Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Stochastic runtime analysis of a Cross-Entropy algorithm for traveling salesman problems

Zijun Wu [a,*,1], Rolf H. Möhring [a,**,2], Jianhui Lai [b,**]

[a] Beijing Institute for Scientific and Engineering Computing (BISEC), Pingle Yuan 100, Beijing, PR China
[b] College of Metropolitan Transportation, Beijing University of Technology, Pingle Yuan 100, Beijing, PR China

A R T I C L E I N F O

Article history:
Received 21 December 2016
Received in revised form 11 October 2017
Accepted 13 October 2017
Available online xxxx
Communicated by C. Blum

Keywords:
Probabilistic analysis of algorithms
Stochastic runtime analysis of evolutionary algorithms
Cross Entropy algorithm
Max-Min Ant System
$(\mu+\lambda)$ EA

A B S T R A C T

This article analyzes the stochastic runtime of a Cross-Entropy algorithm mimicking an Max-Min Ant System with iteration-best reinforcement. It investigates the impact of magnitude of the sample size on the runtime to find optimal solutions for TSP instances.

For simple TSP instances that have a $\{1,n\}$-valued distance function and a unique optimal solution, we show that sample size $N \in \omega(\ln n)$ results in a stochastically polynomial runtime, and $N \in O(\ln n)$ results in a stochastically exponential runtime, where "stochastically" means with a probability of $1-n^{-\omega(1)}$, and n represents number of cities. In particular, for $N \in \omega(\ln n)$, we prove a stochastic runtime of $O(N \cdot n^6)$ with the vertex-based random solution generation, and a stochastic runtime of $O(N \cdot n^3 \ln n)$ with the edge-based random solution generation. These runtimes are very close to the best known expected runtime for variants of Max-Min Ant System with best-so-far reinforcement by choosing a small $N \in \omega(\ln n)$. They are obtained for the stronger notion of stochastic runtime, and analyze the runtime in most cases.

We also inspect more complex instances with n vertices positioned on an $m \times m$ grid. When the n vertices span a convex polygon, we obtain a stochastic runtime of $O(n^4 m^{5+\epsilon})$ with the vertex-based random solution generation, and a stochastic runtime of $O(n^3 m^{5+\epsilon})$ for the edge-based random solution generation. When there are $k \in O(1)$ many vertices inside a convex polygon spanned by the other $n-k$ vertices, we obtain a stochastic runtime of $O(n^4 m^{5+\epsilon} + n^{6k-1} m^{\epsilon})$ with the vertex-based random solution generation, and a stochastic runtime of $O(n^3 m^{5+\epsilon} + n^{3k} m^{\epsilon})$ with the edge-based random solution generation. These runtimes are better than the expected runtime for the so-called $(\mu+\lambda)$ EA reported in a recent article, and again obtained for the stronger notion of stochastic runtime.

1. Introduction

The Cross Entropy (CE) algorithm is a general-purpose evolutionary algorithm (EA) that has been applied successfully to many $\mathcal{NP}$-hard combinatorial optimization problems, see e.g. the book [1] for an overview. It was initially designed for

* Principal corresponding author.
** Corresponding authors.
E-mail addresses: zijunwu@bjut.edu.cn (Z. Wu), rolf.moehring@tu-berlin.de (R.H. Möhring).
[1] The author is also affiliated with the School of Applied Mathematics and Physics at Beijing University of Technology.
[2] The author is a professor emeritus of mathematics at Berlin University of Technology.

https://doi.org/10.1016/j.tcs.2017.10.012
0304-3975/© 2017 Elsevier B.V. All rights reserved.

rare event simulation by Rubinstein [2] in 1997, and thereafter formulated as an optimization tool for both continuous and discrete optimization (see [3]).

CE has much in common with the famous ant colony optimization (ACO, see [4]) and the estimation of distribution algorithms (EDAs, see [5]). They all belong to the so-called *model-based search* paradigm (MBS), see [6]. Instead of only manipulating solutions, which is very typical in traditional heuristics like Genetic Algorithms [7] and Local Search [8] and others, MBS algorithms attempt to optimize the solution reproducing mechanism. In each iteration, they produce new solutions by sampling from a probabilistic distribution on the search space. The distribution is often called a *model* in the literature (see e.g. [6] and [9]). This model evolves iteratively by incorporating information from some elite solutions occurring in the search history, so as to asymptotically model the spread of optimal solutions in the search space. See the recent Thesis [9] for more details on MBS algorithms and their mathematical properties.

An important issue for MBS algorithms is to determine a suitable size for the sampling in each iteration. A large sample size makes each iteration unwieldy, however a small sample size may mislead the underlying search due to the randomness in the sampling. Sample size reflects the *iterative complexity* (computational complexity in each iteration). Whether a large sample size is harmful depends on the required *optimization time* (i.e., the total number of iterations required to reach an optimal solution). This article aims to shed a light on this issue by theoretically analyzing the relation between sample size and optimization time for a CE *variant* that includes also some essential features of the famous Max-Min Ant System ($\mathcal{MMAS}$ [10]). To this end, a thorough runtime analysis is needed.

The theoretical runtime analysis of EAs has gained rapidly growing interest in recent years, see e.g. [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], and [25]. In the analysis, an oracle-based view of computation is adopted, i.e., the *runtime* of an algorithm is expressed as *the total number of solutions evaluated before reaching an optimal solution*. Since the presence of randomness, the runtime of an EA is often conveyed in expectation or with high probability. Due to the famous No Free Lunch Theorem [26], the analysis must be problem-specific. The first steps towards this type of analysis were made for the so-called $(1+1)$ EA [11] on some test problems that use *pseudo boolean functions* as cost functions, e.g., OneMax [15], LeadingOnes [27] and BinVar [11]. Recent research addresses problems of practical importance, such as the computing a minimum spanning trees (MST) [28], matroid optimization [29], traveling salesman problem [30], the shortest path problem [23], the maximum satisfiability problem [31] and the max-cut problem [32].

Runtime analysis generally considers two cases: *expected runtime analysis* and *stochastic runtime analysis*. Expected runtime is the average runtime of an algorithm on a particular problem, see, e.g., the runtime results of $(1+1)$ EA reported in [11]. Expected runtime reflects the oracle-based average performance of an algorithm. A mature technique for expected runtime analysis is the so-called drift analysis [12]. However, this technique requires that the algorithm has a finite expected runtime for the underlying problem. By [33], drift analysis is not applicable to the *traditional* CE [3].

An algorithm with a smaller expected runtime need not be more efficient, see [34] for details. In contrast, stochastic runtime provides a better understanding of the performance of a (randomized) EA. Stochastic runtime is a runtime result conveyed with an overwhelming probability guarantee (see, e.g., the classic runtime result of 1-ANT in [15]), where an overwhelming probability means a probability tending to 1 superpolynomially fast in the problem size. It therefore reflects the efficiency of an algorithm for most cases in the sense of uncertainty. This article is concerned with stochastic runtime analysis, aiming to figure out the relation between stochastic runtime and magnitude of the sample size.

Runtime analysis of CE algorithms has been initiated in [33], where Wu and Kolonko proved a pioneering stochastic runtime result for the traditional CE on the standard test problem LeadingOnes. As a continuation of the study of [33], Wu et al. [34] further investigated the stochastic runtime of the traditional CE on another test problem OneMax. The runtime results reported in [33] and [34] showed that sample size plays a crucial role in efficiently finding an optimal solution. In particular, Wu et al. [34] showed that if the problem size n is moderately adapted to the sample size N, then the stochastic runtime of the traditional CE on OneMax is $O(n^{1.5+\frac{4}{3}\epsilon})$ for arbitrarily small $\epsilon > 0$ and a constant smoothing parameter $\rho > 0$, which beats the best-known stochastic runtime $O(n^2)$ reported in [13] for the classic 1-ANT algorithm, although 1-ANT employs a much smaller sample size (i.e., sample size equals one). Moreover, by imposing upper and lower bounds on the sampling probabilities as was done in $\mathcal{MMAS}$ [10], Wu et al. [34] showed further that the stochastic runtime of the resulting CE can be significantly improved even in a very rugged search space.

The present article continues the stochastic runtime analysis of [34], but now in combinatorial optimization with a study of CE on the traveling salesman problem (TSP). We emphasize the impact of the magnitude of N on the stochastic runtime, put $\rho = 1$, and consider a CE variant resembling an $\mathcal{MMAS}$ with iteration-best reinforcement under two different random solution generation mechanisms, namely, a vertex-based random solution generation and an edge-based random solution generation.

Stochastic runtime for $\mathcal{MMAS}$ with iteration-best reinforcement on simple problems like OneMax has been studied in [20] and [25]. In particular, Neumann et al. [20] showed that to obtain a stochastically polynomial runtime for OneMax, $N/\rho \in \Omega(\ln n)$ is necessary. We shall not only extend this to TSP for the case of $\rho = 1$, but also prove that $N \in \omega(\ln n)$ is already sufficient to guarantee a stochastically polynomial runtime for simple TSP instances.

TSP is a famous $\mathcal{NP}$-complete combinatorial optimization problem. It concerns finding a shortest Hamiltonian cycle on a weighted complete graph. Existing algorithms exactly solving TSP generally have a prohibitive complexity. For instance, the Held–Karp algorithm [35] solves the problem with a complexity of $O(n^2 2^n)$. A well-known polynomial time approximation algorithm for metric TSP is the so-called Christofides algorithm [36], which finds a solution with a cost at most 3/2 times the cost of optimal solutions. As mentioned in [37], this is still the best known approximation algorithm for the general