A COMPARATIVE STUDY ON OPTIMIZATION METHODS FOR THE CONSTRAINED NONLINEAR PROGRAMMING PROBLEMS

OZGUR YENIAY

Received 2 August 2004 and in revised form 12 November 2004

Constrained nonlinear programming problems often arise in many engineering applications. The most well-known optimization methods for solving these problems are sequential quadratic programming methods and generalized reduced gradient methods. This study compares the performance of these methods with the genetic algorithms which gained popularity in recent years due to advantages in speed and robustness. We present a comparative study that is performed on fifteen test problems selected from the literature.

1. Introduction

There are many applications in various branches of engineering field (e.g., mechanical engineering, chemical engineering, electrical engineering, aerospace engineering, etc.) that can be formulated as constrained nonlinear programming problems (NLPs). Typical examples include structural optimization, mechanical design, chemical process control, engineering design, and VLSI design. Quality of the solutions to these applications affects the system performance significantly, resulting in low-cost implementation and maintenance, fast execution, and robust operation [21].

A general constrained nonlinear programming problem (P) can be stated as follows:

(P)

$$\begin{aligned} &\text{Minimize } f(x), \quad x \in F \subseteq S \subseteq \mathbb{R}^n, \\ &\text{subject to} \\ &\qquad h_i(x) = 0, \quad i = 1,\dots,p, \\ &\qquad g_j(x) \le 0, \quad j = p+1,\dots,q, \\ &\qquad a_k \le x_k \le b_k, \quad k = 1,\dots,n, \end{aligned} \tag{1.1}$$

where $x = [x_1,\dots,x_n]$ is a vector of n variables, $f(x)$ is the objective function, $h_i(x)$ $(i = 1,\dots,p)$ is the ith equality constraint, and $g_j(x)$ $(j = p+1,\dots,q;\ q < n)$ is the jth inequality constraint. S is the whole search space and F is the feasible search space. The a_k

Copyright © 2005 Hindawi Publishing Corporation
Mathematical Problems in Engineering 2005:2 (2005) 165–173
DOI: 10.1155/MPE.2005.165

and b_k denote the lower and upper bounds of the variable x_k $(k = 1,\ldots,n)$, respectively. It is assumed that all problem functions $f(x)$, $h_i(x)$, and $g_j(x)$ are twice continuously differentiable. In most of the nonlinear programming problems $f(x)$, $h(x)$, and $g(x)$ are nonconvex and the problems have multiple locally optimal solutions. In the only case where the $f(x)$ is convex, every $h_i(x)$ is linear and every $g_j(x)$ is convex, constrained local minimum is also constrained global minimum.

Although a number of methods for the solution of constrained nonlinear programming problems are available, there is no known method to determine the global minimum with certainty in the general nonlinear programming problem. The methods for constrained optimization can be divided into two categories as deterministic and stochastic methods. According to some comparative studies, the generalized reduced gradient (GRG) methods and the sequential quadratic programming (SQP) methods are two of the best deterministic local optimization methods [8]. These gradient-based methods always look for optimum closest to the starting point whether it is a local or global one. A number of packages, such as Optima, Matlab, GRG, and LSGRG, are based on these widely used methods. In recent years, there has been an increasing interest to employ the stochastic methods, such as genetic algorithms (GA), simulated annealing (SA), and tabu search (TS), in solving complex optimization problems involving even nondifferentiable, discontinuous, highly nonlinear objective, and constraint functions. These methods are stochastic global optimization methods which do not require gradient information unlike GRG and SQP.

In this paper, the performances of SQP and GRG are compared with that of GA, which is the most popular method among the stochastic methods in solving constrained nonlinear programming problems. The experimental study is conducted on several NLPs taken from the literature.

The organization of this paper as follows: we briefly describe sequential quadratic programming methods, generalized reduced gradient methods, and genetic algorithms in Section 2. Section 3 presents the 15 test problems and the optimization results obtained by using SQP, GRG, and GA. The conclusion is drawn in Section 4.

2. Methods used in the study

In this section, the brief summaries of each of the methods, namely sequential quadratic programming, generalized reduced gradient, and genetic algorithms, are given.

2.1. Sequential quadratic programming. SQP methods are iterative methods that solve at the kth iteration a quadratic subproblem (QP) of the form

(QP)

$$\begin{aligned}
&\text{Minimize } \frac{1}{2}d^t H_k d + \nabla f(x_k)^t d,\\
&\text{subject to}\\
&\qquad \nabla h_i(x_k)^t d + h_i(x_k) = 0, \quad i = 1,\ldots,p,\\
&\qquad \nabla g_j(x_k)^t d + g_j(x_k) \le 0, \quad j = p+1,\ldots,q,
\end{aligned} \tag{2.1}$$

where d is the search direction and H_k is a positive definite approximation to the Hessian matrix of Lagrangian function of problem (P). The Lagrangian function is given by

$$L(x,u,v) = f(x) + \sum_{i=1}^{p} u_i h_i(x) + \sum_{j=p+1}^{q} v_j g_j(x), \tag{2.2}$$

where u_i and v_j are the Lagrangian multipliers. The subproblem (QP) can be solved by using the active set strategy. The solution d_k is used to generate a new iterate

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.3}$$

where the step-length parameter $\alpha_k \in (0,1]$ depends on some line search techniques.

At each iteration, the matrix H_k is updated according to any of the quasi-Newton method. The most preferable method to update H_k is Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [5], where H_k is initially set to the identity matrix I and updated using the formula

$$H_{k+1} = H_k + \frac{y_k y_k^t}{s_k y_k^t} - \frac{H_k s_k s_k^t H_k}{s_k^t H_k s_k}, \tag{2.4}$$

where

$$s_k = x_{k+1} - x_k, \qquad y_k = \nabla L(x_{k+1}, u_{k+1}, v_{k+1}) - \nabla L(x_k, u_k, v_k). \tag{2.5}$$

We have only provided the most basic form of the SQP methods here. Detailed description of the SQP method can be found in [2, 3].

2.2. Generalized reduced gradient. The GRG algorithm was first developed by Abadie and Carpentier [1] as an extension of the reduced gradient method.

GRG transforms inequality constraints into equality constraints by introducing slack variables. Hence all the constraints in (P) are of equality form and can be represented as follows:

$$h_i(x) = 0, \quad i = 1,\dots,q, \tag{2.6}$$

where x contains both original variables and slacks. Variables are divided into dependent, x_D, and independent, x_I, variables (or basic and nonbasic, resp.):

$$x = \begin{bmatrix} x_D \\ \cdots \\ x_I \end{bmatrix}. \tag{2.7}$$

The names of basic and nonbasic variables are from linear programming. Similarly, the gradient of the objective function bounds and the Jacobian matrix may be partitioned as follows:

$$a = \begin{bmatrix} a_D \\ \cdots \\ a_I \end{bmatrix}, \qquad b = \begin{bmatrix} b_D \\ \cdots \\ b_I \end{bmatrix}, \qquad \nabla f(x) = \begin{bmatrix} \nabla_D f(x) \\ \cdots \\ \nabla_I f(x) \end{bmatrix},$$
$$J(x) = \begin{bmatrix} \nabla_D h_1(x) \vdots \nabla_I h_1(x) \\ \nabla_D h_2(x) \vdots \nabla_I h_2(x) \\ \vdots \\ \nabla_D h_q(x) \vdots \nabla_I h_q(x) \end{bmatrix}. \tag{2.8}$$

Let x^0 be an initial feasible solution, which satisfies equality constraints and bound constraints. Note that basic variables must be selected so that $J_D(x^0)$ is nonsingular.

The reduced gradient vector is determined as follows:

$$g_I = \nabla_I f(x^0) - \nabla_D f(x^0)\left(J_D(x^0)\right)^{-1} J_I(x^0). \tag{2.9}$$

The search directions for the independent and the dependent variables are given by

$$d_I = \begin{cases} 0, & \text{if } x_i^0 = a_i,\ g_i > 0, \\ 0, & \text{if } x_i^0 = b_i,\ g_i < 0, \\ -g_i, & \text{otherwise}, \end{cases}$$
$$d_D = -\left(J_D(x^0)\right)^{-1} J_I(x^0) d_I. \tag{2.10}$$

A line search is performed to find the step length α as the solution to the following problem:

$$\begin{aligned} &\text{Minimize } f(x^0 + \alpha d), \\ &\text{subject to} \\ &\qquad 0 \le \alpha \le \alpha_{\max}, \end{aligned} \tag{2.11}$$

where

$$\alpha_{\max} = \sup\left\{\frac{\alpha}{a} \le x^0 \le x^0 + \alpha d \le b\right\}. \tag{2.12}$$

The optimal solution α^* to the problem gives the next solution:

$$x^1 = x^0 + \alpha^* d. \tag{2.13}$$

A more detailed description of the GRG method can be found in [10].

2.3. Genetic algorithms. GAs are stochastic optimization algorithms based upon the principles of evolution observed in nature [7, 12]. Because of their power and ease of implementation, the use of GAs has noticeably increased in recent years. Unlike the gradient methods, they have no requirements on convexity, differentiability, and continuity of the objective, and constraint functions. These significant characteristics of GAs increase their popularity in applications.

The basic GA can be summarized by the following steps:

(1) generate an initial population of chromosomes (or possible solutions) randomly,
(2) evaluate the fitness of each chromosome in the initial population,
(3) select chromosomes that will have their information passed on to the next generation,
(4) cross over the selected chromosomes to produce new offspring chromosomes,
(5) mutate the genes of the offspring chromosomes,
(6) repeat steps (3) through (5) until a new population of chromosomes is created,
(7) evaluate each of the chromosomes in the new population,
(8) go back to step (3) unless some predefined termination condition is satisfied.

GAs are directly applicable only to the unconstrained problems. In the application of GAs to constrained nonlinear programming problems, chromosomes in the initial population or those generated by genetic operators during the evolutionary process generally violate the constraints, resulting in infeasible chromosomes. During the past few years, several methods were proposed for handling constraints by GAs.

Michalewicz and Schoenauer grouped the constraint handling methods into the following four categories [14]:

(1) methods based on preserving feasibility of solutions,
(2) methods based on penalty functions,
(3) methods based on a search for feasible solutions,
(4) hybrid methods.

Penalty function methods are the most popular methods used in the GAs for constrained optimization problems. These methods transform a constrained problem into an unconstrained one by penalizing infeasible solutions. Penalty is imposed by adding to the objective function $f(x)$ a positive quantity to reduce fitness values of such infeasible solutions:

$$\hat{f}(x) = \begin{cases} f(x) & \text{if } x \in F, \\ f(x) + p(x) & \text{otherwise,} \end{cases} \tag{2.14}$$

where $\hat{f}(x)$ is the fitness function and $p(x)$ is the penalty function whose value is positive. The design of the penalty function $p(x)$ is the main difficulty of penalty function methods. Several forms of penalty functions are available in the literature.

Nevertheless, most of them have the form

$$p(x) = \sum_{i=1}^{p} r_i [H_i(x)]^{\beta} + \sum_{j=p+1}^{m} c_j [G_j(x)]^{\gamma}, \tag{2.15}$$

Table 3.1. Fifteen constrained nonlinear programming problems.

Problem number	n	LE	LI	NE	NI	Type of objective function	Best known	Source
1	8	0	3	0	3	Linear	7049.25	[6]
2	7	0	0	0	4	Polynomial	680.6300573	[13]
3	6	0	0	3	1	Linear	−0.3888	[22]
4	10	4	0	1	2	Linear	−400	[16]
5	6	3	3	0	0	Nonlinear	−13.401904	[16]
6	5	0	0	0	6	Quadratic	30665.41	[20]
7	10	0	3	0	5	Quadratic	24.3062	[20]
8	13	0	9	0	0	Quadratic	−15	[13]
9	2	0	1	0	1	Polynomial	−118.704860	[16]
10	2	0	2	0	2	Linear	−2.828427	[16]
11	2	0	0	0	2	Linear	−5.50801	[18]
12	5	0	0	3	0	Nonlinear	0.0539498	[18]
13	3	0	0	0	2	Quadratic	11.68	[11]
14	2	0	0	0	2	Quadratic	−79.8078	[19]
15	10	0	3	0	5	Polynomial	−216.602	[17]

where $H_i(x)$ and $G_j(x)$ are functions of the equality constraint $h_i(x)$ and the inequality constraint $g_j(x)$, respectively, and r_i and c_j are positive penalty coefficients. β and γ are positive constants usually set to be 1 or 2. The most general form of H_i and G_j is as follows:

$$H_i(x) = \left|h_i(x)\right|, \qquad G_j(x) = \max\left[0, g_j(x)\right]. \tag{2.16}$$

How to design a penalty function and which penalty function method is the best are still open questions one needs to answer. Comparative studies about the penalty function methods in genetic algorithms can be found in [9, 13, 15]. One important result of these studies is that the quality of the solution severely depends on selected values of penalty coefficients. Hence, determining the appropriate values for penalty coefficients at the beginning of the optimization has vital importance in order to have a robust solution.

3. Test problems and results

In order to test the efficiency of the algorithms in terms of getting closer to the best-known solution, some constrained nonlinear programming problems have been selected from the literature. Several authors from optimization community have used some of these problems to compare the performances of some available penalty function methods and their own methods. These problems have objective functions of various types with different types of constraints. Table 3.1 presents some information of these problems, including the number of variables n, the number of constraints, type of the objective function, the best-known objective function value, and the source of the problem. LE, LI, NE, and NI denote the number of linear equations, linear inequalities, nonlinear equations, and nonlinear inequalities, respectively.

Table 3.2. Results obtained by GRG, SQP, and GA.

Problem number	Methods		
	GRG	SQP	GA
1	7049.248	7049.248	7049.248
2	680.63	680.63	680.63
3	−0,3876	0	−0.3888
4	−400	−400	−400
5	−12.5079	−4.8038	−13.4019
6	30665.4152	30665.4152	30665.4152
7	24.6232	24.3062	24.3062
8	−10.1094	−10.1094	−15
9	10.94	11	−118.7049
10	1.4142	1.4142	−2.8284
11	−4.0537	−4.0537	−5.5080
12	1	1	0.0435
13	6.9418	7.0732	11.6766
14	−47.9977	−79.8078	−79.8078
15	4.1288	197.8580	−216.6025

In our experiments, three different programs have been used to solve the problems above:

(i) Microsoft Excel Solver using GRG algorithm,
(ii) the function fmincon from the MATLAB Optimization Toolbox using SQP algorithm,
(iii) GA Solver of WWW-NIMBUS system.

The GA Solver of WWW-NIMBUS system uses a parameter-free penalty function approach [4]. One of the important advantages of this penalty approach is that no parameter is required in handling constraints unlike other penalty function approaches which require that a large number of parameters must be set right by the user. consequently , we decided to use GA Solver of WWW-NIMBUS for solving 15 test problems. For detailed information about the parameter-free penalty function approach, see [4].

For each of the problems, we started the GRG and SQP methods from five different starting points in order to increase their chances of finding better solutions, and recorded the best solutions, found. The experimental results obtained by each of the three methods are shown in Table 3.2.

From the results of Table 3.2, it is easy to draw a conclusion that GA has better performance than GRG and SQP in constrained nonlinear programming problems. In fourteen problems, GA has been able to find solutions close to or the same as objective function values reported in earlier studies. In problem 12, the solution $(x^* = -1.4730; 1.7546; -1.8568; -0.7956; 0.8190)$ obtained with GA is more accurate than that reported earlier. Although GRG and SQP could give the same solutions with true optimum solutions in some of the problems, they could not find better solutions than those of GA in any of

the problems. It is also interesting to note that GRG and SQP have given solutions rather far from the true optimum solutions in some problems (e.g., problems 8, 9, 12, 13, and 15).

4. Conclusion

Three popular optimization methods were tested on various constrained optimization problems that were found in the literature. Sizes of our test problems were smaller than the problems generally encountered in engineering applications, but they had some common characteristics (e.g., nonconvex and multimodal solution space) with real-world engineering optimization problems. GA Solver of WWW-NIMBUS system was found to be highly efficient for all these problems. Neither GRG nor SQP could give better solutions than those found by using GA. In some problems, these methods get trapped local optimum solutions rather far from the true optimum solutions. We expect that the poor performances of the GRG and SQP methods will also continue in large-scale engineering optimization problems because of the nonconvex solution spaces.

GA Solver of WWW-NIMBUS using penalty-free method makes things easier for the user by removing the problem of setting a large number of the penalty parameter values even in small size problems. Since the GA explores multiple regions of the solution space simultaneously, it can avoid the local optimum problem and identify the global optimum. We conclude that GA is reliable and effective for solving nonlinear and nonconvex engineering problems.

References

[1] J. Abadie and J. Carpentier, *Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints*, Optimization (Sympos., Univ. Keele, Keele, 1968) (R. Fletcher, ed.), Academic Press, London, 1969, with discussion, pp. 37–47.

[2] P. T. Boggs and J. W. Tolle, *Sequential quadratic programming*, Acta Numerica, 1995, Acta Numer., Cambridge University Press, Cambridge, 1995, pp. 1–51.

[3] ———, *Sequential quadratic programming for large-scale nonlinear optimization*, J. Comput. Appl. Math. **124** (2000), no. 1-2, 123–137.

[4] K. Deb, *An efficient constraint handling method for genetic algorithms*, Comput. Methods Appl. Mech. Eng. **186** (2000), no. 2–4, 311–338.

[5] R. Fletcher, *Practical Methods of Optimization*, A Wiley-Interscience Publication, John Wiley & Sons, Chichester, 1987.

[6] C. A. Floudas and P. M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Lecture Notes in Computer Science, vol. 455, Springer, Berlin, 1990.

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

[8] C. Kao, *Performance of several nonlinear programming software packages on microcomputers*, Comput. Oper. Res. **25** (1998), no. 10, 807–816.

[9] A. F. Kuri-Morales and J. Gutiérrez-Garcia, *Penalty functions methods for constrained optimization with genetic algorithms: a statistical analysis*, Proc. 2nd Mexican International Conference on Artificial Intelligence, Springer-Verlag, Heidelberg, Germany, 2001, pp. 108–117.

[10] L. S. Lasdon, A. D. Warren, A. Jain, and M. Ratner, *Design and testing of a generalized reduced gradient code for nonlinear programming*, ACM Trans. Math. Software **4** (1978), no. 1, 34–50.

[11] M. Mathur, S. B. Karale, S. Priye, V. K. Jayaraman, and B. D. Kulkarni, *Ant colony approach to continuous function optimization*, Ind. Engng. Chem. Res **39** (2000), no. 10, 3814–3822.

[12] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin, 1994.

[13] ———, *Genetic algorithms, numerical optimization and constraints*, Proc. 6th International Conference on Genetic Algorithms (L. J. Eshelman, ed.), Morgan Kaufmann, California, 1995, pp. 151–158.

[14] Z. Michalewicz and M. Schoenauer, *Evolutionary algorithms for constrained parameter optimization problems*, Evolutionary Computation **4** (1996), no. 1, 1–32.

[15] K. Miettinen, M. M. Mäkelä, and J. Toivanen, *Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms*, J. Global Optim. **27** (2003), no. 4, 427–446.

[16] H. S. Ryoo and N. V. Sahinidis, *Global optimization of nonconvex NLPs and MINLPs with applications in process design*, Comput. Oper. Res. **19** (1995), no. 5, 551–566.

[17] M. Sakawa and K. Yauchi, *Floating point genetic algorithms for nonconvex nonlinear programming problems: revised GENOCOP III*, Electron. Comm. Japan **83** (2000), no. 8, 1–9.

[18] H. Sarimveis and A. Nikolakopoulos, *A line up evolutionary algorithm for solving nonlinear constrained optimization problems*, Comput. Oper. Res. **32** (2005), no. 6, 1499–1514.

[19] K. Schittkowski, *More Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, vol. 282, Springer, Berlin, 1987.

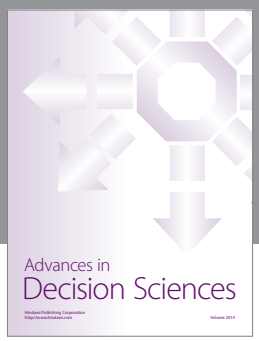
[20] V. S. Summanwar, V. K. Jayaraman, B. D. Kulkarni, H. S. Kusumakar, V. Gupta, and J. Rajesh, *Solution of constrained optimization problems by multi-objective genetic algorithm*, Comput. Oper. Res. **26** (2002), no. 10, 1481–1492.

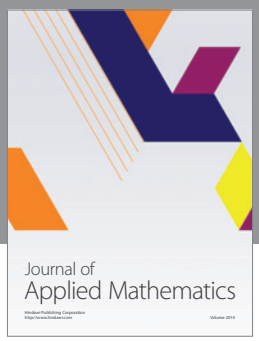
[21] T. Wang, *Global optimization for constrained nonlinear programming*, Ph.D. thesis, Department of Computer Science, University of Illinois, Illinois, 2001.

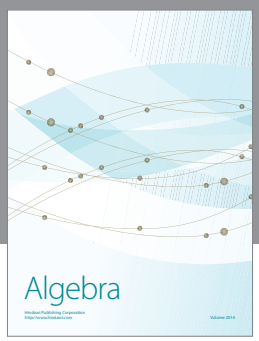
[22] L. Yan and D. Ma, *Global optimization of non-convex nonlinear programs using line-up competition algorithm*, Comput. Oper. Res. **25** (2001), no. 11-12, 1601–1610.

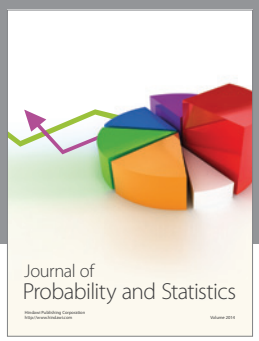
Ozgur Yeniay: Department of Statistics, Faculty of Science, Hacettepe University, 06532 Beytepe, Ankara, Turkey

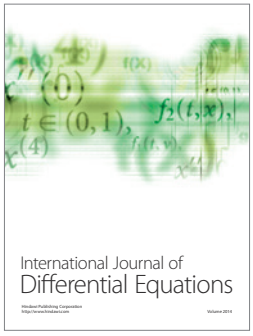
E-mail address: yeniay@hacettepe.edu.tr

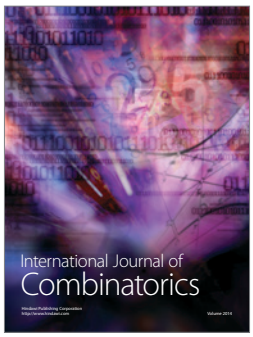


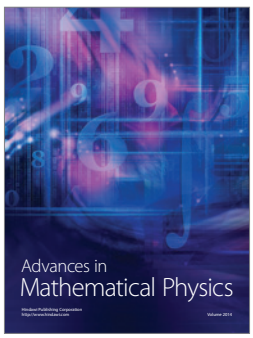


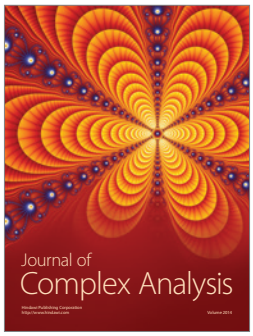


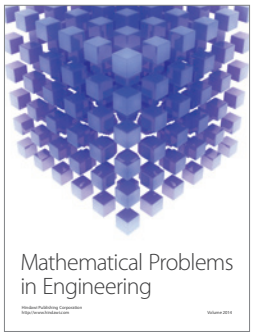


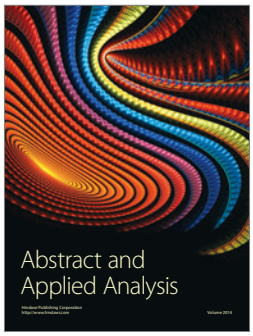


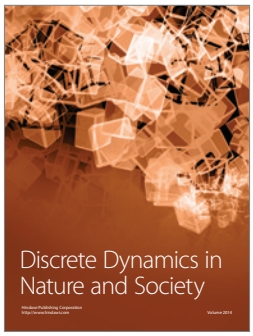


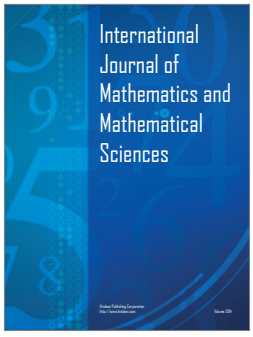


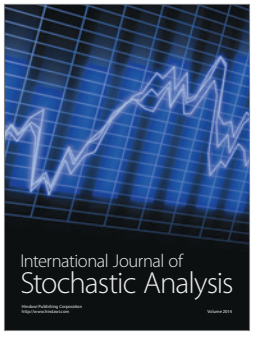












Hindawi
Submit your manuscripts at
http://www.hindawi.com