A Simplex-Genetic method for solving the Klee-Minty cube

JUAN FRAUSTO-SOLIS, RAFAEL RIVERA-LOPEZ
Department of Computer Science
ITESM, Campus Cuernavaca
Av. Paseo de la Reforma 182-A, 62289, Temixco, Morelos
MEXICO

Abstract: - Although the Simplex Method (SM) developed for Dantzig is efficient for solving many linear programming problems (LPs), there are constructions of hard linear programs as the Klee-Minty cubes and another deformed products, where this method has an exponential behavior. This work presents the integration of genetic algorithms (GA) and SM to fastly reach the optimum of this type of problems. This integration, called Simplex Genetic Method (SGM), applies first a GA to find a solution near the optimum and afterwards uses the SM to reach the optimum in a few steps. In the GA phase, the populations are constructed by only basic LP solutions codified as binary chromosomes and the crossover operator uses a tabu approach over infeasible solutions to produce the new offsprings. Based in this binary representation, a translation schema is used to transfer the GA solution as the initial solution of the Simplex search mechanism, avoiding that the SM realizes many iterations and reducing the optimum search time. In this work, several instances of the Klee-Minty cube are evaluated and compared with the traditional SM and the results suggest that for hard linear problems the SGM has better behavior that the SM.

Key-Words: - Genetic algorithms, linear programming, Simplex method, optimization.

1 Introduction

The development in 1947 of the Simplex Method (SM) for Linear Programming problems (LPs) marks the start of the modern era in optimization [1]; many implementations of SM have been applied to real world problems ([2], [3]), and many researchers have examined closely the SM characteristics. SM is still used in most practical problems ([4], [5], [6], [7], [8]), although another methods are attractive to many researchers (interior point methods, for instance). SM has many improvements (see [9]), and several pivot rules have been proposed [10]. Although the SM finds an optimum solution in short time for many practical problems [11], for LPs such as the Klee-Minty cube (KM_d) and another deformed products [12] the deterministic pivot rules have an exponential behavior. A d-dimensional KM_d, for instance, requires 2^d simplex iterations [13].

Genetic Algorithms (GAs) are considered as a global search method based on natural genetics [14], an adaptation process applied over binary strings using both crossover and mutation operators [15]. GAs have been successfully applied in many complex problems ([16], [17], [18] and [19]) and there are a lot of forums exist to exchange ideas and collaborations between researchers (see [20]). However, GAs are unconstrained optimization procedures, and therefore is necessary to develop techniques of incorporating the constraints (normally existing in any real-world application) into the fitness function [21]. The central problem for applying GAs to the constrained optimization is how to handle constraints because genetic operators used to manipulate the chromosomes often yield infeasible solutions [20]. So, several techniques and approaches to handle constraints ([20]-[32]) have been proposed. Among them, they are several works showing that the hybrid genetic approach ([29]-[32]) is a good option to solve hard optimization problems in an efficient way.

In this paper a Simplex Genetic Method (SGM) that uses both a Simplex approach and genetic operators in the search procedure for reaching the optimum of a LP is developed. Genetic operators can be applied because the basic solutions are codified as binary strings. In order to avoid the creation of many infeasible offsprings, the crossover operator uses historical information represented by a tabu list within the grade of infeasibility of each variable on the solutions to create new individuals into the evolution process. This tabu approach, together with a handling technique, improves the generation of chromosomes representing basic solutions. When the genetic procedure is not improving its better solution, this solution is transferred to SM that applies its simplex machine to find the optimum in few iterations. This transfer is based in the binary representation of the solutions. The better GA

solution represents the initial vertex of the simplex search mechanism, avoiding that the SM realizes many iterations and reducing the optimum search time.

2 Preliminaries

A LP method finds a solution **x** for $\{z=\mathbf{c}^T\mathbf{x}: \mathbf{Ax}\leq\mathbf{b}\}$. A feasible solution is a basic solution that is one vertex of the feasible region, a polytope delimited by the constraints. SM exploits the combinatorial structure of the LP to find the optimum and this is the reason why the LP is also catalogued as a combinatorial optimization problem ([33], [34]).

2.1 Simplex Method

SM searches along the edges of the feasible region, moving from one basic feasible solution (BFS) to another adjacent BFS. For the convexity properties of this region, if an LPs has an optimal solution, then one of these BFSs is the optimum. SM has two phases: (I) to obtain an initial solution, and (II) to construct one tableau that identifies the basic and non-basic variables, and to apply some pivot rule for choosing both the entering and the leaving variables of the actual BFS. SM constructs a sequence of tableaux, until the optimum is reached. Revised Simplex Method (RSM) uses a matrix representation for a LP and produces the sequence actualizing at each iteration only a sub-matrix **B** of **A**. Many pivot rules ([10], [35]), schemes for processing **B** [35] and another techniques ([36], [37], [38]) have been developed for improving SM. Klee and Minty showed in [13] that SM has an exponential behavior for KM_d. KM_d represented in (1) is a deformed product of one d-cube. The figure 1 shows a 3-dimensional KM_d using $\varepsilon < {}^1/_3$. KM_d has an exponential behavior for any deterministic pivot rule ([33], [10], [12]).

$$\begin{gathered}\max x_d \\ \text{s.t. } 0\leq x_1\leq 1;\ \varepsilon x_{j-1}\leq x_j\leq 1-\varepsilon x_{j-1} \\ \text{for } j{:}1,\ldots,d \text{ and } 0<\varepsilon<{}^1/_2\end{gathered} \qquad (1)$$

2.2 Genetic Algorithms

GAs, differ from conventional search techniques, as they start with an initial set of random solutions called populations [20]. Michalewicz in [39] resumes the five components of a genetic algorithm.

The central problem for applying GAs in constrained optimization is how to handle constraints because the genetic operators used to manipulate chromosomes often yield infeasible solutions [20]. Many approaches have been proposed for handling solutions that violate one or more constraints, but no one general method has emerged [40]. The existing techniques can be roughly classified as rejecting, repairing, modifying genetic operator and penalty strategies.

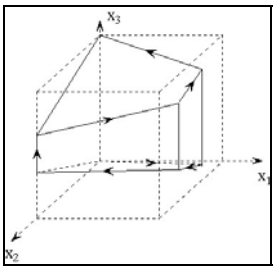

Fig. 1. The Klee-Minty cube for n=3 and $\varepsilon < 1/3$.

Penalty strategies transform the constrained problem into an unconstrained problem by penalized infeasible solutions. The penalty term $p(\mathbf{x})$ can be in addition form, as in (2), or in product form, as is showed in (3). In [21] is provided a survey of techniques to handle constraints in GA.

$$fitness(\mathbf{x})=f(\mathbf{x})+p(\mathbf{x}) \qquad (2)$$
$$fitness(\mathbf{x})=f(\mathbf{x})p(\mathbf{x}) \qquad (3)$$

3 Hybrid approach

A hybrid approach combines GAs with another techniques to develop hybrid genetic algorithms ([29], [30], [31], [32]). A hybrid system aims to take advantage of GAs (to reach fast a solution near to the optimum) and then to apply another search procedure (SM method, for instance) to carry out fine search and to find the optimum. In this paper, this hybrid approach is used: GA is applied for reaching a good solution, and this solution is refined using the RSM.

3.1 Revised Simplex Method

RSM uses a sub-matrix **B** of **A** in its search procedure. Using *max* $\{z=\mathbf{c}^T\mathbf{x}: \mathbf{Ax}\leq\mathbf{b}\}$ and the concepts of basic and non-basic variables, the formulas (4) and (5) can be derivates. Formula (4) represents the LP using basic and non-basic variables, and (5) shows z using only non-basic variables (that is useful for the iteration process).

$$\max \{ z = \mathbf{C_B x_B} + \mathbf{C_{NB} x_{NB}} : [\ \mathbf{B} \mid \mathbf{N}\ ]\ \mathbf{x} = \mathbf{b}\} \qquad (4)$$
$$z = \mathbf{C_B B^{-1} b} - (\mathbf{C_B B^{-1} N} - \mathbf{C_{NB}})\ \mathbf{x_{NB}} \qquad (5)$$

In (4) and (5), $\mathbf{C_B}$ and $\mathbf{C_{NB}}$ are the cost coefficients of basic $\mathbf{x_B}$ and non-basic $\mathbf{x_{NB}}$ variables, respectively, and **N** and **B** are sub-matrices of **A**. **N** has the initial non-basic columns and **B** has the basic columns. $\mathbf{B}^{-1}$ is the only element actualized in the process and both

optimality and feasibility conditions are used. Optimality condition uses the second term of (5) and it tells that the entering variable (x_k) is the non-basic variable with a maximum contribution to the objective function. This occurs when ($\mathbf{C_B B^{-1} A_k} - c_k$)<0. If all contributions are positives, then the optimum is reached and the method finishes [41]. The feasibility condition uses the formula (6) to determine the leaving variable (x_j) for the next solution. x_j should be minimal and positive, otherwise, a solution does not exist. Although there are several implementations of RSM [9], the number of visited vertices is always the same, and so their behavior in the KM_d is exponential for all of them. This is because the progress of RSM with these implementations only improves the amount of calculations by iteration, but the number of iterations remains the same.

$$x_j = \min_{i=1}^{m} \left\{ \frac{(\mathrm{B^{-1}b})_i}{(\mathrm{B^{-1}P_k})_i},\ (\mathrm{B^{-1}P_k})_i > 0 \right\} \quad (6)$$

3.2 Simplex Genetic method

The main idea of this paper is to move quickly to an approximate good solution without having to follow adjacent solutions, using GAs, and then reach the optimum with the RSM. From one LP in its standard form, SGM starts with a random population of basic solutions, and new populations are produced applying genetic operators. They are created until GA converges to its best solution (optimal or sub-optimal). This solution (a binary string) is processed to identify the basic and non-basic variables, and to construct the actual matrix $\mathbf{B^{-1}}$. Then RSM applies their simplex procedure. Figure 2 shows the SGM algorithm. Two important aspects in this approach are the string codification and the definition of crossover and mutation operators. These elements must assure that all chromosomes are basic solutions for the LP. In the next subsections, these aspects are described.

3.3 Components of SGM

In the SGM the main components are the schema for solution representation, the construction of the fitness function, the definition of the genetic operators and the transfer scheme from GA to RSM.

3.3.1 Genetic representation of solutions

Each string represents a basic solution of a LP. A LP with n variables m constraints has m basic and n non-basic variables. String length is $m+n$ bits, where a bit 1 represents a basic variable and a bit 0 is a non-basic variable. All strings have m bits 1 and n bits 0. Figure 3 shows two examples of basic solutions for a LP with n=2 and m=4. In this approach, the artificial variables are avoided.

3.3.2 Fitness function

In this implementation, fitness function is an adaptive penalty function based in [21] and showed in (7), where $\Delta b_i(x)$ is the violation for each constraint and $\Delta b_i^{\max}$ is the maximum of violation for constraint i in the population.

$$p(x) = 1 - \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\Delta b_i(x)}{\Delta b_i^{\max}}\right)^{\alpha} \quad (7)$$

procedure *Simplex_Genetic*;
begin
  *{*** Genetic phase ***}*
  iteration← 0;
  Pick randomly an initial population *P(0)*;
  Evaluate the initial population *P(0)*;
  s_0^* ← better of *P(0)*; $f_{\text{better}}{}^{0}$ ←*f*(s_0^*);
  repeat
    iteration← *iteration*+1;
    Recombine *P*(*iteration*-1) to yield *P*(*iteration*);
    Evaluate *P*(*iteration*);
    $s_{\text{iteration}}^*$ ←better of *P*(*iteration*);
    $f_{\text{better}}{}^{\text{iteration}}$ ←*f*($s_{\text{iteration}}^*$);
  until $f_{\text{better}}{}^{\text{iteration-1}} = f_{\text{better}}{}^{\text{iteration}}$;
  *{***Transition phase***}*
  Determine original $\mathbf{B^{-1}}$ ($\mathbf{B^{-1}}_0 = \mathbf{I}$);
  $I_{\text{basic}} \leftarrow \{i | i \in s_{\text{iteration}}^* \wedge (s_{\text{iteration}}^*)_i = 1\}$;
  Construct a matrix **T** of **A** such that $\{\mathbf{T}_j | j \in I_{\text{basic}}\}$;
  Apply *i* row operations in matrix $[\mathbf{T}|\mathbf{B^{-1}}_0]$ such that $\mathbf{T}_i = \mathbf{I}$;
  $\mathbf{B^{-1}}_{\text{better}} \leftarrow \mathbf{B^{-1}}_i$, the right side of matrix $[\mathbf{T}_i|\mathbf{B^{-1}}_i]$;
  Actualize **C** such that $\{c_j = 0 | \mathbf{j} \in I_{\text{basic}}\}$;
  Actualize **b**, $\mathbf{x_B}$ and $\mathbf{x_{NB}}$ with the *i* row operations,;
  *{***RSM phase***}*
  optimality← *feasibility*← **true**;
  while not *optimality*∧*feasibility* **do**
    $\mathbf{R_N} \leftarrow \mathbf{C_B B^{-1} N} - \mathbf{C_{NB}}$;
    if $\mathbf{R_N} > 0$ **then** *optimality*←**true**;
    else begin
      select *k* as the entering variable such that
        $(\mathbf{R_N})_k$ is the minimum;
      select *j* as the leaving variable as in (6);
      if $\mathbf{x_j}$ not exist **then** *feasibility*←**false**;
      else Update $\mathbf{x_B}$, $\mathbf{x_{NB}}$ and $\mathbf{B^{-1}}$;
    end
  end
  if *optimality* **then** $z \leftarrow \mathbf{C_B x_B}$;
end

Fig. 2. The Simplex Genetic Method.

3.3.3 Genetic operators

To create a new generation, a new chromosome is formed by merging two parents from **a** current

generation using a tabu approach for the crossover operator and modifying a chromosome using a mutation operator.

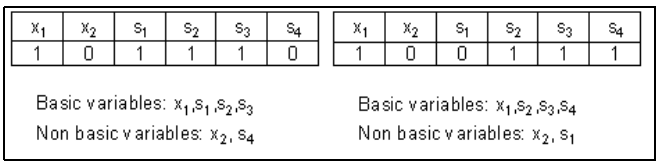

Fig. 3. Two examples of basic solution codified as binary string: bits 1 represent basic variables and bits 0 represent non-basic variables.

Tabu approach for crossover operator. Crossover operator applies two operations: (a) Identical bits in parents are copied into the child in the same site, and (b) empty sites in the child are filled using a tabu list constructed according to the values of variables in infeasible solutions. A tabu list is a set of forbidden moves allocated in a memory function that provide strategic forgetting [42]. In this case, tabu list is constructed by the average of the value of variables in infeasible solutions (variables with negative value). In figure 4, five infeasible solutions, the tabu list produced by them and the construction of a new solution are shown. Figure, 4-(a) shows five strings of infeasible solutions, while 4-(b) shows the values of these five solutions, and 4-(c) the tabu list (for instance, 1.5/5=0.3 is the average of negative values of the solutions). In 4-(d) two parents are shown, and 4-(e) shows that the identical bits in the parents are copied. Using the tabu list, the empty sites in the child are filled in 4-(f); in this example, two zeroes are allocated in the sites where the tabu list has a high value, because this value indicates that these positions have been used by several infeasible solutions. This high value indicates that this variable, in infeasible solutions, is farther from the feasible region, too. This approach allows new solutions to be produced into the feasible region nearer to it. Tabu list is actualized in each generation.

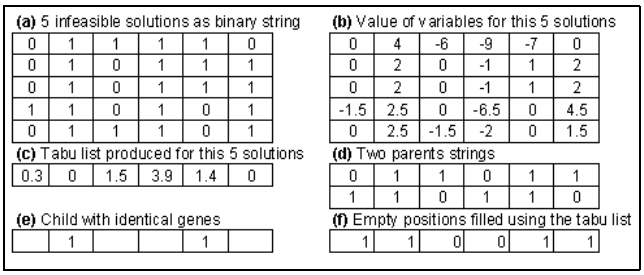

Fig. 4. (a), (b) and (c) shows the construction of tabu list, (d), (e) and (f) shows the use of tabu list for the creation of new chromosome.

Mutation operator. This operator replaces a randomly chosen bit of a string with a different value. Mutation uses a permutation of two bits randomly selected in the string. In figure 5, a string with 4 basic and 3 non-basic variables is selected for applying the mutation. Two sites are selected (1 and 3, in this case), and the bits are permuted.

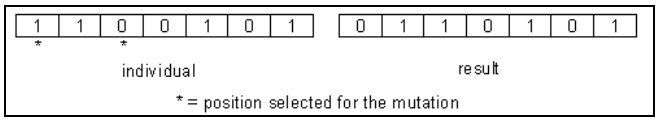

Fig. 5. Mutation operator over a chromosome.

3.3.4 Transfer scheme from GA to RSM (Gauss operator)

The better solution produced by the GA will be the initial solution for the RSM. GA solution (binary string $s_{iteration}^{*}$ in the algorithm shows in figure 2) is transformed as the initial elements for RSM ($\mathbf{B}^{-1}_{better}$, $\mathbf{x}_B$, $\mathbf{x}_{NB}$, $\mathbf{C}$ and $\mathbf{b}$). The original $\mathbf{B}^{-1}_0$ is an identity matrix $\mathbf{I}$ (constructed for the coefficients of the slack variables in the restrictions). Matrix $\mathbf{T}$ is then constructed using several columns of the original matrix $\mathbf{A}$, this selection is based on the binary string: an $\mathbf{A}_j$ column is used in $\mathbf{T}$ if the site in binary string is 1. Matrix $\mathbf{T}$ is the actual $\mathbf{B}_{better}$. Then, reducing $\mathbf{T}$ to $\mathbf{I}$ and applying the same row operations in the original $\mathbf{B}^{-1}_0$, the actual $\mathbf{B}^{-1}_{better}$ is calculated [43]. This is the function of a new operator called Gauss operator. Formulas (8) and (9) show how $\mathbf{B}^{-1}_{better}$ can be calculated using this Gauss operator. Figure 6 shows an example of this operator.

$$\mathbf{B}_{better}\mathbf{x} = \mathbf{I}\mathbf{b} = \mathbf{B}^{-1}_0\mathbf{b} \quad (8)$$

$$\mathbf{x} = \mathbf{B}^{-1}_{better}\mathbf{b} \quad (9)$$

The value of new $\mathbf{C}$ is produced changing the c_j values by zeroes, if j is the index of a new basic variable. Vector $\mathbf{b}$ is actualized applying the same row operations used in the $\mathbf{T}$ reduction process. Finally, vectors $\mathbf{x}_B$ and $\mathbf{x}_{NB}$ are actualized. Gauss operator avoids the direct calculus of $\mathbf{B}^{-1}_{better}$.

Fig. 6. An example of the gauss operator over the values of matrix $\mathbf{B}^{-1}$ and to calculate its new value that is passed to RSM.

4 Results

The principal results obtained with the SGM are shown in figure 7. All tests were executed in an Intel Pentium II, 200 MHz PC. Figure 7 shows an exponential behavior of RSM with the number of variables, and the same for SGM; however the

increased rate in the later is much smaller that the former, even for the worst case. It is clear that SGM has a substantial improvement over RSM for KM_d. For this comparison, SGM was executed for many tests, and we reported the best and the worst case. From the experimental result for the KM_d, it is observed that:

1. RSM has a better behavior only when the number of variables is very small.
2. The population size of SGM affects noticeably its execution time, so it is impractical for a few number of variables.
3. SGM has a better behavior of RSM when the number of variables is bigger.

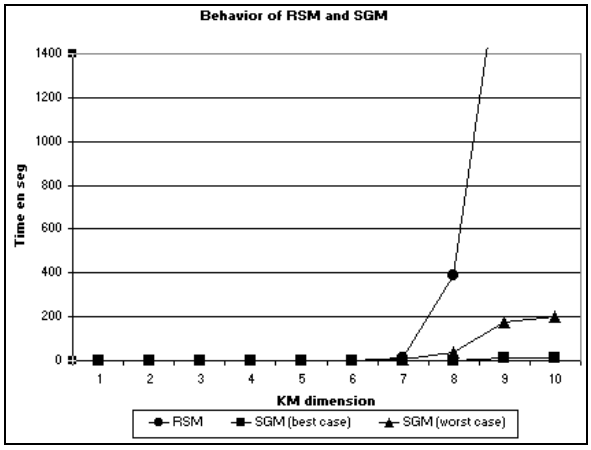

Fig. 7. Behavior of RSM and SGM.

5 Conclusions and future work

This paper presents a hybrid approach for the solution of hard LP (as KM_d), using a GA inside optimum solution search procedure. Both RSM and SGM were implemented and the results suggest that SGM has better behavior that SM for this type of problems. The codification scheme developed for SGM allows representing a basic solution as a string that can be used with a genetic approach. In the SGM is avoided the first phase of the original method, because the genetic phase finds the initial solution for the simplex procedure, and this implies that the artificial variables are not used.

As an extension of this work, concerning the application of genetic operators, an additional effort can be conducted for tuning up the algorithm, finding the values of operators probabilities for improving the SGM performance. Another extension is the study of better procedures to obtain the function objective value.

The comparison of SGM and RSM can be extended including other approach in SGM, as those given by criss-cross methods or random methods for linear programming.

References:

[1] J. Nocedal and S.J. Wright, *Numerical Optimization*, Springer-Verlag, 1999.

[2] C. MacMillan, *Mathematical Programming*, John Wiley, 1970.

[3] A.D. Belegundu and T.R. Chandrupatla. *Optimization, Concepts and Applications in Engineering*, Prentice-Hall, 1999.

[4] M. Sakawa, K. Kato and H. Mohara, Efficient of a decomposition method for large-scale multi-objective fuzzy linear programming problems with block angular structure, in *Proc. IEEE Int. Conf. on Knowledge-Based Intelligent Electronic Systems*, Vol. 1, 1998, pp. 80-86.

[5] K. Murakati and H. S. Kim, Optimal capacity and flow assignment for self-healing ATM networks based on line and end-to-end restoration, in *IEEE/ACM Trans. on Networking*, Vol. 6, No. 2, 1998, pp. 207-221.

[6] Y. H. Liu, Qualitative test and force optimization of 3-d frictional form-closure grasp using linear programming, in *IEEE Trans. on Robotics and Automation*, Vol. 15, No. 1, 1999, pp. 163-173.

[7] S. Nordebo and Z. Zang, Semi-infinite linear programming: a unified approach to digital filter design with time- and frequency-domain specifications, in *IEEE Trans. on Circuits and Systems II*, Vol. 4, No. 6, 1999, pp. 765-775.

[8] K. Yamamura and S. Tanaka, Finding all solutions of piecewise-linear resistive circuits using the dual SM, in *Proc. IEEE Int. Symp. on Circuits and Systems*, Vol. 4, 2000, pp. 165-168.

[9] S. S. Morgan, A Comparison of Simplex Method Algorithms. *Master Thesis, U. Florida*, 1997.

[10] T. Terlaky and S. Zhang, Pivot Rules for Linear Programming, in *Annals of Operations Research*, Vol. 46, 1993, pp. 203-233.

[11] K. H. Borgwardt, *The Simplex Method. A Probabilistic Analysis*, Vol. 1, Springer-Verlag, 1987.

[12] N. Amenta and G. Ziegler, Deformed products and maximal shadows of polytopes, in *Advances in Discrete and Computational Geometry*, Amer. Math. Soc., Vol. 223, 1999, pp. 57-90.

[13] V. Klee and G. J. Minty, How good is the simplex algorithm?, in *Inequalities III*, Academic Press, 1972, pp. 159-175.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems*, U. Michigan Press, 1975.

[16] F. Ramos, V. de la Cueva and S. Hsia, Path Planning using reference configuration and

genetic algorithms, in *Proc. IASTED Int. Conf. in Robotics and Manufacturing*, 1996, pp. 403-406.

[17] V. de la Cueva and F. Ramos, Cooperative Genetic Algorithms: A new approach to solve the path planning problem for cooperative robotic manipulators sharing the same workspace, in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Vol. 1, 1998, pp. 267-272.

[18] F.J. Ares-Peña, J.A. Rodríguez-González, E. Villanueva-López and S.R. Rengerajan, Genetic algorithms in the design and optimization of antenna array patterns, in *IEEE Trans. on Antennas and Propagation*, Vol. 47, No. 3, 1999, pp. 506-510.

[19] J. Pittman and C. A. Murthy, Fitting optimal piecewise linear functions using genetic algorithms, in *IEEE Trans. on Pattern Analysis and Mach. Intelligence*, Vol. 22, No. 7, 2000, pp. 701-718.

[20] M. Gen and R. Cheng. *Genetic Algorithms and Engineering Design*, John Wiley, 2000.

[21] C. A. Coello Coello, A Survey of Constraint Handling Techniques used with Evolutionary Algorithms, *Tech. Report LANIA RI-99-04*, 1999.

[22] M. Schoenauer and S. Xanthakis, Constrained GA Optimization, in *Proc. Int. Conf. on Genetic Algorithms*, 1993, pp. 573-580.

[23] J. A. Joines and Ch. R. Houk, On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's, in *Proc. IEEE Conf. On Evolutionary Computation*, 1994, pp. 579-584.

[24] F. Jimenez and J. L. Verdegay, Evolutionary Techniques for Constrained Optimization Problems, in *European Congress on Intelligent Techniques and Soft Computing*, 1999.

[25] B. W. Wah and Y. X. Chen, Constrained Genetic Algorithms and their Application in Nonlinear Constrained Optimization, in *Proc. IEEE Int. Conf. On Tools with Artificial Intelligence*, 2000, pp. 286-293.

[26] Z. Michalewicz, A Survey of Constraint Handling Techniques in Evolutionary Computation Methods, in *Proc. Annual Conf. on Evolutionary Programming*, 1995, pp. 135-155.

[27] Z. Michalewicz and M. Schoenauer, Evolutionary Algorithms for Constrained Parameter Optimization Problems, in *Evolutionary Computation*, Vol. 4, No. 1, 1996, pp. 1-32.

[28] J-H. Kim and H. Myung, Evolutionary Programming Techniques for Constrained Optimization Problems, in *IEEE Trans. on Evolutionary Computation*, Vol. 1, No. 2, 1997, pp. 129-140.

[29] J. Renders and S. Flasse, Hybrid methods using genetics algorithms for global optimization, in *IEEE Tran. on Systems, Man and Cybernetics*, Vol. 26, No. 2, 1996, pp. 243-258.

[30] A. Ruiz-Andino, L. Araujo, F. Saenz and J. Ruz, A Hybrid Evolutionary Approach for Solving Constrained Optimization Problems over Finite Domains, in *IEEE Trans. on Evolutionary Computation*, Vol. 4, No. 4, 2000, pp. 353-372.

[31] D. Whitley, Modeling Hybrid Genetic Algorithms, in *Genetic Algorithms in Engineering and Computer Science*, John Wiley, pp. 191-201.

[32] D.G. Sotiropoulos, E.C. Stavropoulos and M.N. Vrahatis, A New Hybrid Genetic Algorithm for Global Optimization, in *Nonlinear Analysis, Theory, Methods and Applications*, Elseiver, Vol. 30, No. 7, 1997, pp. 1529-1538.

[33] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Dover, 1998.

[34] B. Korte and J. Vygen, *Combinatorial Optimization*, Springer Verlag, 2000.

[35] I. Maros and G. Mitra, Simplex Algorithms, in *Advances in linear and integer programming*, Oxford University Press, 1996.

[36] K. Fukuda and T. Terlaky, Criss-Cross Methods, in *Mathematical Programming*, Vol. 79, 1997, pp. 369-396.

[37]B. Gärtner and G.Ziegler, Randomized Simplex Algorithms on Klee-Minty Cubes, in *Combinatorica*. 1998.

[38] M. Goldwasser, A Survey of Linear Programming in Randomized Subexponential Time, in *SIGACT News*, Vol. 26, No. 2, pp. 96-104.

[39] Z. Michalewicz, *Genetic Algorithms + Data Structures=Evolution Programs*, Springer-Verlag, 1996.

[40] S. E. Carlson, A General Method for Handling Constraints in Genetic Algorithms, in *Proc. Annual Joint Conf. on Information Science*, 1995, pp. 663-667.

[41] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison Wesley, 1984.

[42] F. Glover, Tabu Search – Part I, in *ORSA Journal on Computing*, Vol. 1, No. 3, 1989, pp. 190-206.

[43] S. I. Grossman, *Linear Algebra*, Grupo Editorial Iberoamericana, 1984.