A Parameterized Runtime Analysis of Simple Evolutionary Algorithms for Makespan Scheduling

Andrew M. Sutton and Frank Neumann

School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia

Abstract. We consider simple multi-start evolutionary algorithms applied to the classical NP-hard combinatorial optimization problem of Makespan Scheduling on two machines. We study the dependence of the runtime of this type of algorithm on three different key hardness parameters. By doing this, we provide further structural insights into the behavior of evolutionary algorithms for this classical problem.

1 Introduction

Evolutionary algorithms and other types of bio-inspired computing techniques have been extensively used for a wide range of combinatorial optimization problems. Understanding the behavior of evolutionary algorithms on NP-hard combinatorial optimization problems from a theoretical point of view is still a challenging task. Results on the runtime of evolutionary algorithms for different combinatorial optimization problems have been obtained during the last ten years. We refer the interested reader to the textbook of Neumann and Witt [10] for an overview on this area of research. One of the first runtime analyses of evolutionary algorithms for NP-hard combinatorial optimization problems has been carried out by Witt [14] by considering the Makespan Scheduling problem. Witt has studied the approximation and average-case behavior of simple evolutionary algorithms for this problem. Gunia [7] later extended this work to the case of multiple machines. Other recent works have studied the multiple machine case in terms of convergence to solutions corresponding to Nash equilibria in multiplayer non-cooperative games [4,6].

We consider the analysis of evolutionary algorithms in the context of fixed-parameter tractability by expressing their runtime as a function of both problem size and an additional hardness parameter that attempts to isolate the exponential complexity of the instance. This approach, which is widely used in the classical analysis of algorithms and problem hardness [3], has recently been introduced into the analysis of evolutionary algorithms. It facilitates the understanding of which features in an instance of a given problem makes the problem hard to solve. Parameterized runtime results have been obtained in the context of evolutionary computation for the vertex cover problem [9], the computation of maximum leaf spanning trees [8], the MAX-2-SAT problem [12], and the Euclidean TSP [13].

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part I, LNCS 7491, pp. 52–61, 2012.

Our goal is to provide further insights into the optimization process of evolutionary algorithms for the Makespan Scheduling problem by carrying out parameterized runtime analyses. We show that multi-start variants of two simple evolutionary algorithms are fixed-parameter evolutionary algorithms for a parameterization of Makespan Scheduling that takes into account the value of the optimal schedule above its theoretical lower bound. We then study their runtime in dependence of the critical path size of an optimal schedule. Finally, we investigate a parameterization that considers the machine load discrepancy in an optimal schedule. We show that, with a minor modification to the mutation procedure, the resulting multi-start variant of RLS is a Monte Carlo fixed-parameter tractable algorithm for Makespan Scheduling. This indicates that instances with large discrepancies will be easier to solve by randomized local search.

2 Preliminaries

We investigate the classical NP-hard Makespan Scheduling problem on two identical machines. In this problem, we have a set of n jobs where each job j requires a nonzero integral processing time p_j on either machine. We define the *load* of a machine to be the sum of processing times of the jobs that are assigned to it. The makespan of a schedule is the maximum load over both machines. The objective is to find an assignment that minimizes the makespan.

An arbitrary schedule can be represented as a binary length-n decision vector where the j-th component specifies to which machine job j is assigned in a schedule. For a given instance of Makespan Scheduling, the makespan of a schedule corresponding to a binary decision vector $x \in \{0,1\}^n$ is captured by the pseudo-Boolean function

$$f : \{0,1\}^n \to \mathbb{N} := x \mapsto \max \left\{ \sum_{j=1}^{n} x_j p_j, \sum_{j=1}^{n} (1 - x_j) p_j \right\}.$$

We will denote $P = \sum_{j=1}^{n} p_j$. Thus $P/2 \leq f(x) \leq P$. Without loss of generality, we will assume that the processing times are sorted in nonincreasing order, i.e., $p_1 \geq \cdots \geq p_n$. We denote $f^* = \min_{x \in \{0,1\}^n} f(x)$ as the value of the optimal makespan for an instance.

We will carry out parameterized runtime analyses of evolutionary algorithms for Makespan Scheduling. Let Σ be a finite alphabet. A *parameterized problem* over Σ is a pair (L, κ) where $L \subseteq \Sigma^*$ is a language over Σ and $\kappa : \Sigma^* \to \mathbb{N}$ is a map called a *parameterization* of Σ. Letting $n = |x|$ and $k = \kappa(x)$, a parameterized problem (L, κ) is *fixed-parameter tractable* if there is an algorithm that decides $x \in L$ in time bounded by $g(k) \cdot poly(n)$ where g is an arbitrary recursive function that depends only on k. We call such an algorithm an fpt-algorithm. The class of parameterized problems (L, κ) that can be decided by an fpt-algorithm is called FPT. A *Monte Carlo fpt-algorithm* for (L, κ) is a randomized fpt-algorithm with runtime bounded by $g(k) \cdot poly(n)$ that will accept an input $x \in \Sigma^*$ with probability at least $1/2$ if $x \in L$, otherwise it accepts with

probability zero. An XP-algorithm for a parameterized problem (L, κ) is an algorithm that runs in worst-case time $n^{g(k)}$. We define a *Monte Carlo* XP*-algorithm* analogously.

We consider two classical mutation-only evolutionary algorithms, the (1+1) EA and RLS. In particular, we will analyze repeating runs of length $\ell(n) = O(poly(n))$ and take the best solution found during any run. A run of length $\ell(n)$ for the (1+1) EA and RLS is explicitly defined in Algorithms 1 and 2, respectively. In each case, we will investigate the probability that a single run solves the parameterized problem. Observing that each run is an independent Bernoulli trial, it will be straightforward to bound the failure probability after a prescribed number of runs. We will make use of the following technical lemma.

Lemma 1. *Let h be a positive function. The probability that an arbitrary (but nonempty) set of $k < n$ bits is never changed during a run of length $\ell(n) = n \cdot h(n)$ is bounded by $\Omega\big(e^{-k \cdot h(n)}\big)$ for the (1+1) EA, and $\Omega\big(e^{-(k \log k) \cdot h(n)}\big)$ for RLS.*

Proof. For the (1+1) EA, the probability that none of the specified k bits are mutated during a single iteration is $(1 - 1/n)^k$. After $\ell(n)$ iterations, the probability is $(1-1/n)^{kn \cdot h(n)} = \Omega(e^{-k \cdot h(n)})$. For RLS, the probability that none of the specified k bits are changed in a single iteration is $(1 - k/n)$. Here, we must also consider the rate k grows as a function of n. If $k = o(n)$, the bound is obviously the same for the (1+1) EA, otherwise, $k = \Theta(n)$. In the case that $c_1 n \leq k \leq c_2 n$ for some constants $0 < c_1 \leq c_2 < 1$, then we have $(1-k/n)^n \geq (1-c_2)^{k/c_1} = e^{-k\epsilon}$ where ϵ is a positive constant. Finally, in the case that $k \sim n$, since k is at most $n-1$, it must hold that $(1-k/n)^n \geq e^{-n \log n}$ and since in this case $n = (1+o(1))k$, the asymptotic bound holds. □

Algorithm 1. A single run of the (1+1) EA

```
   input  : A run length ℓ(n)
   output: A candidate decision vector x
1  Choose x uniformly at random from {0,1}^n;
2  for i ← 1 to ℓ(n) do
3  |  x' ← x;
4  |  Flip each bit of x' independently with probability 1/n;
5  |  if f(x') ≤ f(x) then x ← x'
6  end
```

3 Parameterized Analysis for Optimal Makespan Value

The *standard parameterization* of a combinatorial optimization problem is (assuming minimization), given an instance and a parameter k, is the value of the optimal solution *at most* k? Fernau [5] has shown that the standard parameterization of Makespan Scheduling[1] is fixed-parameter tractable. The proof relies on the following straightforward kernelization technique. If $k < P/2$, the

[1] Makespan Scheduling is referred to as Minimum Partition in Fernau's work.

Algorithm 2. A single run of RLS

```
input  : A run length ℓ(n)
output: A candidate decision vector x
Choose x uniformly at random from {0,1}^n;
for i ← 1 to ℓ(n) do
    x' ← x;
    Choose j uniformly at random from {1,...,n};
    x'_j ← (1 − x'_j);
    if f(x') ≤ f(x) then x ← x'
end
```

answer is always "no" since clearly f is bounded below by $P/2$. On the other hand, if $k \geq P/2$, it follows that $2k \geq P \geq n$, the rightmost inequality coming from the fact that the processing times are positive integers. Hence there are at most 2^{2k} schedules which can be search exhaustively in time bounded by $O(4^k)$.

To provide stronger insights into the difficulty of Makespan Scheduling as a function of the value of the optimal makespan, we will consider a more detailed parameterization that captures the difference between the makespan of an optimal schedule and the theoretical lower bound. In particular, we show that if the optimal schedule has a makespan much larger than $P/2+P/n$, the problem is easier to solve by the (1+1) EA and RLS using a multi-start approach. We show that the multi-start variants of both the (1+1) EA and RLS are Monte Carlo fpt-algorithms for Makespan Scheduling by showing they are capable of simulating a polynomial-time approximation scheme (PTAS).

We will hereafter assume that $p_1 \leq P/2$, otherwise it is easy to show that RLS always runs in expected polynomial time simply by collecting all the smaller jobs onto the other machine. A move could result in an improving solution if it shifts a job from the fuller machine to the emptier machine. We follow Witt [14] and define the *critical job size* $s(x)$ with respect to a decision vector x as the processing time of the smallest job on the fuller machine. If $f(x) > (P + s(x))/2$, then it is possible to construct an improving schedule by moving at least one job from the fuller machine to the emptier machine. The optimal solution parameterization is, given an instance of Makespan Scheduling and an integer k, is $f^* \leq P/2 + P/k$?

Lemma 2 (due to Witt [14]). *Let x be the current search point. Suppose the critical job size is bounded above by s^* for all following search points of value greater than $L + s^*/2$ where $L \geq P/2$. Then for any $\gamma > 1$ and $0 < \delta < 1$, both the (1+1) EA and RLS can compute a decision vector with makespan at most $L + s^*/2 + \delta P/2$ in at most $\lceil en \ln(\gamma/\delta) \rceil$ steps with probability at least $1 - \gamma^{-1}$.*

Lemma 3. *Given some $1 \leq k \leq n$, let x' be a decision vector such that the contribution of jobs $1, \ldots, k$ is minimal. The probability that after a run of length $\lceil en \ln(2k) \rceil$ the (1+1) EA or RLS has discovered a schedule with makespan at most $P/2 + P/k$ is bounded below by $\Omega(e^{-k\lceil e \ln(2k) \rceil})$ for the (1+1) EA, and $\Omega(e^{-(k \log k)\lceil e \ln(2k) \rceil})$ for RLS.*

Proof. As long as no move involves the first k bits, the critical job size s^* is bounded above by p_k. Furthermore, since $kp_k \leq p_1 + \cdots + p_k \leq P$, it follows that p_k is at most P/k. By Lemma 2, by setting L to $P/2$, s^* to P/k, $\gamma = 2$, and δ to $1/k$, the probability that we reach a solution $\hat{x}$ where

$$f(\hat{x}) \leq L + P/(2k) + (1/k)(P/2) = P/2 + P/k$$

in $\lceil en \ln(2k) \rceil$ steps is at least $1/2$, as long as none of the first k jobs are moved.

Thus, if q denotes the probability that none of the first k bits are mutated during a run of length $\lceil en \ln(2k) \rceil$, then the solution is reached with probability at least $q/2$. The proof is completed by appealing to Lemma 1 for the lower bound on q and using the fact that $\lceil en \ln(2k) \rceil \leq n \lceil e \ln(2k) \rceil$.

Theorem 1. *The multi-start (1+1) EA (RLS) using runs of length* $\ell(n) = \lceil en \ln(2k) \rceil$ *is a Monte Carlo fpt-algorithm for the optimal makespan parameterization of* Makespan Scheduling.

Proof. Consider an arbitrary instance of Makespan Scheduling. If $f^* > P/2 + P/k$ the proof is complete since the output of the algorithm in this case is irrelevant. Thus we suppose that $f^* \leq P/2 + P/k$.
The probability that a random initial solution to any run contains the first k jobs properly fixed is at least 2^{-k+1}. Given such a solution, let $q(n)$ denote the probability that, after a run of length $\lceil en \ln(2k) \rceil$, the algorithm has found a schedule $\hat{x}$ where $f(\hat{x}) \leq P/2 + P/k$. The probability that t consecutive runs are all unsuccessful is at most $(1 - q(n)/2^{k-1})^t$. Setting $t = \lceil 2^{k-1} q(n)^{-1} \rceil$ gives a failure probability of at most $1/e$. Since each run consists of $O(n \log k)$ evaluations, the total runtime is $O(tn \log k)$. Due to Lemma 3, $q(n)$ is bounded by a function depending only on k for both the (1+1) EA and RLS. Thus by setting $g(k) = 2^{k-1} q(n)^{-1}$, the total runtime is bounded by $O(g(k) \cdot n \log k)$ and the success probability is at least $1 - 1/e > 1/2$.

4 Parameterized Analysis for Critical Path Size

In general machine scheduling problems, the *critical path* of a schedule is a set of consecutive jobs in which the first job starts at time zero, the completion time of the last job is the makespan of the schedule, and the completion time of each job is equal to the starting time of the next [11]. For the two-machine Makespan Scheduling problem, we define the critical path of a schedule as the set of jobs scheduled on the fuller machine. Formally, the critical path of a schedule x is the set $\mathcal{C}(x) \subseteq [n]$ such that for all $i, j, \in \mathcal{C}(x)$, $x_i = x_j$ and $\sum_{i \in \mathcal{C}(x)} p_i = f(x)$. In the ambiguous case (when the machines balance) we define the critical path as the smallest such set with ties in cardinality broken arbitrarily. We define the *critical path size* of a schedule x as $|\mathcal{C}(x)|$. The *critical path size parameterization* of Makespan Scheduling is, given an integer k, is there a schedule with critical path size at most k?

Lemma 4. *Consider an instance of* Makespan Scheduling *such that there exists a schedule z with $|\mathcal{C}(z)| \leq k$. Suppose x' corresponds to a schedule such that for all $i, j \in \mathcal{C}(z)$, $x'_i = x'_j$. We call a run of the (1+1) EA (RLS) a* success *if it discovers a schedule with critical path size at most k. Then starting with x' as the initial decision vector, for any constant $c > 1$, the success probability of a run of the (1+1) EA of length $\lceil cen\,(\ln n + \ln p_1 + 1)\rceil$ is bounded by $\Omega\big((enp_1)^{-cek}\big)$. Moreover, the success probability of a run of RLS of length $\lceil cn\,(\ln n + 1)\rceil$ starting from x' is bounded by $\Omega\big((en)^{-ck\log k}\big)$.*

Proof. Without loss of generality, suppose that for all $i, j \in \mathcal{C}(z)$, $x'_i = x'_j = 0$. If, for any $\ell \in [n]$, $x'_\ell = 0 \implies \ell \in \mathcal{C}(z)$, then the proof is complete. Otherwise, machine zero (i.e., the machine that corresponds to a zero bit in the decision vector) obviously must have the highest load since it contains every job in $\mathcal{C}(z)$. Let $S(x) = \{i : i \notin \mathcal{C}(z) \wedge x_i = 0\}$ be the set of jobs on machine zero that do not belong to $\mathcal{C}(z)$.
During a run of either the (1+1) EA or RLS, as long as none of the jobs in $\mathcal{C}(z)$ are not moved off machine zero, any move that reduces the number of jobs not in $\mathcal{C}(z)$ on machine zero is accepted. Furthermore, as long as the jobs in $\mathcal{C}(z)$ remain on machine zero, its load is at least the load of machine one. Thus, no moves which increase the number of jobs on machine zero are accepted.

For the (1+1) EA, let $d(x) = \sum_{i \in S(x)} p_i$. Suppose that no jobs from $\mathcal{C}(z)$ are moved off machine zero during a run of the (1+1) EA. In this case, any mutation involving an element of $S(x)$ is accepted and decreases the makespan (and the d value) by its processing time. Such a move occurs with probability at least $1/(en)$. By the multiplicative drift theorem [2], the expected number of steps until the d value has reached zero conditioned on the event that no bits corresponding to $\mathcal{C}(z)$ are flipped is at most $en\,(1 + \ln d(x')) \leq en\,(\ln n + \ln p_1 + 1)$ since $d(x') \leq np_1$. Consider a run of the (1+1) EA of length $t = cen(\ln n + \ln p_1 + 1)$. By the Markov inequality, the success probability of such a run conditioned on the event that no bit in $\mathcal{C}(z)$ is flipped is at least $1 - 1/c = \Omega(1)$. Hence the bound on the success probability is $\Omega\big((enp_1)^{-cek}\big)$ by Lemma 1.

For RLS, we set the run length to $\ell(n) = \lceil cn(\ln n + 1)\rceil$. The probability that RLS takes fewer than $\ell(n)$ steps to move the remaining $|S(x)|$ jobs conditioned on the event that no jobs in $\mathcal{C}(z)$ are moved is at least $1 - (ne)^{-c+1} = 1 - o(1)$. This result comes from the classical coupon collector analysis (see Theorem 1.23 in Chapter 1 of [1]). The bound on the success probability of such a run of RLS then follows directly from Lemma 1. □

Theorem 2. *For any constant $c > 1$, a multi-start (1+1) EA procedure using a run length of $\ell(n) = \lceil cen(\ln n + \ln p_1 + 1)\rceil$ solves the critical path size parameterization in at most $O\big(2^k (enp_1)^{cek} \cdot n(\log n + \log p_1)\big)$ evaluations with probability at least $1/2$. Moreover, a multi-start RLS procedure using a run length of $\ell(n) = \lceil cn\,(\ln n + 1)\rceil$ solves the critical path size parameterization in at most $O\big(2^k (en)^{ck\log k} \cdot n \log n\big)$ evaluations with probability at least $1/2$.*

Proof. Consider an arbitrary instance of Makespan Scheduling. If there is no schedule z such that $|\mathcal{C}(z)| \leq k$, the proof is complete. Otherwise, suppose there exists such a schedule.
With probability at least $2 \cdot 2^{-k}$, the initial schedule x' of a run of the (1+1) EA (RLS) has $x'_i = x'_j$ for all $i, j \in \mathcal{C}(z)$. Let $q(n)$ denote the probability that a (1+1) EA run of length $\lceil cen(\ln n + \ln p_1 + 1) \rceil$ starting from x' generates a schedule with critical path size at most k. By Lemma 4, $q(n) = \Omega\big((enp_1)^{-cek}\big)$.

The probability that t consecutive runs of the required size of the (1+1) EA all fail to find such a schedule is at most $(1 - q(n)/2^{k-1})^t$. Hence, after $2^{k-1}q(n)^{-1} = O\big(2^k(enp_1)^{cek}\big)$ such runs of the (1+1) EA, the failure probability is at most $1/e$ and the parameterization is solved with probability $1 - 1/e > 1/2$. Since each run of the (1+1) EA costs $O(n(\log n + \log p_1))$ evaluations, we have the claimed runtime. The proof for RLS is analogous. □

It immediately follows from Theorem 2 that the multi-start RLS is a Monte Carlo XP-algorithm for the critical path size parameterization of Makespan Scheduling. We must, however, be slightly more careful in the case of the multi-start (1+1) EA since p_1 can be exponential in n. In this case, it follows that the multi-start (1+1) EA is a Monte Carlo XP-algorithm for inputs where all processing times are polynomially bounded in n.

5 A Monte Carlo fpt-Algorithm for Discrepancy

Following the terminology of Witt [14] we define the absolute difference in load across machines the *discrepancy* of a schedule, i.e., $\Delta(x) = 2f(x) - P$. Denoting as $\Delta^* = 2f^* - P$ the discrepancy of the optimal solution of an instance, we consider the following parameterized problem (for notational convenience, we set $p_{n+1} = 0$). Given an instance of Makespan Scheduling and an integer k, is $p_k \geq \Delta^* \geq p_{k+1}$?

We will consider two evolutionary algorithms, called k-biased (1+1) EA and k-biased-RLS which differ from the (1+1) EA and RLS by using a slightly modified mutation operator. We then consider the efficiency of these variants for solving the discrepancy parameterization. For the k-biased (1+1) EA, the mutation step in line 4 of Algorithm 1 is replaced with the following lines of code.

for $j \leftarrow 1$ **to** k **do** flip x'_j with probability $1/(kn)$;
for $j \leftarrow k+1$ **to** n **do** flip x'_j with probability $1/n$;

For the k-biased-RLS, the mutation step in lines 4 and 5 of Algorithm 2 are replaced with the following lines of code.

if $r < 1/n$ **then** choose j uniformly at random from $\{1, \ldots, k\}$;
else choose j uniformly at random from $\{k+1, \ldots, n\}$;
$x'_j \leftarrow (1 - x'_j)$;

These biased mutation operators have a smaller probability of flipping the bits on the first k positions compared to the ones presented in Section 2.

Lemma 5. *Let h be a positive function. The probability that the k-biased (1+1) EA (k-biased-RLS) does not change the first k bits during a run of length $\ell(n) = n \cdot h(n)$ is bounded by $\Omega(e^{-h(n)})$.*

Proof. For the k-biased (1+1) EA, the probability that none of k bits are selected for mutation in a single step is $(1 - 1/(kn))^k$. After $\ell(n)$ steps the probability that none of the first k bits have changed is at least $(1 - 1/(kn))^{kn \cdot h(n)}$. For k-biased-RLS, the probability that any of the first k bits are selected for mutation is $1/n$. After $\ell(n)$ steps, the first k bits have not changed with probability at least $(1-1/n)^{n \cdot h(n)}$. In both cases, the asymptotic bound follows from $(1-1/x)^{x \cdot f(x)} = \Omega(e^{-f(x)})$. □

Lemma 6. *Let k be such that $p_{k+1} \leq \Delta^*$ where $p_{n+1} = 0$. Let x' be a decision vector such that the contribution of jobs $1, \ldots, k$ to the makespan is minimal. We call a run of the k-biased (1+1) EA (k-biased-RLS) a* success *if it discovers an optimal schedule. Then starting with x' as the initial decision vector, the success probability for a run of the k-biased (1+1) EA of length $2en(\ln n + \ln p_1 + 1)$ is bounded below by $\Omega\big((np_1)^{-2e}\big)$. Moreover, the success probability for a run of k-biased-RLS of length $2n(\ln n + 1)$ is bounded below by $\Omega\big(n^{-2}\big)$.*

Proof. We assume $\Delta(x') > \Delta^* \geq 0$ since otherwise x' is already optimal. In this case there is a machine with a higher load. Let $S = \{k+1, k+2, \ldots, n\}$. We first show that as long as x' is not optimal and there are jobs from S on the fuller machine, moving any such job to the emptier machine results in a strictly improving move. Suppose not. Then there is a job $j > k$ on the fuller machine and $\Delta(x') \leq p_j$, otherwise moving p_j results in an improvement. But by definition, we have $p_j \leq \Delta^*$ which contradicts the non-optimality of x'. It follows that if the first k jobs already contribute minimally to the makespan, as long as no mutation involves the first k bits, the optimal schedule can be found by moving all jobs from S on to the emptier machine.

For the k-biased (1+1) EA, let $d(x) = \Delta(x) - \Delta^*$. The probability that a mutation removes a job in S from the fuller machine is at least

$$(1 - 1/(kn))^k \, (1 - 1/n)^{n-k-|S|} \, |S|^{-1} \geq (1 - 1/n)^{n-|S|} \, |S|^{-1} \geq 1/(en).$$

The expected time until the d value has reduced to zero conditioned on the event that no bits of index at most k are flipped follows from the multiplicative drift theorem of Doerr et al. [2] and is at most $t = en\,(1 + \ln d(x'))$. By the Markov inequality, the probability that this occurs after $2t$ steps (again, conditioned on the event that no bits with index at most k are flipped) is at least $1/2 = \Omega(1)$. The bound on the success probability of a run of length $2t$ follows from Lemma 5.

For k-biased-RLS, suppose there are i jobs from S on the fuller machine. The probability that k-biased-RLS moves one of these jobs to the emptier machine is at least

$$(1 - 1/n) \cdot i/(n-k) = \frac{n-1}{n} \cdot \frac{i}{n-k}$$

for $n > 1$. The expectation until all jobs from S are moved off the fuller machine conditioned on the event that no jobs in $[n] \setminus S$ are moved is at most

$$\frac{n}{n-1} \cdot (n-k) \sum_{i=1}^{n-k} 1/i \leq n(\ln n + 1)$$

since $k \geq 1$. By the Markov inequality, the probability that this occurs in a run of $2n(\ln n + 1)$ steps is at least $1/2 = \Omega(1)$. The final bound on the success probability comes from Lemma 5. □

We now prove that the k-biased (1+1) EA (on inputs with polynomially bounded processing times) and k-biased-RLS (for general processing times) are Monte Carlo fpt-algorithms for this parameterization. At this point, it might be tempting to assume that we require instance-specific knowledge in order to choose the appropriate value for k. Instead, we are interested in the following question. For a given and fixed k, is there a class of Makespan Scheduling instances for which k-biased-RLS and the k-biased (1+1) EA are efficient? We now prove that such a class must include instances where $p_k \geq \Delta^* \geq p_{k+1}$.

Theorem 3. *A multi-start k-biased-RLS procedure that uses a run length of $\ell(n) = \lceil 2n(\ln n + 1) \rceil$ is a Monte Carlo fpt-algorithm for the discrepancy parameterization of* Makespan Scheduling. *In particular, if the instance is a yes instance (that is, $p_k \geq \Delta^* \geq p_{k+1}$), it solves the problem after $O(2^k n^3 \log n)$ steps with probability $1 - 1/e$.*
Similarly, the multi-start (1+1) EA is a Monte Carlo fpt-algorithm for the discrepancy parameterization for inputs where the processing times are polynomially bounded in n.

Proof. Consider an arbitrary instance of Makespan Scheduling. If it is not the case that $p_k \geq \Delta^* \geq p_{k+1}$, the proof is complete since, in this case, the output of the algorithm is arbitrary. Thus we can assume the bounds on Δ^*. A single run of k-biased-RLS starts with the first k jobs contributing minimally to the makespan with probability at least 2^{-k+1}. Let $q(n)$ denote the probability that a k-biased-RLS run of length $\lceil 2n(\ln n + 1) \rceil$ is successful. The failure probability for t consecutive runs is at most $(1 - q(n)/2^{k-1})^t$. Setting $t = \lceil 2^{k-1} q(n)^{-1} \rceil$ gives a failure probability of at most $1/e$. By Lemma 6, $q(n) = \Omega(n^{-2})$. Thus, the probability that the algorithm solves the discrepancy parameterization of Makespan Scheduling in $t = O(2^k n^2)$ runs of length $O(n \log n)$ evaluations each is at least $1 - 1/e > 1/2$.
The proof for the multi-start k-biased (1+1) EA is identical, except we set $\ell(n) = \lceil 2en(\ln n + \ln p_1 + 1) \rceil$ and apply Lemma 6 to get $q(n) = \Omega\big((np_1)^{-2e}\big)$. Thus after $O\big(2^k (np_1)^{2e} n(\log n + \log p_1)\big)$ steps, the algorithm has solved the discrepancy parameterization with probability at least $1 - 1/e$.

6 Conclusion

The parameterized analysis of evolutionary algorithms allows for a deeper understanding of which structural parameters of an instance of a combinatorial

optimization problem makes it easy or hard to solve. With this paper, we have contributed to the parameterized runtime analysis of evolutionary algorithms. We studied the Makespan Scheduling problem previously analyzed by Witt from a worst case and average case perspective. Our results provide further insights into the behaviour of evolutionary algorithms for this classical problem. We have shown that multi-start variants of the (1+1) EA and RLS are Monte Carlo fpt-algorithms for a parameterization which considers the value of the optimal solution above its lower bound. We have performed a runtime analysis in dependence of the critical path size of an optimal solution, and shown that a multi-start variant of RLS is a Monte Carlo fpt-algorithm for a parameterization that considers the discrepancy in load across machines.

References

1. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics: Foundations and Recent Developments. World Scientific Publishing Company (2011)
2. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. In: Pelikan, M., Branke, J. (eds.) GECCO, pp. 1449–1456. ACM (2010)
3. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
4. Even-Dar, E., Kesselman, A., Mansour, Y.: Convergence time to Nash equilibrium in load balancing. ACM Transactions on Algorithms 3(3) (2007)
5. Fernau, H.: Parameterized Algorithmics: A Graph Theoretic Approach. Habilitationsschrift (English), Universität Tübingen (2005)
6. Goldberg, P.W.: Bounds for the convergence rate of randomized local search in a multiplayer load-balancing game. In: Chaudhuri, S., Kutten, S. (eds.) PODC, pp. 131–140. ACM (2004)
7. Gunia, C.: On the analysis of the approximation capability of simple evolutionary algorithms for scheduling problems. In: Beyer, H.G., O'Reilly, U.M. (eds.) GECCO, pp. 571–578. ACM (2005)
8. Kratsch, S., Lehre, P.K., Neumann, F., Oliveto, P.S.: Fixed Parameter Evolutionary Algorithms and Maximum Leaf Spanning Trees: A Matter of Mutation. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 204–213. Springer, Heidelberg (2010)
9. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. In: Rothlauf, F. (ed.) GECCO, pp. 293–300. ACM (2009)
10. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity. Springer (2010)
11. Pinedo, M.: Scheduling: theory, algorithms, and systems. Springer (2012)
12. Sutton, A.M., Day, J., Neumann, F.: A parameterized runtime analysis of evolutionary algorithms for MAX-2-SAT. In: GECCO. ACM (to appear, 2012)
13. Sutton, A.M., Neumann, F.: A parameterized runtime analysis of evolutionary algorithms for the Euclidean traveling salesperson problem. In: AAAI. AAAI Press (to appear, 2012)
14. Witt, C.: Worst-Case and Average-Case Approximations by Simple Randomized Search Heuristics. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 44–56. Springer, Heidelberg (2005)