Archives of Control Sciences
Volume 21(LVII), 2011
No. 3, *pages 299–312*

Hybrid Mesh Adaptive Direct Search and Genetic Algorithms Techniques for industrial production systems

PANDIAN VASANT

In this paper, computational and simulation results are presented for the performance of the fitness function, decision variables and CPU time of the proposed hybridization method of Mesh Adaptive Direct Search (MADS) and Genetic Algorithm (GA). MADS is a class of direct search of algorithms for nonlinear optimization. The MADS algorithm is a modification of the Pattern Search (PS) algorithm. The algorithms differ in how the set of points forming the mesh is computed. The PS algorithm uses fixed direction vectors, whereas the MADS algorithm uses random selection of vectors to define the mesh. A key advantage of MADS over PS is that local exploration of the space of variables is not restricted to a finite number of directions (poll directions). This is the primary drawback of PS algorithms, and therefore the main motivation in using MADS to solve the industrial production planning problem is to overcome this restriction. A thorough investigation on hybrid MADS and GA is performed for the quality of the best fitness function, decision variables and computational CPU time.

Key words: mesh adaptive direct search, genetic algorithms, fitness function, degree of possibility, level of satisfaction

1. Introduction

Audet and Dennis [1] presented and analyzed a new mesh adaptive direct search (MADS) class of algorithms for minimizing a nonsmooth function $f : R^n \to R \cup \{+\infty\}$ under general constraints $x \in \Omega \neq \varnothing \subseteq R^n$.

A key advantages of mesh adaptive direct search (MADS) over pattern search (PS) is that local exploration of the space of variables is not restricted to finite number of directions (call poll directions). This is the primary drawback of Pattern Search (PS) algorithms, and Audet and Dennis [1] was motivated in defining MADS to overcome this restriction. MADS algorithms are frame-based methods. Audet and Dennis [1] proposed a less general choice of frames than the choices allowed by Coope and Price [6]. Audet

The Author is with University Technology Petronas, 31750 Tronoh, Malaysia. E-mail: vasant0001@yahoo.com

The author would like to sincerely thank the editorial board of ACS (Archives of Control Sciences) for their valuable comments, suggestions and support on this research paper publications.

Received 15.08.2011.

and Dennis [1] MADS frames are easy to implement, and they are specifically aimed at ensuring an asymptotically dense set of polling directions.

The genetic algorithms (GA), is an approach that mimics biological processes in evolving optimal or near optimal solutions to problems (Goldberg [11]). In this paper a practical applications to challenging industrial problems of production planning has been investigated to exploit the full potential of GA to look for the promising global optimal solution.

The paper is organized in the following orders. Section 2 provides the methodology for MADS and GA approaches. The problem statement and formulation of fuzzy model are illustrated in Section 3. Section 4 gives detail information on the analysis of experimental results and findings. The paper ends with conclusions and future research directions.

2. Methodology

Given an initial iterate $x_0 \in \Omega$, a MADS algorithm attempts to locate a minimizer of the function f over Ω by evaluating f at some trial points. The algorithm does not require any derivative information for f. This is useful when there are several local optima. But it is essential when ∇f is unavailable, either because it does not exist, or it cannot be accurately estimated due to noise in f or other reasons.

MADS is an iterative algorithm where at each iteration a finite number of trial points are generated, and the infeasible trial points are discarded. The objective function values at the feasible trial points are compared with the current incumbent value $f_\Omega(x_k)$, i.e., the best feasible objective function value found so far. Each of these trial points lies on the current mesh, constructed from a finite set of n_D directions $D \subset R^n$ scaled by a mesh size parameter $\Delta_k^m \in R_+$.

There are two restrictions on the set D. First, D must be a positive spanning set (Davis [7]), i.e., nonnegative linear combination of its element must span R^n. Second, each direction $d_j \in D$ (for $j = 1,2,\ldots,n_D$) must be the product Gz_j of some fixed non-singular generating matrix $G \in R^{n\times n}$ by an integer vector $z_j \in Z^n$. For the convenience, the set D is also viewed as a real $n \times n_D$ matrix.

The mesh is conceptual in the sense that it is never actually constructed. In practice, one can easily make sure that the strategy for generating trial points is such that they all belong to the mesh. The objective of the iteration is to find a trial mesh point with a lower objective function value than the current incumbent value $f_\Omega(x_k)$. Such a trial point is called an improved mesh point, and the iteration is called a successful iteration. There are no sufficient decrease requirements on the objective function value.

The evaluation of f_Ω at a trial point x is done as follows. First, the constraints defining Ω are tested to determine if x is feasible or not. Indeed, since some of the constraints defining Ω might be expensive or inconvenient to test, one would order the constraints to test the easiest first. If $x \notin \Omega$, then $f_\Omega(x)$ is set to $+\infty$ without evaluating $f(x)$, and perhaps without evaluation all the constraints defining Ω. In effect, this means we discard

the infeasible trial points. On the other hand, if $x \in \Omega$, then $f(x)$ is evaluated. This remark seem obvious, but is saves the computation work.

Each iteration is divided into two steps. The first, called the search step, has the sane flexibility as in PS. It allows evaluation of $f_\Omega(x)$ at any finite number of mesh points. Any strategy can be used in the search step to generate a finite number of trial mesh points. Restricting the search points to lie on the mesh is a way in which MADS is less general than the frame methods of Coope and Price [6]. The search is said to be empty when no trial points are considered. The drawback to the search flexibility is that it cannot be used in the convergence analysis - except to provide counterexamples as in (Audet [2]). The detail discussion of search steps is given in (Abramson, Audet and Dennis [3]).

When an improved mesh point is generated, then the iteration may stop, or it may continue if the user hopes to find a better improved mesh point. In either case, the next iteration will be initiated with a new incumbent solution $x_{k+1} \in \Omega$ with $f_\Omega(x_{k+1}) < f_\Omega(x_k)$ and with a mesh size parameter Δ^m_{k+1} equal to or larger than Δ^m_k. Coarsening the mesh when improvements in f_Ω are obtained can be convergence.

Whenever the search step fails to generate an improved mesh point, then the second step, called the poll, is invoked before termination of the iteration. The difference between the MADS and the PS algorithms lies exactly in this poll step. For this reason, the numerical comparison (Audet and Dennis [1]) in the sequel used empty, or very simple, search steps in order to isolate the value of the MADS poll step.

When the iteration fails in generating an improved mesh point, then the next iteration is initiated from any point $x_{k+1} \in S_{k+1}$ where S_k is the set points where the objective function f had been evaluated by the start of iteration k with $f_\Omega(x_{k+1}) = f_\Omega(x_k)$; though there is usually a single such incumbent solution, and then x_{k+1} is set to x_k. The mesh size parameter Δ^m_{k+1} is reduced to increase the mesh resolution, and therefore to allow the evaluation of f at trial points closer to the incumbent solution. More precisely, given a fixed rational number $\tau > 1$, and two integers $w^- \leqslant -1$ and $w^+ \geqslant 0$, the mesh size parameter is updated as follows:

$$\Delta^m_{k+1} = \tau^{w_k}\Delta^m_k \text{ for some } w_k \in \begin{cases} \{0,1,...,w^+\} & \textit{if an improved mesh point is found} \\ \{w^-,w^-+1,...,-1\} & \textit{otherwise} \end{cases} \tag{1}$$

For MADS, Audet and Dennis [1] introduced the poll size parameter $\Delta^p_k \in R_+$ for iteration k. This new parameter dictates the magnitudes of the distance from the trial points generated by the poll step to the current incumbent solution x_k. In PS, there is a single parameter to represent these quantities: $\Delta_k = \Delta^p_k = \Delta^m_k$. In MADS, the strategy for updating Δ^p_k must be such that $\Delta^m_k \leqslant \Delta^p_k$ for all k, and it must satisfy

$$\lim_{k\in K}\Delta^m_k = 0 \text{ if and only if } \lim_{k\in K}\Delta^p_k = 0 \text{ for any infinite subset of indices } K. \tag{2}$$

The set trial points considered during the poll step is called a frame. The frames of Coope and Price [6] can be more general than MADS frames.

The MADS frame is constructed using a current incumbent solution x_k (called frame center 0 and the poll and mesh size parameters Δ_k^p and Δ_k^m to obtain a positive direction D_k. Unlike GPS, generally the MADS set of direction D_k is not a subset of D. If the poll step fails to generate an improved mesh point then the frame is called a minimal frame, and the frame center x_k is said to be a minimal frame center. This leads to mesh refinement. At each iteration, the columns of D_k are called the poll directions.

The algorithm of MADS is very similar to PS, with differences in the poll step, and in the new poll size parameter.

MADS algorithm

- Initialization: Let $x_0 \in \Omega$, $\Delta_0^m \leqslant \Delta_0^p$, D, G, τ, w^- and w^+ satisfy (2). Set the iteration counter $k \leftarrow 0$.
- Search and poll step: Perform the search and possibly the poll steps (or only part of them) until an improved mesh point x_{k+1} is found on the mesh M_k (Audet and Dennis [1]).

$$M_k = \bigcup_{s \in S_k} \{x + \Delta_k^m D z : z \in N^{n_D}\}$$

  where S_k is the set of points where the objective function f had been evaluated by the start of iteration k.

  - Optional search: Evaluate f_Ω on a finite subset of trial points on the mesh M_k.
  - Local poll: Evaluate f_Ω on the frame P_k (Audet and Dennis [1]).

$$P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k$$

    where D_k is a positive spanning set such that $0 \notin D_k$ and for each $d \in D_k$,

    * d can be written as a nonnegative integer combination of the direction in D: $d = Du$ for some vector $u \in N^{n_{D_k}}$ that may depend on the iteration number k
    * the distance from the frame center x_k to a frame point $x_k + \Delta_k^m d \in P_k$ is bounded by a constant times the poll size parameter:

$$\Delta_k^m \|d\| \leqslant \Delta_k^p \max\left\{\left\|d^l\right\| : d^l \in D\right\}$$

    * limit (as defined in Coope and Price [6]) of the normalized sets D_k are positive spanning sets.

- Parameter update: Update Δ_{k+1}^m according to (1), and Δ_{k+1}^p according to (2).
- Set $k \leftarrow k+1$ and go back to the search and poll step.

The crucial distinction and advantage of MADS over PS is that the MADS mesh size parameter Δ_k^m may go to zero more rapidly than Δ_k^p. Consequently, the direction in D_k used to define the frame may be selected in away so that asymptotically they are not confined to a finite set. Note that in PS both Δ_k^m and Δ_k^p are equal: a single parameter plays the role of the mesh and poll size parameters, and therefore, the number of positive spanning sets that can be formed by subsets of D is constant over all iterations.

In PS, the set D_k is a subset of the finite set D. There is more flexibility in MADS.

PS is a valuable algorithm, but the application of non smooth analysis techniques in (Audet and Dennis [4]) showed its limitation due to the finite choice of directions in (Audet [2]). MADS removes the PS restriction to finitely many poll directions. In MADS the poll directions change at each iteration, but they are static in PS.

Genetic algorithms

The crossover operator is an important component of GA. The crossover operation generates offspring from randomly selected pairs of individuals within the mating pool, by exchanging segments of the chromosome strings from the parents.

The purpose of the mutation is to ensure that diversity is maintained in the population. It gives random movement about the search space thus preventing the GA becoming trapped in 'blind corners' or 'local optima' during the search. Evolver performs order-based mutation. In this mutation, two tasks are selected at random and their positions are swapped. The 'mutation rate' determines the probability that mutation is applied after a crossover. The number of swaps performed is increased or decreased proportionately to the increase and decrease in the mutation rate setting.

The performance of a GA depends upon population size, scaling function, selection, reproduction, elite count, crossover, mutation, migration and stopping criteria. Detailed experiments have been carried out to study the effect of these parameters. The application of the GA to the fuzzy optimization is repeated here for each of the combinations of the following parameter levels:

Population Size: 100-1000

Mutation operator: Adaptive feasible - this parameter randomly generates directions that are adaptive to the last successful or unsuccessful generations. A step length is chosen along each direction so that linear constraints and bounds are satisfied.

Crossover operator: Crossover combines two individual, or parents, to form a new individual, or child, for the next generation. Arithmetic crossover creates children that are the weighted arithmetic mean of two parents. Children are feasible with respect to linear constraints and bounds.

Scaling function: Rank scales the raw scores based on the rank of each individual, rather tan its score. The rank of the individual is its position on the sorted scores. The rank of

the fittest individual is 1, the next fittest is 2 and so on. Rank fitness scaling removes the effect of the spread of the raw scores.

Creation function: It's specifies the function that creates the initial population. Uniform function creates a random initial population with a uniform distribution.

Initial scores: The algorithm creates one using the uniform creation function.

Initial range: [0; 1] - Specifies the lower and upper bounds for the entries of the vectors in the initial population.

Selection: The selection function chooses parents for the next generation based on their scaled values from the fitness scaling function. Stochastic uniform lays out a line in which each parent corresponds to a section of the line of length proportional to its expectation. The algorithm moves along the line in steps of equal size, one step for each parent. At each step, the algorithm allocates a parent from the sections it lands on. The first step is a uniform random number less than the step size.

Reproduction: Reproduction parameter determines how the genetic algorithm creates children at each new generation. Elite count has been provided to specify the number of individuals that are guaranteed to survive to the next generation. The elite count should be a positive integer and less than or equal to population size.

Elite count = 8-100.

Migration: Migration is the movement of individuals between subpopulations, which the algorithm creates if we set population size to be a vector of length greater than 1. Every so often, the best individuals from one subpopulation replace to the worst individuals in another subpopulation. We can control how migration occurs by the following three parameters.

(a) *Direction:* Migration can take place in one direction or two. If we select the direction to be forward, migration takes place toward the last subpopulation. That is the nth subpopulation migrates into the $(n+1)$'th subpopulation. If we select the direction to be both, then the *n*th subpopulation migrates into the $(n-1)$th and the $(n+1)$th subpopulation.

(b) *Fraction:* Fraction controls how many individuals move between subpopulations. Fraction is the fraction of the smaller of the two subpopulations that moves. If individual migrate from a subpopulation of 100 individuals into a population of 1000 individuals and the fraction is 0.2, 20 individuals (0.2·100) migrate. Individuals that migrate from one subpopulation to another are copies. They are not removed from the source subpopulation.

(c) *Interval:* Interval controls how many generations pass between migrations. If we select the interval to be 500, for example, then the migration between subpopulations takes place every 500 generations.

Stopping criteria: The stopping criteria determine the caused of algorithms to terminate. The following are the stopping criteria's that have been used in this research paper.

(i) *Generations:* Generations specifies the maximum number of iterations the genetic algorithm performs.

(ii) Time limit - Time limit specifies the maximum time in seconds the genetic algorithm runs before stopping.

(iii) *Fitness limit:* If the best fitness value is less than or equal to the value of fitness limit, the algorithm stops.

(iv) *Stall generations:* If the weighted average change in the fitness function value over stall generations is less than function tolerance, the algorithm stops.

(v) *Stall time limit:* If there is no improvement in the best fitness value for an interval of the time in seconds specified by stall time limit, the algorithm stops.

(vi) *Function tolerance:* If the cumulative change in the fitness function value over stall generations is less than function tolerance, the algorithm stops.

Genetic Algorithms is a global stochastic method based on the mechanism of nature selection and evolutionary genetics and used in some different fields (Gen and Cheng [9]). For the specific problem such as nonlinear programming, the combination of genetic algorithm and other method such as the MADS method can outperform genetic algorithms, which can be illustrated by the experimental results (Honggang and Jianchao [10]). This idea of hybrid genetic algorithms has been adopted in this research work.

3. Problem statement

Optimization techniques are primarily used in production planning problems in order to achieve optimal profit, which maximizes certain objective function by satisfying a number of constraints. The first step in an optimal production planning problems is to formulate the underlying nonlinear programming (NLP) problem by writing the mathematical functions relating to the objective and constraints.

Given a degree of satisfaction value μ, the fuzzy constrained optimization problem can be formulated (Jiménez, Cadenas, Sánchez, Gómez-Skarmeta and Verdegay [12]; Vasant and Barsoum [14]) as the non linear constrained optimization problem shown below in the following section.

Problem formulation

$$\text{Maximize } \sum_{i=1}^{8} (c_i x_i - d_i x_i^2 - e_i x_i^3)$$

Subject to:

$$\begin{aligned}
&\sum_{i=1}^{8} \left[a_{ij}^{l} + \left(\frac{a_{ij}^{h} - a_{ij}^{l}}{\alpha} \right) \ln \frac{1}{C} \left(\frac{B}{\mu} - 1 \right) \right] x_i - b_j \leqslant 0, \quad j = 1,2,...,17 \\
&\sum_{i=7}^{8} r_i x_i - 0.15 \sum_{i=1}^{6} r_i x_i \leqslant 0 \\
&x_1 - 0.6 x_2 \leqslant 0 \\
&x_3 - 0.6 x_4 \leqslant 0 \\
&x_5 - 0.6 x_6 \leqslant 0 \\
&0 \leqslant x_i \leqslant u_i, \quad i = 1,2,...,8
\end{aligned} \tag{3}$$

In the above non-linear programming problem, the variable vector x represents a set of variables x_i, $i = 1,2,\ldots,8$. The above optimization problem contains eight continuous variables and 21 inequality constraints (Vasant and Barsoum [16]). A test point x_i satisfying constrains is called feasible, if not infeasible. The set satisfying constrains is called the feasible domain. The aim of the optimization is to maximize the total production profit for the industrial production planning problems. The formulation of the new non-linear cubic function for this particular problem has been refereed to Vasant [15]. The cubic objective function has 24 coefficients for eight decision variables. This problem considered one of the most challenging problems in the research area of industrial production planning.

4. Analysis on experimental results

Parameter settings for MADS algorithm are as follows:
Poll: MADS position basis 2N
Complete poll
Polling order: Random
Mesh: Initial size 1 and Max mesh size Infinite
Expansion factor: 2
Contraction factor: 0.5
Stopping criteria: Mesh tolerance - 1e-006; Max iteration - 100·8;
Max function evaluation - 2000·8; Time limit - infinite;
Constraint tolerance - 1e-006 and Function tolerance - 1e-006.

The results for the industrial production planning problems were obtained by using the following genetic algorithms.

Genetic Algorithms

1. *Initialization*. Generate an initial population. Initialize parameters and define fitness function.
2. *Crossover*. Perform crossover using arithmetic crossover.
3. *Mutation*. Perform mutation using adaptive feasible.
4. *Selection*. Evaluate the fitness of each individual in the population. Choose the N best chromosomes to form the next generation.
5. *Termination criteria*. If stopping criteria's are satisfied, then terminate. Otherwise, select the next generation and go to Step 2.
6. Output the best solutions.

Notation used in the MADSGA method is as follows:
μ = Degree of possibility
γ = Level of satisfaction
α = Vagueness factor
f = objective function or fitness function
x_i = Decision variables
Time in seconds = CPU (s)

In this paper, the results for the hybrid and non-hybrid techniques of various optimization approaches have been investigated thoroughly in the form of 2D, 3D plots and tables (Chapman [5]; Gilat [8]). The industrial production planning problem which was illustrated in chapter three was solved successfully by most of the hybrid optimizations methods. The simulation and computational results are discussed in form of detail explanation. The formulation coded in MATLAB Version 2.2 (R2007b) [13] was run on Intel Pentium M 1.6 GHz, 512 MB Ram, 40 GB HDD PC. The results are analyzed along with the optimal profit function (Objective or Fitness function) with level of satisfaction, decision variables, vagueness factor and computational time (CPU).

Fig. 1 depicts the best fitness function solution for $\alpha = 13.813$ and $\gamma = 0.001$ to $\gamma = 0.99$. The best optimal fitness function value is 197364.2 obtained at $\gamma = 0.99$ after three iteration with 8251 number of function evaluations. CPU time for running MADS and GA is 39.427 s and 39.4275 s respectively. Total CPU time for running at $\alpha = 13.813$ and $\gamma = 0.001$ to $\gamma = 0.99$ is 8m 49.32 s. Table 1 reports feasible optimal solution for eight decision variables with γ and fitness function values.

The feasible solutions in Tab. 1 for the eight decision variables reflect the real world situation for the industrial production planning problems. The best optimal solution for

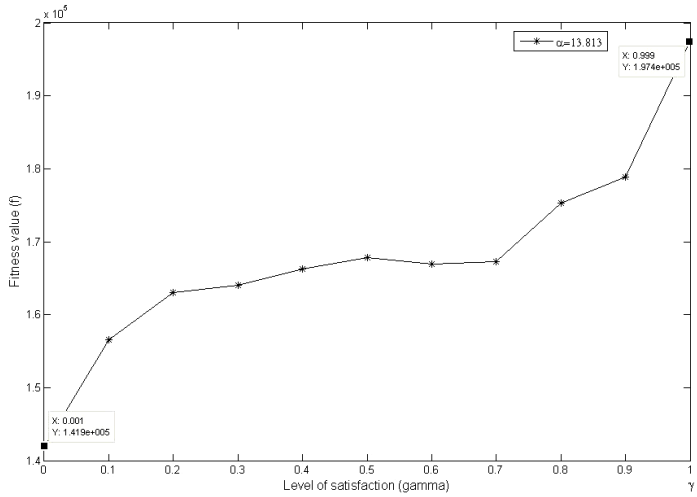

Figure 1. 1 Fitness value versus γ.

Table 8. Optimal value for Fitness Function

γ	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	f
0.001	255.5	545.2	102.3	341.9	92.4	233.5	66.5	41.8	141916.7
0.1	286.3	644.3	213.6	459.1	125.0	208.4	67.3	65.6	156521.4
0.2	270.8	540.7	285.9	488.6	114.4	192.3	109.4	41.0	163081.0
0.3	249.0	496.2	269.2	522.5	188.9	314.8	99.4	41.6	164043.3
0.4	259.7	669.3	237.4	408.7	185.8	309.6	114.3	30.6	166244.4
0.5	326.3	664.2	185.0	425.8	161.8	269.7	133.8	22.3	167783.4
0.6	352.8	588.7	284.1	473.5	91.4	169.4	94.8	77.0	166901.8
0.7	300.7	598.5	242.5	566.8	187.7	312.9	72.0	78.2	167212.6
0.8	322.8	574.7	284.4	479.2	161.3	268.9	123.0	51.2	175329.0
0.9	368.3	653.2	240.2	402.1	167.3	278.8	157.2	30.8	178822.2
0.99	444.0	740.4	358.9	598.2	149.3	248.8	178.0	81.0	197364.2

the fitness function value at $\gamma = 0.99$ is less than fitness function value in the MADS method alone. This is due to MADS stopped the optimization process earlier than GA searching process. Further more the CPU time for this hybrid approach is almost double

than MADS alone. The major draw back of this method is, it is suffering from slow convergence and it is wandering around optimal solution if high accuracy needed.

From the above explanation, it suspected that further experiment be carried on for several of α in order to investigate improved optimal solution for the fitness function value. Fig. 2 depicts the simulation results for $\alpha = 1$ to $\alpha = 41$.

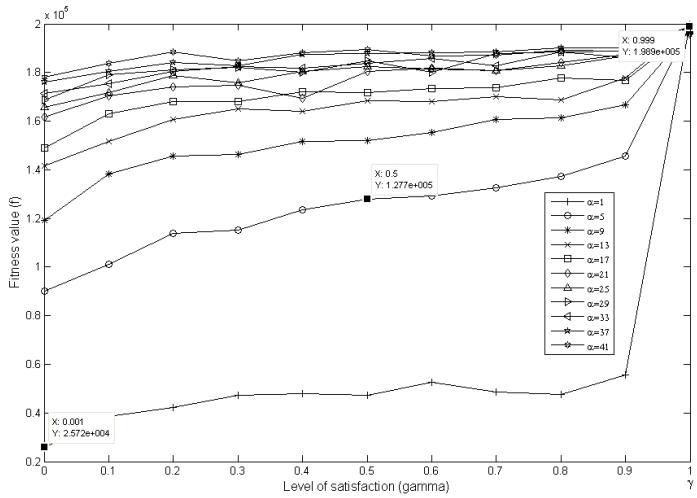

Figure 2. 1 Fitness value versus γ.

The best optimal fitness function value is 199813.94 obtained at $\gamma = 0.99$ and $\alpha = 1$. CPU time for running MADS and GA is 6.1055 s and 6.1059 s respectively for $\gamma = 0.99$ and $\alpha = 41$. Number of iterations and function evaluations is 3 and 6001 respectively. Total CPU time for $\alpha = 1$ to $\alpha = 41$ is 1 h 54 m 48.87 s.

Tab. 2 reports the best optimal fitness function values for $\alpha = 1$ to $\alpha = 41$ at $\gamma = 0.99$. From the Tab. 2, it is concluded that this hybrid approach reaches the best optimal fitness value 199813.94 at $\alpha = 1$. This is the major significant contribution of MADS and GA hybrid approach in this case studies. On the other hand, MADS alone reaches optimal fitness value 198002 at $\alpha = 1$ and $\gamma = 0.99$.

Further experiment carried out for the investigation of simulation results for the fitness function respect to α and γ. Fig. 3 depicts the results. Total CPU time for this simulation is 1 h 45 m 17.79 s. Optimal fitness function value at $\alpha = 41$ and $\gamma = 0.99$ is 197902.8. CPU time running PS and GA for this result is 83.3417 s and 83.3422 s respectively.

Table 9. Optimal value for Fitness Function at $\gamma = 0.99$

α	f
1	199813.94
5	199171.89
9	197168.54
13	199357.83
17	197303.77
21	195918.66
25	198955.51
29	197928.65
33	198738.09
37	198915.82
41	196138.06

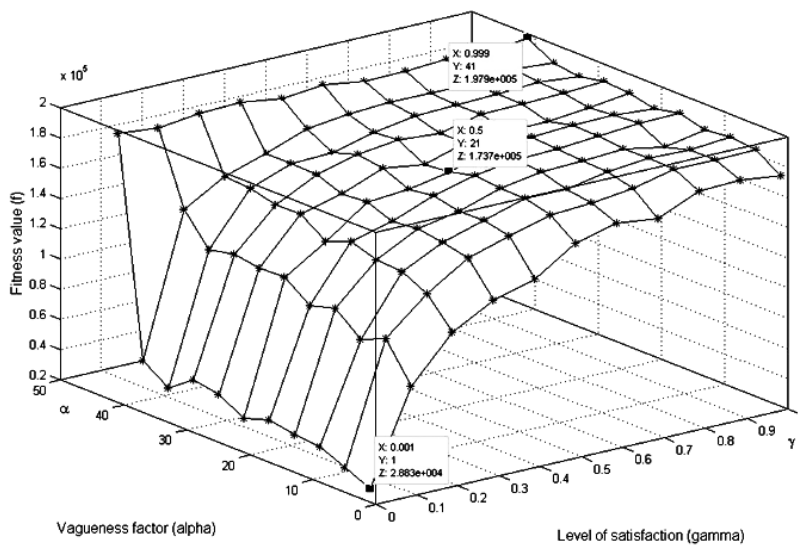

Figure 3. 1 Fitness value versus α and γ.

5. Conclusions

The findings of the computational and simulation results reveal that the significant contribution of MADS and GA hybrid approach is in finding a reasonable quality solu-

tion for the feasible decision variables and fitness function value. The incorporation of GA into MADS exhibit far more superior speedy solution in CPU computational time compare to HPSGA techniques. On the other hand, its major draw back is on the longer computational CPU time in running MADS and GA algorithms. This is due to the role of GA in this optimization process as searching technique for the best initial solution and MADS played as an optimizer. Nevertheless, there is a very strong possibility of improving this draw back with other additional hybrid optimizations techniques.

References

[1] C. AUDET and J.E. DENNIS: Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, **17**(1), (2007), 188-217.

[2] C. AUDET: Convergence results for pattern search algorithms are tight. *Optimization and Engineering*, **5**(2), (2004), 101-122.

[3] M.A. ABRAMSON, C. AUDET and J.E. DENNIS: Generalized pattern searches with derivative information. *Mathematical Programming*, **100** (2004), 3-25.

[4] C. AUDET and J.E. DENNIS: . Analysis of generalized pattern searches. *SIAM Journal on Optimization*, **13**(3), (2003), 889-903.

[5] S.J. CHAPMAN: MATLAB programming for engineers. USA: Brooks & Cole, 2002.

[6] I.D. COOPE and C.J. PRICE: On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization*, **11**(4), (2001), 859-2001.

[7] C. DAVIS: Theory of positive linear dependence. *American J. of Mathematics*, **76** (1954), 733-746.

[8] A. GILAT: MATLAB. An introduction with applications. USA, John Wiley & Sons, Inc., 2005.

[9] M. GEN and R. CHENG: . Genetic Algorithms and Engineering Design. New York, Wiley, 1996.

[10] W. HONGGANG and Z. JIANCHAO: The hybrid genetic algorithm for solving nonlinear programming. *IEEE Int. Conf. on Intelligent Processing Systems*. Beijing, China, (1997).

[11] D.E. GOLDBERG: . Genetic Algorithms in search optimization and machine learning. Toronto, Addison Wesley, 1989.

[12] F. JIMÉNEZ, J.M. CADENAS, G. SÁNCHEZ, A.F. GÓMEZ-SKARMETA and J.L. VERDEGAY: . Multi-objective evolutionary computation and fuzzy optimization. *Int. J. of Approximate Reasoning*, **43** (2006), 59-75.

[13] MATLAB user's Guide. The MathWorks, 2007.

[14] P. VASANT and N. BARSOUM: Hybrid genetic algorithms and line search method for industrial production planning with non-linear fitness function. *Engineering Applications of Artificial Intelligence*, **22**(4-5), (2009), 767-777.

[15] P. VASANT: . Hybrid simulated annealing and genetic algorithms for industrial production management problems. *Int. J. of Computational Methods*, **7**(2), (2010), 279-297.

[16] P. VASANT and N. BARSOUM: Hybrid pattern search and simulated annealing for fuzzy production planning problems. *Computers and Mathematics with Application*, **60**(4), (2010), 1058-1067.