Syrian Arab Republic
Higher Institute for Applied Sciences and Technology
Informatics Department
Fourth Year

Using Genetic Algorithms to Find Approximations for the Minimum Vertex Cover Problem

Keywords: genetic algorithms, NP-complete, combinatorial optimization, non-deterministic algorithms, approximation algorithms, minimum vertex cover.

Author: *Farouk Hjabo*
Academic Supervisor: *Dr. Said Desouki*
General Supervisor: *Dr. Kadan Aljoumaa*
Langauge Supervisor: *Mr. Fahmi Alammareen*

February 22, 2018

Abstract

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

"It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change."

— Charles Darwin

Contents

1 Introduction

Once the NP-hardness of a combinatorial optimization problem is established, the search for an optimal solution is abandoned. The goal then becomes one of finding a good heuristic, i.e. a polynomial running time algorithm that can find solutions close to the optimal. In most cases, traditional heuristics are problem dependent; a heuristic is tailored to the specific problem it is trying to solve.

In this paper, we present an alternative approach that uses genetic algorithms as a generalized heuristic for solving NP-hard combinatorial optimization problems. The application of a genetic algorithm is demonstrated here for the *minimum vertex cover* problem. These algorithms have been successfully applied to a broad range of problems. This wide range can be tackled by genetic algorithms mainly due to the fact that they work with an encoding of the domain rather than with the problem domain itself.

2 Genetic Algorithms

2.1 The Intuition Behind GAs

Genetic Algorithms (GAs) [3] are population based search algorithms, where by repeated use of genetic operations, such as ***mutation***, ***selection***, ***crossover***, etc... Successive new generations of better populations in the direction of search objectives are created. They are inspired by Darwinian principles based on natural evolution. In other words, the main idea behind genetic algorithms is that only the ***fittest*** individuals will survive.
Main advantage of genetic algorithms is that they ideally do not make any assumption about the underlying problem, hence they are suitable to tackle a wide range of diverse problems in engineering, art, biology, economics, marketing, genetics, operations research, robotics, social sciences, physics, politics, chemistry, etc...

So what is a GA? A typical GA consists of the following:

1. a number, or ***population***, of candidate solutions to the problem.
2. a way of calculating how good or bad the individual solutions within the population are. i.e. how ***fit*** an individual is?

3. a method for mixing fragments of the better solutions to form new, on average even better solutions. Which permits the population to ***evolve*** naturally.
4. a ***mutation*** operator to avoid permanent loss of diversity within the solutions. This allows introducing new information in the population.

2.2 Typical GA Schema

The basic iteration cycle of a genetic algorithm proceeds on a population of individuals, each of which represents a search point in the space of potential solutions of a given optimization problem. In case of a canonical genetic algorithm, each individual is a binary vector $\vec{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ of fixed length n. The fitness function $f : \{0,1\}^n \to \mathbb{R}$ provides a quality measure which is used by the selection procedure to direct the search towards regions of the search space where the average fitness of the population increases. The recombination operator allows for the exchange of information between different individuals, and mutation introduces innovation into the population.
This cycle is repeated until a termination criterion is fulfilled. In most cases, the algorithm is terminated after a certain number of iterations of the basic cycle or when a satisfactory fitness level has been reached. refer to figure 1 for a visual representation of this process.

The population size (denoted as u) depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Often, the initial population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be 'seeded' in areas where optimal solutions are likely to be found.

Normally, selection in genetic algorithms is a probabilistic operator which uses the relative fitness $p_s(\vec{x_i}) = f(\vec{x_i}) / \sum_{j=1}^{u} f(\vec{x_j})$ to serve as selection probabilities (u denotes the population size). This selection operator is called proportional selection. If the problem under consideration is a minimization one, or if the fitness function can take negative values, then $f(\vec{x_i})$ has to be linearly transformed before calculating selection probabilities. This technique known as linear dynamic scaling is commonly used in genetic algorithms (see [13], pp. 123-124, or [14]).

The recombination (crossover) operator allows for the exchange of information between different individuals. The original one-point crossover [9] works on two parent individuals (which are randomly chosen from the population, see the next paragraph)

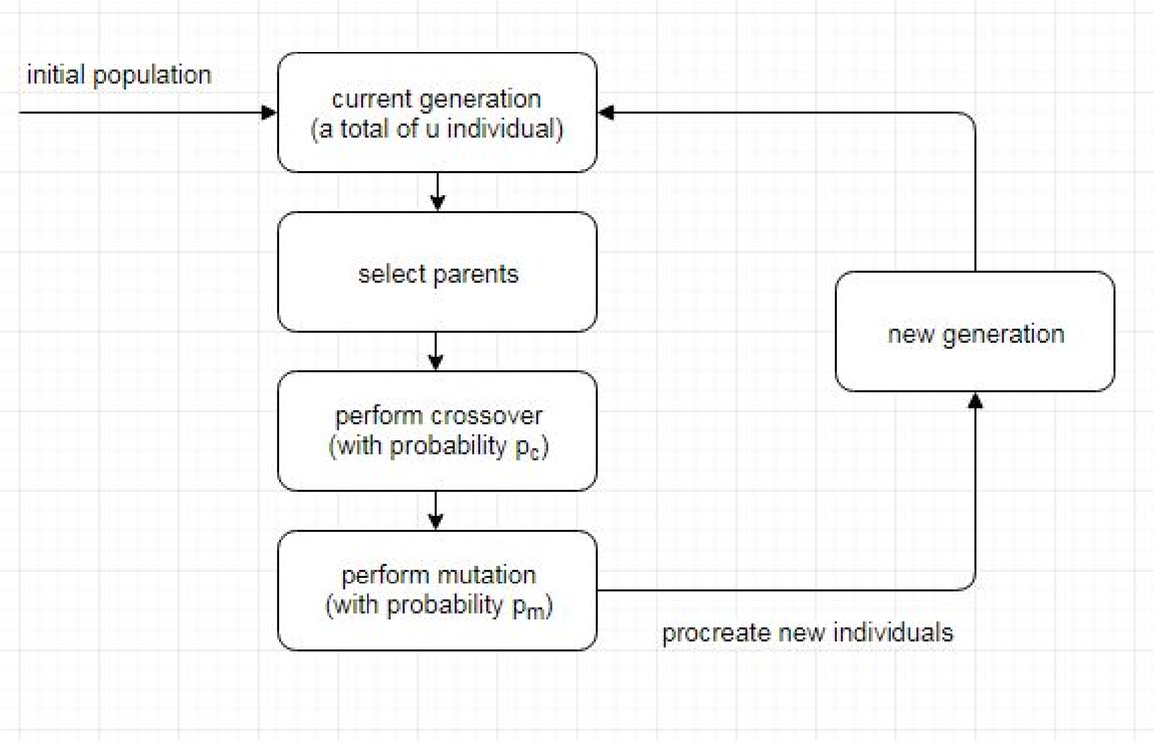

Figure 1: general schema for genetic algorithms

by choosing a crossover point $\chi \in \{1, \ldots, n-1\}$ at random and exchanging all bits after the χ^{th} one between both individuals. The crossover rate p_c (e.g. $p_c \approx 0.6$) determines the probability to undergo crossover (see todo for more details). The crossover operator can be extended to a generalized multi-point crossover [10] or even to uniform crossover, where an it is randomly decided for each bit whether to exchange it or not [19]. The strong mixing effect introduced by uniform crossover is sometimes helpful to overcome local optima.

Innovation, i.e. new information, is introduced into the population by means of mutation, which works by inverting bits with a small probability p_m (e.g. $p_m \approx 0.001$) Though mutation is often interpreted as a rather unimportant operator in genetic algorithms [9], recent theoretical work gives strong evidence for an appropriate choice of a mutation rate $p_m = 1/n$ on many problems [2; 12].

3 The Minimum Vertex Cover Problem

Lorem [2] ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4 Experimental Results

Lorem [1] ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5 Conclusion

sit amet, consectetur adipisicing elit, sed do eiusmod

References

[1] P. A. M. Dirac. *The Principles of Quantum Mechanics*. International series of monographs on physics. Clarendon Press, 1981.

[2] A. Einstein. “Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]”. In: *Annalen der Physik* 322.10 (1905), pp. 891–921.

[3] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Artificial Intelligence. Addison-Wesley Publishing Company, 1989.