Final IGR Report for EPSRC Grant GR/R52541/01
Average Computation Time of Evolutionary Algorithms for Combinatorial Optimisation Problems

Xin Yao
School of Computer Science, The University of Birmingham
Edgbaston, Birmingham B15 2TT. Email: x.yao@cs.bham.ac.uk

1 Background

Evolutionary algorithms (EAs in short) have been used widely in solving a number of combinatorial optimisation problems, e.g., VLSI cell placement, routing, job-shop scheduling, cutting stocks, time tabling, etc. Successes have been reported for many such applications. However, theoretical work on EAs has been lagging behind applications. Most of the theoretical work so far has been on the convergence analysis of EAs. Few theoretical results exist on the average computation time or time complexity of EAs for combinatorial optimisation problems. It is still unclear where EA's real power is, theoretically, in solving combinatorial optimisation problems.

Theoretical work on EAs, such as schema theorems, analysis of convergence and (local) convergence rates, fitness landscape analysis, dynamical system models, etc., has greatly increased our understanding of how and why EAs work. However, such studies contrast sharply to the analysis of (conventional) algorithms by most computer scientists, where the time complexity of an algorithm is a key measurement of the algorithm's performance for large problems. Unfortunately, none of the above theoretical work on EAs lends itself to such kind of results [1]. There is an urgent need to establish a theoretical framework for studying time complexity of EAs on different problems so that comparisons among different algorithms (either EAs or not) can be made and insights can be gained.

Early work on the analysis of time complexity of EAs was carried out using $(1 + 1)$ EAs only on some toy problems, e.g., the ONE-MAX problem [2]. Recently, there has been an increasing interest in the topic from researchers in both theoretical computer science and evolutionary computation. Some initial results have been obtained [3, 4]. However, there are two major weakness in the current study of EA's time complexity on different problems. Firstly, existing methods and techniques used in the analysis are very complicated and difficult to be applied to different EAs on different problems. For example, even the analysis of a very simple $(1 + 1)$ EA on the linear function is quite complicated [5]. Alternative and simpler methods and techniques for analysing EA's computation time are needed. Secondly, most existing studies are limited to $(1 + 1)$ EAs on simple artificial fitness functions. Population-based EAs have not been studied in any depth. Yet population is a distinct feature of EAs in practice and is often credited (as one of the factors) for EA's good performance in many successful applications. This project addresses these two issues.

In this project, a unified approach, based on drift analysis, to analyse EA's mean computation time for different problems is studied. Different from other problem-specific approaches, this approach is intuitive, easy to understand and straightforward to use in the analysis of EA's mean computation time. We have been able to simplify the proofs of previous results as well as deriving new results. Our achievements in this project have showed the advantage of using this approach in studying EA's mean computation time.

Armed with a set of techniques in drift analysis, we have been able to start cracking the hard nut of analysing population-based EAs, where the population size is great than 1. We have also been able to move away from simple and artificial pseudo-binary problems to classical combinatorial optimisation problems. The results we have obtained in this project have given us some very useful insights as to when and why a problem is hard for an EA.

2 Key Advances and Supporting Methodology

2.1 Exact Average Computation Time of EAs and Its Applications

Markov chains can be used to model most EAs. If the chain is homogeneous, then its expected first hitting time to the optimal set will satisfy a linear system. We can obtain an explicit solution to the linear system for some

simple EAs on certain problems. We have showed how this is done through two case studies, i.e., when the probability transition matrix of the Markov chain is a tridiagonal matrix or a lower triangular matrix. Some EAs and their accurate average computation time on certain problem were described in detail in our papers [6, 7] .

An important application of the exact computation time is the study of $(1+1)$ EAs with elitism selection strategy accepting only higher fitness individuals [7]. We have established *sufficient and necessary* conditions under which a problem can be solved in polynomial or exponential time by these EAs. These conditions lead to a natural classification of hard problems for EAs. Intuitively, such hard problems can be classified into two categories from a geometric point of view.

Wide Gap Problems: Starting from some state in a subspace (in which all states have the same fitness value), the transition probability for the EA moving to "upper" subspaces with a higher fitness is very small. The EA is trapped in the subspace, because there is a *wide gap* between the current subspace and upper subspaces with a higher fitness.

Long Path Problems: Starting from each subspace, the transition probability for the EA moving to upper subspaces with a higher fitness is reasonably large, but the number of subspaces (i.e., "length") is exponential in the problem size.

2.2 Bounds on the Average Computation Times of EAs

Exact computation time is often very hard to obtain in reality because the linear system mentioned in the previous section is too complex to solve in most cases. In such cases, we can derive the bounds on EA's computation time using drift analysis [7]. The main idea of our approach can be explained as follows.

First, we define a function $d(x)$ to measure the distance between a population and the optimal set (i.e., the set of optimal states). Then we study the drift conditions for estimating the bounds on the average computation time of EAs. The drift conditions for estimating the lower bound of the mean first hitting time $m(x)$ can be stated as: Denote $\mathbf{d} = [d(x)]_{x \in Y}$ and $\mathbf{m} = [m_(x)]_{x \in Y}$ where Y is the population set, if $\mathbf{d - Td} \leq \mathbf{1}$, then $\mathbf{m} \geq \mathbf{d}$, where $\mathbf{T}$ is the transition probability matrix. Similarly, the drift condition for the upper bound can be described as: if $\mathbf{d - Td} \geq \mathbf{1}$, then $\mathbf{m} \leq \mathbf{d}$.

Because the cardinality of state space S is often very large, it is very difficult to define a distance for every state. In other words, drift analysis as mentioned above may not be applicable for large and complex problems. For such cases, we have introduced a method of reducing the number of states in the state space. That is, the state space is decomposed into a number of subspaces. All states within a subspace will have the same distance to the optimal set.

2.3 Martingale Model of EAs and Drift Conditions

The martingale model is an alternative model for analysing EAs, which was used to study the convergence of non-elitism EAs. Although its use is not as common as the Markov chain model, we have found it to be a good model in analysing the time complexity of EAs by drift analysis. Since an EA can be modelled by a supermartingale [8], we have been able to improve our previous work [4] on drift conditions [8].

The first drift condition is a *sufficient and necessary* condition to determine whether an EA can solve a problem in polynomial time: Given a problem, the mean first hitting time $m(x)$ is polynomial in the problem size n for all population $x \in Y$ *if and only if* there exists a distance function $d(x)$, and

1. $d(x) \leq D_1$, where D_1 is polynomial in the problem size n,
2. the process $\{d(\xi_t), t = 0, 1, \cdots)\}$ is a supermartingale, and
3. the one-step mean drift satisfies: $E[d(\xi_t) - d(\xi_{t+1}) \mid \xi_0] \geq c_{low}$, where $c_{low} > 0$ is a constant.

The second drift condition is a *sufficient and necessary* condition to determine whether an EA solves a problem in exponential time: Given a problem, for some population $x \in Y$, the mean first hitting time $m(x)$ is exponential in the problem size n *if and only if* there exists a distance function $d(x)$, and

1. $d(x)$ satisfies: for some population $x \in Y : d(x) \geq D_2$, where D_2 is exponential in the problem size n,
2. the process $\{d(\xi_t), t = 0, 1, \cdots\}$ is a supermartingale,
3. the one-step mean drift satisfies: for any initial population ξ_0 with $d(\xi_0) > 0$, $E[d(\xi_t) - d(\xi_{t+1}) \mid \xi_0] \leq c_{up}$, where $c_{up} > 0$ is a constant.

Different from our previous results [4], these are sufficient and necessary conditions to determine EA's computation time. They are applicable to any EAs.

2.4 Fitness Landscape Classification Based on Time Complexity of EAs

The classification of fitness landscapes is an important topic in evolutionary computation. Although there are many attempts in characterising fitness landscapes, e.g. deception, multimodality, fitness distance correlation, epistasis variance, etc., none of them can characterise EA easy or hard problems very well [9].

Based on drift analysis, we have introduced a rigorous landscape classification using the mean computation time of EAs [8]. It differentiates EA hard problems from easy ones clearly. It is a direct conclusion from the sufficient and necessary drift conditions given above. All fitness landscapes can be classified into two basic convergence classes: *short-distance landscapes*, i.e., the polynomial time class, and *far-distance landscapes*, i.e., the exponential time class.

Short-Distance Landscapes: There exists a distance function, and

1. under the distance, all populations are not far (a short distance) away from the optimal set, and
2. the one-step expected drift towards the optimal set is large and always greater than a positive constant.

Far-Distance Landscapes: There exists a distance function, and

1. under this distance, some populations are very far (exponential in the problem size) from the optimal set, and
2. the one-step mean drift towards the optimal set is limited and always less than a positive constant.

2.5 Comparison Between EAs With and Without a Population

As mentioned briefly in Section 1, theoretical results on the mean computation time of population-based EAs are few [10]. However, the vast majority of applications of EAs use a population size that is greater than one. The use of population brings robustness and efficiency to EAs. It is important to compare $(1+1)$ and population-based EAs theoretically.

In spite of the common wisdom in the evolutionary computation community that a population ought to benefit evolutionary search, few theoretical results are available on the existence of such benefit and how much benefit there is if it exists.

We have compared the mean computation time of $(1+1)$ and $(N+N)$ EAs on certain problems [6], and proved that [6], in some cases,

- a population of size N $(N > 1)$ can turn an exponential time $(1+1)$ EA into a polynomial time $(N+N)$ EA;
- a population of size N $(N > 1)$ may make little difference to a polynomial time $(1+1)$ EA since the $(N+N)$ EA is also polynomial in time;
- a population of size N $(N > 1)$ could turn a polynomial time $(1+1)$ EA into an exponential time $(N+N)$ EA, especially when the selection pressure becomes too high.

We have also analysed the impact of population on the first hitting probability and showed that population increases the first hitting probabilities [6].

2.6 Comparison Between EAs With and Without Crossover

Crossover is used in many EAs as a primary search operator. It is an important feature of EAs and makes it rather different from other search algorithms. However, little theoretical analysis has been done in analysing the role of crossover in terms of EA's computation time. In this project, we have compared rigorously the mean computation time of EAs with and without crossover [7].

An interesting question that often arises in the evolutionary computation community is: Does an EA with a crossover always perform better than its competitor without the crossover? Of course, "always" is a strong word and the statement cannot hold in light of the no-free-lunch theorem [15]. However, we have shown an interesting case where the EA with crossover is no worse than the one without any crossover. The problem used in this study was a jump function. Two different forms of crossover were used. Both of them were shown to be helpful in reducing the mean first hitting time of the EA.

The first crossover we considered is:

Crossover 1: Let x and y be two individuals (coded by binary strings) in a population. Their offspring are produced by $x' = (x \text{ AND } y)$ and $y' = (x \text{ OR } y)$. If x' or y' is infeasible, then replace it by x or y.

We showed that for the EA using the above crossover, its average computation time for the jump function is polynomial in the problem size, but the EA without any crossover needs an exponential time [7].

The second crossover we considered is:

Crossover 2: Let x and y be two individuals in a population, then independently for all positions $i \in \{1, \cdots, n\}$, exchange s_{xi} and s_{yi} with probability 0.5 and obtain two offspring x' and y'. If x' or y' is infeasible, replace it by x or y.

We showed that the EA using Crossover 2 used less mean computation time than one without any crossover on the jump function, but they both have exponential time [7].

Our results [7] show that search operators such as crossover can cause dramatic changes to an EA's computation time, from one class to another.

2.7 Analysis of Different Selection Schemes

We have analysed three different selection schemes for $(1+1)$ EAs: elitist selection that accepts only the higher fitness individual, elitist selection that accepts higher or equal fitness individuals, and non-elitist selection [7].

The first EA was taken as the basis of our comparison, because we were able to obtain the explicit expression of the mean first hitting time for the EA [7]. We then compared the second and third EAs (using corresponding selection schemes as given above) against it.

We have shown that the second selection strategy is better than the first one [7]. More specifically, given any fitness function and subspace with the same fitness, if the initial individual is taken from the subspace, then the EA using the second selection strategy will arrive at a subspace with a higher fitness no slower than the EA using the first selection strategy. This is true for both the worst and average cases [7].

It is interesting to compare EAs using the elitist and non-elitist selection strategies. Intuitively a non-elitist selection strategy appears to be worse than an elitist one because it may lose a good (or even optimal) solution. However, for some functions in the class of wide-gap landscapes, a non-elitist selection strategy might be better, because it can "climb" out of the absorbing area of a local optimum [7].

2.8 Analysis of Maximum Matching Problem

Maximum matching is probably one of the hardest easy problems. It's easy because it is a P-class problem. It is hard because the polynomial algorithm is quite complicated. It seems to be an good test problem for analysing EA's performance on P-class combinatorial optimisation problems. In [11], we have obtained some of the first steps towards analysing the mean computation time of an population-based EA for the maximum matching problem.

The EA for the maximum matching problem uses the following crossover and mutation:

Crossover: Let M_1 and M_2 be two individuals in a population. Denote $Q_{M_1}(M_2)$ (and $Q_{M_2}(M_1)$) to be the set of edges in M_1 (and M_2) that are matchable to M_2 (and M_1). Then the crossover can be described as:

$$\begin{aligned} M_1' &= M_1 + Q_{M_2}(M_1), \\ M_2' &= M_2 + Q_{M_1}(M_2), \end{aligned}$$

where M_1' and M_2' are two offspring.

Mutation 1: For each individual M in a population, let $V_{exposed}$ be the set of exposed nodes with respect to M. Choose a node v uniformly at random from $V_{exposed}$ and denote $N_{neighbour}(v)$ to be its neighbours, but excluding nodes in M. A mutant is generated as follows:

- If there is one or more nodes in $N_{neighbour}(v)$ which is exposed, then choose an exposed node v' in $N_{neighbour}(v)$ at random and add edge (v, v') to M, i.e., $M' = M + (v, v')$.
- If none of the nodes in $N_{neighbour}(v)$ is exposed, then choose a node v' in $N_{neighbour}(v)$ at random. Let v'' be the node in M which is connected to v', generate a new matching by delete edge (v', v'') from M and adding edge (v, v') to M, i.e., $M' = M - (v', v'') + (v, v')$.

By using drift analysis, we have shown [11] that the above $(N+N)$ EA needs at least exponential average time to find the exact maximum matching for a family of bipartite graphs although the problem can be solved by a deterministic algorithm in polynomial time. However, the EA can find a near maximum matching for *any* graph in polynomial time [11]. Our study [11] confirms theoretically a couple of important empirical observations. First, the real power of EAs lies in their ability in finding a near, rather than an exact, optimal solution efficiently. Second, without problem specific knowledge, it is hard for EAs to find the exact optimal solution to even an "easy" problem where there exists a polynomial deterministic algorithm.

We have also considered another mutation [14]:

Mutation 2: For each individual M in a population, choose an edge e uniformly at random from E and generate a mutant as follows:

$$M' = \begin{cases} M + e, & \text{if } e \text{ is not in } M \text{ but matchable to } M, \\ M - e \text{ with probability } p > 0, & \text{if } e \text{ is in } M, \\ M, & \text{otherwise.} \end{cases}$$

In this case, the EA can still find a near maximum matching for any graph in polynomial time, although the mutation is less efficient than the first mutation [14], because it only utilises the definition of matching and but not the concept of exposed node.

3 Research Plan Review

We have made no changes to the original research plan and have achieved all original objectives except for one, i.e., objective 2(b), which was to analyse EA's time complexity on an NP-hard problem (the subset sum problem). We did not pursue this line of research because of the suggestion from one reviewer of our grant proposal. The reviewer suggested that "PI should give up this objective and concentrate on the other issues." We have taken the comment on board, thought it over carefully, and decided that the reviewer is correct in his/her suggestion. As a result, we have been able to go much deeper into other issues, e.g., the necessary and sufficient conditions that we have established, the impact of populations, etc. [7, 8, 11].

4 Research Impact and Benefits to Society

Although our publications from this project only started appearing last year (2002). They have already attracted attentions from the international community. Our papers have been cited by others [12, 13] already. We have received invitations to give seminars on this topic in both UK and overseas. We are also organising a special session on this topic at the 2003 Congress on Evolutionary Computation in Canberra, Australia, on 8-12 December 2003.

Although our work has no direct benefits to our society, it has made significant contributions to the theory of evolutionary computation and benefits researchers in evolutionary computation as well as theoretical computer science. It is also beneficial to practitioners because the theoretical results have revealed some insights into when and why EAs perform well/poorly in some cases. A deeper understanding of the theoretical issues will eventually benefits the practical applications of EAs.

5 Explanation of Expenditure

The expenditure is the same as that planned in the original grant proposal. The entire grant money was used to support the RA, Dr. Jun He. There were no equipment, travel, or other budget items.

6 Dissemination Activities

We made a decision in the early stage of this project that we would target our results at the top international journals, rather than conferences, for two reasons. First, good journal papers get more exposure and impact than conference papers in general. They also enable us to give more detailed proofs than short conference papers. Second, we did not have a budget for conference travel.

So far we have two papers published or in press in *IEEE Transactions on Evolutionary Computation* and *Artificial Intelligence*, and two more under review by *IEEE Transactions on Evolutionary Computation*. One paper was published in the Proceedings of the 2002 UK Workshop on Computational Intelligence. The workshop was held in Birmingham and did not incur any travel costs.

We are now organising a special session on *Computational Complexity of Evolutionary Algorithms* at the *2003 IEEE Congress on Evolutionary Computation* on 8-12 December 2003, Canberra, Australia, one of leading international conferences in evolutionary computation. The special session will both report new results and discuss new ideas and future research directions.

We have also given seminars at Wuhan University, P. R. China (December 2002), and will give one at the University of Essex (April 2003). We have been invited to collaborate with Spanish researchers from the University of the Basque Country to explore whether our analytical methods could be applied to estimation of distribution algorithms.

7 Further Research

The analysis of EA's time complexity on combinatorial optimisation problems is a very challenging task, partly because of its stochastic nature and population-based. Our research so far is only the first step towards a better understanding of this issue. It is also the first step towards studying different analytical methods and techniques that are useful in the analysis of EAs. However, there are still many open questions that need to be answered. For example, Can an EA find an approximate solution to a hard combinatorial optimisation problem more efficiently than conventional deterministic algorithms? What is the impact of different search operators on EA's mean computation time on different problems? Given two EAs (with different performance) on the same problem, how can we tease out exactly what makes one EA more efficient than the other? How such information could be used to derive new results for other EAs on different problems? For EA-hard problems, i.e., such as those we identified as wide-gap and long-path problems, what kind of algorithms will be most efficient? ...

References

[1] H.-G. Beyer, H.-P. Schwefel, and I. Wegener. How to analyse evolutionary algorithms. *Theoretical Computer Science*, 287(1):101–130, 2002.

[2] G. Rudolph. Finite Markov chain results in evolutionary computation: A tour d'Horizon. *Fundamenta Informaticae*, 35(1-4):67–89, 1998.

[3] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In R. Sarker, M. Mohammadian, and X. Yao, editors, *Evolutionary Optimisation*, chapter 14, pages 349–370. Kluwer Academic Publisher, Boston, 2002.

[4] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, 2001.

[5] S. Droste, T. Jansen, and I. Wegener. A rigorous complexity analysis of the (1+1) evolutionary algorithm for linear functions with Boolean inputs. *Evolutionary Computation*, 6(2):185–196, 1998.

[6] J. He and X. Yao. From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):495–511, 2002.

[7] J. He and X. Yao. Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 2003. (in press).

[8] J. He and X. Yao. A rigorous analysis for convergence quality classes of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*. (submitted).

[9] B. Naudts and L. Kallel. A comparison of predictive measure of problem difficulty in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, 4(1):1–15, 2000.

[10] S. Droste, T. Jansen, and I. Wegener. On the analysis of the $(1 + 1)$ evolutionary algorithms. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[11] J. He and X. Yao. Time complexity of maximum matching by an $(n + n)$ evolutionary algorithm. *IEEE Transactions on Evolutionary Computation*. (submitted).

[12] O. Giel and I. Wegener. Evolutionary algorithms and the maximum matching problem. In H. Alt and M. Habib, editors, *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2003), LNCS 2607*, pages 415–426, Berlin, Germany, February 2003. Springer.

[13] I. Wegener and C. Witt. On the optimization of monotone polynomials by simple randomized search heuristics. *Combinatorics, Probability and Computing*, 2003. (to appear).

[14] J. He and X. Yao. Maximum cardinality matching by evolutionary algorithms. In J. A. Bullinaria, editor, *Proc. of the 2002 U.K. Workshop on Computational Intelligence (UKCI'02)*, pages 53–60, 2002. (ISBN 0704423685)

[15] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.