Integer Programming With Genetic Algorithms

Dr. Tang Van To (tvto@s-t.au.ac.th)
Kamal Kumar Dua (kkd@preciousshipping.com)
Faculty of Science & Technology
Assumption University Bangkok, Thailand

ABSTRACT

Integer programming is a NP hard problem[1]. Genetic Algorithms is relatively newer field, which has proven useful in function optimizations. However there has little research been done to use GA for Integer Programming problems. We have tried to use GA for solving Integer Programming problems by modifying traditional GA. In our method simplex method is used to find some vertex of feasible region and most of initial population is chosen with help of these vertex coordinates. We have also used a variable mutation probability.

KEYWORDS: Genetic Algorithms, Integer Programming, Linear Programming, and Knapsack problem

1. Introduction

A Linear Programming (LP) is a problem that can be expressed as follows (the so-called Standard Form): [7]

minimize or maximize cx
subject to Ax = b
x >= 0

Where x is the vector of variables to be solved for, A is a matrix of known coefficients, and c and b are vectors of known coefficients. The expression "cx" is called the objective function, and the equations "Ax=b" are called the constraints.

LP problem in which all variables are required to be integer is called *integer-programming [IP] problem*. The case where the integer variables are restricted to be 0 or 1 is called *0-1 knapsack problems*.

There are two common approaches to solving Integer programming problems. Historically, the first method developed was based on cutting planes (adding constraints to force integrality). However, the most effective technique is *branch and bound.*[2]. Over the last decade, genetic algorithms (GAs) have been extensively used as search and optimization tools in various problem domains, including sciences, commerce, and engineering. The primary reasons for their success are their broad applicability, ease of use, and global perspective. John Holland of the University of Michigan, Ann Arbor, first conceived the concept of a genetic algorithms [3].

Holland's genetic algorithm attempts to simulate nature in the following manner. The first step is to represent a legal solution to the problem you are solving by a string of genes that can take on some value from a specified finite range or alphabet. This string of genes, which represents a solution, is known as a chromosome. Then an initial population of legal chromosomes is constructed at random. At each generation, the fitness of each chromosome in the population is measured. The fitter chromosomes are then selected to produce offspring for the next generation, which inherit the best characteristics of both the parents. After many generations of selection for the fitter chromosomes, the result is hopefully a population that is substantially fitter than the original.

2. Related work:

GA are often considered for search problems in which the solution space is described by non continuous or multimodal functions [4]. Some research have been done on IP using GA but It has been on use of some specific real world problem. Moreover GA has been modified to suit on particular problem and is not generic. Jaffrey et al. [4] used GA to solve IP problem for cell designing manufacturing industry. They used modified chromosome representation for Integer variables and also used new GA operators such as swap crossover and multipoint crossover to improve the results. This was successfully implemented in the field of cell designing manufacturing industry.

Deb et al [5]used binary coded and real coded GA to solve Mixed Integer Programming (MIP) problem for scheduling of casting industry. This problem involved a large numbers of variables . However results were not comparable to other

Linear programming (LP) methods. Deb et al. then modified the GA to a problem specific approach and results obtained were much better. Key feature used here was new problem specific recombination operator. However this problem is of MIP problem.

Andrea Toniolo [6] used a hybrid approach thus by combining LP and GA by which a set of solution is first obtained by LP and an optimized solution is searched using GA. Above approach is also relate to particular Industry and is of MIP type.

Another approach as suggested by Gunther [7] for multiconstrained problem is similar to our approach where pre optimization of initial population is suggested. Further a local improvement and a repair operator is suggested. This algorithm claims to work better with Knapsack problems but can not be used for IP which contain variable values other than 0 and 1.

In his paper Yury Potroinu [8] mentioned that GA can be used for solving IP problems but has no details as how this can be achieved except applying traditional approach.

For many years principal technique used in practice to solve MIP problems remains unchanged. Linear Programming based Branch and bound (LPBB) method has been most popular which was introduced by Land and Doig in 1960 [2]. Many computational experiments in the past twenty years have revealed that LPBB algorithms with cuts added are currently the most successful tool for attacking combinatorial and IP without any given structure. Such algorithms are known as branch-and-cut and may be interpreted as dual type methods.

Current implementations of branch-and-cut algorithms heavily rely on the polyhedral theory for designing the cutting phase of such an algorithm [9]. Utz-Uwe Haus[9] used a technique which involves lattice operations only and, hence, explicitly relies on the discrete structure of the feasible set. There are many publications and papers which show various methods to solve IP. The one which was found most comprehensive and useful is book by Robert J. Vanderbei [10].

3. Proposed Algorithms

Our proposed algorithm have been a combination of simplex method and genetic algorithm. Range for initial population is calculated with simplex algorithm. Binary representation of genes have been choosen so Integer results can obtained easily. In our approach constraints are handled using parameter less approach as defined by Deb[11].

- **Initial Population**

We have used Simplex method to find the some vertices of feasible region . With help of these few vertices, range for initial population is computed. This method helps us to find a better range for each variable. After applying simplex method to get some hints about the ranges of each variable we choose 75% of initial population within this ranges and rest 25% of population is generated randomly.

- **Selection**

The modified Tournament selection method, which works as follows:

- Shuffle the population.
- Pickup two genes at random
- If both are feasible genes one with higher fitness is selected
- If both are non feasible then one with lower penalty is selected
- If one is feasible and one is not then feasible one is selected
- Second parent is also selected same way.

Further if both selected parents are not feasible then one with higher penalty is dropped and instead one with best fitness is selected. Best gene is always carried to next generation.

- **Crossover**

We used the single point crossover on whole chromosome string for knapsack problems and single point crossover on each variable for IP problems. Reason for using crossover on each variable for IP problem is that if we only use crosso ver on whole string then each variable may not come to optimal value as in many cases it may never get a chance to reach to optimal value. While generating new population best of two parents and best of two children produced is included in the new generation.

- **Mutation**

Bitwise mutation is done. Since bitwise mutation can change many chromosome in a gene hence a mutation probability is very low say 0.001 was used initially. But this low mutation probability had one disadvantage of forming local cluster and no improvements in objective functions were noticed if GA get stuck in local optima.

Following mutation approach was adopted which proved good. In this case evolution is started with a low mutation probability which increases after each generation till first one forth of total generations are reached. After this mutation probability starts reducing and becomes original at end of first half of run. During second half of evolution same pattern is followed by mutation probability. This ensures that in case when local optimal is reached or about to be reached then

slightly high mutation probability diversifies the population hence breaking apart from a local optimal solution.

- **New population**

Best fitness gene(of course one which is feasible) is always carried to next generation.

- **Termination**

The GA is terminated when specified number of generations are completed.

4. Experimental Results

Using above approaches we have tested our algorithm on Integer programming problems obtained from public domain[12]. These test data files are for which are for multi constrained Knapsack problems. For IP problems no other test data files were available hence above data files are used for IP problems as well. Results obtained were compared with LP_SOLVE[13] program which solves LP, Integer programming and knapsack problems. Due to limitation of randomness of GA our algorithm was used three times and best of these three results are chosen as results.
Some of results obtained are listed below in tables.

- **Integer Programming Results:**

Table1: Variables: 30 Constraints:5, Generations: 100, population Size: 1000

File	Results Found in Gen #	Best Results out of 3 runs	Results by LP_SOLVE	Variation	Average Variation
weish01.dat	9	8836	8836	0.0%	
weish02.dat	10	10228	10356	1.2%	
weish03.dat	11	11034	11034	0.0%	0.24%
weish04.dat	21	7350	7350	0.0%	
weish05.dat	20	7083	7085	0.0%	

Table2: Variables: 40 Constraints:5, Generations: 100, population Size: 1000

File	Results Found in Gen #	Best Results out of 3 runs	Results by LP_SOLVE	Variation	Average Variation
weish06.dat	7	12936	13206	2.0%	
weish07.dat	15	12524	12524	0.0%	
weish08.dat	14	13688	13972	2.0%	1.02%
weish09.dat	27	12590	12590	0.0%	

Table3: Variables: 50 Constraints:5, Generations: 100, population Size: 1000

File	Results Found in Gen #	Best Results out of 3 runs	Results by LP_SOLVE	Variation	Average Variation
weish10.dat			NR		
weish11.dat	12	17180	17230	0.3%	
weish12.dat			NR		0.15%
weish13.dat	16	20008	20008	0.0%	

NR: No results found by LP_SOLVE in 12 hours on Intel PII 300 Mhz with 192MB RAM & Windows 98.

Table4: Variables: 60 Constraints:5, Generations: 100, population Size: 1000

File	Results Found in Gen #	Best Results out of 3 runs	Results by LP_SOLVE	Variation	Average Variation
weish14.dat	20	24020	24158	0.6%	
weish15.dat	30	17312	17312	0.0%	
weish16.dat	33	20589	20945	1.7%	0.65%
weish17.dat	20	36924	37024	0.3%	

Table5: Variables: 70 Constraints:5, Generations: 100, population Size: 1000

File	Results Found in Gen #	Best Results out of 3 runs	Results by LP_SOLVE	Variation	Average Variation
weish18.dat	29	32186	32384	0.6%	
weish19.dat	23	27456	28022	2.0%	
weish20.dat	22	25612	26262	2.5%	2.20%
weish21.dat	31	24406	25340	3.7%	

Table6: Variables: 80 Constraints:5, Generations: 100, population Size: 1000

File	Results Found in Gen #	Best Results out of 3 runs	Results by LP_SOLVE	Variation	Average Variation
weish22.dat	25	33956	33956	0.0%	
weish23.dat	38	32294	32294	0.0%	
weish24.dat	37	36460	37232	2.1%	1.31%
weish25.dat	27	29220	30174	3.2%	

Table7: Variables: 90 Constraints:5, Generations: 100, population Size: 1000

File	Results Found in Gen #	Best Results out of 3 runs	Results by LP_SOLVE	Variation	Average Variation
weish26.dat	10	37338	37546	0.6%	
weish27.dat	20	37786	38164	1.0%	
weish28.dat	23	36910	37114	0.5%	0.91%
weish29.dat	33	37444	37444	0.0%	
weish30.dat	41	46841	48024	2.5%	

We grouped test problems according to number of variables best results were obtained by running our method 3 times with varying populations of 100, 200, 500 and 1000. Optimal results were obtained using LP_SOLVE program. Average errors are compared for all test problems and plotted on graph. Following figures show average variations from optimal values for some test cases.

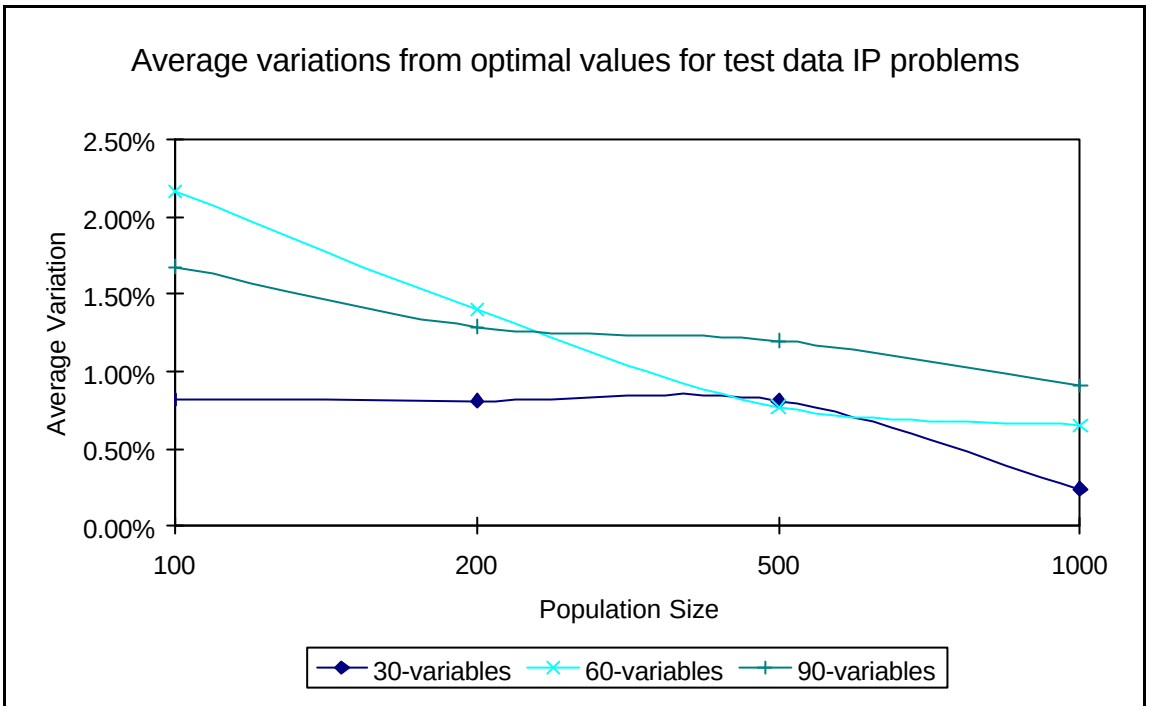

Fig 1: Average variations from optimal values for test data IP problems

As can be seen from Fig 1 that average error is getting smaller when the population size is increased. On the other hand error is generally more in case number of variables are increased in the problem. Again due to randomness of GA this phenomena can not be guaranteed.

Average variation from optimal values for knapsack problems is very lower than that of IP problems because for knapsack problems each variable can have only two possible values i.e. 0 or 1. In other words we exactly know the variable range. Even for knapsack problems crossover is performed on whole chromosome strings and not on each variable due to only two values involved for each variable, which makes solving knapsack problem faster than IP problems.

5. Limitations

As no system can be considered perfect our proposed GA method also have limitations. Among these some are inherited from GA itself. These are briefly summarized as follows:

Working of GA depends on system generated random numbers hence results obtained for a problem on two different occasions may not be same. Best results can not be guaranteed by running our method at single time hence best results need to be chosen by running our GA for more than one time on same problem. Other known methods for solving IP problems may produce better results. Results obtained greatly depend on choosing right values of population size, maximum number of generations, and Crossover and mutation probability along with right value of mutation variable constant.

6. Conclusion

Genetic Algorithms is relatively newer field, which has proven useful in function optimizations. However there has little research been done to use GA for Integer Programming problems. We have tried to use GA for solving Integer Programming problems as discussed above.

To conclude it is necessary to emphasize that good results are obtained by our research , although accurate and consistent results can not be guaranteed by our me thod due to limitations of randomness of GA. Nevertheless IP with GA can be very useful in obtaining results where no results can be obtained by other known methods in given frame of time. This has been demonstrated that in our test data weish10.dat and weish12.dat each with only 50 variables could not be solved in 12 hours on Intel PII 300 Mhz with 192MB RAM & Windows 98 by LP_SOLVE program. But with GA results can be found using same amount of time as with other 50 variable test problems. Hence IP with GA can be very useful in areas where accuracy is not needed and approximate results are useful.

Hence there is a clear advantage of using GA for IP problems where results can not be obtained in polynomial time with other known methods.

We have used different approaches to solve IP with GA. It has been observed that better results still can be obtained if we can initialize all initial population with in feasible region. We have used simplex method to find few co-ordinates of feasible region and generate initial population somewhat within feasible region. There is a clear scope for research where one can initialize the population with in feasible region so that better and faster results can be obtained.

7. References:

[1] Yuh-Chyun Luo et al., [2001] An Efficient Approach Integrating Genetic Algorithm, Linear Programming, and Ordinal Optimization for Linear Mixed Integer Programming Problems, Smart Engineering System Design, vol. 00, pp1-12.

[2] Robert E. Bixby et al.,[2000] MIP Theory and practice closing the Gap.Department of Computational and Applied Mathametics, Rice University , TX.

[3] Kalyanmoy Deb,[1999] An Introduction to Genetic Algorithms , Technical Paper, Kanpur Genetic Algorithms Laboratory (KanGAL).

[4] Jeffrey A. Joines et al.,[1996] Manufacturing Cell Design; An Integer Programming Model Employing Genetic Algorithms , , Department of Industrial Engineering, North Carolina State University.

[5] Deb Kalyanmoy et al.,[2002] Optimal Scheduling of Casting Sequence Using Genetic Algorithms , KanGal report No. 2002002 , IIT Kanpur.

[6] Andrea Toniolo Staggemeir,[2002] A hybrid genetic algorithm to solve a lot-sizing and scheduling problem, Internal Research Report MS-2002-4, Intelligent Computer Systems Centre Group, School of Mathematical Sciences University of West of England.

[7] Gunther R. Raidl, [1998] An Improved Genetic Algorithm for the Multiconstrained 0-1 Knapsack Problem, Proceedings of the 5th IEEE International Conference on Evolutionary Computation, pages 207-211. IEEE Press.

[8] Yuri Potroinu,[2001] Usage of genetic Algorithms in Solving tasks of Integer programming, Department of Information Technologies, State Economic University, Belarus.

[9] Utz-Uwe Haus, et al.,[2001] The Integral Basis Method for Integer Programming, Otto-von-Guericke-Universit.at Magdeburg, Department of Mathematics/IMO, Universit.atsplatz 2, 39106 Magdeburg, Germany.

[10] Robert J. Vanderbei,[1998] Linear Programming: Foundations and Extensions Second Edition, Kluwer Academic Publishers.

[11] Deb, K. [2000]. An Efficient Constraint Handling Method for Genetic Algorithms . Computer Methods in Applied Mechanics and Engineering.

[12] http://ftp.zib.de/mp-testdata/ip/sac94-suite/index.html

[13] http://www.cs.sunysb.edu/~algorith/implement/lpsolve/implement.shtml