$.

To show this we exploited that less diversity can mean to be trapped, but it also gives advantages if there is migration between the islands. Note that a ring delays migration, since in every migration step an individual can only proceed by one island along the ring, so the total time to inform all islands is highly concentrated around the expectation. This is different in the complete topology with a high value of τ: the expected time to inform all individuals might be the same, but the concentration is weaker.

We showed when a ring topology outperforms a topology with faster dissemination of individuals due to the increase in diversity. It would be interesting to see what other properties of the search space can also gain from a ring topology. Furthermore, one could wonder whether always one of the two extremes, complete and ring, is the best choice. There may be effects that can benefit for example a two-dimensional torus over both the ring and the complete graph.

We also discussed the method of composing different fitness functions, based on nesting one fitness function in another. We focused on the case that the outer fitness function is LeadingOnes and all inner fitness functions are Fork of the same length, but in principle one can consider inner fitness function that

differ from each other, possibly also in their length and also other options for out fitness functions such as OneMax. We gave a general but precise tool to calculate run times on LeadingOnes-composite fitness functions when using the (1+1) EA.

References

1. Alba, E.: Parallel evolutionary algorithms can achieve super-linear performance. Inf. Process. Lett. **82**, 7–13 (2002)
2. Bäck, T., Fogel, D.B., Michalewicz, Z.: Handbook of evolutionary computation. Release **97**(1), B1 (1997)
3. Badkobeh, G., Lehre, P.K., Sudholt, D.: Black-box complexity of parallel search with distributed populations. In: Proceedings of the 2015 FOGA XIII. pp. 3–15. ACM (2015)
4. Doerr, B., Fischbeck, P., Frahnow, C., Friedrich, T., Kötzing, T., Schirneck, M.: Island models meet rumor spreading. In: Proceedings of the GECCO 2017, pp. 1359–1366. ACM (2017)
5. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. Algorithmica **65**(1), 224–250 (2013)
6. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)
7. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory Comput. Syst. **39**(4), 525–544 (2006)
8. Eisenberg, B.: On the expectation of the maximum of IID geometric random variables. Stat. Probab. Lett. **78**(2), 135–143 (2008)
9. Gießen, C.: Hybridizing evolutionary algorithms with opportunistic local search. In: Proceedings of the GECCO 2013, pp. 797–804. ACM (2013)
10. Jansen, T., Wiegand, R.P.: Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA. In: Cantú-Paz, E. (ed.) GECCO 2003. LNCS, vol. 2723, pp. 310–321. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45105-6_37
11. Lässig, J., Sudholt, D.: Design and analysis of migration in parallel evolutionary algorithms. Soft Comput. **17**(7), 1121–1144 (2013)
12. Lässig, J., Sudholt, D.: General upper bounds on the runtime of parallel evolutionary algorithms. Evol. Comput. **22**, 405–437 (2014)
13. Lissovoi, A., Witt, C.: A runtime analysis of parallel evolutionary algorithms in dynamic optimization. Algorithmica **78**(2), 641–659 (2017)
14. Neumann, F., Oliveto, P.S., Rudolph, G., Sudholt, D.: On the effectiveness of crossover for migration in parallel evolutionary algorithms. In: Proceedings of the GECCO 2011, pp. 1587–1594 (2011)
15. Ruciński, M., Izzo, D., Biscani, F.: On the impact of the migration topology on the Island model. Parallel Comput. **36**, 555–571 (2010)
16. Sudholt, D.: General lower bounds for the running time of evolutionary algorithms. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 124–133. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_13