SBSE

650046219

May 11, 2016

Abstract

This is a literature review briefly looking at Search Based Software Engineering, its applications benefits and drawbacks as well as a more through examination of how it is conducted through the use of Nature Inspired Computing.

I certify that all material in this document which is not my own work has been identified.

1 Introduction

With the increase in popularity of Software systems, new methods to optimise their development and release have been produced. With a relatively small increase in the total number of possible features a software house could add to their development schedules, the number of different permutations increases rapidly. There are many management frameworks and methodologies that are unique to software development however many advances are currently being made in the area of optimising software development. Organisational advantages of optimising Software development include more effective use of resources such as developers or memory management, advantages to businesses such as retaining revenue streams and especially to the intended users of the software as better iterations can be produced faster. The example given for software users is known as the Next Release Problem.

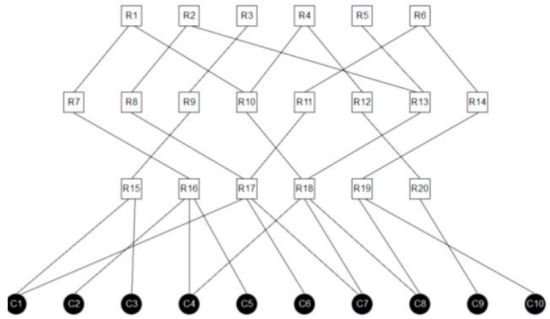
In an attempt to find better techniques for Software Development and Computer Architecture, as with many other fields of Science, Academics look to Nature for inspiration in problem solving. Nature Inspired Computing has played a large part in Artificial Intelligence and man of the same techniques have shown promising results when applied to the same problems Search Based Software Engineering attempts to solve. Creating links between options similar to the way in Neural Networks and analysing Ant Colonies are some of the ways Computer Scientists are finding solutions to the new problems we all face.

This paper will briefly look at some of the Problems Search Based Software Engineering is being applied to, as will as the technique of Genetic Algorithms works in more detail.

2 Next Release Problem

Updating software can easily become one of the most time consuming parts of developing new sofftwareas its the software can become used more widley or requiments such as features or hardware become updated. [1, 2]From a productivity or business point of view choosing which aspects to work on becomes in itself an optimisation problem for example an online game will need to balance retaining reliability as well as increasing revenue streams in the form of add ons in much the same way large software houses will have to work to ensure their packages remain competitive in a rapidly changing market.
Search Based Software Engineering is increasinly used to assit in this decision making process by treating the problem as a search space of a number of solutions and highlting the pros and cons of these choices. [3, 1, 2]

Figure 1: Node diagram showing possible release pathways 'C' with requirments and requirments 'R' with dependancies [4]

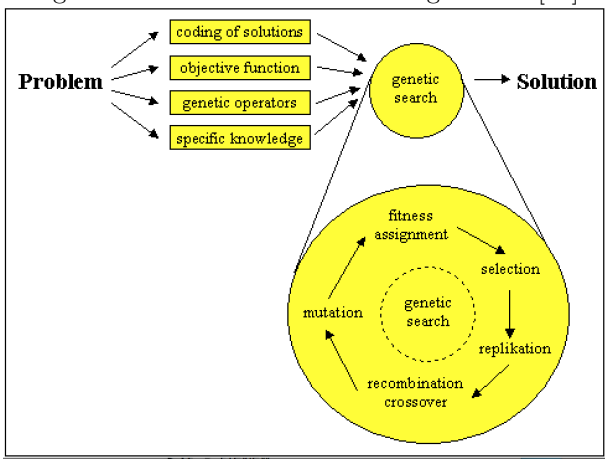


3 Genetic Algorithms

A genetic algorithm is a method used in Search Based Software Engineering as well as other fields of Science such as Artificial Intelligence to create solutions to problems based on an advanced form of trial and error or Evolution. a fitness function is first calculated which determins how relevant a given solution is for example, how quickly does this program solve the problem.[5, 6] This allows different solutions to be compared and imporved upon or discared as nessesary. [7, 8, 9, 10, 11] In the context of a program the search space could involve language syntax or specific packages such as graphics or mathematics.

The results from the search space with a higher fitness function are then optimised by being combined with other results. this closely mimics the way evolution has been observed in nature with successful gene expressions replacing less useful or exisitng traits. In the context of Software Engineering a code snippet can be thought of as a gene and it is the natural selection of the fitness functions that detrmines which expressions will continue to evlove or die out as they are replaced. [12, 10, 13, 11] The duration of inclusion of previous generations could also be factored into the fitness function to ensure that effective solutions are not cast aside before they are given a chance to develop.
In the following subject the general stages of a genetic algorithm will be outlined.

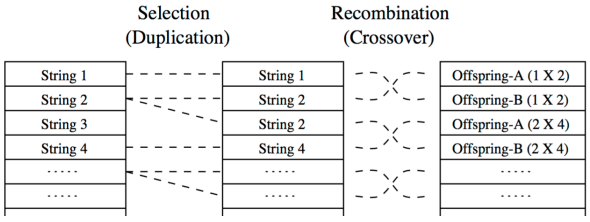
Figure 2: Flow chart of Genetic algorithms [14]



3.1 Selection

The inital population is a range of solutions found within the search space of total possible combination of solutions. Each canidate is then analysed by the fitness function and options with a higher fitness are duplicated. this gives a higher probability that the ofspring will have improved fitness over the population members or parents that have come before it[5, 10]. IThere are a number of different selection algorithms, this one mentioned is closer to 'roulette-wheel' which is also known as Sampling with replacement.

Figure 3: Selection, recombonation or crossover stage



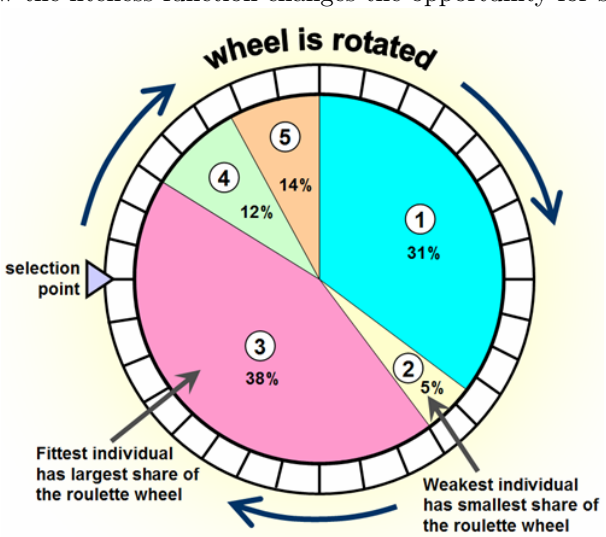
[7]

[15] The resulting population is refered to as an intermediate generation and becomes a new generation after the next stage, recombination.[7, 8, 11]

3.2 Recombination

Two popular methods of recombination or crossover are PMX and GPX which are partial matching crossover and generalised position recombination respectivly.[10] the main difference between these two methods is found in the information retained by each operator in the resulting combinations.PMX focuses more on keeping track of the relative relation between variables while GPX has more of an emphasis on the absolut relation.[14]

Figure 4: How the fiteness function changes the opportunity for selection



[16]

3.3 Mutation

Another technique inspired from evolution is the use of random mutations. an arbitrary method of selecting a bit to invert or swap with another is usally employed. This is a useful way of ensuring that the solutions produced by the algorithm can adapt in a way not initally covered by the fitness function. In nature random mutations in a species can often become dominant traits.[10, 17, 11] Introducing radom variations is also prevents the algorithm from producing repated ofspring and can also produce new variations without having to repeat all the previous stages.[18]

3.4 Advantages and Disadvantages

The benefits of progressing in Genetic Algorithms have applications in any field of Engineering and provide a number of solutions to problems found in this fields. multiple fitness functions can also be used to discard less useful solutions. Despite the numerous benefits of Genetic Algorithms, as with any new method there are still a number of ways in which it can be improved. Real time applications of this technique largely depend on a computers speed, a solution may not be adequately found if the search space is too large or changes rapidly and is very dependant on a good definition of a fitness function which can still require Human intervention.[19, 20]

Figure 5: Example pusedocode of a genetic algorithm

1. Initialise search population: $population = \{\mathbf{x}_i\}_{i=1}^{n}$
2. evaluate(population)
3. while stopping criteria not reached
4. parents = select_parents(population)
5. children = recombine(parents)
6. children = mutate(children)
7. evaluate(children)
8. population = merge_and_reduce(children, population)

[10] 9. end while 10. return best solution found

4 Conclusion

One particularly interesting application of Software based Engineering in the near future is is automatic code generation. If the large amount of Open Source programs currently available online from academic institutions and online repositories was able to labelled in an effective way, categorised and searched through in a timely fashion, an algorithm or artificial intelligence could be developed which could easily not only predict the best strategies for development but also write code with less human intervention.

References

[1] Omolade Saliu and Guenther Ruhe. Supporting software release planning decisions for evolving systems. University of Calgary http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.458.5725&rep=rep1&type=pdf.

[2] Francisco J. Orellana Jose del Sagrado, Isabel M. del Aguila. Requirements interaction in the next release problem. University of Calgary https://pdfs.semanticscholar.org/8acd/f1ade1aad983e8f89c90e2179e67a813e22d.pdf.

[3] Enrique Alba Juan J. Durillo Mark Harman, Yuanyuan Zhang and Antonio J. Nebro. A study of the bi-objective next release problem pages 1 - 4. University of Malaga http://www0.cs.ucl.ac.uk/staff/mharman/durillo.et.al-emse.pdf.

[4] Jerffeson T. Souza Fabricio G. Freitas, Daniel P. Coutinho. Software next release planning approach through exact optimization. State University of Ceara http://www.ijcaonline.org/volume22/number8/pxc3873636.pdf.

[5] Darrell Whitley. A genetic algorithm tutorial, pages 4-6. Colorado State University http://www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf.

[6] Phil McMinn. Search-based software testing: Past, present and future. University of Sheffield https://philmcminn.staff.shef.ac.uk/publications/c18.pdf.

[7] Darrell Whitley. A genetic algorithm tutorial, pages 1-2. Colorado State University http://www.cs.colostate.edu/~genitor/MiscPubs/tutorial.pdf.

[8] Mitchell Melanie. An introduction to genetic algorithms pages 1, 4. A Bradford Book The MIT Press http://www.boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf.

[9] K. Matous M. Leps J. Zeman M. Sejnoha. Applying genetic algorithms to selected topics commonly encountered in engineering practice pages 1 - 2. Czech Technical University https://www3.nd.edu/~kmatous/Papers/CMAME_AGA.pdf.

[10] Jonathan Fieldsend. Frontiers of computer science search-based software engineering ii slides 4 - 13. University of Exeter http://vle.exeter.ac.uk/pluginfile.php/696690/mod_resource/content/2/ECM1412_SBSE2.pdf.

[11] SethBling. Mari/o - machine learning for video games. https://www.youtube.com/watch?v=qv6UVOQ0F44.

[12] Mitchell Melanie. An introduction to genetic algorithms page 5. A Bradford Book The MIT Press http://www.boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf.

[13] Mark Harman. The relationship between search based software engineering and predictive modeling chapter 2. University College London http://www0.cs.ucl.ac.uk/staff/mharman/promise-keynote.pdf.

[14] Hartmut Polheim. Evolutionary algorithms 10 combination of operators and options to produce evolutionary algorithms chapter 10. GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab http://www.geatbx.com/docu/algindex-09.html.

[15] Hartmut Polheim. Genetic and evolutionary algorithms: Principles, methods and algorithms section 1, section 3. Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms http://www.pg.gda.pl/~mkwies/dyd/geadocu/algindex.html.

[16] John Dalton. Roulette wheel selection. Newcastle University http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php.

[17] Gregor Mendel. Experiments in plant hybridization chapter 4. http://www.mendelweb.org/Mendel.html.

[18] Nikita Upadhyay Manish Dikit and Sanjay Silakari. An exhaustive survey on nature inspired optimization algorithms. http://www.sersc.org/journals/IJSEIA/vol9_no4_2015/11.pdf.

[19] Mark Harman Phil McMinn Jerffeson Teixeira de Souza and Shin Yoo. Search based software engineering: Techniques, taxonomy, tutorial. University College London, University of Sheffield, State University of Ceara `http://www0.cs.ucl.ac.uk/staff/mharman/laser.pdf`.

[20] Mujahid Tabassum and Kuruvilla Mathew. A genetic algorithm analysis towards optimization solutions. Swinburne University of Technology, International Journal of Digital Information and Wireless Communications.