The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

Study of computational performance of Genetic Algorithm for 3-satisfiability problem

QingLian Ma, Yu-an Zhang, Makoto Sakamoto, Hiroshi Furutani
Faculty of Engineering, University of Miyazaki
Miyazaki City 889-2192 Japan

Abstract

In order to improve the computing performance of Genetic Algorithms (GAs), it is important to study the effects of crossover and mutation. In this study, we examine the relations of first hitting time T of optimum solution in population, success probability S, mutation rate p_m and crossover rate p_c by GA experiments, which are carried out on the 3-satisfiability (3-SAT) problem. Here, S is defined as that there is at least one optimum solution in a population at the stationary distribution. We found that, when mutation rate is small, the effects of crossover on T and S are large. S with crossover is larger than that without crossover, and T is smaller than that without crossover. We also observed the relation between T and a/S when mutation rate becomes large, and found that $T = a/S$. When $p_m = 0.02$, $T \approx 1/S$.

1 Introduction

The satisfiability (SAT) problem is a core of a large family of computationally intractable NP-complete problems [1] with relevant practical applications such as automated reasoning, computer-aided design, machine vision, database, robotics, computer network design and so on. Methods to solve the SAT problem play an important role in the fields of efficient computing systems [2].

During the last two decades, several improved algorithms have been developed, and important progress has been achieved. These algorithms have considerably enlarged our capacity of solving large SAT instances. It can be divided into two main classes: complete and incomplete algorithms. The first category includes approaches based on the Davis-Putnam-algorithm [3]. Most incomplete algorithms include approaches based on local search [4] and evolutionary algorithms (EAs) [5, 6]. De Jong and Spears (1989) proposed a classical GA for SAT problem, and observed that the GA may not outperform highly tuned and problem-specific algorithms. Their result was confirmed experimentally by Fleurent and Ferland (1996), who reported poor performance of classical GAs when compared to local search methods. Marchiori designed a rather successful GA-based algorithm for hard 3-SAT problems by combing a native GA with a local search algorithm [7]. GA for SAT employ heuristic information into the fitness function or into the GA operations (selection, crossover, and mutation) [2, 8]. In this paper, we study the computational performance of genetic algorithm on a 3-SAT problem. GA can solve SAT problems. However it usually needs high performance computing. To overcome this problem, we study the mean first hitting time of optimum solution T, success probability S, the mean survival time a and their relations.

2 3-Satisfiablity problem

The SAT problem is a task to determine whether there exists an assignment of truth values to a set of Boolean variables that make a conjunctive normal form (CNF) formula to be true [2]. The SAT problem can be formulated as follows: given a set of clauses $C_1, C_2, \ldots, C_m$ on the Boolean variables $x_1, x_2, \ldots, x_n$, determine if there is an assignment for the variables such that the formula $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ evaluates to true, where $\wedge$ is a logical connector *and*. A clause is a disjunction of literals, e.g., $x_1 \vee \overline{x}_2 \vee x_3$, where a literal is a Boolean variable x or its negation $\bar{x}$, and $\vee$ is a logical connector *or*. If each C_i contains exactly z distinct literals, then the problem belongs to the z-SAT class [7]. In this paper we consider formulas in conjunctive normal form, and each clause has exactly 3 literals.

We use the following notations, let:

- F be a CNF Boolean formula,
- m be the number of clauses in F,
- n be the number of variables in F,
- x_k be the kth variable in $F(1 \leq k \leq n)$,

• C_i be the ith clause in $F(1 \leq i \leq m)$,
• l be the clause length in C_i,
• $Q_{i,j}$ be the jth literal in the ith clause in C_i $(1 \leq j \leq l)$.

We use bit variable $x_k = 1 \text{ or } 0$, corresponding to $true$ or $false$, respectively. We count the number of clauses that are fulfilled by the variable assignment. A chromosome is a candidate solution, which is a string of bits having length equal to n.

A common formulation for CNF formulas exists [9]. The fitness function of F is expressed by

$$f(F) = \sum_{i=1}^{m} C_i, \tag{1}$$

where C_i is denoted as

$C_i = Q_{i,1} \vee Q_{i,2} \vee Q_{i,3}$, $Q_{i,j} = x_k \, or \, \overline{x}_k (j = 1, 2, 3)$. If all clauses are satisfied, then $f(F) = m$.

In this study a set of benchmark instance was used. Problems in this benchmark are downloaded from the SATLIB-benchmark Problems [10]. The instance provided here is cnf formulae encoded in DIMACS cnf format. We use a uniform random-3-SAT problem, uf20-91, with 20 variables and 91 clauses. Each of clauses is constructed from 3 literals. The input of our program is data file format with 91 rows, 3 columns. We evolve a population of variable assignments until we find an assignment that makes the formula true.

In the following we define some variables for evaluating the GA computing performance. The success probability $S(t)$ is defined as the probability that there is at least one optimum solution at generation t in stationary distribution [11]. S is the time average of $S(t)$ defined by

$$S = \sum_{t=1000}^{3000} S_t. \tag{2}$$

T is defined as the mean first hitting time of the optimum solution in a population. The mean survival time a is the average consecutive generations containing optimum solution in the stationary distribution.

3 Numerical experiments

In this paper, we study the relation between the success probability S and the mean first hitting time T by experiments. We performed numerical calculations of GA with roulette wheel selection on the 3-SAT problem. We carried out the experiments with string length $L = n = 20$, number of clause $m = 91$ and clause length $l = 3$. Crossover was done with the uniform crossover. The calculations were performed repeatedly, and results were averaged over 10000 runs.

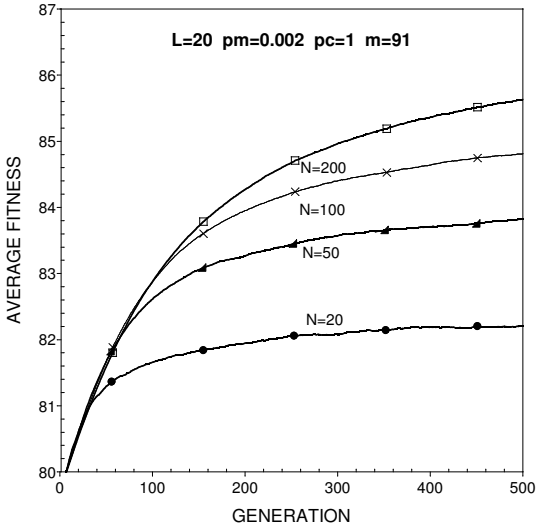

Figure 1: Dependence of average fitness on N. The horizontal axis represents generation.

Figure 1 shows the dependence of average fitness on N with $p_c = 1$ and $p_m = 0.002$. By comparing results with $N = 20, 50, 100, 200$, we observe the strong N dependence of average fitness.

Figure 2 shows the dependence of average fitness on p_m with $p_c = 1$ and $N = 200$. By comparing results with $p_m = 0.05, 0.005, 0.0005$, we find strong p_m dependence of average fitness. When mutation becomes strong, the average fitness is low, and converges into a smaller value.

Figure 3 shows N-dependence of success probability S with $p_m = 0.02$. As show in this figure, the success probability shows little difference between calculations of $p_c = 0$ and $p_c = 1$. The effect of crossover on success probability is small in this case.

Figure 4 shows p_m-dependence of success probability S with population size $N = 50$. The solid line is success probability with crossover rate $p_c = 1$, and the dotted line is success probability with crossover rate $p_c = 0$. We find that the success probability with crossover is larger than that without crossover when mutation rate is small. However, when mutation rate becomes large, the effect of crossover goes to small, and the success probabilities are almost equal.

Figure 5 shows p_m-dependence of the mean first hitting time of optimum solution T with population size $N = 50$. The value of T without crossover is larger than that of with crossover, which means crossover accelerates the speed of evolution in GA. When muta-

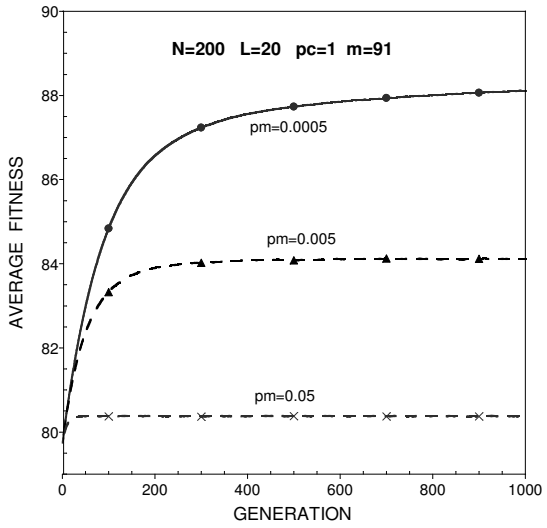

Figure 2: Dependence of average fitness on p_m. The horizontal axis represents generation.

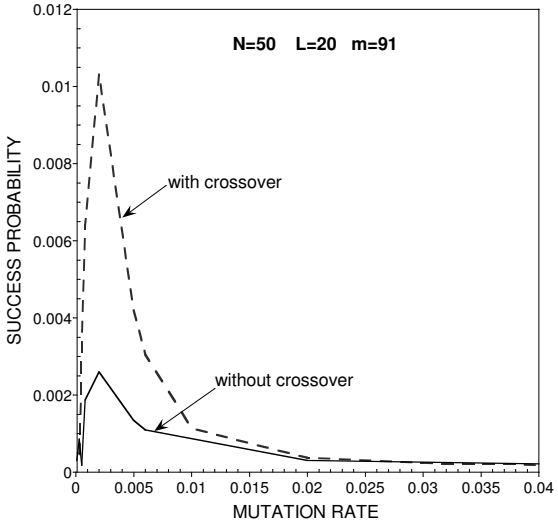

Figure 4: p_m-dependence of success probability S with crossover rates $p_c = 1$ and 0. The horizontal axis represents mutation rate p_m.

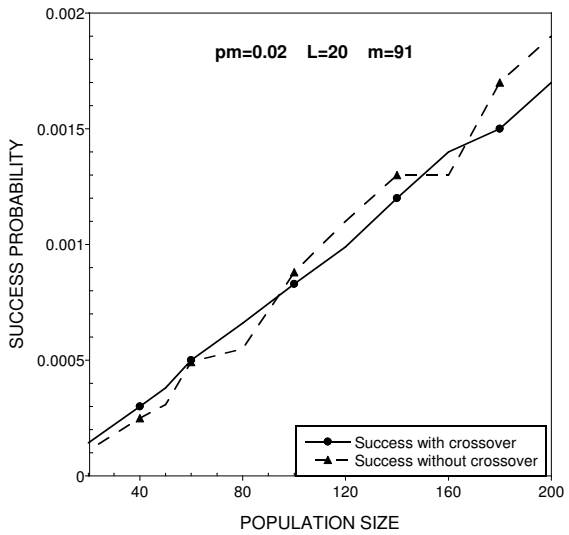

Figure 3: N-dependence of success probability S with crossover rates $p_c = 1$ and 0. The horizontal axis represents population size N.

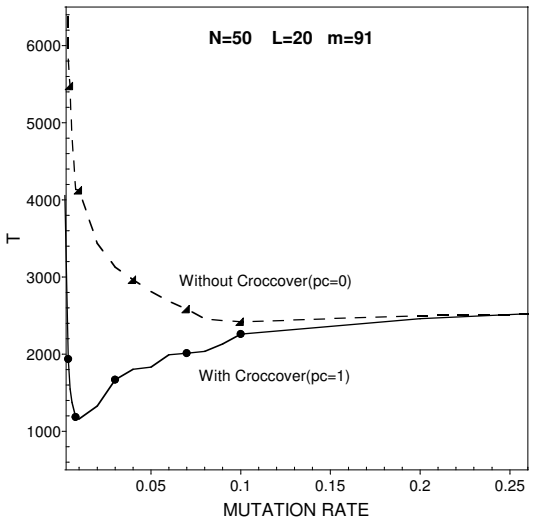

Figure 5: p_m-dependence of T with crossover rates $p_c = 1$ and 0. The horizontal axis represents mutation rate p_m.

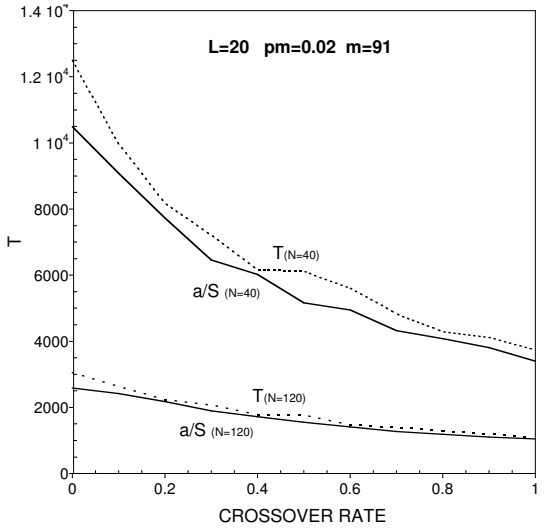

Figure 6: The realation of T and a/S with $N = 40, 120$. The horizontal axis represents crossover rate p_c.

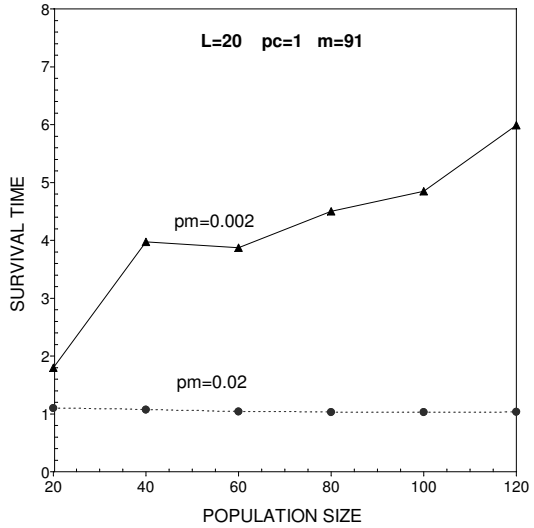

Figure 7: The dependence of a on N with $p_m = 0.02$ and 0.002. The horizontal axis represents population size N.

tion rate is large, the effect of crossover on T is small.

Figure 6 shows the relation between T and a/S when p_m is 0.02. a is the mean survival time. From this figure we observe the relation of $T = a/S$. There is strong N dependence of first hitting time of optimum solution T.

Figure 7 shows the dependence of a on N with $p_m = 0.02$ and 0.002. When $p_m = 0.02$, $a \approx 1$, while $p_m = 0.002$, a is large. Therefore, the conclusion of $T \approx 1/S$ is established if $p_m = 0.02$.

4 Summary

The GA parameters, for example, population size N, string length L, crossover rate p_c and mutation rate p_m, influence the GA performance. In this study of GA on 3-SAT problem, the effects of crossover and mutation are investigated. The results show that : when mutation rate is small,

1) The effects of crossover on S and T are large.
2) The success probability S with crossover is larger than that without crossover, which means crossover plays an important role on S.
3) The value of T with crossover is smaller than that without crossover, which means crossover accelerates the speed of evolution in GA.

We also observed the relation between T and a/S when mutation rate becomes large, and that $T \approx a/S$. When $p_m = 0.02$, $a \approx 1$ and $T \approx 1/S$.

References

[1] J. Gu: Randomized and Deterministic Local Search for SAT and Scheduling Problems. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, Vol.43, 61–62 (1999).

[2] J. Gu: Parallel Algorithms for Satisfiability (SAT) Problem. *DIMACS Volume Series in Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, Vol.22, 105–161 (1995).

[3] M. Davis, H. Putnam: A Computing Procedure for Quantification Theory. *Journal of the ACM*, Vol.7, Issue 3, 201-215 (1960).

[4] J. Hansen, B. Jaumard: Algorithms for the maximum satisfiability problem. *Computing*, Vol.44, Issue 4, 279-303 (1990).

[5] K. De Jong, W. Spears: Using genetic algorithms to solve NP-complete problems. *Proceedings of the Third International Conference on Genetic Algorithms*, 124-132 (1989).

[6] C. Fleurent, J. Ferland: Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. *Discrete Mathematics and Theoretical Computer Science*, Vol.26, 619-652(1996).

[7] E. Marchiori, C. Rossi: A Flipping Genetic Algorithm for Hard 3-SAT Problems. *In Genetic and Evolutionary Computation Conference*, Vol.1, 393-400 (1999).

[8] J. Gottlieb, E. Marchiori, C. Rossi: Evolutionary Algorithms for the Satisfiability Problem. *Evolutionary Computation*, Vol.10, Issue 1, 35-50 (2002).

[9] J. Gu: Effecient local search for very large-scale satisfiability problem. *SiGART Bulletin*, Vol.3, Issue 1, 8-12(1992).

[10] *http://www.cs.ubc.ca/ hoos/SATLIB/benchm.html.*

[11] Y. Zhang, M. Sakamoto, H. Furutani: Effects of Population Size and Mutation Rate on Results of Genetic Algorithm. *Proceedings of Third International Conference on Natural Computation*, Vol.3, 70-75 (2008).