Runtime Analysis of (1 + 1) Evolutionary Algorithm Controlled with Q-learning using Greedy Exploration Strategy on OneMax+ZeroMax Problem

Denis Antipov[1], Maxim Buzdalov[1], and Benjamin Doerr[2]

[1] ITMO University, 49 Kronverkskiy av.,
Saint-Petersburg, Russia, 197101
antipovden@yandex.ru, mbuzdalov@gmail.com
[2] LIX, École polytechnique, 91128 Palaiseau Cedex
doerr@lix.polytechnique.fr

Abstract. There exist optimization problems with the target objective, which is to be optimized, and several extra objectives. The extra objectives may or may not be helpful in optimization process in terms of the number of objective evaluations necessary to reach an optimum of the target objective.
OneMax+ZeroMax is a previously proposed benchmark optimization problem where the target objective is OneMax and a single extra objective is ZeroMax, which is equal to the number of zero bits in the bit vector. This is an example of a problem where extra objectives are not good, and objective selection methods should ignore the extra objectives.
The EA+RL method is a method which selects objectives to be optimized by evolutionary algorithms (EA) using reinforcement learning (RL). Previously it was shown that it runs in $\Theta(N \log N)$ on OneMax+ZeroMax when configured to use the randomized local search algorithm and the Q-learning algorithm with the greedy exploration strategy.
We present the runtime analysis for the case when the $(1 + 1)$-EA algorithm is used. It is shown that the expected running time is at most $3.12eN \log N$.

1 Introduction

Single-objective optimization can often benefit from multiple objectives [6, 7, 9]. Different approaches are known from the literature. Some researchers introduce additional objectives to escape from the plateaus [1]. Decomposition of the primary objective into several objectives also helps in many problems [5, 7]. Additional objectives may also arise from the problem structure [8].

1.1 Approaches for Extra Objectives

Different approaches may be applied to a problem with the "original" objective, which can be called the *target* objective, and some extra objectives. The

multi-objectivization approach is to optimize *all* extra objectives at once using a multi-objective optimization algorithm [5, 7]. The *helper-objective* approach is to optimize simultaneously the target objective and some (not necessarily all, in some cases, only one is preferrable) extra objectives, switching between them from time to time [6].

The approaches above are designed in the assumption that the extra objectives are crafted to help optimizing the target objective. However, this is not always true, especially when their properties are unknown. In fact, the extra objectives may support or obstruct the process of optimizing the target objective. The EA+RL method [3] was developed to cope with such situations. The idea of this method is to use a single-objective optimization algorithm and switch between the objectives (which include the target one and the extra ones). To find the most suitable objective for the optimization, reinforcement learning algorithms are used [10].

1.2 Reinforcement Learning and EA+RL

Reinforcement learning [10] uses the concepts of *state*, *action* and *reward*. A reinforcement learning algorithm is often thought to control an *agent* which interacts with a certain *environment*. The agent receives the current state from the environment as input. It should return an action to apply on the environment. For that action, it receives a reward. The aim of the reinforcement learning algorithm is to maximize the total reward by choosing appropriate actions in different states. The total reward can be treated as a sum of all rewards received by the algorithm, or as a *discounted* reward, when a reward for i-th step from the end is taken with a weight of γ^i, where $0 < \gamma < 1$.

In the EA+RL method, actions are objectives to choose, while states and rewards are defined depending on the problem. A good choice of the reward can be the value of the target objective after the selection of an objective minus the value of the target objective before it. The sum of all rewards during the optimization is equal to the difference between the final value of the target objective and its initial value, so optimization of the reward leads to optimization of the target objective.

2 Analyzed Problem and Algorithm

OneMax is a well-known optimization problem widely used in theoretical research on evolutionary computation. It can be defined as "maximize the number of one-bits in a bit vector of length N". It is known that simple evolutionary algorithms, such as randomized local search (RLS) or $(1+1)$ evolutionary algorithm ($(1+1)$-EA), solve this problem in $\Theta(N \log N)$ function evaluations [11].

We define ZeroMax, a counterpart of OneMax, as follows: the number of zero-bits in a bit vector needs to be maximized. Clearly, the maximum point of ZeroMax is the same as the minimum point of OneMax, and vice versa. Moreover, any change that increases the OneMax fitness decreased the ZeroMax fitness at the same time.

In paper [2], OneMax+ZeroMax was defined as an optimization problem with extra objectives where OneMax is the *target* objective and ZeroMax is the extra objective. Clearly, for this problem every objective selection algorithm should eventually manage to ignore the offensive extra objective. It was shown that the EA+RL method [3], the objective selection method based on reinforcement learning, indeed learns to ignore the ZeroMax objective when randomized local search (RLS) is used as an optimizer and finds the OneMax optimum in $\Theta(N \log N)$, more precisely, at most twice slower than RLS itself does when optimizing OneMax. The proof in [2] was done by analysing the Markov chain which modeled the optimization process. Since this Markov chain possessed a simple linear structure, the proof was relatively easy. However, when the optimizer is changed to $(1+1)$-evolutionary algorithm, which is able to flip more than one bit at a time, using Markov chains becomes insanely complicated.

In this paper, we consider the $(1+1)$ evolutionary algorithm with the fixed probability of flipping a bit $p = 1/N$ as a single-objective optimization algorithm. To select which objective to optimize at each iteration, we use the EA+RL method [3], which internally uses a reinforcement learning (RL) algorithm [10] to do this. The *actions* of the RL algorithm are the possible objectives, so the set of possible actions for the considered problem is $\{\text{OneMax}, \text{ZeroMax}\}$. The choices for the EA+RL method are fixed in this paper to the following values:

- RL algorithm: the Q-learning algorithm with greedy exploration strategy [10];
  - the learning rate: an arbitraty $\alpha \in (0; 1)$;
  - the discount factor: an arbitrary $\gamma \in [0; 1]$;
- RL state: the value of the OneMax fitness;
- RL reward for the action: the difference of the OneMax fitness values after the action and before the action.

The pseudocode for the $(1+1)$-EA controlled by the EA+RL method with the parameters listed above is shown on Fig. 1.

3 Learning Lemma

The Q-learning algorithm stores estimations of action rewards as a $Q(s, a)$ matrix, where $Q(s, a)$ is the expected reward for applying an action a in a RL state s. When randomized local search is used, it was shown in [2] that EA+RL learns to ignore the offensive ZeroMax objective in the following way: once it leaves each RL state for the first time, it maintains $Q(s, 1) > Q(s, 0)$, which makes it select OneMax each time it enters the same state the next time. The same idea is true in the current situation, but in a more complicated manner.

Lemma 1 (Learning lemma). *If the algorithm has not reached the terminal state (where $s = N$) and $\gamma < \frac{1}{N-1}$ then for every non-terminal state s it is true that:*

$$Q(s, 0) \leq 0 \leq Q(s, 1) < N - 1 - s.$$

Additionally, if the algorithm has never left a state s, it is true that $Q(s, 0) = Q(s, 1) = 0$. Otherwise, $Q(s, 0) < Q(s, 1)$.

1: $X \leftarrow$ current individual, vector of N zeros
2: $Q \leftarrow$ transition quality matrix, $N \times 2$, filled with zeros
3: $f_1 \leftarrow$ OneMax, the target objective
4: $f_0 \leftarrow$ ZeroMax, the extra objective
5: Mutate$(X) \leftarrow$ mutation operator: inverts each bit with $p = 1/N$
6: **while** $f_1(X) < N$ **do**
7: $\quad s \leftarrow f_1(X)$
8: $\quad Y \leftarrow$ Mutate(X)
9: $\quad f, i$: chosen fitness function and its index
10: $\quad$ **if** $Q(s, 0) > Q(s, 1)$ **then**
11: $\quad\quad i \leftarrow 0$
12: $\quad$ **else if** $Q(s, 0) < Q(s, 1)$ **then**
13: $\quad\quad i \leftarrow 1$
14: $\quad$ **else**
15: $\quad\quad i \leftarrow$ Random$(0, 1)$
16: $\quad$ **end if**
17: $\quad f \leftarrow f_i$
18: $\quad$ **if** $f(Y) \geq f(X)$ **then**
19: $\quad\quad X \leftarrow Y$
20: $\quad$ **end if**
21: $\quad s' \leftarrow f_1(X)$
22: $\quad r \leftarrow s' - s$
23: $\quad Q(s, i) \leftarrow (1 - \alpha)Q(s, i) + \alpha(r + \gamma \cdot \max_j Q(s', j))$
24: **end while**

Fig. 1. $(1 + 1)$-EA controlled by EA+RL using the greedy Q-learning algorithm

Proof. We use mathematical induction. The base is obvious: in the very beginning, all $Q(i, j)$ are zeros and the algorithm has never left any state, so the lemma statement is true. Assume that the lemma statement was true for all previous algorithm iterations. The current iteration can have the following forms:

1. The algorithm has never left the current state and remains there.
2. The algorithm has left the current state for the first time.
3. The algorithm has left the current state before.

Below we denote the state before the current iteration as s, the state after the current iteration as s', the Q-values before the current iteration as $Q(i, j)$ and the Q-values after the current iteration as $Q'(i, j)$.

Case 1. By induction hypothesis, $Q(s, 0) = Q(s, 1) = 0$. If the algorithm remains at the state s, the reward is zero, so all the components of the expression at line 23 are zeros. Whatever objective i was selected, $Q'(s, i)$ is set to zero, and $Q'(s, 1 - i)$ remains zero as well, so the induction hypothesis is proven for the current iteration.

Case 2. By induction hypothesis, $Q(s, 0) = Q(s, 1) = 0$. This means that the objective is OneMax with the probability of 0.5 and ZeroMax with the prob-

ability of 0.5. As $s' \neq s$, the change was accepted by the selected objective, so for OneMax ($i = 1$) $s' \geq s + 1$, while for ZeroMax ($i = 0$) $s' \leq s - 1$.

By induction hypothesis, $Q(s', 0) \leq 0 \leq Q(s', 1) < N - s' - 1$. This means that for the chosen objective i the following will be true:

$$Q'(s, i) = (1 - \alpha)Q(s, i) + \alpha \left(s' - s + \gamma \max_j Q(s', j) \right) = \alpha(s' - s + \gamma Q(s', 1)).$$

The upper bound on $s' - s + \gamma Q(s', 1)$, provided that $s' < N$, is:

$$\begin{aligned} s' - s + \gamma Q(s', 1) &< s' - s + \frac{N - s' - 1}{N - 1} = s' \left(1 - \frac{1}{N - 1} \right) + 1 - s \\ &= s' \frac{N - 2}{N - 1} + 1 - s \leq \frac{(N - 1)(N - 2)}{N - 1} + 1 - s = N - s - 1. \end{aligned}$$

It follows that $Q'(s, i) < N - s - 1$ as well. The lower bound is $Q'(s, i) \geq \alpha(s' - s)$. For $i = 1$ these two bounds immediately yield that $0 < Q'(s, 1) < N - s - 1$. For $i = 0$ we should additionally use the fact that $s' \leq s - 1$, which brings:

$$Q'(s, 0) < s' \frac{N - 2}{N - 1} + 1 - s \leq s' + 1 - s \leq 0.$$

To sum up, after an iteration which leaves a state s for the first time it will be that either $0 < Q'(s, 1) < N - s - 1$ and $Q'(s, 0) = 0$ or $Q'(s, 0) < 0$ and $Q'(s, 1) = 0$. This proves the induction hypothesis for the current iteration in the considered case.

Case 3. By induction hypothesis, $Q(s, 0) < Q(s, 1)$, so the OneMax objective is selected. As a result, $s' \geq s$. Using the upper bound on $s' - s + \gamma Q(s', 1)$ proven in the previous case (which still holds under assumptions of the current case), the fact that it is non-negative and the induction assumption that $Q(s', 0) \leq Q(s', 1)$, we get the bounds on $Q'(s, 1)$:

$$\begin{aligned} Q'(s, 1) &= (1 - \alpha)Q(s, 1) + \alpha \left(s' - s + \gamma Q(s', 1) \right) \\ &< (1 - \alpha)(N - s - 1) + \alpha(N - s - 1) = N - s - 1. \\ Q'(s, 1) &\geq (1 - \alpha)Q(s, 1). \end{aligned}$$

In any case, $Q'(s, 1) < N - s - 1$. If $Q(s, 1) > 0$, then $Q'(s, 1) > 0$. If $Q(s, 1) = 0$, then $Q'(s, 1) \geq 0$. This proves the induction hypothesis for the current iteration in the considered case.

In all three possible cases the induction hypothesis is proven, which completes the proof. □

This lemma lets us describe each RL state in any moment of time either as *learned* or *unlearned*. In the learned state the algorithm always selects the correct objective, OneMax. In the unlearned state, it selects either OneMax or ZeroMax with equal probabilities. Each unlearned state becomes learned when the algorithm leaves this state and enters another one. All these considerations are true when $\gamma < 1/(N - 1)$, so we consider it to be so in the rest of the paper until explicitly noted.

4 Transition Probabilities

What is the exact probability that the independent bit-flip mutation (with a probability of flipping each bit equal to $1/N$) constructs a bit string with j one-bits from a bit string with i one-bits? Consider the situation where $i < j$: this means that $j-i+k$ zeros and k ones are flipped. The exact expressions for these probabilities are given below.

$$P^{i,j} = \begin{cases} \sum_{k=0}^{\min(N-j,i)} \binom{N-i}{j-i+k}\binom{i}{k} \left(\frac{1}{N}\right)^{j-i+2k} \left(1-\frac{1}{N}\right)^{N-(j-i+2k)} & \text{if } i < j, \\ \sum_{k=0}^{\min(N-i,j)} \binom{i}{i-j+k}\binom{N-i}{k} \left(\frac{1}{N}\right)^{i-j+2k} \left(1-\frac{1}{N}\right)^{N-(i-j+2k)} & \text{if } i > j, \\ \sum_{k=0}^{\min(N-i,i)} \binom{N-i}{k}\binom{i}{k} \left(\frac{1}{N}\right)^{2k} \left(1-\frac{1}{N}\right)^{N-2k} & \text{if } i = j. \end{cases} \quad (1)$$

In an unlearned state, OneMax or ZeroMax is chosen with the probability of 0.5. Together with the probabilities given above, transition probability $P_U^{i,j}$ from an unlearned state i to a state j is $\frac{1}{2}P^{i,j}$ if $i \neq j$ and $1 - \frac{1}{2}\sum_{k\neq i} P^{i,k} = \frac{1}{2}\left(1 + P^{i,i}\right)$ if $i = j$.

In a learned state, OneMax is always chosen, so the transition probability $P_L^{i,j}$ from a learned state i to a state j is $P^{i,j}$ if $i < j$, $1 - \sum_{k=i+1}^{j} P^{i,j}$ if i = j and zero otherwise.

4.1 Lower and Upper Bound on $P^{i,j}$

The expressions for $P^{i,j}$ are rather complex. The following theorem gives a lower and an upper bound on $P^{i,j}$.

Theorem 1. *Assume that $i \neq j$. Let $S^{i,j}$ be the following:*

$$S^{i,j} = \begin{cases} \binom{N-i}{j-i}\left(\frac{1}{N}\right)^{j-i}\left(1-\frac{1}{N}\right)^{N-(j-i)} & \text{if } i < j, \\ \binom{i}{i-j}\left(\frac{1}{N}\right)^{i-j}\left(1-\frac{1}{N}\right)^{N-(i-j)} & \text{if } i > j. \end{cases} \quad (2)$$

Then $S^{i,j} \leq P^{i,j} \leq \frac{8}{7}S^{i,j}$.

Proof. The lower bounds are proven easily, since $S^{i,j}$ are the addends for $k = 0$ in (1), and all these addends are positive.

We denote as $S_k^{i,j}$ the k-th addend of the sum in (1) corresponding to $P^{i,j}$. Specifically, $S^{i,j} = S_0^{i,j}$. Consider the case of $i < j$. The ratio of the k-th addend to the $(k+1)$-th addend is:

$$\frac{S_k^{i,j}}{S_{k+1}^{i,j}} = \frac{\binom{N-i}{j-i+k}\binom{i}{k}\left(\frac{1}{N}\right)^{j-i+2k}\left(1-\frac{1}{N}\right)^{N-(j-i+2k)}}{\binom{N-i}{j-i+k+1}\binom{i}{k+1}\left(\frac{1}{N}\right)^{j-i+2k+2}\left(1-\frac{1}{N}\right)^{N-(j-i+2k)-2}} =$$

$$= \frac{(j-i+k+1)(k+1)}{(N-j-k)(i-k)}N^2\left(1-\frac{1}{N}\right)^2 = \frac{(j-i+k+1)(k+1)}{(N-j-k)(i-k)}(N-1)^2.$$

When i and j are fixed, this ratio grows as k grows, so

$$\frac{S_k^{i,j}}{S_{k+1}^{i,j}} \geq \frac{S_0^{i,j}}{S_1^{i,j}} = \frac{j-i+1}{(N-j)i}(N-1)^2.$$

When i is fixed, this ratio grows as j grows, so we replace j with its minimum possible value $i+1$ and then minimize the result with $i = \frac{N-1}{2}$:

$$\frac{S_k^{i,j}}{S_{k+1}^{i,j}} \geq \frac{2(N-1)^2}{(N-i-1)i} \geq \frac{2(N-1)^2}{\frac{N-1}{2}\frac{N-1}{2}} = \frac{8(N-1)^2}{(N-1)^2} = 8.$$

This means that $P^{i,j}$ can be bounded by a sum of geometric progression:

$$P^{i,j} = \sum_{k=0}^{\min(N-j,i)} S_k^{i,j} \leq \sum_{k=0}^{\min(N-j,i)} \left(\frac{1}{8}\right)^k S_0^{i,j} \leq \sum_{k=0}^{\infty} \left(\frac{1}{8}\right)^k S_0^{i,j} = \frac{8}{7} S_0^{i,j} = \frac{8}{7} S^{i,j}.$$

The case of $i > j$ is proven in the similar way. □

In the rest of the paper, we denote the $\frac{8}{7}$ constant from this theorem by R.

4.2 Lower and Upper Bounds on Partial Sums of $P^{i,j}$

Theorem 2. *If* $V_i = \left(\frac{N-1}{N}\right)^{N-i} - \left(\frac{N-1}{N}\right)^N$, *then* $V_i \leq \sum_{j=0}^{i-1} P^{i,j} \leq RV_i$.

Proof. Considering the definition of $S^{i,j}$ from Theorem 1, we get that:

$$\begin{aligned}
\sum_{j=0}^{i-1} S^{i,j} &= \sum_{j=0}^{i-1} \binom{i}{i-j} \left(\frac{1}{N}\right)^{i-j} \left(1-\frac{1}{N}\right)^{N-(i-j)} \\
&= \left(1-\frac{1}{N}\right)^N \sum_{j=0}^{i-1} \binom{i}{i-j} \left(\frac{1}{N-1}\right)^{i-j} \\
&= \left(1-\frac{1}{N}\right)^N \left(\left(\frac{N}{N-1}\right)^i - 1\right) = V_i.
\end{aligned}$$

As $P^{i,j} \geq S^{i,j}$, it follows that $\sum_{j=0}^{i-1} P^{i,j} \geq \sum_{j=0}^{i-1} S^{i,j} = V_i$. Similarly, as $P^{i,j} \leq RS^{i,j}$, it follows that $\sum_{j=0}^{i-1} P^{i,j} \leq R\sum_{j=0}^{i-1} S^{i,j} = RV_i$. □

Theorem 3. *If* $W_i = \left(\frac{N-1}{N}\right)^i - \left(\frac{N-1}{N}\right)^N$, *then* $W_i \leq \sum_{j=i+1}^{N} P^{i,j} \leq RW_i$.

4.3 Lower and Upper Bounds on Other Expressions

Theorem 4. *If* $Y_i = \frac{i}{N-1}\left(\frac{N-1}{N}\right)^{N-i+1}$, *then* $Y_i \leq \sum_{j=0}^{i-1} P^{i,j}(i-j) \leq RY_i$.

Proof. Consider $\sum_{j=0}^{i-1} S^{i,j}(i-j)$:

$$\begin{aligned}
\sum_{j=0}^{i-1} S^{i,j}(i-j) &= \sum_{j=0}^{i-1} \binom{i}{i-j} \left(\frac{1}{N}\right)^{i-j} \left(1-\frac{1}{N}\right)^{N-(i-j)} (i-j) \\
&= \left(1-\frac{1}{N}\right)^{N} \sum_{j=0}^{i-1} \binom{i}{j} \frac{i-j}{(N-1)^{i-j}} \\
&= \left(1-\frac{1}{N}\right)^{N} (1-N) \sum_{j=0}^{i-1} \binom{i}{j} \left(\frac{1}{(N-1)^{i-j}}\right)_N' \\
&= \left(1-\frac{1}{N}\right)^{N} (1-N) \left(\sum_{j=0}^{i-1} \binom{i}{j} \frac{1}{(N-1)^{i-j}}\right)_N' \\
&= \left(1-\frac{1}{N}\right)^{N} (1-N) \left(\left(\frac{N}{N-1}\right)^{i} - 1)\right)_N' \\
&= \left(1-\frac{1}{N}\right)^{N} \frac{i}{N-1} \left(\frac{N}{N-1}\right)^{i-1} = \frac{i}{N-1} \left(\frac{N-1}{N}\right)^{N-i+1} = Y_i.
\end{aligned}$$

Similarly to Theorem 2, we prove the bounds on the required sum. □

Theorem 5. *If* $Z_i = \frac{N-i}{N-1}\left(\frac{N-1}{N}\right)^{i+1}$, *then* $Z_i \le \sum_{j=i+1}^{N} P^{i,j}(j-i) \le RZ_i$.

5 Drift analysis

We analyse the running time of the algorithm using the additive drift theorem [4]. To do that, we construct the following potential function:

$$\Phi(i,l) = \sum_{t=i}^{N-1} \frac{N}{N-t} + CN \sum_{t=0}^{N-1} \frac{1-l(t)}{N-t},$$

where i is the current state (equal to the number of one-bits), $l(t)$, the *learn indicator*, is equal to one if the state t is a learned state and to zero otherwise, and C is a constant.

Such function rewards the algorithm not only for getting closer to the optimum, but for learning a state as well. Note that each time $\Phi(i,l) = 0$, the algorithm is at the optimum, however, the opposite is not true: the optimum can be reached, but not all states become learned. This does not hurt anything: the additive drift theorem gives an upper bound on the number of iterations until the condition that the optimum is reached *and* all the states are learned, which, in turn, is an upper bound on the actual running time of the algorithm.

We can treat $\Phi(i,l)$ as a sum of two functions, $\Phi_1(i) = \sum_{t=i}^{N-1} \frac{N}{N-t}$ and $\Phi_2(l) = CN \sum_{t=0}^{N-1} \frac{1-l(t)}{N-t}$. As Φ_1 is upwards convex, from Jensen's inequality it follows that $\Phi_1(i) - E(\Phi_1(i')) \ge \Phi_1(i) - \Phi_1(E(i'))$, if Φ_1 is extended to non-integer arguments by linear interpolation.

5.1 Drift from a learned state

In a learned state the situation resembles how (1+1)-EA works on OneMax [11]. In this case, the learn indicator l does not change, so drift of Φ_2 is zero. The lower bound on $E(i')$ is (using Theorem 5):

$$E(i') = i + \sum_{j=0}^{N}(j-i)P_L^{i,j} = i + \sum_{j=i+1}^{N}(j-i)P^{i,j} \geq i + \frac{N-i}{N-1}\left(\frac{N-1}{N}\right)^{i+1}.$$

The lower bound on $E(i') - i$ is at most one. The drift of Φ_1 (and thus Φ) is:

$$\Phi_1(i) - E(\Phi_1(i')) \geq \Phi_1(i) - \Phi_1(E(i')) \geq \frac{N}{N-i}\frac{N-i}{N-1}\left(\frac{N-1}{N}\right)^{i+1} \geq e^{-1}.$$

5.2 Drift from an unlearned state

In an unlearned state, the expected drift of $\Phi_2(l)$ can be estimated as the reward for learning the current state i multiplied by the probability the algorithm leaves the state i (using Theorems 2 and 3). $\Phi_2(l) - E(\Phi_2(l))$ is at least:

$$C\frac{N}{N-i}\sum_{j\neq i}\frac{P^{i,j}}{2} \geq \frac{C}{2}\frac{N}{N-i}\left(\left(\frac{N-1}{N}\right)^{N-i} + \left(\frac{N-1}{N}\right)^{i} - 2\left(\frac{N-1}{N}\right)^{N}\right).$$

The lower bound on $E(i')$ is (using Theorems 4 and 5):

$$\begin{aligned}
E(i') &= i + \sum_{j=0}^{N} P_U^{i,j}(j-i) = i + \frac{1}{2}\sum_{j\neq i} P^{i,j}(j-i) \\
&= i + \frac{1}{2}\left(\sum_{j=i+1}^{N} P^{i,j}(j-i) - \sum_{j=0}^{i-1} P^{i,j}(i-j)\right) \\
&\geq i + \frac{1}{2}\left(\frac{N-i}{N-1}\left(\frac{N-1}{N}\right)^{i+1} - R\frac{i}{N-1}\left(\frac{N-1}{N}\right)^{N-i+1}\right).
\end{aligned}$$

As $R \leq \frac{8}{7}$, the lower bound on $E(i') - i$ is at most $-\frac{4}{7}$, so the drift of Φ_1 is:

$$\begin{aligned}
\Phi_1(i) - E(\Phi_1(i')) &\geq \Phi_1(i) - \Phi_1(E(i')) \\
&\geq \frac{1}{2}\frac{N}{N-i+1}\left(\frac{N-i}{N-1}\left(\frac{N-1}{N}\right)^{i+1} - \frac{Ri}{N-1}\left(\frac{N-1}{N}\right)^{N-i+1}\right) \\
&= \frac{1}{2}\frac{N-i}{N-i+1}\left(\frac{N-1}{N}\right)^{i} - \frac{R}{2}\frac{i}{N-i+1}\left(\frac{N-1}{N}\right)^{N-i}.
\end{aligned}$$

The total value of Φ, namely, $D = \Phi(i,l) - E(\Phi(i',l'))$, is bounded from below by sum of drifts for Φ_1 and Φ_2:

$$D \geq \frac{C}{2}\frac{N}{N-i}\left(\left(\frac{N-1}{N}\right)^{N-i} + \left(\frac{N-1}{N}\right)^{i} - 2\left(\frac{N-1}{N}\right)^{N}\right)$$

$$
\begin{aligned}
&+\frac{1}{2}\frac{N}{N-i+1}\left(\frac{N-i}{N}\left(\frac{N-1}{N}\right)^{i}-\frac{Ri}{N}\left(\frac{N-1}{N}\right)^{N-i}\right)\\
&\geq\frac{C}{2}\frac{N}{N-i+1}\left(\left(\frac{N-1}{N}\right)^{N-i}+\left(\frac{N-1}{N}\right)^{i}-2\left(\frac{N-1}{N}\right)^{N}\right)\\
&+\frac{1}{2}\frac{N}{N-i+1}\left(\frac{N-i}{N}\left(\frac{N-1}{N}\right)^{i}-\frac{Ri}{N}\left(\frac{N-1}{N}\right)^{N-i}\right)\\
&=\frac{\left(\frac{N-1}{N}\right)^{N-i}(CN-Ri)+\left(\frac{N-1}{N}\right)^{i}(CN+N-i)-\left(\frac{N-1}{N}\right)^{N}(2CN)}{2(N-i+1)}.
\end{aligned}
$$

If $C \geq R$, then $CN - Ri$ is positive. As $\left(\frac{N-1}{N}\right)^{x}$ is convex downwards, we can use Jensen's inequality to simplify a part of the latter expression, which we call G:

$$
\begin{aligned}
G&=\left(\frac{N-1}{N}\right)^{N-i}(CN-Ri)+\left(\frac{N-1}{N}\right)^{i}(CN+N-i)\\
&=\frac{\left(\frac{N-1}{N}\right)^{N-i}\frac{CN-Ri}{(2C+1)N-(R+1)i}+\left(\frac{N-1}{N}\right)^{i}\frac{CN+N-i}{(2C+1)N-(R+1)i}}{((2C+1)N-(R+1)i)^{-1}}\\
&\geq((2C+1)N-(R+1)i)\left(\frac{N-1}{N}\right)^{\frac{(N-i)(CN-Ri)+i(CN+N-i)}{(2C+1)N-(R+1)i}}.
\end{aligned}
$$

To find a lower bound on G one needs to find an upper bound on the exponent in the expression above, assuming that $0 \leq i \leq N$. Recall that $R = \frac{8}{7}$, which makes the exponent be equal to $\frac{i^2-iN+7CN^2}{14CN+7N-15i}$. The derivative by i of this exponent have no roots in $[0; N]$ at least when $C \geq 1$. The value for $i = 0$ is $N\frac{C}{2C+1}$, and for $i = N$ it is $N\frac{7C}{14C-8}$, the second one is the biggest of two. We continue with the lower bound on D:

$$
\begin{aligned}
D&\geq\frac{((2C+1)N-(R+1)i)\left(\frac{N-1}{N}\right)^{N\frac{7C}{14C-8}}-\left(\frac{N-1}{N}\right)^{N}(2CN)}{2(N-i+1)}\\
&=\frac{\left(\left(1+2C\left(1-\left(\frac{N-1}{N}\right)^{N\frac{7C-8}{14C-8}}\right)\right)N-(R+1)i\right)}{2(N-i+1)}\left(\frac{N-1}{N}\right)^{N\frac{7C}{14C-8}}.
\end{aligned}
$$

If we choose C such that $1+2C\left(1-\left(\frac{N-1}{N}\right)^{N\frac{7C-8}{14C-8}}\right) > R+1$, then starting from some N the fraction will be greater than one. For these needs we approximate $\left(\frac{N-1}{N}\right)^{N}$ by e^{-1}. This problem can be reduced to finding a minimum C such that $1-\frac{4}{7C} > e^{\frac{7C-8}{8-14C}}$. This can be done by a binary search which yields $C = 2.115188060\ldots \approx 2.12$. Consequently, when C is at least this large, the drift from an unlearned state is at least $\left(\frac{N-1}{N}\right)^{N\frac{7C}{14C-8}} \approx \left(\frac{N-1}{N}\right)^{0.6845N} \geq e^{-1}$. Together with all previous analysis, this proves the following theorem:

Table 1. Experiment results. $C \approx 2.12$ is a constant which was proven.

N	Average FF calls		Average false	$2eN \log N$	$(1+C)eN \log N$	Ratio to
	$\gamma = 1/N$	$\gamma = 1$	queries, $\gamma = 1$			$\gamma = 1/N$
$1 \cdot 10^1$	$9.892 \cdot 10^1$	$9.648 \cdot 10^1$	$3.000 \cdot 10^{-2}$	$1.252 \cdot 10^2$	$1.950 \cdot 10^2$	1.97
$3 \cdot 10^1$	$4.673 \cdot 10^2$	$4.855 \cdot 10^2$	$1.900 \cdot 10^{-1}$	$5.547 \cdot 10^2$	$8.640 \cdot 10^2$	1.85
$1 \cdot 10^2$	$2.382 \cdot 10^3$	$2.389 \cdot 10^3$	$7.800 \cdot 10^{-1}$	$2.504 \cdot 10^3$	$3.900 \cdot 10^3$	1.64
$3 \cdot 10^2$	$9.340 \cdot 10^3$	$9.335 \cdot 10^3$	$1.690 \cdot 10^0$	$9.303 \cdot 10^3$	$1.449 \cdot 10^4$	1.55
$1 \cdot 10^3$	$3.910 \cdot 10^4$	$3.925 \cdot 10^4$	$8.280 \cdot 10^0$	$3.755 \cdot 10^4$	$5.849 \cdot 10^4$	1.50
$3 \cdot 10^3$	$1.389 \cdot 10^5$	$1.441 \cdot 10^5$	$2.414 \cdot 10^1$	$1.306 \cdot 10^5$	$2.034 \cdot 10^5$	1.46
$1 \cdot 10^4$	$5.585 \cdot 10^5$	$5.461 \cdot 10^5$	$7.443 \cdot 10^1$	$5.007 \cdot 10^5$	$7.799 \cdot 10^5$	1.40
$3 \cdot 10^4$	$1.882 \cdot 10^6$	$1.901 \cdot 10^6$	$2.330 \cdot 10^2$	$1.681 \cdot 10^6$	$2.619 \cdot 10^6$	1.39
$1 \cdot 10^5$	$7.225 \cdot 10^6$	$7.108 \cdot 10^6$	$7.780 \cdot 10^2$	$6.259 \cdot 10^6$	$9.749 \cdot 10^6$	1.35
$3 \cdot 10^5$	$2.376 \cdot 10^7$	$2.376 \cdot 10^7$	$2.325 \cdot 10^3$	$2.057 \cdot 10^7$	$3.204 \cdot 10^7$	1.35
$1 \cdot 10^6$	$8.726 \cdot 10^7$	$8.648 \cdot 10^7$	$7.632 \cdot 10^3$	$7.511 \cdot 10^7$	$1.170 \cdot 10^8$	1.34

Theorem 6. *The expected running time of the* $(1+1)$ *evolutionary algorithm controlled by greedy Q-learning on the* OneMax+ZeroMax *problem with a value of the discount factor* $\gamma < 1/(N-1)$ *is at most:*

$$(1+C)eN \log N \approx 3.12eN \log N.$$

6 Experimental evaluation

We conducted experimental evaluation on big problem sizes to see how precise our estimations are. Table 1 presents the results. For each N, there were 100 runs, and the numbers of fitness function calls were averaged. We tested two values of the discount factor γ, namely $1/N$ and 1. For the latter value, we additionally track the number of situations when Q values were tuned wrong and ZeroMax was chosen intentionally (the "Average false queries" column). Additionally, we track the value of $2eN \log N$, which was the common belief for the "right" expression on the algorithm's runtime.

First, it turned out that for large N the algorithm needs more than $2eN \log N$ iterations to find an optimum, which was surprising. Second, the runtimes for different values of γ seem to be the same. The false queries column provides an insight for this phenomenon: for $\gamma = 1$, the number of mismatches with the learning lemma seems to be $\Theta(N)$ with a constant approximately equal to $7.6 \cdot 10^{-3}$. We have no proof for this yet, but it seems that the probability of "learning the wrong way" is very small and in most cases the algorithm can later "self-heal" from wrong decisions. Finally, our estimation appears to be quite a good one, as our estimations are only 35% pessimistic for large N.

7 Conclusion

We presented a proof that the (1+1) evolutionary algorithm, when controlled by the EA+RL method which uses Q-learning with a greedy exploration strategy,

small values of the discount factor $\gamma < 1/(N-1)$, difference in target fitness function as a reward and reinforcement learning states determined by the target fitness function, solves the OneMax+ZeroMax problem in $O(N \log N)$ — more precisely, in at most $3.12eN \log N$ fitness function calls in expectation.

Experiments show that early thoughts on the actual expected running time, $2eN \log N$, are wrong for $N \geq 10^5$. The current upper bound seems to be only 35% worse than the "real life". What is more, the influence of big values of γ is shown to be negligible. We hope that these results will show the way for proving bounds on the running time for the EA+RL method on more complex problems.

This work was partially financially supported by the Government of Russian Federation, Grant 074-U01.

References

1. Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: On the Effects of Adding Objectives to Plateau Functions. Transactions on Evolutionary Computation 13(3), 591–603 (2009)
2. Buzdalov, M., Buzdalova, A., Shalyto, A.: A First Step towards the Runtime Analysis of Evolutionary Algorithm Adjusted with Reinforcement Learning. In: Proceedings of the International Conference on Machine Learning and Applications. vol. 1, pp. 203–208. IEEE Computer Society (2013)
3. Buzdalova, A., Buzdalov, M.: Increasing Efficiency of Evolutionary Algorithms by Choosing between Auxiliary Fitness Functions with Reinforcement Learning. In: Proceedings of the International Conference on Machine Learning and Applications. vol. 1, pp. 150–155 (2012)
4. Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. Advances in Applied Probability 14(3), 502–525 (1982)
5. Handl, J., Lovell, S.C., Knowles, J.D.: Multiobjectivization by Decomposition of Scalar Cost Functions. In: Parallel Problem Solving from Nature Parallel Problem Solving from Nature X, pp. 31–40. No. 5199 in Lecture Notes in Computer Science, Springer (2008)
6. Jensen, M.T.: Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimisation: Evolutionary Computation Combinatorial Optimization. Journal of Mathematical Modelling and Algorithms 3(4), 323–347 (2004)
7. Knowles, J.D., Watson, R.A., Corne, D.: Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization. pp. 269–283. Springer-Verlag (2001)
8. Lochtefeld, D.F., Ciarallo, F.W.: Helper-Objective Optimization Strategies for the Job-Shop Scheduling Problem. Applied Soft Computing 11(6), 4161–4174 (2011)
9. Neumann, F., Wegener, I.: Can Single-Objective Optimization Profit from Multiobjective Optimization? In: Multiobjective Problem Solving from Nature, pp. 115–130. Natural Computing Series, Springer Berlin Heidelberg (2008)
10. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA (1998)
11. Witt, C.: Optimizing Linear Functions with Randomized Search Heuristics – the Robustness of Mutation. In: Proceedings of the 29th Annual Symposium on Theoretical Aspects of Computer Science. pp. 420–431 (2012)