An evolutionary algorithm for the Minimum Hitting Set Problem

V. CUTELLO, E. MASTRIANI, F. PAPPALARDO
Department of Mathematics and Computer Science
University of Catania
V.le A. Doria 6, 95125 Catania
ITALY
{cutello, mastriani, francesco}@dmi.unict.it

Abstract: - In this paper we propose an evolutionary algorithm to approximate optimal solutions to instances of the Minimum Hitting Set Problem, a well known $\mathcal{NP}$-complete problem (see [3]). Our genetic algorithm (see [2]) will use the idea of *viruses* which infect chromosomes and change one of their bits. A special dynamic fitness function has been also used to improve overall performances.

Key-Words: Minimum Hitting Set, Minimum Set Cover, Genetic Algoritms, Approximation Algorithms, NP-complete Problems.

1 Introduction

The Minimum Hitting Set is one of the better known combinatorial problems which belong to the class of NP-complete problems. Therefore, the problem is very unlikely to have a polynomial solution and, in turn, it is worthwhile to explore heuristics that might give near-optimal solutions. We propose here a genetic algorithm for the problem, which contains two novel ideas:

- viruses (see [10]) and anti-viruses which co-evolve with the system, and, to some extent, play the role of mutation; and
- dynamic fitness evaluation, i.e. the way we evaluate the fitness of all the individuals change as soon as a specific event has happened. In our case, that event is the presence in the population of a *hitting* set.

In the following, we will give a detailed description of our algorithm and, subsequentely, we will show its performance on a series of test cases of different dimensions. We will now give for completeness the definition of the Minimum Hitting Set Problem (MHSP).

1.1 The Minimum Hitting Set Problem

The formal definition of the MHSP is the following

- **Instance**: a collection $\mathcal{C}$ of subsets of a finite set S.
- **Solution**: a hitting set for $\mathcal{C}$, i.e. a subsets $S' \subseteq S$ such that S' intersects each and every element of $\mathcal{C}$.
- **Objective Function**: the cardinality $|S'|$ of S'
- **Optimal Solution**: a hitting set S' that minimizes $|S'|$.such that S'

It is worthwhile to mention another NP-complete problem, Minimum Set Cover Problem (see [4, 7]), which is completely equivalent to the above described MHSP. Such a problem, which has been more widely studied, presents good approximation results (see [6, 8]), within $1 + \log|S|$ (see [9, 5]) of the optimal solution. Such results carry over the MHSP.

2 Formalizing our Genetic Algorithm

Given a finite set S of cardinality n, we can think of it as the set $\{0, 2, \ldots, n-1\}$. Therefore, any subset $T \subseteq S$ can be seen as a binary array of length n such that $T[i] = 1$ if and only if $i \in T$. In turn, $\mathcal{C}$ is a finite collection of binary arrays all of length n.

A hitting set is also a subset of S and therefore can also be represented as a binary array of length n.

Thus, it is very natural to formalize the MHSP in the classical terms of a genetic algorithm search.

2.1 Fitness

We need to come up with a good definition of fitness for a tentative solution. We have two parameters to take into considerations: first the tentative solution must intersect all the members of $\mathcal{C}$; second, it must be of minimal cardinality. Any evolutionary approach which just takes into consideration the first paramenter, will tend to produce populations of individuals *very rich* of ones. Thus, not minimal. If instead, we take into consideration just the second parameter, we will have populations of individuals full of zeroes, and thus not hitting set. Any fitness function therefore must be a two-variable function of type $f(\rho, \alpha)$ where ρ is the number of sets hit by T, and α is the number of zeroes of T. Also, f must be increasing on both parameters, i.e.

$$\begin{aligned} f(\rho,\alpha) &> f(\rho,\alpha') \text{ iff } \alpha > \alpha' \\ f(\rho,\alpha) &> f(\rho',\alpha) \text{ iff } \rho > \rho' \end{aligned}$$

Among the infinite possible choices, we selected the following two functions:

$$f(\rho,\alpha) = \rho \times n + \alpha \quad (1)$$

$$f(\rho,\alpha) = \rho + \alpha \quad (2)$$

with $n = |S|$. The reasons for our choices are the following:

- although definition (1) takes into account the number of zeroes of a particular individual, it gives a lot more weight to the number of sets of $\mathcal{C}$ that the individual intersects. Indeed, as it is easy to see, given two individuals c_1 and c_2 if c_1 hits more sets than c_2 it will have a higher fitness value no matter what the number of zeroes in the two individuals are. With this choice of fitness function, evolution quickly produces populations of individuals with high values of ρ.
- Once ρ has reached the maximal possible value $m = |\mathcal{C}|$, evolution should favour individuals with high α values and $\rho = m$ value. To this end, definition (2) seems to be a very good choice.

Therefore our GA will have a 2-phase fitness function. In the first phase, when no solution has been found yet, the fitness function (1) is used. As soon as a solution is found, the fitness function (2) is used for the whole population.

2.2 Genetic richness

To guarantee genetic variety in the population, we chose not to use a mutation operator. Instead, we decided to associate to each individual, an extra binary string of $2 + \log n$ bits. We called these bits *a virus*. Viruses act differently in the two phases (solution not found, solution found) above mentioned. In the first phase, they tend to weaken individuals by increasing the number of 1 (α decreases) but at the same time they are helping them in becoming solutions). In the second phase, they tend to weaken individuals by increasing the number of 0 (ρ may decrease) but at the same time α increases. The reason for the $2 + \log n$ bits is the following:

- any string of $\log n$ bits uniquely identify a number between 0 and $n - 1$ and therefore, uniquely identifies a position within the individual string (a locus). If the virus hits the individual, that bit will be put to 1 (in the first phase) or to 0 in the second phase.
- The virus hits the individual if the remaining two bits, called control bits, are both 1. As is the case for many diseases in nature, individuals can be divided into 3 groups: healthy (control bits are both 0); disease-carrier (one and only one of the control bits is 1); sick (control bits are both 1). Any individual, even an healthy one, carries with itself a virus, inherited along with its genetic patrimony, from its parents. Two diseas-carrier individuals which are chosen for crossover, will produce a sick child with 1/4 probability.

2.3 Crossover and other parameters

We used the uniform crossover method. As a selection method we used the tournament selection (see

[1]). A selected individual will mate with probability 1. No mutation is introduced and we used a mild form of elitism: the best fitness individual, and the best candidate solution (the two migth be different) are carried onto the next generation. A population is formed by 200 individuals.

3 Experimental results

We decided to test our algorithm on a variety of test cases. We generated randomly four different sets of test cases: denoted by f, g, h, j, which will be described below. For each test set we ran our genetic algorithm on nine different test cases, generated with cardinalities 100, 150 and 200 for S, and different values for the cardinality of $\mathcal{C}$. Each test set is characterized by how the sets in $\mathcal{C}$ are generated randomly. To randomly generate a subset of S we acted as follows:

- We fixed two integer interval parameters $[a_1, \ldots, a_2]$ and $[b_1, \ldots, b_2]$.
- To assign a random value to an individual bit, we draw randomly two numbers $a' \in [a_1, \ldots, a_2]$ and $b' \in [b_1, \ldots, b_2]$. If $a' < b'$ the bit is given the value 1 otherwise is given the value 0.

The test sets are characterized by different choices for the integer interval parameters. We chose the integer interval parameters so the expected cardinalities of the minimum hitting sets are high for the test set g, medium for the test set f and low for the test set h. Below we will define the different test sets and then, we will show the results of our experiment using tables with 7 columns:

Column A gives the test case name;

Column B gives the cardinality of S;

Column C gives the cardinality of $\mathcal{C}$

Column D gives the total number of virus hits

Column E gives the generation number where the best solution is found

Column F gives the cardinality of the best hitting set found

Column G since a genetic algorithm is ran three times on each test case, gives how many times the best solution is found

A	B	C	D	E	F	G
f_0	100	20000	6457	125	33	1/3
f_1	100	30000	8095	102	38	1/3
f_2	100	50000	6845	127	42	1/3
f_3	150	80000	12597	200	46	1/3
f_4	150	90000	9392	142	48	1/3
f_5	150	100000	6752	142	50	2/3
f_6	200	150000	15855	188	51	1/3
f_7	200	180000	10989	184	55	1/3
f_8	200	200000	11909	161	55	1/3

Table 1: Results for the test set f

A	B	C	D	E	F	G
g_0	100	20000	6115	108	37	2/3
g_1	100	30000	7845	122	39	2/3
g_2	100	50000	8143	153	43	1/3
g_3	150	80000	10762	159	49	1/3
g_4	150	90000	12392	148	50	3/3
g_5	150	100000	10185	158	51	1/3
g_6	200	150000	15918	187	54	1/3
g_7	200	180000	17442	186	57	1/3
g_8	200	200000	10021	177	57	2/3

Table 2: Results for the test set g

3.1 Test set f

The test set f is characterized by the intervals $[0, \ldots, 9]$ and $[1, \ldots, 10]$. What is the probability that $a' < b'$? The event space is made of the 100 possible pairs of values $[a', b']$. Of these 100 pairs, the ones for which $a' < b'$ are exactly 55. Therefore with probability $\frac{55}{100}$ a bit is set to 1. Any set in $\mathcal{C}$ will therefore have a little ofver one half of its bits equal to 1. Table 1 shows the results obtained for the test set f.

3.2 Test set g

The test set g is characterized by the intervals $[2, \ldots, 9]$ and $[1, \ldots, 10]$. What is the probability that $a' < b'$? Of the 80 pairs of values in the event space, the ones for which $a' < b'$ are exactly 36. Therefore with probability $\frac{9}{20}$ a bit is set to 1. Any set in $\mathcal{C}$ will therefore have a little less than a half of its bits equal to 1. Table 2 shows the results obtained for the test set g.

A	B	C	D	E	F	G
h_0	100	20000	64583	88	14	1/3
h_1	100	30000	7918	118	15	1/3
h_2	100	50000	4297	78	17	3/3
h_3	150	80000	9430	112	18	1/3
h_4	150	90000	3651	83	19	3/3
h_5	150	100000	6483	111	19	2/3
h_6	200	150000	9083	84	20	1/3
h_7	200	180000	8593	102	21	2/3
h_8	200	200000	2435	50	21	3/3

Table 3: Results for the test set h

A	B	C	D	E	F	G
j_0	100	20000	7549	107	32	1/3
j_1	100	30000	6497	101	36	2/3
j_2	100	50000	6341	109	39	2/3
j_3	150	80000	16982	180	43	1/3
j_4	150	90000	10301	150	46	1/3
j_5	150	100000	9337	176	46	1/3
j_6	200	150000	1223	179	50	1/3
j_7	200	180000	15298	175	50	1/3
j_8	200	200000	14754	182	53	1/3

Table 4: Results for the test set j

3.3 Test set h

The test set h is characterized by the intervals $[0,\ldots,7]$ and $[1,\ldots,10]$. What is the probability that $a' < b'$? Of the 80 pairs of values in the event space, the ones for which $a' < b'$ are exactly 52. Therefore with probability $\frac{52}{80}$ a bit is set to 1. Any set in $\mathcal{C}$ will therefore have about two thirds of its bits equal to 1. Table 3 shows the results obtained for the test set h.

3.4 Test set j

Finally, The test set j is generated as f for one third, g for another third, and h for the last third. Table 4 shows the results obtained for the test set j.

4 Summary and Conclusions

We have proposed an evolutionary algorithm to approximate optimal solutions to instances of the Minimum Hitting Set Problem, a well known $\mathcal{NP}$-complete problem. In our approach, we have used the idea of a virus which co-evolves with the individual to which is attached, and a dynamic fitness function to improve performances. In a first phase when viruses infect chromosomes they tend to weaken individuals by increasing the number of 1, but in doing so they make them stronger in terms of being candidate solutions. When a candidate solution has been found, viruses tend to increase the number of 0. In doing so, candidate solutions may no longer be such, but also it migth be the case that candidate solutions remain so with a lower number of 1. The choice of dynamic two-phase fitness function provided better overall performances. The obtained positive results in a long series of experiments, prove that such ideas are worthwhile exploring in greater depth for such kind of problems.

References

[1] Golberg, D. E., and Deb, A comparative analysis of selection schemes used in genetic algorithms, *In G. Rawlins, ed., Foundations of Genetic Algorithms*, Morgan Kaufmann, 1991.

[2] M. Mitchell, *An Introduction to Genetic Algorithm,* A Bradford Book, The MIT Press, 1996.

[3] Garey M. R., and Johnson, D. S., *Computers and Intractability: A guide to the theory of NP-completeness,* W. H. Freeman and Company, San Francisco, 1979.

[4] Ausiello, G., D'Atri, A., and Protasi, M. "Structure preserving reduction among convex optimization problems," *J. Comput. System Sci.*, Vol. 21, pp. 136-153, 1980.

[5] Zuckerman, D., "NP-complete problems have a version that's hard to approximate," *Proc. Eight Ann. Structure in Complexity Theory Conf. IEEE Computer Society*, pp. 305-312, 1993.

[6] Hassin, R., and Megiddo, N., "Approximation algorithms for hitting objects with straight lines," *Disc. Appl. Math.*, Vol. 30, pp. 29-42, 1991.

[7] Srimivasan, A., Improved approximations of packing and covering problems, *Proc. 27th Ann. ACM Symp. on Theory of Comp.*, pp. 268-276, 1995.

[8] Feige U., A threshold of $\log n$ for approximating set cover, *J. ACM*, Vol. 45, pp. 634-652, 1998.

[9] Johnson, D. S., Approximation algorithms for combinatorial problems, *J. Comput. System Sci. 9*, pp. 256-278, 1974.

[10] Susumy Saito and Tooshi Sako, A Genetic Algorithm By Use Of Virus Evolutionary Theory For Combinatorial Problems, *http://www.ise.nus.edu.sg/proceedings/apors2000/fullpapers/02-04.htm*, Science University of Tokyo School of Management Kukishi 346-8512, Japan.