Improving the Run Time of the (1 + 1) Evolutionary Algorithm with Luby Sequences

Tobias Friedrich
Hasso Plattner Institute
Potsdam, Germany

Timo Kötzing
Hasso Plattner Institute
Potsdam, Germany

Francesco Quinzan
Hasso Plattner Institute
Potsdam, Germany

Andrew M. Sutton
University of Minnesota Duluth
Duluth, MN, USA

ABSTRACT

In the context of black box optimization, one of the most common ways to handle deceptive attractors is to periodically restart the algorithm. In this paper, we explore the benefits of combining the simple (1+1) Evolutionary Algorithm (EA) with the Luby Universal Strategy - the (1 + 1) $EA_{\mathcal{U}}$, a meta-heuristic that does not require parameter tuning.

We first consider two artificial pseudo-Boolean landscapes, on which the (1 + 1) *EA* exhibits exponential run time. We prove that the (1 + 1) $EA_{\mathcal{U}}$ has polynomial run time on both instances.

We then consider the Minimum Vertex Cover on two classes of graphs. Again, the (1 + 1) *EA* yields exponential run time on those instances, and the (1 + 1) $EA_{\mathcal{U}}$ finds the global optimum in polynomial time.

We conclude by studying the Makespan Scheduling. We consider an instance on which the (1 + 1) *EA* does not find a $(4/3 - \epsilon)$-approximation in polynomial time, and we show that the (1 + 1) $EA_{\mathcal{U}}$ reaches a $(4/3 - \epsilon)$-approximation in polynomial time. We then prove that the (1 + 1) $EA_{\mathcal{U}}$ serves as an Efficient Polynomial-time Approximation Scheme (EPTAS) for the Partition Problem, for a $(1 + \epsilon)$-approximation with $\epsilon > 4/n$.

CCS CONCEPTS

• **Mathematics of computing** → **Combinatorial optimization**;
• **Theory of computation** → *Theory of randomized search heuristics*;

KEYWORDS

Restart Strategy, Deceptive Attractors, Combinatorial Optimization

ACM Reference Format:
Tobias Friedrich, Timo Kötzing, Francesco Quinzan, and Andrew M. Sutton. 2018. Improving the Run Time of the (1 + 1) Evolutionary Algorithm with Luby Sequences. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205525

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205525

1 INTRODUCTION

In the context of real-world optimization, there is usually very little analytical knowledge of the problem at hand, and classical numerical methods often fail. Problems of this kind can be approached with *black-box* optimizers - algorithms that access the function to be optimized only via the evaluation of possible solutions.

Optimizers of this kind often exhibit a large variance in run time, due to a strong dependence of the run time on initial conditions. Typically, a significant number of iterations are wasted trapped in deceptive basins of attraction: sets of states that lead the search away from the optimal solution. A simple and effective strategy for handling this problem is to use *restarts*. In this scenario, an algorithm is periodically re-initialized after some time (the *restart time*), with the hope that if it has already become trapped in a deceptive basin then it can be allotted another chance to discover the global optimum. The success of a restart strategy depends on the structure of the attractor basins in the search space, as well as the length of the period between restarts. These two properties are intimately related. Restart policies for discrete black box algorithms have been studied for decades from an experimental perspective [6, 16]. In these works, good restart times are often proposed dynamically by some process based on the fitness of the solutions seen by the algorithm so far.

The utility of re-initializing with a uniformly chosen random point depends strongly on the function under consideration. Let $\mathcal{X}$ be a fixed domain, and consider any family of functions $\mathcal{F} \subseteq \{f \mid f : \mathcal{X} \to \mathbb{R}\}$ s.t. for all $x \in \mathcal{X}$ there is a $f \in \mathcal{F}$ that attains a unique global optimum at $f(x)$. Then the restart strategy that uses uniform measure on the samples is the optimal choice [9]. On the other hand, if there are more restrictions on the global structure of the space $\mathcal{F}$, different restart strategies can be employed. This is the idea behind *iterated local search* [14] in which restarts are nonuniform, and it is assumed that smaller perturbations can escape deceptive attractor basins. In this paper, we shall assume the former situation, in which no information is known and so the best policy is restarting uniformly at random. Such function families are a common object of study, for example in the field of black-box complexity.

A *restart strategy* for a black-box algorithm is an infinite sequence $(t_1, t_2, \ldots, t_t, \ldots)$ that specifies the algorithm should be run from

a uniformly random starting point for t_1 steps, then restarted uniformly at random and run again for t_2 steps and so forth. Note that strategies of this kind are always set "a priori", and do not directly make use of possible best solutinos found at a given time step. The *optimal restart strategy* is the sequence that minimizes the expected number of runs until a run returns the optimal solution. Given any Las Vegas algorithm, it is possible to derive theoretically the *optimal restart time* - the value that yields the fastest possible convergence over all such restart times. This result is due to Luby, Sinclair, and Zuckermann [15], but was also discovered in the context of backpropagation training for neural networks [17].

The optimal restart time is often unknown or difficult to compute. Therefore, Luby et al. describe a *universal strategy* that on any given Las Vegas algorithm yields run time of at most a logarithmic factor worse than the one obtain with the optimal strategy. This strategy is not tailored to any specific problem, and it has the advantage that no parameter tuning is needed. Our work consists of adapting the results mentioned above to commonly studied local search heuristics.

In the framework given by Luby et al. [15], only *a priori* restart strategies are considered, although it has been observed that other approaches can be beneficial cf. de Perthuis de Laillevault et al. [1], Gyorgy and Kocsis [7]. Another strategy that is not covered in this setting is the Bet-and-Run, first introduced by Fischetti and Monaci [3]. This strategy has been particularly successful in improving state-of-the-art solvers to approach combinatorial problems (cf. Friedrich et al. [4]), and has been studied theoretically in the context of EAs (cf. Lissovoi et al. [13]). In contrast to all strategies mentioned above, Luby sequences require *no parameter tuning*.

We explore the benefits of using the universal strategy with EAs on various fitness landscapes. In Section 2 we give an account of the algorithms and restart strategies studied in this paper. We then test the benefits of using the universal strategy, and draw a comparison with the Bet-and-Run in Section 3. We then study theoretically some combinatorial optimization problems in Section 4, and experimentally in Section 5. We conclude the paper in Section 6.

2 DEFINITIONS AND TECHNICAL TOOLS

2.1 Algorithms and Framework

The $(\mu+1)$ *EA* is a simple population-based Evolutionary Algorithm (cf. Algorithm 1). Initially, a population of size μ is generated u.a.r. A parent is then selected u.a.r. and an offspring is generated via uniform mutation. The offspring is then added to the population and the element with worst fitness is discarded. This algorithm uses no diversity-preserving mechanism and the selection pressure is quite low (cf. Friedrich et al. [5]). The $(1+1)$ *EA* is a specialization of the $(\mu+1)$ *EA* with $\mu = 1$. It requires as input an individual of fixed length. An offspring is generated with an operator that resembles asexual reproduction. The fitness is then computed, and the less desirable result is discarded. In this paper we only perform the analysis on the $(1+1)$ *EA*, and compare it with the $(\mu+1)$ *EA* for $\mu > 1$, using pre-existing results.

Given a fitness function $f : \{0,1\}^n \longrightarrow \mathbb{R}_{\geq 0}$, we describe the run time of the $(1+1)$ *EA* as a Markov chain $\{X_t\}_{t\geq 0}$ with X_t the f-value reached at time step t. We study the run time as the first hitting time $\tau = \inf_t\{X_t = OPT\}$, with OPT the global optimum

Algorithm 1: $(\mu+1)$ *EA*

```
t ← 0;
Choose population P_t of μ individuals u.a.r.;
while convergence criterion not met do
    select parent x ∈ P_t u.a.r.;
    generate y by flipping each bit of x w.p. 1/n;
    select z ∈ P_t with worst fitness;
    if f(z) ≤ f(y) then
        P_{t+1} ← P_t \ {z} ∪ {y};
    else
        P_{t+1} ← P_t;
    t ← t + 1;
```

of f. Since the $(1+1)$ *EA* requires only a single fitness evaluation per step, then any upper bound on the number of steps τ yields an upper bound on the expected number of fitness evaluations. In this paper, we also study the $(1+1)$ *EA* as an *approximation algorithm*. To this end, we consider the following definition.

Definition 2.1. Consider a function $f : \{0,1\}^n \longrightarrow \mathbb{R}_{\geq 0}$, and denote with OPT the global minimum. An ϵ-approximation of OPT is any solution $x \in \{0,1\}^n$ s.t. $f(x)/OPT \leq \epsilon$.

We analyze the run time of the $(1+1)$ *EA* as an approximation algorithm, by looking at the first hitting time $\tau_\epsilon = \inf_t\{X_t \leq \epsilon OPT\}$. Note that if $\Pr(\tau < +\infty) = 1$, then $\Pr(\tau_\epsilon < +\infty) = 1$ for all $\epsilon \geq 1$. The definitions above are only valid for minimization problems. However, symmetric definitions hold for maximization problems.

2.2 The Luby Universal Strategy

In this context, a *restart strategy* for a black-box algorithm is an infinite sequence $(t_1, t_2, \ldots, t_t, \ldots)$ that specifies that the algorithm should be run from a uniformly random starting point for t_1 steps, then restarted uniformly at random and run again for t_2 steps and so forth. Thus, we consider only *a priori* restart strategies, although it has been observed that other kinds of restart techniques may fasten the expected run time of randomized algorithms (cf. de Perthuis de Laillevault et al. [1] and Lissovoi et al. [13]).

The *optimal restart strategy* is the sequence that minimizes the expected number of runs until a run returns the optimal solution. In the context of Las Vegas algorithms, when the density function of the convergence probability over time is known, then the optimal strategy can be computed exactly (cf. Luby et al. [15]). More formally, let $\mathcal{A}$ be any Las Vegas algorithm, and with an abuse of notation let τ denote the r.v. that returns the number of calls to the fitness function, until either the global optimum or an approximation of it is reached (cf. Definition 2.1). Following the definition of Luby et al. [15], we call an algortthm *Las Vegas* if it always converges in finite time. Let $p(t) = \Pr(\tau = t)$ be the probability of convergence at step t, and $q(t) = \sum_{t'\leq t} p(t') = \sum_{t'\leq t} \Pr(\tau = t')$ the probability of search termination before step t, i.e. the cumulative distribution function. Then the following theorem holds.

Algorithm 2: The Universal Strategy $\mathcal{U}$ for $\mathcal{A}$.

```
i ← 1, t ← [ ];
while convergence criterion not met do
    if ∃k ∈ ℕ : i = 2^k − 1 then
        run 𝒜 for 2^{k−1} steps;
        t[i] ← 2^{k−1};
    else if ∃k ∈ ℕ : 2^{k−1} ≤ i < 2^k − 1 then
        run 𝒜 for t[i − 2^{k−1} + 1] steps;
        t[i] ← t[i − 2^{k−1} + 1];
    i ← i + 1;
```

Theorem 2.2 (Theorem 1 in Luby et al. [15]). *For any Las Vegas algorithm $\mathcal{A}$, let $p(t)$ and $q(t)$ be as above. Consider the function*

$$\ell(t) := \frac{1}{q(t)}\left(t - \sum_{t'<t} q(t')\right).$$

The optimal restart strategy for $\mathcal{A}$ is the repeating sequence $\mathcal{S} = (t^*, \dots, t^*, \dots)$ with t^* defined as $t^* := \operatorname{argmin}_{t\geq 0}\{\ell(t)\}$.

Intuitively, this theorem tells us that for any black-box algorithm that converges in finite time, there exists an optimal restart strategy, and this strategy always consists of consecutively running for t^* steps. Note that the optimality of $\mathcal{S}$ holds both for finite t^* and for $t^* = +\infty$. In the latter case, the optimal strategy is to not restart the process at all. The proof of the theorem above is based on the following lemma.

Lemma 2.3 (Lemma 1 in Luby et al. [15]). *For any Las Vegas algorithm $\mathcal{A}$, let $p(t)$ and $q(t)$ be as above, and consider a restart strategy $\mathcal{R} = (t, \dots, t, \dots)$. Let $\tau_{\mathcal{R}}$ be the run time of $\mathcal{A}$ following the restart strategy $\mathcal{R}$. Then there holds*

$$\mathbb{E}[\tau_{\mathcal{R}}] = \frac{1}{q(t)}\left(t - \sum_{t'<t} q(t')\right) \leq \frac{t}{q(t)}.$$

While being theoretically relevant, Theorem 2.2 cannot be easily used in practise, because $p(t)$ and $q(t)$ are often unknown or hard to compute. For these reasons, Luby et. al. give a *universal strategy*, that has expected run time of a logarithmic factor worse then the one achieved by the optimal restart strategy. This strategy is indicated by $\mathcal{U} = (t_1, t_2, \dots, t_n, \dots)$ and it is defined recursively as

$$t_i = \begin{cases} 2^{k-1} & \text{if } i = 2^k - 1; \\ t_{i-2^{k-1}+1} & \text{if } 2^{k-1} \leq i < 2^k - 1; \end{cases}$$

The restart strategy $\mathcal{S}$ for an algorithm $\mathcal{A}$ is presented in Algorithm 2. The following theorem holds.

Theorem 2.4 (Theorem 5 in Luby et al. [15]). *For any Las Vegas algorithm $\mathcal{A}$, let τ^* be the run time of $\mathcal{A}$ following the optimal restart strategy, and denote $\tau_{\mathcal{U}}$ the run time of $\mathcal{A}$ following the universal strategy $\mathcal{U}$. Then it holds $\mathbb{E}[\tau_{\mathcal{U}}] \leq 192\mathbb{E}[\tau^*](\log_2(\mathbb{E}[\tau^*]) + 5)$.*

The theorem above only makes the assumption that $\mathbb{E}[\tau^*]$ is finite. In particular, the run time of $\mathcal{A}$ without restarts needs not be finite. From Theorem 2.4 it trivially follows that the run time of the universal strategy can be upper-bounded with the run time of any non-optimal restart strategy. More formally,

Corollary 2.5. *For any Las Vegas algorithm $\mathcal{A}$, let $\tau_{\mathcal{U}}$ be the run time of $\mathcal{A}$ following the universal restart strategy, and denote $\tau_{\mathcal{R}}$ the run time of $\mathcal{A}$ following any given restart strategy $\mathcal{R}$. Then there holds $\mathbb{E}[\tau_{\mathcal{U}}] \leq 192\mathbb{E}[\tau_{\mathcal{R}}](\log_2(\mathbb{E}[\tau_{\mathcal{R}}]) + 5)$.*

Since the (1 + 1) *EA* on any fitness function $f : \{0,1\}^n \longrightarrow \mathbb{R}$ has expected run time at most n^n (cf. Droste et al. [2]), then the (1 + 1) *EA* fulfils the definition of Las Vegas algorithm given in Luby et al. [15]. In the remaining part of this paper we denote with (1 + 1) $EA_{\mathcal{U}}$ the (1 + 1) *EA* following the Luby universal strategy as given in Algorithm 2.

3 PSEUDO-BOOLEAN LANDSCAPES

3.1 Deceptive Basins of Attraction

We analyze the run time of the (1 + 1) $EA_{\mathcal{U}}$, and comparing with the run time of the (μ + 1) *EA*, on the following fitness

$$\mathsf{TwoMax}(x_1, \dots, x_n) = \max\{|x|_1, n - |x|_1\} + \prod_{i=1}^{n} x_i,$$

where $|x|_1$ returns the number of 1s in the input string. Among all search points with more than $\frac{n}{2}$ 1-bits, this function increases with the number of ones. Among all search points with less than $\frac{n}{2}$ 1-bits, it increases with the number of zeros. This function has two branches and it is symmetric w.r.t. the underlying hypercube. The point 0^n is a *local* optimum, while the point 1^n is a *global* optimum. The leftmost branch is a basin of attraction for the local optimum. The following theorem gives an estimate of the related expected value for the (μ + 1) *EA*.

Theorem 3.1 (Theorem 1 in Friedrich et al. [5]). *The probability that the (μ+1) EA with no diversity-preserving mechanism and $\mu = o(n/\log n)$ optimises* TwoMax *in time n^{n-1} is at most $1/2 - o(1)$. Its expected optimization time is $\Omega(n^n)$.*

This result is intuitively motivated by the fact that 0^n is a deceptive attractor: if the algorithm reaches 0^n, then the probability of hitting the global optimum afterwards is n^{-n}. In the remaining part of this section we show that the (1 + 1) $EA_{\mathcal{U}}$ yields better run time than the (μ + 1) *EA* on the TwoMax. To this end, we consider the following lemma.

Lemma 3.2. *The* (1+1) *EA reaches a* local *optimum of the* TwoMax *in expected $O(n \log n)$ many fitness evaluations. Moreover, the probability that the solution is only a local optimum is* 1/2.

This lemma follows immediately from the fact that the (1+1) *EA* optimizes the OneMax within $O(n \log n)$ fitness evaluations (cf. Droste et al [2] and Mühlenbein [18]), and from the fact that the problem is symmetric w.r.t. the underlying hypercube. The lemma above is useful in proving an upper bound on the run time of the (1+1) $EA_{\mathcal{U}}$ on the function TwoMax. The following theorem holds.

Theorem 3.3. *The* (1 + 1) $EA_{\mathcal{U}}$ *reaches the global maximum of* TwoMax *in expected $O(n \log^2 n)$ many fitness evaluations.*

Proof. We first prove the upper bound on the run time of the (1 + 1) $EA_{\mathcal{U}}$, and then we prove the tail bound. We proceed by identifying a finite restart strategy that yields better expected run time than the (1 + 1) *EA* with no restarts, and then we use Theorem

2.4 to obtain the desired upper bound. To this end, denote with $\bar{\tau}$ the run time of the $(1+1)$ *EA* until a *local* optimum is reached. We consider the restart strategy $\mathcal{R} = (\bar{t}, \dots, \bar{t}, \dots)$ with $\bar{t} := 2\mathbb{E}[\bar{\tau}]$. Using Lemma 3.2 together with Markov's inequality, we conclude that

$$q(\bar{t}) \geq \frac{1}{2}\Pr(\bar{\tau} < 2\mathbb{E}[\bar{\tau}]) \geq \frac{1}{4}.$$

Thus, if we denote with $\tau_{\mathcal{R}}$ the run time of the $(1+1)$ *EA* following the strategy $\mathcal{R}$, from Lemma 2.3 it follows that

$$\mathbb{E}[\tau_{\mathcal{R}}] \leq \frac{\bar{t}}{q(\bar{t})} \leq 8\mathbb{E}[\bar{\tau}] = O(n \log n).$$

We conclude by applying Theorem 2.4 to obtain that $\mathbb{E}[\tau_{\mathcal{S}}] \leq 1536\mathbb{E}[\bar{\tau}](\log_2 \mathbb{E}[\bar{\tau}] + 5) = O(n \log^2 n)$. □

3.2 Slopes and Plateaus

We analyze the run time of the $(1+1)$ $EA_{\mathcal{U}}$ on the following function [13].

$$f_h(x) = \begin{cases} |x|_1 & \text{if } |x|_1 \geq n/2; \\ h & \text{otherwise.} \end{cases} \quad (1)$$

This function depends on the parameter h. For $h > n/2$ this function exhibits a plateau in the leftmost region of the hypercube, and a slope in the rightmost region. The plateau is a deceptive basin, whereas the slope guides the algorithm toward the global optimum. We consider the function f_h, to replicate the setting that Lissovoi et al. proposed for their analysis on the Bet-and-Run strategy. The following theorem gives a lower bound for the run time of the $(1+1)$ *EA* on this instance. As noted in Lemma 3.3 in Lissovoi et al. [13], the expected optimization time of the $(1+1)$ *EA* on the instance f_h is at least $(h - n/2)!$, for $h > n/2$. Note that h needs not be constant for increasing problem size. For example, a choice of $h = 3n/4$ yields a lower-bound on the ru time of the $(1+1)$ *EA* as $(n/4)!$. We consider the case of the $(1+1)$ *EA* using the universal strategy on the function f_h as defined above, for $h > n/2$. We observe that these algorithms with the universal strategy yield expected polynomial run time. In order to perform the analysis we use the following lemma.

Lemma 3.4. *Consider the* $(1+1)$ *EA maximizing the function* f_h *for any choice of* $h > n/2$*, and denote with* τ *its run time. Then we have* $\Pr(\tau \leq cn \log n) = \Omega(1)$*, for an appropriate constant* c*.*

Proof. We first observe that for an initial solution x_0 sampled u.a.r. it holds $|x_0|_1 \geq n/2 + \pi/(4e)\sqrt{n}$ w.p. at least $1/4$ (cf. Lemma 1 in Kötzing et al. [11]). Suppose that an initial solution x_0 is sampled s.t. $|x_0|_1 \geq n/2 + \pi/(4e)\sqrt{n}$. Then in order to reach the leftmost plateau at least $\Omega(\sqrt{n})$ bit flips are necessary. Also, any upper-bound on the probability that x_0 reaches the plateau is also an upper-bound on the probability that any subsequent point further up the slope does not reach the plateau. Therefore, given an initial point $|x_0|_1 \geq n/2 + \pi/(4e)\sqrt{n}$, we can roughly upper-bound the probability of reaching the plateau in the next iteration with an upper-bound for the probability of performing a jump greater than $\left\lceil \frac{\pi}{4e}\sqrt{n} \right\rceil$, which is

$$\sum_{j=\left\lceil \frac{\pi}{4e}\sqrt{n} \right\rceil}^{n} \frac{1}{n}^j \left(1 - \frac{1}{n}\right)^{n-j} \leq n^{-\Omega(\sqrt{n})}$$

Therefore, the probability that the next individual remains on the slope is at least $1 - n^{-\Omega(\sqrt{n})}$. We conclude that in a phase of length $\Theta(n \log n)$ the $(1+1)$ *EA* samples only points on the slope w.p. at least $(1 - n^{-\Omega(\sqrt{n})})^{\Theta(n \log n)} = \Omega(1)$. Assuming that the $(1+1)$ *EA* only samples points on the slope, then the $(1+1)$ *EA* climbs the slope within $O(n \log n)$ expected steps with single bit-flips, as in the case of OneMax (cf. Droste et al [2] and Mühlenbein [18]). □

We remark that the lemma above, as well as the underlying ideas for the proof, is inspired by the work of Lissovoi et al. [13]. Note that in the lemma above we do not use the requirement that $h - n/2$ is constant for increasing problem size. We can use the lemma above to prove that an upper bound on the runtime of the $(1+1)$ *EA* using the universal strategy. The following theorem holds.

Theorem 3.5. *Consider the* $(1+1)$ $EA_{\mathcal{U}}$ *optimizing the instance* f_h *as defined above, and denote with* $\tau_{\mathcal{U}}$ *its run time. Then it holds* $\mathbb{E}[\tau_{\mathcal{U}}] = O(n \log^2 n)$*.*

Proof. We first prove the upper bound on the expected run time, with an argument similar to the one given in Theorem 3.3. We proceed by identifying a finite restart strategy that yields better expected run time than the $(1+1)$ *EA* with no restarts, and then we use Theorem 2.4 to obtain the desired upper bound. As usual, we denote with τ the run time of the $(1+1)$ *EA* with no restarts. We consider the restart strategy $\mathcal{R} = (\bar{t}, \dots, \bar{t}, \dots)$ with $\bar{t} = cn \log n$, for an appropriate constant c as in Lemma 3.4. Note that by Lemma 3.4 we have $q(\bar{t}) = \Pr(\tau \leq cn \log n) = \Omega(1)$. If we denote with $\tau_{\mathcal{R}}$ the run time of the algorithm following the restart strategy $\mathcal{R}$, then there holds

$$\mathbb{E}[\tau_{\mathcal{R}}] \leq \frac{\bar{t}}{q(\bar{t})} = \frac{cn \log n}{\Omega(1)} = O(n \log n).$$

The claimed theorem follows from Theorem 2.4, since $\mathbb{E}[\tau_{\mathcal{U}}] = O(\mathbb{E}[\tau_{\mathcal{R}}] \log \mathbb{E}[\tau_{\mathcal{R}}]) = O(n \log^2 n)$. □

4 COMBINATORIAL OPTIMIZATION

4.1 On the Minimum Vertex Cover

Given a graph $G = (V, E)$ the Minimum Vertex Cover problem (MVC) consists of finding a minimal subset of vertices s.t. each edge is incident to at least one vertex in the set. The MVC is NP-complete and has no polynomial-time approximation algorithm, unless P = NP. In this section, we investigate the benefits of using restarts with the $(1+1)$ *EA* when searching for a MVC. Given a graph $G = (V, E)$ of order n and an indexing of V, each subset of vertices $U \subseteq V$ is stored in memory as a pseudo-Boolean array $(x_1, \dots, x_n)$ where $x_i = 1$ iff. the i-th element is in U. We use a fitness function as given in Khuri [10] and He et al. [8], defined as

$$f(x) = \sum_{i=1}^{n} \left(x_i + n(1 - x_i) \sum_{j=1}^{n} (1 - x_j) e_{ij} \right) \quad (2)$$

with e_{ij} the coefficients of the adjacency matrix

$$e_{ij} = \begin{cases} 1 & \text{if there is an edge } (i, j) \in E; \\ 0 & \text{otherwise;} \end{cases}$$

We focus on two classes of graphs s.t. the run time of the $(1+1)$ *EA* is exponential, and show that the run time of the $(1+1)$ $EA_{\mathcal{U}}$ is

polynomial on those instances. The first class of graphs that we consider is the case of a complete bipartite graph. More formally,

Definition 4.1. We say that a graph $G = (V, E)$ is complete bipartite if there exists a partition $\{V_1, V_2\}$ of V such that V_1 and V_2 are independent sets, and every pair of vertices $(u, v) \in V_1 \times V_2$ is adjacent.

Let $G = (V, E)$ be any finite complete bipartite graph with partitions $\{V_1, V_2\}$. Suppose that $|V_1| < |V_2|$ and consider the run time of the (1 + 1) *EA* on the corresponding fitness f, as given in Equation 2. Then both V_1 and V_2 are local optima, but only V_1 is a global optimum. Suppose that at any point in time the solution V_2 is produced. Then no neighbouring solutions have equal or smaller fitness, and the (1 + 1) *EA* cannot easily escape the local optimum. This phenomenon has already been observed in the literature. In fact, the following theorem holds.

Theorem 4.2 (Theorem 5 in Oliveto et al. [19]). *Consider a bipartite graph with partitions $|V_1| = \epsilon n$ and $|V_2| = (1 - \epsilon)n$ respectively with $\epsilon \leq 1/2$. The expected optimization time of the (1+1) EA to find the optimal solution of the bipartite graph is $\Omega(n^{n/\epsilon})$. With probability at least $1/(2e) - 2^{-\Omega(n)}$ the (1 + 1) EA finds the global optimum in time $O(n \log n)$.*

We prove that the (1 + 1) $EA_{\mathcal{U}}$ finds the MVC on any complete bipartite graph in expected polynomial fitness evaluations. The following theorem holds.

Theorem 4.3. *On complete bipartite graph the (1 + 1) $EA_{\mathcal{U}}$ finds a MVC after expected $O(n \log^2 n)$ fitness evaluations.*

Proof. From Theorem 4.2 we have that there exists a constant $c > 0$ s.t. the (1 + 1) *EA* finds a MVC within $cn \log n$ fitness evaluations, with constant probability. Fix $\bar{t} = cn \log n$, and consider the restart strategy $\mathcal{R} = (\bar{t}, \ldots, \bar{t}, \ldots)$. If we denote with $\tau_{\mathcal{R}}$ the run time of the (1 + 1) *EA* following the strategy $\mathcal{R}$, from Lemma 2.3 it follows that

$$\mathbb{E}[\tau_{\mathcal{R}}] \leq \frac{\bar{t}}{q(\bar{t})} = O(n \log n),$$

where we have used that $q(\bar{t}) = \Omega(1)$, again from Theorem 4.3. We conclude by applying Theorem 2.4 to obtain that $\mathbb{E}[\tau_{\mathcal{U}}] = O(\mathbb{E}[\tau_{\mathcal{R}}] \log \mathbb{E}[\tau_{\mathcal{R}}]) = O(n \log^2 n)$. □

The second class of vertex-cover instances we consider is the Papadimitriou-Steiglitz (PS) class, as defined as follows. Fix an integer n of the form $n = 3k + 4$, for some $k > 0$. A PS_n graph of order n consists of three rows (disjoint subsets) of vertices V_1, V_2, V_3 s.t. in the first two rows there are $k + 2$ vertices, whereas in the third row there are only k nodes. Each element in V_1 is connected to a single element in V_2 and each element in V_2 is connected to a single element in V_1. Each node in V_3 is connected to all nodes in V_2, and there are no connections between V_1 and V_3. The expected time for the (1 + 1) *EA* to optimize a PS_n is $2^{\Omega(\sqrt[3]{n})}$. With constant probability the (1 + 1) *EA* finds a global optimum after expected $O(n \log n)$ fitness evaluations (cf. Theorem 2 in Oliveto et al. [19]).

We show that the run time of the (1 + 1) $EA_{\mathcal{U}}$ is polynomial on this instance. Again, we define a restart strategy $\mathcal{R} = (\bar{t}, \ldots, \bar{t}, \ldots)$ with $\bar{t} = cn \log n$ for a sufficiently large constant $c > 0$ and s.t. $q(\bar{t}) = \Omega(1)$. We then upper-bound the run time of the (1 + 1) $EA_{\mathcal{U}}$ in terms of the expected run time of the (1 + 1) *EA* following the restart $\mathcal{R}$. The following lemma holds.

Lemma 4.4. *On the PS_n graph the (1 + 1) $EA_{\mathcal{U}}$ finds a MVC after expected $O(n \log^2 n)$ fitness evaluations.*

We omit a formal proof as the Lemma above can be proven as in the case of Theorem 4.3.

4.2 On the Makespan Scheduling Problem

We analyze the run time of the (1 + 1) *EA* and (1 + 1) $EA_{\mathcal{U}}$ on the *makespan sheduling* problem, which consists of assigning jobs to resources at particular times. The most basic version works as follows: Assign n jobs $J_1, J_2, \ldots, J_n$ to m machines with varying processing power, and minimize the total elapsed time. The case of $m = 2$ machines with equal processing power is an instance of the *Partition Problem*, which consists of finding a partition $\{S_1, S_2\}$ of a base set S s.t. the sum of the numbers in S_1 is equal to the sum of the numbers in S_2. The partition problem is a well-known NP-complete problem. In this section, we analyze the run time of the (1 + 1) *EA* until an approximation of the global optimum is reached.
For a partition problem over a set of n elements $\{w_1, \ldots, w_n\}$, each solution is represented via a pseudo-boolean array of length n, where the ith coefficient is $x_i = 1$ if assigned to one partition, and $x_i = 0$ to the other one. We search for an approximation of the absolute minimum of the function

$$f(x) := \max\left\{\sum_{j=1}^{n} W_j x_j, \sum_{j=1}^{n} W_j (1 - x_j)\right\}.$$

For problems of this kind some positive results have already been presented. In Witt [20] it is shown that the (1 + 1) *EA* reaches a $(4/3 + \delta)$-approximation ratio in expected $O(n)$ many fitness evaluations, whereas it reaches a 4/3-approximation in expected $O(n^2)$ many fitness evaluations (cf. Table 1). In this section we study a worst-case example by which the (1 + 1) *EA* does not reach an approximation better than $4/3 - \epsilon$ within polynomial many fitness evaluations, for each $0 < \epsilon < 1/3$. We show that on the same instance the (1 + 1) $EA_{\mathcal{U}}$ finds the optimal solution in polynomial time. We consider the following definition.

Definition 4.5. Let n be even and fix a constant $0 < \epsilon < 1/3$. We define the makespan scheduling instance $W_\epsilon = \{J_1, \ldots, J_n\}$ by setting

$$J_1 = J_2 := \frac{1}{3} - \frac{\epsilon}{4} \quad \text{and} \quad J_i = \frac{1}{n-2}\left(\frac{1}{3} + \frac{\epsilon}{2}\right)$$

for all $3 \leq i \leq n$.

It can be shown (cf. Witt [20]) that for each $\epsilon > 0$ the (1 + 1) *EA* does not find a $4/3 - \delta$ approximation of a global optimum in polynomially many steps, on the instance W_ϵ. In fact, let n be even and fix a constant $0 < \epsilon < 1/3$. Then the (1 + 1) *EA* needs at least $n^{\Omega(n)}$ to generate a solution with better approximation ratio than $4/3 - \epsilon$ on the instance W_ϵ. Despite the negative result above, we prove that the (1 + 1) *EA* reaches a $(4/3 - \epsilon)$-approximation with constant probability $\Omega(1)$ on that instance. We then use this information to argue that the (1 + 1) *EA* with the universal strategy finds the global optimum in polynomial time. Useful in the analysis is the following definition.

Definition 4.6. Consider an instance of the the Partition problem $W = (w_1, \dots, w_n)$. For any characteristic vector x the critical job size $s(x)$ consists of the processing time w_j of the smallest job on the fuller machine.

On any instance of the Partition problem it is possible to obtain an upper bound on the f-value reached after a phase of length that is linear in the problem size, given that the critical job size is upper-bounded by a constant. The following technical lemma holds.

Lemma 4.7 (Lemma 2 in Witt [20]). *For an instance of the partition problem W, let ℓ be a lower bound on the optimal solution, and define $P = \sum_i w_i$. Suppose that the largest job is smaller than $P/2$, and suppose that from some time t_* the critical job size of the current search point of the $(1+1)$ EA is upper-bounded by a constant s^*. Then for any $\gamma > 1$ and $0 < \delta < 1$ the $(1+1)$ EA reaches an f-value at most $\ell + s^*/2 + \delta P/2$ in at most $\lceil en\log(\gamma/\delta)\rceil$ steps w.p. at least $1-\gamma^{-1}$. Moreover, the expected number of steps is at most $2\lceil en\log(2/\delta)\rceil$.*

We use the lemma above to prove some upper bounds on the run time of the $(1+1)$ EA on the worst case instances W_ϵ as defined above. We first show that the $(1+1)$ EA reaches a 4/3-approximation after expected $O(n)$ fitness evaluations for all $0 < \epsilon < 1/3$. We then prove that the $(1+1)$ EA reaches a $(4/3-\epsilon)$-approximation on the instance W_ϵ at least with constant probability. Combining these results we can finally prove that on the instance W_ϵ the $(1+1)$ $EA_{\mathcal{U}}$ reaches a $(4/3-\epsilon)$-approximation after expected $O(n\log n)$ fitness evaluations.

Lemma 4.8. *Fix a constant $0 < \epsilon < 1/3$. On the instance W_ϵ the $(1+1)$ EA reaches a 4/3-approximation after $O(n)$ expected fitness evaluations.*

Proof. We observe that if the two big jobs are assigned to a single machine and all small jobs to the second machine, then the corresponding f-value is at least

$$J_1 + J_2 = \frac{2}{3} - \frac{\epsilon}{2},$$

and this configuration yields the desired approximation ratio. Thus, w.l.o.g. we can assume that the critical volume is upper-bounded as $1/(n-2)(1/3+\epsilon/2)$. We can apply Lemma 4.7 with $\ell = 1/2$ and $P = 1$ to obtain that the $(1+1)$ EA reaches an f-value of at most $1/2 + \delta/4 + 1/2(1/(n-2)(1/3+\epsilon/2))$ after expected $2\lceil en\log(2/\delta)\rceil$ steps. In the remaining part of the proof we give an upper bound of δ in order to reach the desired approximation ratio. Since $OPT = 1/2$ we have that

$$\frac{1}{OPT}\left(\frac{1}{2} + \frac{\delta}{4} + \frac{1}{2}\left(\frac{1}{n-2}\left(\frac{1}{3}+\frac{\epsilon}{2}\right)\right)\right) \le \frac{4}{3}$$

from which it follows that the equation above is satisfied by taking

$$\delta \le \frac{2n - 3(\epsilon+2)}{3(n-2)}$$

for n sufficiently large. Therefore, from Lemma 4.7 we obtain an upper bound on the expected number of steps as

$$2\left\lceil en\log\left(\frac{6(n-2)}{2n-3(\epsilon+2)}\right)\right\rceil$$

for any constant $0 < \epsilon < 1/3$, and for n sufficiently large. □

Using a similar argument we show that the $(1+1)$ EA reaches a $(4/3-\epsilon)$-approximation after $O(n)$ fitness evaluations at least with constant probability $\Omega(1)$. Again, using Lemma 4.7 the following lemma holds.

Lemma 4.9. *Fix a constant $0 < \epsilon < 1/3$. On the instance W_ϵ the $(1+1)$ EA reaches a $(4/3-\epsilon)$-approximation after $\lceil en\log\gamma\rceil$ many fitness evaluations, w.p. at least $(\gamma-1)/(4\gamma^3)$, for all $\gamma > 1$.*

Proof. Suppose that the two big jobs are assigned to different machines at the beginning of the process. This event occurs w.p. 1/2. Assuming that the two big jobs are never moved in this phase, then the critical job size is bounded from above as

$$s^* \le \frac{1}{(n-2)}\left(\frac{1}{3}+\frac{\epsilon}{2}\right). \tag{3}$$

We use Lemma 4.7 to obtain that the algorithm reaches an f-value of at most $1/2 + \delta/4 + 1/2(1/(n-2)(1/3+\epsilon/2))$ after $\lceil en\log(\gamma/\epsilon)\rceil$, w.p. at least $1-\gamma^{-1}$. We solve the inequality

$$\frac{1}{OPT}\left(\frac{1}{2} + \frac{\delta}{4} + \frac{1}{2}\left(\frac{1}{n-2}\left(\frac{1}{3}+\frac{\epsilon}{2}\right)\right)\right) \le \frac{4}{3} - \epsilon$$

w.r.t. the variable δ, to obtain that any choice

$$\delta \le \frac{2n - 6n\epsilon + 9\epsilon - 6}{3n-6}$$

yields the desired approximation ratio, for n sufficiently large. Therefore, the $(1+1)$ EA reaches the desired approximation ratio within at most

$$\left\lceil en\log\left(\frac{\gamma(3n-6)}{2n-6n\epsilon+9\epsilon-6}\right)\right\rceil \le \lceil en\log\gamma\rceil$$

steps w.p. at least $(1/2)(1-\gamma^{-1})$, and for n sufficiently large. We conclude by estimating a lower bound on the probability that the two big jobs are not moved in this phase. Since the probability of performing a chosen bit-flip is approximately $1/en$, then we can obtain the desired lower bound as

$$\left(1-\frac{2}{en}\right)^{\lceil en\log\gamma\rceil} \ge \frac{1}{2\gamma^2}.$$

for n sufficiently large. We conclude that the $(1+1)$ EA reaches at least the desired approximation ratio after $\lceil en\log\gamma\rceil$ w.p. at lest $1/4(1-\gamma^{-1})\gamma^{-2} = (\gamma-1)/(4\gamma^3)$. □

We remark that in the proof given above the choice $\gamma = 1/2$ is arbitrary. A similar result holds by choosing γ to be any constant $0 < \gamma < 1$. We can use the theorem above to give an upper bound on the run time of the $(1+1)$ $EA_{\mathcal{U}}$. We first identify a non-optimal improving restart strategy and then apply Theorem 2.4. The following theorem holds.

Theorem 4.10. *Fix a constant $0 < \epsilon < 1/3$. On the instance W_ϵ the $(1+1)$ $EA_{\mathcal{U}}$ reaches a $(4/3-\epsilon)$-approximation after expected $O(n\log n)$ fitness evaluations.*

Proof. We proceed by first defining a (non-optimal) strategy that yields polynomial run time, and then use this information to find an upper bound on the run time of the $(1+1)$ $EA_{\mathcal{U}}$. For all $\gamma > 1$, we define the strategy $\mathcal{R} := (\bar{t}, \dots, \bar{t}, \dots)$ with $\bar{t} = \lceil en\log\gamma\rceil$. From Lemma 4.9 we can lower-bound the cumulative distribution

function $q(\bar{t})$ as $q(\bar{t}) \geq (\gamma - 1)/(4\gamma^3)$. If we fix $\gamma = \Theta(1)$ then from Lemma 2.3 it follows that

$$\mathbb{E}[\tau_{\mathcal{R}}] \leq \frac{\bar{t}}{q(\bar{t})} \leq \frac{4\gamma^3}{\gamma - 1}\lceil en \log \gamma \rceil = O(n)$$

where $\tau_{\mathcal{R}}$ denotes the run time of the (1 + 1) *EA* following the strategy $\mathcal{R}$. We apply Corollary 2.5 to conclude that $\mathbb{E}[\tau_{\mathcal{U}}] = O(\mathbb{E}[\tau_{\mathcal{R}}] \log \mathbb{E}[\tau_{\mathcal{R}}]) = O(n \log n)$. □

In the analysis above, the key observation is that if larger jobs are placed evenly on the two machines at the beginning of the process, then the run time significantly improves. It is possible to generalize this idea, to obtain that the (1+1) *EA* reaches a $(1+\epsilon)$-approximation on any instance of the partition problem in expected $\lceil en \log(2/\epsilon) \rceil$ steps, w.p. at least $2^{(-e \log e + e)\lceil 2/\epsilon \rceil \ln(2/\epsilon) - \lceil 2/\epsilon \rceil}$ (cf. Theorem 3 in Witt [20]). We can use this result to derive the following upper-bound on the run time of the (1 + 1) $EA_{\mathcal{U}}$ on any instance of the Partition problem.

Lemma 4.11. *On any instance of the Partition Problem, the* (1 + 1) $EA_{\mathcal{U}}$ *finds a* $(1 + \epsilon)$*-approximation after expected*

$$O\left(n \log\left(\frac{1}{\epsilon}\right)\left(\log n + \left\lceil \frac{1}{\epsilon} \right\rceil \log\left(\frac{1}{\epsilon}\right)\right) 2^{\left\lceil \frac{2}{\epsilon} \right\rceil \left((e \log e - e) \ln\left(\frac{2}{\epsilon}\right) + 1\right)}\right)$$

fitness evaluations, for all $\epsilon > 4/n$.

Proof. Again, we first define a (non-optimal) restart strategy, and then we use Corollary 2.5 to give an upper-bound on the run time of the (1 + 1) $EA_{\mathcal{U}}$. We consider the restart strategy $\mathcal{R} = (\bar{t}, \ldots, \bar{t}, \ldots)$ with $\bar{t} = \lceil en \log(2/\epsilon) \rceil$. From Theorem 3 in Witt [20] it follows that

$$q(\bar{t}) \geq 2^{(-e \log e + e)\lceil 2/\epsilon \rceil \ln(2/\epsilon) - \lceil 2/\epsilon \rceil}.$$

We denote with $\tau_{\mathcal{R}}$ the run time of the (1 + 1) *EA* following the restart strategy $\mathcal{R}$. Then it follows that

$$\mathbb{E}[\tau_{\mathcal{R}}] \leq \frac{\lceil en \log(2/\epsilon) \rceil}{q(\bar{t})} \leq \lceil en \log(2/\epsilon) \rceil 2^{(e \log e - e)\lceil 2/\epsilon \rceil \ln(2/\epsilon) + \lceil 2/\epsilon \rceil}.$$

Again, from Corollary 2.5 it follows that the algorithm reaches the desired approximation ratio after $O(\mathbb{E}[\tau_{\mathcal{R}}] \log \mathbb{E}[\tau_{\mathcal{R}}])$ fitness evaluations. □

Note that from Lemma 4.11 it follows that the (1+1) $EA_{\mathcal{U}}$ serves as an EPTAS for the Partition Problem, for a $(1 + \epsilon)$-approximation with $\epsilon > 4/n$. In other words, for the (1 + 1) $EA_{\mathcal{U}}$ an increase in problem size has the same relative effect of $O(n \log n)$ on the Run Time regardless of the approximation ratio, for $\epsilon > 4/n$.

5 EXPERIMENTS

Given the positive results of Section 3.1, we experimentally compare the performance of the $(\mu+1)$ *EA* with the (1+1) $EA_{\mathcal{U}}$ (cf. Algorithm 2), on the MVC problem.

We consider a network that was collected from survey participants using a Facebook app (cf. McAuley and Leskovec [12]). Each node in the dataset represent a user. Two nodes are connected if the respective users are Facebook friends. The dataset was anonymized by replacing the Facebook-internal ids for each user with a new value. The resulting graph has 2888 vertices and 2981 edges. Its maximum node degree is $\Delta = 769$. A visualization of this network is presented in Figure 1. We choose this network because some of its characteristics, such as community structure, are commonly observed on a larger scale.

We search for the MVC of the network described above with the (1 + 1) *EA*, $(\mu + 1)$ *EA* with $\mu = 10, 30, 70, 80$, and (1 + 1) $EA_{\mathcal{U}}$. We approach the problem by minimizing the function $(u(x), |x|_1)$ in lexicographical order, with $u(x)$ the function that returns the number of uncovered edges. We let each algorithm run for a given time budget and look at the sample mean and sample standard deviation of the best solution found in that time frame. The time budget is given in number of calls to the fitness function. The results are displayed in Figure 2. We observe that the (1 + 1) $EA_{\mathcal{U}}$ outperforms the $(\mu + 1)$ *EA* for any choice of μ hereby considered. Note that the $(\mu + 1)$ *EA* with $\mu = 70$ outperforms the $(\mu + 1)$ *EA* with $\mu = 80$. This indicates that increasing μ for $\mu > 70$ may not result in a better performance.

All tests are performed on MacBook Pro (13" Retina, Beginning 2015), with operating system Mac OS X Version 10.13.2, processor 2.7GHz dual-core Intel Core i5 (Turbo Boost up to 3.1GHz) with 3MB shared L3 cache, and memory 8GB of 1866MHz LPDDR3. All algorithms are implemented in C++ on Xcode Version 9.2 (9C40b), and implemented as OSX command line executables.

approximation ratio	general case	worst case no restarts	worst case w/ restarts
$4/3 + \epsilon$	$O(n)$	$O(n)$	$O(n \log n)$
$4/3$	$O(n^2)$	$O(n)$	$O(n \log n)$
$4/3 - \epsilon$	$n^{\Omega(n)}$	$n^{\Omega(n)}$	$O(n \log n)$

Table 1: Run time of the (1 + 1) *EA* **and** (1 + 1) $EA_{\mathcal{U}}$ **on the Makespan Sheduling, for a set of size** n **and a fixed constant** $0 < \epsilon < 1/3$**, as given in Lemma 4.8 and Theorem 4.10. The bounds for the general case, and the worst case for a** $(4/3-\epsilon)$**-approximation follow from Witt [20].**

6 CONCLUSIONS

In this paper we adapt the work on restart strategies by Luby et al. [15] to the of case of the simple (1 + 1) *EA*. We show that the universal restart strategy - that does not require any parameter tuning - can be an effective tool to escape local optima. We discuss some instances of commonly studied problems by which the run time of the (1 + 1) *EA* is exponential, whereas the run time of the (1 + 1) $EA_{\mathcal{U}}$ is polynomial on those instances.

We first study two pseudo-boolean landscapes, the TwoMax and function f_h as defined in Equation 1. In both cases the (1 + 1) *EA* performs poorly (cf. Friedrich et al. [5] and Lissovoi et al. [13]), whereas the run time of the (1 + 1) $EA_{\mathcal{U}}$ is upper-bounded as $O(n \log^2 n)$ (cf. Theorem 3.3 and Theorem 3.5).

We then consider the Minimum Vertex Cover problem. We focus on complete bipartite graphs and on Papadimitriou-Steiglitz (PS) instances. In the first case the (1 + 1) *EA* yields run time at least $\Omega(n^{n/\epsilon})$ with ϵn the size of the smallest partition; in the second case the (1 + 1) *EA* has run time at least $2^{\Omega(\sqrt[3]{n})}$ (cf. Oliveto et al.

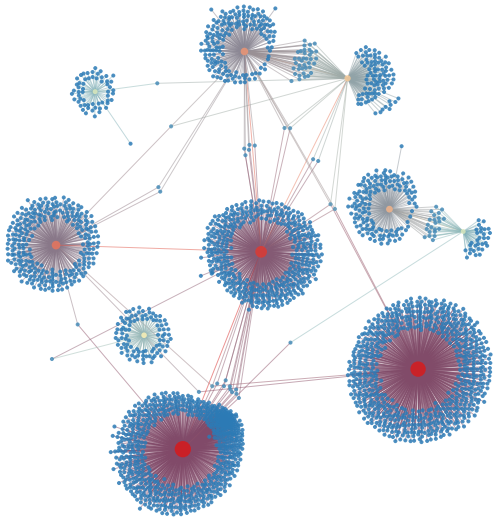

Figure 1: A visualization of the Facebook dataset presented in McAuley and Leskovec [12]. Each node consists of a Facebook user. For every two nodes there is a connecting edge if the corresponding users are Facebook friends.

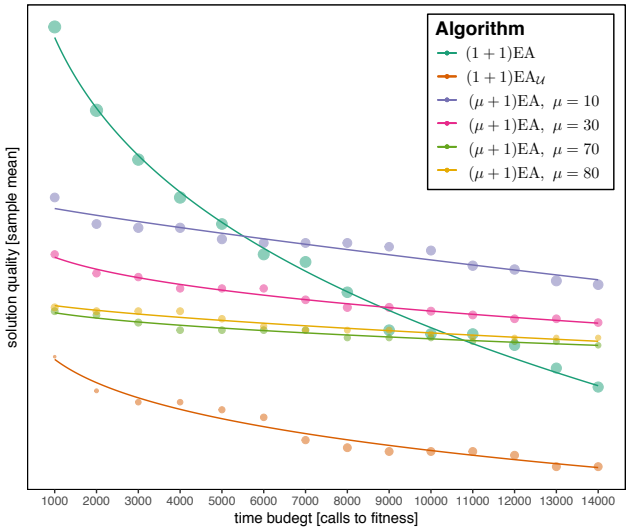

Figure 2: Run time of the $(1+1)$ EA and $(\mu+1)$ EA with $\mu = 10, 30, 70, 80$, and the $(1+1)$ $EA_{\mathcal{U}}$ for a fixed time budget, on the network displayed in Figure 1. Each dot corresponds to the sample mean of 50 runs, and its size is proportional to the sample standard deviation. The trend of each set of experiments is highlighted with a possible fitting curve. We see that the $(1+1)$ $EA_{\mathcal{U}}$ outperforms the $(1+1)$ EA and the $(\mu+1)$ EA for any choice of μ.

[19]). On both classes of instances the $(1+1)$ $EA_{\mathcal{U}}$ has run time $O(n \log^2 n)$ (cf. Theorem 4.4 and Lemma 4.3).

We study the Makespan Scheduling on two machines with equal computational power. Following the work of Witt [20], we prove that the $(1+1)$ $EA_{\mathcal{U}}$ outperforms the $(1+1)$ EA on a worst-case instance (cf. Theorem 4.10 and Table 1). We show that the $(1+1)$ $EA_{\mathcal{U}}$ serves as a EPTAS for the Partition problem, for sufficiently large approximation (cf. Lemma 4.11).

We experimentally compare the run time of the $(1+1)$ EA, the $(1+1)$ $EA_{\mathcal{U}}$, and the $(\mu+1)$ EA. We consider a network taken from Facebook (cf. McAuley and Leskovec [12]), and we search for the Minimum Vertex Cover. We observed that for fixed time budget the $(1+1)$ $EA_{\mathcal{U}}$ outperforms all other algorithms (cf. Figure 2).

We plan to further explore the relationship between randomized algorithms, fitness landscapes and restart strategies in the future.

7 ACKNOWLEDGEMENTS

The research leading to these results has received funding from the German Science Foundation (DFG) under grant agreement FR 2988 (TOSU).

REFERENCES

[1] Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. 2015. Money for Nothing: Speeding Up Evolutionary Algorithms Through Better Initialization. In *Proc. of GECCO '15*. 815–822.
[2] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276, 1-2 (2002), 51–81.
[3] Matteo Fischetti and Michele Monaci. 2014. Exploiting Erraticism in Search. *Operations Research* 62, 1 (2014), 114–122.
[4] Tobias Friedrich, Timo Kötzing, and Markus Wagner. 2017. A Generic Bet-and-Run Strategy for Speeding Up Stochastic Local Search. In *Proc. of AAAI*. 801–807.
[5] Tobias Friedrich, Pietro Simone Oliveto, Dirk Sudholt, and Carsten Witt. 2009. Analysis of Diversity-Preserving Mechanisms for Global Exploration. *Evolutionary Computation* 17, 4 (2009), 455–476.
[6] Alex S. Fukunaga. 1998. Restart Scheduling for Genetic Algorithms. In *Proc. of PPSN*. 357–366.
[7] András György and Levente Kocsis. 2011. Efficient Multi-Start Strategies for Local Search Algorithms. *Journal of Artificial Intelligence Research* 41 (2011), 407–444.
[8] Jun He, Xin Yao, and Jin Li. 2005. A comparative study of three evolutionary algorithms incorporating different amounts of domain knowledge for node covering problem. *IEEE Trans. Systems, Man, and Cybernetics, Part C* 35, 2 (2005), 266–271.
[9] Xiaohua Hu, Ronald Shonkwiler, and Marcus C. Spruill. 1994. *Random restarts in global optimization*. Technical Report 110592-015. School of Mathematics, Georgia Institute of Technology.
[10] Sami Khuri and Thomas Bäck. 1994. An Evolutionary Heuristic for the Minimum Vertex Cover Problem. In *Proc. KI-94 Workshop Genetic Algorithms Within Framework Evol. Comput.* 86–90.
[11] Timo Kötzing, Dirk Sudholt, and Madeleine Theile. 2011. How crossover helps in pseudo-boolean optimization. In *Proc. of GECCO*. 989–996.
[12] Jure Leskovec and Julian J. Mcauley. 2012. Learning to Discover Social Circles in Ego Networks. In *Proc. of NIPS*. 539–547.
[13] Andrei Lissovoi, Dirk Sudholt, Markus Wagner, and Christine Zarges. 2017. Theoretical Results on Bet-and-run As an Initialisation Strategy. In *Proc. of GECCO*. 857–864.
[14] Helena Ramalhinho Lourenço, Oliver C. Martin, and Thomas Stützle. 2003. *Iterated Local Search*. 320–353.
[15] Michael Luby, Alistair Sinclair, and David Zuckerman. 1993. Optimal Speedup of Las Vegas Algorithms. *Inform. Process. Lett.* 47, 4 (1993), 173–180.
[16] Sean Luke. 2001. When Short Runs Beat Long Runs. In *Proc. of GECCO*. 74–80.
[17] Malik Magdon-Ismail and Amir F. Atiya. 2000. The Early Restart Algorithm. *Neural Computation* 12, 6 (2000), 1303–1312.
[18] Heinz Mühlenbein. 1992. How Genetic Algorithms Really Work: Mutation and Hillclimbing. In *Proc. of PPSN-II*. 15–26.
[19] Pietro Simone Oliveto, Jun He, and Xin Yao. 2009. Analysis of the (1+1) -EA for Finding Approximate Solutions to Vertex Cover Problems. *IEEE Trans. Evolutionary Computation* 13, 5 (2009), 1006–1029.
[20] Carsten Witt. 2005. Worst-Case and Average-Case Approximations by Simple Randomized Search Heuristics. In *Proc. of STACS*. 44–56.