Using Self-learning and Automatic Tuning to Improve the Performance of Sexual Genetic Algorithms for Constraint Satisfaction Problems*

Hu Xu[1], Karen Petrie[2], and Iain Murray[3]

1 **Computing School,**
QMB 1.10, University of Dundee
huxu@computing.dundee.ac.uk
2 **Computing School,**
QMB 2.10, University of Dundee
karenpetrie@computing.dundee.ac.uk
3 **Computing School,**
QMB 2.21, University of Dundee
irmurray@computing.dundee.ac.uk

Abstract

Currently the parameters in a constraint solver are often selected by hand by experts in the field; these parameters might include the level of preprocessing to be used and the variable ordering heuristic. The efficient and automatic choice of a preprocessing level for a constraint solver is a step towards making constraint programming a more widely accessible technology. Self-learning sexual genetic algorithms are a new approach combining a self-learning mechanism with sexual genetic algorithms in order to suggest or predict a suitable solver configuration for large scale problems by learning from the same class of small scale problems. In this paper, Self-learning Sexual genetic algorithms are applied to create an automatic solver configuration mechanism for solving various constraint problems. The starting population of self-learning sexual genetic algorithms will be trained through experience on small instances. The experiments in this paper are a proof-of-concept for the idea of combining sexual genetic algorithms with a self-learning strategy to aid in parameter selection for constraint programming.

1998 ACM Subject Classification I.2.6 Learning

Keywords and phrases Self-learning Genetic Algorithm, Sexual Genetic algorithm, Constraint Programming, Parameter Tuning

Digital Object Identifier 10.4230/OASIcs.ICCSW.2013.128

1 Introduction

The selection of suitable preprocessing levels for a given constraint problem is an important part of constraint programming (CP); efficiently tuning a constraint solver will shorten the search time and reduce the running cost. One significant method of increasing the search speed for a constraint solver is by tuning the solver's parameters [8]. Currently, the job of tuning the parameters is done manually; a skilled user selects the most suitable preprocessing method using previous experience from similar classes of problems. In most cases, the best preprocessing method used in similar classes of problems will provide a useful clue to aid the

* This work was funded by the School of Computing, University of Dundee.

2013 Imperial College Computing Student Workshop (ICCSW'13).
Editors: Andew V. Jones, Nicholas Ng; pp. 128–135

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

user's selection. However, this learning curve could be a barrier to novice users in learning how to efficiently use a CP solver [9].

Genetic algorithms (GA) are a classic global optimisation method posed by John Holland [6], which mimic the competition of organisms in nature and the mechanisms of evolution. GAs are usually implemented in a computer simulation in which a population of abstract representations of candidate solutions to an optimisation problem evolves towards better solutions. GAs are widely applied to optimisation problems such as function optimisation. Ansótegui et al. have proposed a gender-based genetic algorithm for the automatic configuration of algorithms[1] ; it shows that the sexual genetic algorithm (SGA) is feasible for automatic configuration.

In this paper, sexual genetic-based algorithms were chosen to select a preprocessing method for constraint satisfaction problems. There are three main reasons to choose SGAs to optimise preprocessing selection:

- SGAs have a powerful ability to tackle optimisation problems which lack auxiliary information
- SGAs perform parallel search rather than linear search; each chromosome (solution to the problem) competes against others in each generation
- SGAs are more efficient than standard GAs in preprocessing selection; it is not necessary for the SGA to evaluate the fitness of all chromosomes, which is a considerable consumer of CPU time.

Therefore the idea of combining SGAs and constraint programming seems worth exploring further and it is expected that automatic tuning will lead to improvements over manual tuning by users. ParamILS and CALIBRA [7] have demonstrated the practicality and efficiency of automatic configuration for constraints solvers. However, the general framework of combining GAs with constraint programming and the exploration of parameter sensitivity of GAs to any problems, has not been achieved. In light of this situation, a self-learning sexual genetic-based method for tuning Minion [4], which is one of the most efficient constraint solvers in the world, was proposed.

This paper firstly investigates an SGA and explores its features. The efficiency of this SGA will be tested by comparison with a standard GA for the Travelling Salesman Problem. The self-learning genetic algorithm (SLGA) will then be introduced and applied to select the preprocessing level. Finally, the efficiency of the self-learning sexual genetic algorithm (SLSGA) will be analysed. This paper also provides a proof for one possibility of improving sexual genetic algorithms for preprocessing selection in constraint satisfaction problems [2].

2 Sexual Genetic Algorithm

The basic concepts and features of SGAs are similar to standard GAs. As in standard GAs, there are three basic operators in sexual genetic algorithms: selection, crossover and mutation. The selection, which decides the parents for mating, is very important for evolution. In standard genetic algorithms, there is just one selection strategy during evolution. In nature, male individuals try to spread their gene information as widely as possible and female individuals try to select the fittest males to mate with [13]. Inspired by the natural behaviour of male vigor and female choice, sexual genetic algorithms [10] [12] apply two different selection mechanisms: male group (competitive) and female group (co-operative).

The first step of a genetic algorithm (GA) is called the encoding which is to construct a suitable starting chromosome for the optimisation problem and which can transfer solutions of the optimisation problem to the child chromosomes, where each child chromosome presents

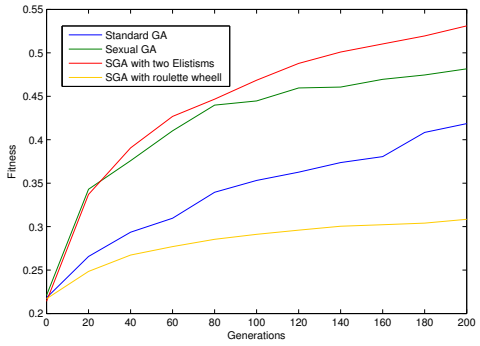

Figure 1 The efficiency of sexual genetic algorithm in solving the Travelling Salesman Problem.

one possible solution. Fitness describes the ability of an individual to reproduce in biology; the fitness function evaluates the difference between the desired result and the actual result.

Selection in genetic algorithms is the strategy which allows the best parents (with highest fitness) to have an increased chance of being selected to generate the next generation. Roulette wheel selection is a commonly-used way of choosing individuals from the population of chromosomes in a way that is proportional to their fitness. Roulette does not guarantee that the fittest member goes through to the next generation, merely that it has a better chance of doing so. In sexual genetic algorithms, the population is randomly divided into to two groups: male (competitive) and female (co-operative) as in nature. The male chromosomes have to compete for the chance of mating (the elitisms (the chromosomes with better fitness) will confer an increased chance of mating), while the female chromosomes have the same opportunity for mating. The running time of the fitness evaluation is substantial when automated tuning is used, because the fitness (searching time) of each chromosome has to be calculated with a given set of preprocessing for the constraint problem. The pseudocode (Algorithm 1) shows that half of the population is selected as male, meaning that half of the fitness evaluation time is saved while the variety of the population is maintained. This is the most important reason that sexual genetic algorithms were selected for the experiment in this paper.

Crossover can improve the fitness of the whole population quickly by mating parents to produce an offspring. Single-point crossover is the most common crossover in genetic algorithms because it can be easily understood and realized. The single crossover will be applied in sexual genetic algorithm.

Mutation, which changes one or more genes in an individual, is another operator used in GA. Mutation can help genetic algorithms escape the local maximum state by creating a new gene string. All of the mutations in this paper's experiments are single-point mutations.

3 Sexual Genetic Algorithm vs. Standard Genetic Algorithm by Solving TSP

The Travelling Salesman Problem (TSP) seeks the shortest Hamiltonian cycle path between n given cities and is a classic NP-complete problem. To prove the efficiency of sexual genetic algorithms and explore their features, a sexual genetic algorithm was applied to solve the TSP with different elitism percentages and this was compared with a standard genetic algorithm. There are three elitism strategies: one elitism, two elitisms and roulette wheel selection elitism.

Table 1 The solving times of different constraint satisfaction problems using of a sexual genetic algorithm.

	BIBD (Solving Time)	Langford (Solving Time)
Sexual GA	1.7 s	3.2 s
Standard GA	3.5 s	4.3 s

Figure 1 shows the efficiency of an SGA to solve the TSP. There are four curves in the graph: standard GA, SGA with one elitism, two elitisms and roulette wheel selection elitisms in male competition. It clearly shows that the SGA with two elitisms in male competition is most the efficient in solving the TSP; however, more elitisms does not mean better evolutionary speed because more elitisms could lead to high convergence. Compared with the standard GA, the SGA with optimised elitisms in male competition always approaches a better fitness.

Finally, the SGA was used to choose the best preprocessing method for constraint problems. In this paper, two classic constraint optimisation problems, balanced incomplete block design (BIBD) and the Langford's Number problem[1] , were chosen as the optimisation problem for testing the SGA. Following David's MicroGA Settings[3], the crossover rate is 0.5 and the mutation rate is 0.04 in all experiments. Each trial was run 100 times and the average of the minima was recorded.

Table 1 shows the solving times of the SGA for different problems in comparison with a standard GA. It shows that the SGA could find better preprocessing methods for both the BIBD and Langford's Number problems than standard GA for a small number of generations.

4 Self-learning Sexual Genetic Algorithm vs. Self-learning Genetic Algorithm

Generally, machine learning makes predictions by training, validation and testing itself against existing data [11] . Self-learning, which learns its own inductive bias based on previous experience, is one of the typical algorithms in the machine learning domain. Self-learning could avoid repetition of searching and computation in the previous experiments. In genetic algorithms, the generation of the starting population is a considerable factor, as are the crossover rate and mutation rate.

To check the influence of the fitness value in the starting population of genetic algorithms, two different starting populations (set to high fitness and low fitness) were applied to optimise the same function $F(x) = x^{10}$[5] where $0 < x < 1$. The size of both starting populations was 30 chromosomes. Figure 2 shows that the genetic algorithm with a starting population with high fitness could approach better fitness than the one with a starting population with poor fitness. This demonstrates that a suitable starting population for a genetic algorithm could lead to a more rapid approach to best fitness, and suggests that this could be applied to self-learning in machine learning.

A self-learning genetic algorithm (SLGA) [14] is an algorithm which makes the preprocessing prediction by using previous experience on the same classes of constraint satisfaction problem. An SLGA can improve the search speed by defining a specific starting population

[1] All From www.csplib.org

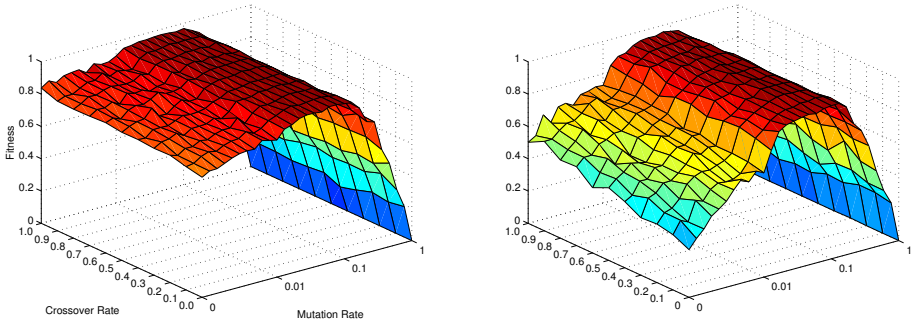

Figure 2 The evolutionary speed comparison with different starting populations. The X axis is the mutation rate and the Y axis the Crossover rate. The Z axis is the best fitness after 50 generations with different mutation rate and crossover rate. The fitness of the starting population of the left graph is lower than 0.1. The fitness of the starting population in the right graph is randomly generated between 0 and 1.

(instead of a random starting population as normally used), with the starting population taken from the training data of the same class of small instance problems.

4.1 Self-learning Sexual Genetic Algorithm for Constraint Satisfaction Problem

The SLSGA pseudocode (Algorithm 1) introduces SLSGA's working principle and clearly shows how self-learning combines with the sexual genetic algorithm. Compared with the self-learning standard genetic algorithm, the self-learning sexual genetic algorithm halves the time spent on fitness evaluation and selects k elitisms from the male group, instead of from the whole population. Compared with the SGA, the SLSGA also has an optimised starting population; the SLSGA is trained using small scale problems to select the starting population for large scale problems.

Algorithm 1 Self-learning Sexual Genetic Algorithm

Produce the starting populations P_i for small instance problem from the experience of small instance ▷ i is the population size
for $j = 1$ to n **do** ▷ j is the generation
  repeat
    Randomly select m chromosomes of population as male
    The rest chromosomes is selected as female ▷ $m = i/2$ in our SGA for tuning constraint programming solver
    Evaluate the fitness of male chromosomes
    Select k elitisms from male chromosomes to mating pool
    Each female chromosomes has the same possibility for mating
    New generation is created from k elitisms and mother chromosomes by crossover and mutation
  until λ =The best preprocessing found or the searching time is out of time limit
end for
return λ

Table 2 The solving times of Self-Learning Sexual Genetic Algorithm. "CSP Problems" refers to the training instance and the optimised problems. "Training Instance" is the search time in seconds and total search nodes for small instances without preprocessing. "Target Instance" is the search time and total search nodes for optimised (large) instances without preprocessing. "Sexual GA" and "SLSGA" are the search time and total search nodes with preprocessing that had been optimised with Sexual Genetic Algorithm and Self-learning Sexual Genetic Algorithm.

CSP Problems	Training Instance		Target Instance		Sexual GA		SLSGA	
	Running Time	Nodes	Running Time	Nodes	Running Time	Nodes	Running Time	Nodes
BIBDline1 BIBDline6	0.75	4332	3.67	48502	3.671	48502	2.5	32773
BIBDline1 BIBDline8	0.75	4332	6.56	90432	5.21	11	3.8	11
BIBDline2 BIBDline8	1.21	10952	6.56	90432	3.81	25269	3.7	25269
OpenStack1 OpenStack2	0.156	106	13.2	835567	9.9	15828	9.9	15828
Lanford(3,10) Lanford(2,20)	0.16	60	22	280968	1.95	22	0.21	696
Lanford(3,10) Lanford(3,17)	0.16	60	13.2	105873	2.13	11840	1.59	8729
BIBDline4 BIBDline11	2.3	29257	207	2287940	136	1670535	136	1670535

The experimental results of SLSGA show that an optimised starting population easily and quickly leads to a better evolutionary result. The aim of automatic tuning is to use the shortest time possible to find the optimal preprocessing settings. To prove the correctness of the hybridization idea (combining self-learning with SGA) and the efficiency of SLSGA, it was then applied to solve some constraint satisfaction problems with different instances: BIBD, Langford's Number and Open Stack problems. All the settings used were the same as others described in this paper.

Table 2 shows the efficiency of SLSGA for solving various constraint satisfaction problems. The table shows the number of search nodes and running time for the solver to find the solution after optimisation of SGA and SLSGA. It clearly shows that SLSGA could arrive at acceptable result more efficiently than SGA, improving the evolutionary speed and deriving useful results at the same time. This shows that self-learning is a feasible algorithm for preprocessing selection in constraint satisfaction problems, although the optimal result from SLSGA is not significantly better than that from SGA.

5 Conclusions and Future work

To prove the concept and potential effectiveness of self-learning sexual genetic algorithms, this paper has firstly shown the efficiency of a sexual genetic algorithm by solving the TSP and comparing this with a standard genetic algorithm. It has demonstrated that the elitisms percentage in the SGA is very important and that selecting suitable elitisms percentages leads to an ideal optimisation speed. Self-learning was proposed to improve the tuning efficiency from previous experience (in the same way as human behaviour) rather than by logical inference. The experimental results showed that the large scale problem could be properly solved using an optimised starting population which was trained using data from small scale problems. Thus the self-learning sexual genetic algorithm was able to achieve satisfactory results by combining two strategies.

The results show that self-learning genetic algorithms can be efficient methods for selecting preprocessing of constraint-solving problems. However, a number of challenges remain for future exploration. In this paper, four classic problems were used to verify the efficiency of a self-learning sexual genetic algorithm on large problems. More and larger-scale problems such as the car sequence problem will be used to explore the efficiency and limitations of SLSGAs. With regard to self-learning, the convergence of the starting population is worth further exploration, for example whether having many optimised individuals in the starting population can lead to local traps. The application of the self-learning strategy to deal with multiple instances is also of interest.

Currently the best model to solve a constraint satisfaction problem is selected by hand by the user; this paper has shown that improved performance may be obtained by applying self-learning sexual genetic algorithms to constraint satisfaction problems.

References

1 C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Principles and Practice of Constraint Programming-CP 2009: 15th International Conference, CP 2009 Lisbon, Portugal, September20-24, 2009 Proceedings*, page 142. Springer, 2009.

2 David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study.* Princeton University Press, 2011.

3 David L. Carroll. Chemical laser modeling with genetic algorithms. *Aiaa Journal*, 34:338–346, 1996.

4 Ian P. Gent, Christopher Jefferson, and Ian Miguel. Minion: A fast scalable constraint solver. In *Proceedings of the 17th Eureopean Conference on Artificial Intelligence (ECAI'06)*, pages 98–102, 2006.

5 David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

6 John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. 1992.

7 Frank Hutter, Holger H. Hoos, and Thomas Stützle. Automatic algorithm configuration based on local search. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 2*, AAAI'07, pages 1152–1157. AAAI Press, 2007.

8 Lars Kotthoff, Ian Miguel, and Peter Nightingale. Ensemble classification for constraint solver configuration. In *Principles and Practice of Constraint Programming–CP 2010*, pages 321–329. Springer, 2010.

9 Tony Lambert, Carlos Castro, Eric Monfroy, María Riff, and Frédéric Saubion. *Hybridization of Genetic Algorithms and Constraint Propagation for the BACP*, volume 3668 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005.

10 M.M. Raghuwanshi and O.G. Kakde. Genetic algorithm with species and sexual selection. In *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*, pages 1–8, 2006.

11 Simon Rogers and Mark Girolami. *A First Course in Machine Learning*. Chapman & Hall/CRC, 1st edition, 2011.

12 M. Jalali Varnamkhasti and MasoumehVali. Sexual selection and evolution of male and female choice in genetic algorithm. *Scientific Research and Essays*, 7(31):2788–2804, 2012.

13 Stefan Wagner and Michael Affenzeller. Sexualga: Gender-specific selection for genetic algorithms. In *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, volume 4, pages 76–81. Citeseer, 2005.

14 Hu Xu and Karen Petrie. Self-Learning Genetic Algorithm For Constrains Satisfaction Problems. In Andrew V. Jones, editor, *2012 Imperial College Computing Student Workshop*, volume 28 of *OpenAccess Series in Informatics (OASIcs)*, pages 156–162, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.