International Journal of Research in Computer Science
eISSN 2249-8265 Volume 4 Issue 3 (2014) pp. 1-11
www.ijorcs.org, A Unit of White Globe Publications
doi: 10.7815/ijorcs.43.2014.084

HYBRID SIMULATED ANNEALING AND NELDER-MEAD ALGORITHM FOR SOLVING LARGE-SCALE GLOBAL OPTIMIZATION PROBLEMS

Ahmed Fouad Ali

Suez Canal University, Dept. of Computer Science, Faculty of Computers and Information, Ismailia, EGYPT
ahmed_fouad@ci.suez.edu.eg

Abstract: *This paper presents a new algorithm for solving large scale global optimization problems based on hybridization of simulated annealing and Nelder-Mead algorithm. The new algorithm is called simulated Nelder-Mead algorithm with random variables updating (SNMRVU). SNMRVU starts with an initial solution, which is generated randomly and then the solution is divided into partitions. The neighborhood zone is generated, random number of partitions are selected and variables updating process is starting in order to generate a trail neighbor solutions. This process helps the SNMRVU algorithm to explore the region around a current iterate solution. The Nelder- Mead algorithm is used in the final stage in order to improve the best solution found so far and accelerates the convergence in the final stage. The performance of the SNMRVU algorithm is evaluated using 27 scalable benchmark functions and compared with four algorithms. The results show that the SNMRVU algorithm is promising and produces high quality solutions with low computational costs.*

Keywords: *Global optimization, Large-scale optimization, Nelder-Mead algorithm, Simulated annealing.*

I. INTRODUCTION

Simulated annealing (SA) applied to optimization problems emerge from the work of S. Kirkpatrick et al. [18] and V. Cerny [3]. In these pioneering works, SA has been applied to graph partitioning and VLSI design. In the 1980s, SA had a major impact on the field of heuristic search for its simplicity and efficiency in solving combinatorial optimization problems. Then, it has been extended to deal with continuous optimization problems [25] and has been successfully applied to solve a variety of applications like scheduling problems that include project scheduling [1, 4], parallel machines [5, 17, 21, 34]. However, implementing SA on the large scale optimization problems is still very limited in comparison with some other meta-heuristics like genetic algorithm, differential evolution, particle swarm, etc. The main powerful feature of SA is the ability of escaping from being trapped in local minima by accepting uphill moves through a probabilistic procedure especially in the earlier stages of the search. In this paper, we produce a new hybrid simulated annealing and Nelder-Mead algorithm for solving large scale global optimization problems. The proposed algorithm is called simulated Nelder-Mead algorithm with random variables updating (SNMRVU). The goal of the SNMRVU algorithm is construct an efficient hybrid algorithm to obtain optimal or near optimal solutions of a given objective functions with different properties and large number of variables. In order to search the neighborhood of the current solution with large variables, we need to reduce the dimensionality. The proposed SNMRVU algorithm searches neighborhood zones of smaller number of variables at each iteration instead of searching neighborhood zones of all the n variables.

Many promising methods have been proposed to solve the mentioned problem in Equation 1, for example, genetic algorithms [14, 24], particle swarm optimization [23, 28], ant colony optimization [31], tabu search [9, 10], differential evolution [2, 6] scatter search [15, 20], and variable neighborhood search [11, 27]. Although the efficiency of these methods, when applied to lower and middle dimensional problems, e.g., $D < 100$, many of them suffer from the curse of dimensionality when applied to high dimensional problems.

The quality of any meta-heuristics method is its capability of performing wide exploration and deep exploitations process, these two processes have been invoked in many meta-heuristics works through different strategies, see for instance [8, 12, 135].

SNMRVU algorithm invoked these two processes and combined them together in order to improve its performance through three strategies. The first strategy is a variable partitioning strategy, which allows SNMRVU algorithm to intensify the search process at each iteration. The second strategy is an effective neighborhood structure, which uses the neighborhood area to generate trail solutions. The last strategy is final intensification, which uses the Nelder-Mead algorithm as a local search algorithm in order to improve the best solutions found so far. The performance of the SNMRVU algorithm is tested using 27 functions, 10 of them are classical function [16] and they are reported in Table 3, the reminder 17 are hard functions [22], which are reported in Table 10. SNMRVU is compared with four algorithms as shown in Section IV, Section V. The numerical results show that SNMRVU is a promising algorithm and faster than other algorithms, and it gives high quality solutions. The paper is organized as follows. In Section II, we define the global optimization problems and give an overview of the Nelder-Mead algorithms. Section III describes the proposed SNMRVU algorithm. Sections IV and V discuss the performance of the proposed algorithm and report the comparative experimental results on the benchmark functions. Section VI summarizes the conclusion.

II. PROBLEM DEFINITION AND OVERVIEW OF THE NELDER-MEAD ALGORITHM

In the following subsections, we define the global optimization problems and present an overview of the Nelder-Mead algorithms and the necessary mechanisms in order to understand the proposed algorithm.

A. Global optimization problems definition

Meta-heuristics have received more and more popularity in the past two decades. Their efficiency and effectiveness to solve large and complex problems has attracted many researchers to apply their techniques in many applications. One of these applications is solving the global optimization problems, this problem can express as follows:

$$\begin{aligned} &Minimize \quad f(x) \\ &Subject\ to \quad l \le x \le u, \end{aligned} \tag{1}$$

Where f(x) is a nonlinear function, $x = (x_1, ..., x_n)$ is a vector of continuous and bounded variables, x, l, u $\in R^n$.

B. The Nelder-Mead algorithm

The Nelder-Mead algorithm [29] is one of the most popular derivative free nonlinear optimization algorithm. The Nelder-Mead algorithm starts with n + 1 point (vertices) as x_1, x_2, ..., x_{n+1}. The algorithm evaluates, order and re-label the vertices. At each iteration, new points are computed, along with their function values, to form a new simplex. Four scalar parameters must be specified to define a complete Nelder-Mead algorithm; coefficients of reflection ρ, expansion χ, contraction γ , and shrinkage σ. These parameters are chosen to satisfy $\rho > 0$, $\chi > 1$, $0 < \gamma < 1$ and $0 < \sigma < 1$.

III. THE PROPOSED SNMRVU ALGORITHM

The proposed SNMRVU starts with an initial solution generated randomly, which consists of n variables. The solution is divided into η partitions, each partition contains υ variables (a limited number of dummy variables may be added to the last partition if the number of variables n is not a multiple of υ). At a fixed temperature, random partition(s) is/are selected, and trail solutions are generated by updating random numbers of variables in the selected partition. The neighbor solution with the best objective function value is always accepted. Otherwise, the neighbor is selected with a given probability that depends on the current temperature and the amount of degradation ΔE of the objective value. ΔE represents the difference in the objective value between the current solution and generated neighboring solution. At a particular level of temperature, many trails solutions are explored until the equilibrium state is reached, which is a given number of iterations executed at each temperature T in our in SNMRVU algorithm. The temperature is gradually decreased according to a cooling schedule. The scenario is repeated until T reached to T_{min}. SNMRVU uses the Nelder-Mead algorithm as a local search algorithm in order to refine the best solution found so far.

A. Variable partitioning and trail solution generating

An iterate solution in SNMRVU is divided into η partitions. Each partition contains υ variables, i.e., η = n/υ. The partitioning mechanism with υ = 5 is shown in Figure 1. The dotted rectangular in Figure 1 shows the selected partition with its variables. Trail solutions are generating by updating all the variables in the selected partition(s). The number of generated trail solutions is μ, the number of the selected partition(s) at each iteration in the algorithm inner loop (Markove chain) is w, where w is a random number, w $\in$ [1, 3]. Once the equilibrium state is reached (a given number

of iterations equal to μ), the temperature is gradually decreased according to a cooling schedule, and the operation of generating new trail solutions is repeated until stopping criteria satisfied, e.g., $T \leq T_{min}$.

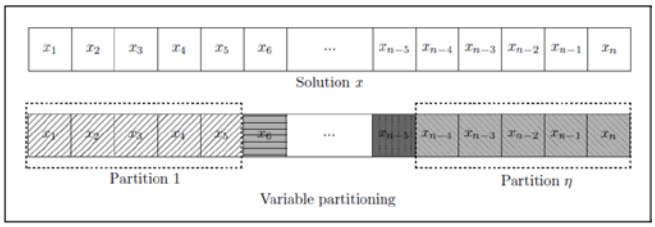


Figure 1: Variable Partitioning

B. The SNMRVU algorithm

SNMRVU algorithm starts with an initial solution x^0 generated randomly. At each iteration the solution is divided into η partitions. The neighborhood zone is generated in order to generate a trail neighborhood solutions. The generated neighbor solution that improves the objective function is always selected. Otherwise the solution is accepted with a given probability $e^{-\Delta E/T}$, where T is the current temperature, and ΔE is the amount of the degradation of the objective function. The scenario is repeated until the equilibrium state is reached. In SNMRVU algorithm, the equilibrium state is a given number of iterations executed at each temperature, this number is equals to μ, μ is a user predefined number. Once the equilibrium state is reached, the temperature is decreased gradually according to a cooling schedule. This process is repeated until the stopping criteria satisfied, which is in our algorithm $T \leq T_{min}$. In order to refine the best solution, SNMRVU uses the Nelder-Mead algorithm as a local search algorithm in the final stage. The structure of the SNMRVU with the formal detailed description is given in Algorithm 1, all variables in Algorithm 1 and its values are reported in Table 1, 2.

Algorithm 1 SNMRVU Algorithm

1. Choose an initial solution x^0 .
2. Set initial values for T_{max}, T_{min}, β, μ, υ , z_0.
3. Set $z= z_0$, $T= T_{max}$, $x=x^0$.
4. **Repeat**
5. k:=0.
6. **Repeat**
7. Partition the solution x into η partitions, where η=n/ υ.
8. Generate neighborhood trail solutions $y^1,...y^n$ around x.
9. Set x' equal to the best trail solution from $y^1,...y^n$.
10. ΔE =f(x')-f(x).
11. **If** ΔE ≤ 0 **Then**
12. x=x'.
13. $z=\tau z_{max}$, τ>1
14. **Else**
15. $z=\alpha z_{min}$, α<1
16. **If** rand( )< $e^{-\Delta E/T}$ **Then**
17. x=x'.
18. **EndIf**
19. **EndIf**
20. k=k+1.
21. **Until** k ≤ μ
22. T=T-β.
23. **Until** $T \leq T_{min}$
24. Apply Nelder-Mead algorithm at N_{elite} solutions.

Table 1: Parameter used in SNMRVU Algorithm

Parameters	Definitions
n	No. of variables
υ	Partition size
η	No. of total partitions
μ	No of maximum trail solutions
ζ	Id no, of the selected partition
w	No of the selected updated partition(s)
z	Radius of neighborhood
z_0	Initial radius of neighborhood
z_{max}	Maximum setting of z
z_{min}	Minimum setting of z
T_{max}	Initial temperature
T_{min}	Final temperature
ΔE	The different objective value between the current solution and the generated solution
β	Temperature reduction value
N_{max}	No. of maximum iteration in the Nelder-Mead algorithm
Nelite	No. of best solutions for intensification

Table 2: Parameter Setting

Parameter	Value
υ	5
η	n/ υ
μ	3
z_{max}	(u-l)/2
z_{min}	(u-l)/50
z_0	$(z_{max+} z_{min)/2}$
T_{max}	n
β	1
N_{max}	5n
Nelite	1

IV. NUMERICAL EXPERIMENTAL

This section presents the performance of our proposed SNMRVU algorithm and the comparison results between it and other four benchmark algorithms. SNMRVU uses a selected two sets of benchmark functions with different properties. The

first set of functions contains of 10 classical functions which listed in Table 3, the second set of functions contains of 17 hard functions from special session on real parameter optimization of IEEE congress on evolutionary computations, CEC2005 [22], the results of the benchmark algorithms and our SNMRVU algorithm are averaged over 30 runs. The numerical results of all tested functions are reported in Tables 4, 5, 6, 7, 9, and 10.

A. Performance analysis

In the following subsection, we present the efficiency of the applied strategies in our proposed SNMRVU algorithm.

Table 3: Classical Functions

f	Function name	Bounds	Global minimum
F1	Branin Function	[-5,10]	$F(x^*)= 0.397887$
F2	Booth Function	[-10,10]	$F(x^*)= 0$
F3	Matyas Function	[-5,10]	$F(x^*)= 0$
F4	Rosenbrock Function	[-5,10]	$F(x^*)= 0$
F5	Zakhrof Function	[-5,10]	$F(x^*)= 0$
F6	Trid Function	$[-n^2,n^2]$	at n=6, $F(x^*)= -50$
F7	SumSqure function	[-5,10]	$F(x^*)= 0$
F8	Sphere Function	[-100,100]	$F(x^*)= 0$
F9	Staircased Rosenbrock	[-5,10]	$F(x^*)= 0$
F10	Staircased LogAbs Function	[-5,10]	$F(x^*)= 0$

1. The efficiency of variables partitioning:

In order to check the efficiency of variable partitioning, we compare our SNMRVU algorithm with the basic SA algorithm using the same termination criteria and the same SA parameters, the results are shown in Figure 2. The dotted line represents the results of the basic SA (without partitioning mechanism), the solid line represents the result SNMRVU algorithm with the variables partitioning process. f_2 is selected with dimensions 100, 500 and 1000 by plotting the number of iterations versus the function values. Figure 2 shows that the function values of SNMRVU are rapidly decreases as the number of iteration increases rather than the performance of the basic SA. We can conclude from Figure 2 that exploring the neighborhood of all variables at the same time as the basic SA do, can badly effects the progress of the search. However, exploring the neighborhood gradually through partition with small number of variables as applied in the SNMRVU algorithm can give better results.

2. The performance of the Nelder-Mead algorithm

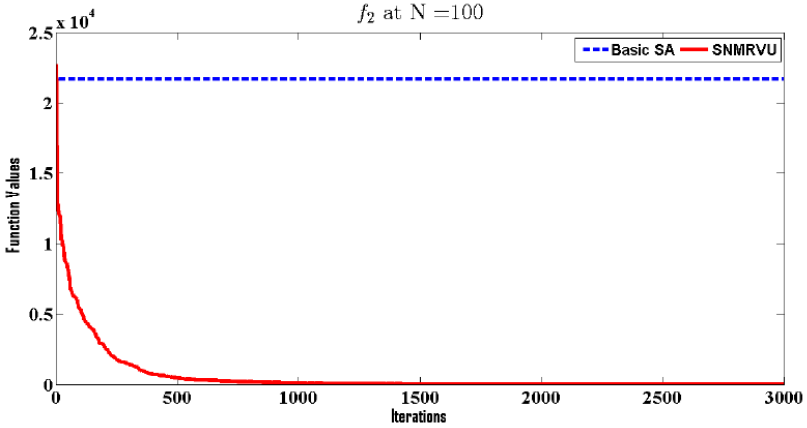
SNMRVU uses the Nelder-Mead algorithm as an intensification process in order to accelerate the convergence in the final stage instead of let the algorithm run for several iterations without significant improvement of the function values. Figure 3 shows the performance of the Nelder-Mead algorithm on two functions f_3, h_2 by plotting the number of iterations versus the function values. The dotted line in Figure 3 represents the behavior of the final intensification process.

B. SNMRVU and other algorithms.

We compared our SNMRVU algorithm with four benchmark algorithm. The results of all algorithms are taken from its original papers.

1. SSR (Scatter search with randomized subset generation).

In [16] two scatter search methods were proposed, they called SSR, SSC, and some computational testes were applied on classical direct search methods [7, 19, 30, 32, 33, 36, 37], and joined them with scatter search method. In SSR method, a subset is created in two steps. First, the subset size is decided, and second selecting the solutions from the reference set randomly until the required subset size is obtained. SSR uses biased randomized method in order to select the subset size. The different size probabilities are adjusted dynamically based on search history.



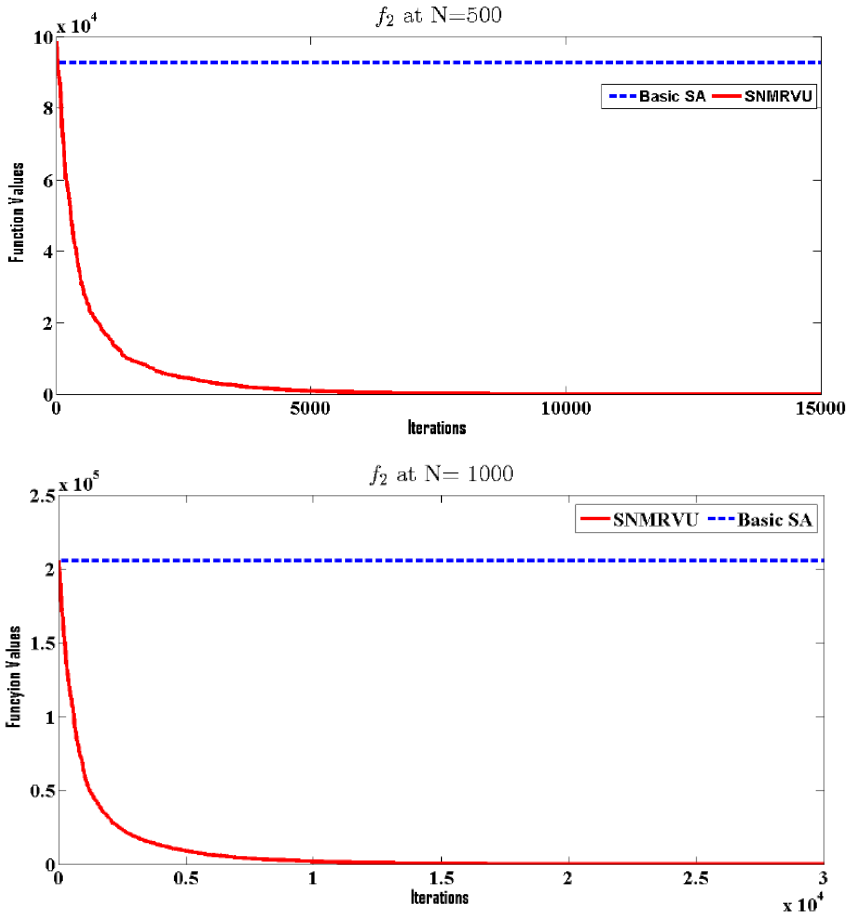


Figure 2: Basic SA vs. SNMRVU.

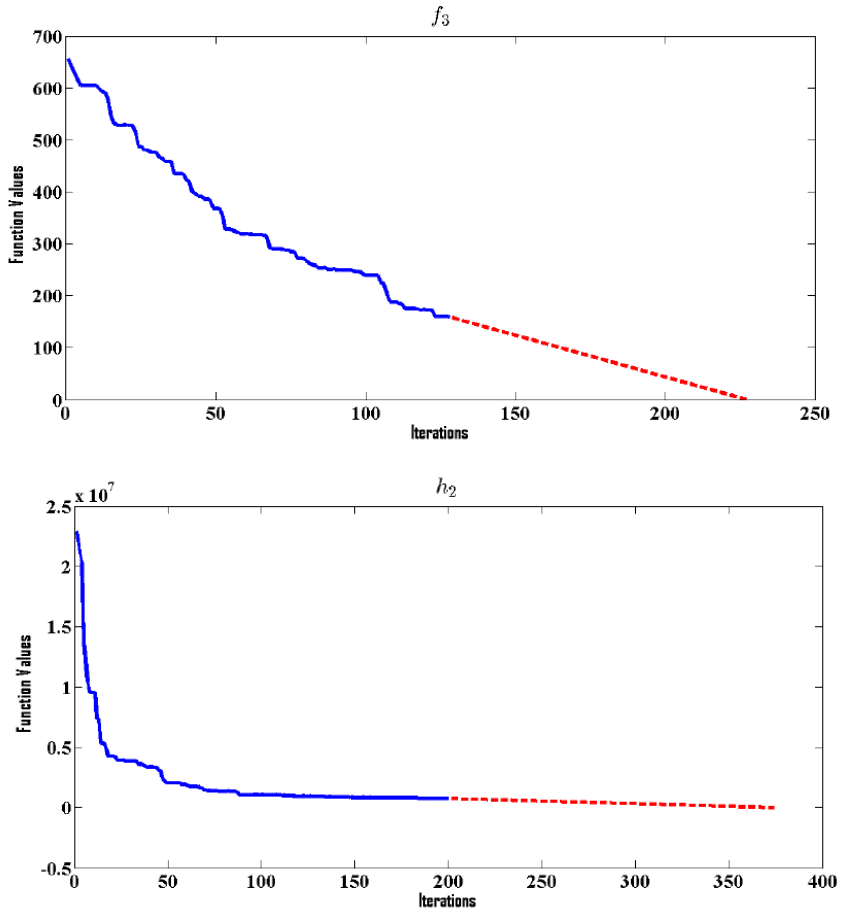


Figure 3: The Performance of the Nelder-Mead Algorithm.

2. *SSC (Scatter search with clustering subset generation).*

SSC method uses k clustering algorithm [26] in order to create subsets by dividing the reference set points into k clusters. The cluster are then ranked by the quality of the points, which they contain, and the cluster are selected as subset sequentially.

C. *Comparison between SNMRVU, SSR and SSC on functions with 16-512 dimensions.*

The proposed SNMRVU algorithm was tested and compared with SSR, SSC algorithm on 10 benchmark functions, which reported in Table 3. The dimensions of functions are 16, 32, 64, 128, 256, and 512, respectively, the results of the three algorithms are reported in Tables 4, 5, 6. SNMRVU was run 30 times with different initial solutions, the average of function evaluation values of each test function is reported. In the case where at least one run fails to the global solution, to produce a converged point, the ratio of successful run is recorded in parentheses. The experimental results for f_5 in Table 6 at 256, 512 dimensional and f_6 in Table 5 at 128 dimensional were not reported in its original paper [16]. The best mean values of the four methods are marked in bold face. SNMRVU algorithm can obtain better function values if we apply one termination criterion, which is the number of evaluation function values is $\leq 50,000$.

Table 4: Function Evaluations of SSR, SSC and SNMRVU with 16, 32 Dimensions.

	16 dimensions			**32 dimensions**		
F	SSR	SSC	SNMRVU	SSR	SSC	SNMRVU
F_1	**406.6**	374.2	456.4	**779.7**	783.2	1086.2
F_2	806.4	737.2	**218.3**	1732.4	1436.9	**393.6**
F_3	1109.8	902.1	**217.3**	2422.8	1834.6	**327.1**
F_4	(0.9)	12500.4	**5134**	(0.9)	21397.7	**13213.4**
F_5	6827.8	4030.5	**439.4**	32626.4	16167.1	**888.3**
F_6	4357.8	3447.5	**643.1**	(0.8)	(0.3)	**2076**
F_7	**298.3**	344.7	312.6	**622.5**	647.5	734.5
F_8	227.2	323.8	**133.4**	462.5	463.2	**261.5**
F_9	31978.3	13575.7	**2645.3**	26315.3	22706.1	**5112.3**
F_{10}	441.8	445.8	**395.4**	893.7	896.3	**698.7**

Table 5: Function Evaluations of SSR, SSC and SNMRVU with 64, 128 Dimensions.

	64 dimensions			**128dimensions**		
F	SSR	SSC	SNMRVU	SSR	SSC	SNMRVU
F_1	1636.6	**1626.4**	1996.1	3448.8	**3443.7**	4130.2
F_2	3154.8	3148.5	**842.8**	6737.8	7516.3	**1545.4**
F_3	3854.2	3827.1	**777.4**	8575	9687.9	1674
F_4	**39193.2**	(0.2)	39212.3	(0.0)	(0.0)	(0.0)
F_5	(0.0)	(0.1)	**1654.2**	(0.0)	(0.0)	**4125.3**
F_6	(0.0)	(0.0)	**7407.3**	-	-	**34440.3**
F_7	1369	**1348.4**	1996.4	2840.7	**2808.8**	4135.2
F_8	958	983.9	**517.3**	1979.4	2111.7	**1029.1**
F_9	**37090.5**	(0.0)	(0.2)	(0.0)	(0.0)	(0.0)
F_{10}	1858.8	1861	**1559.7**	3880.2	3899.9	**2128.9**

Table 6: Function Evaluations of SSR, SSC and SNMRVU with 256, 512 Dimensions.

	256 dimensions			**512 dimensions**		
F	SSR	SSC	SNMRVU	SSR	SSC	SNMRVU
F_1	**7173.4**	7184.6	8273.7	**15206.9**	15221.8	1734.8
F_2	16293.8	16915.3	**3081.3**	36840	36635.4	**7179.2**
F_3	24207	24903.1	**3338.1**	(0.0)	(0.0)	**6656.8**
F_4	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)
F_5	-	-	**9113.1**	-	-	**18312.5**
F_6	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)
F_7	**6057.1**	6105.4	7934.3	**12866.2**	13021.4	23215.7
F_8	4236.8	4258.6	**2310.3**	8890.4	8890.2	**4614.1**
F_9	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)	(0.0)
F_{10}	8070.3	8212.7	**6811.2**	17180.3	16947.2	**12593.3**

Table 7 shows the mean number of function values. For function f_1, we obtained the exact global minimum value for all given dimensions with function evaluation values ≤ 50, 000. Also we obtained the exact global minimum values for functions f_6, h_2 till n = 128, we can reach to the global minimum of these functions for n > 128 by increasing the number of function evaluation values, for example for function f_6 our SNMRVU algorithm can obtains the value -2828800, which is the global minimum of it within a cost of 97400 at dimension n = 256. For the other functions, we obtained very close results to their global minimum. We can conclude that, the SNMRVU algorithm is robust and can obtains better function values with lower computational cost than the other two algorithms values with lower computational cost than the other two algorithms.

Table 7: Mean Number of Function Values with 16- 512 Dimensions.

	Mean number of function values		
F	16	32	64
F_1	0	0	0
F_2	2.15e-11	5.33e-12	2.7e-11
F_3	1.73e-13	2.6e-08	1.31e-12
F_4	2.34e-09	6.86e-10	3.17e-08
F_5	2.1e-10	0	1.34e-08
F_6	0	1.77e-08	0
F_7	1.44e-09	0	7.88e-08
F_8	0	1.75e-03	0
F_9	5.3e-04	1.44e-09	(0.0)
F_{10}	4.12e-09		2.45e-07
F	128	256	512
F_1	0	0	0
F_2	7.14e-11	9.06e-10	3.34e-09
F_3	5.57e-11	6.56e-11	2.12e-09
F_4	(0.0)	(0.0)	(0.0)
F_5	6.01e-07	1.54e-05	4.32e-05
F_6	0	(0.0)	(0.0)
F_7	3.85e-07	2.11e-06	9.85e-06
F_8	4.9e-07	1.42e-14	2.81e-14
F_9	(0.0)	(0.0)	(0.0)
F_{10}	4.3e-07	3.12e-04	1.12e-04

V. HARD BENCHMARK TEST FUNCTIONS

In this section, we evaluate the SNMRVU algorithm with another type of functions with various properties provided by CEC2005 special session [23]. These functions are based on classical benchmark functions after applying some modifications. These modifications make these functions harder to solve and resistance to simple search. We applied our SNMRVU algorithm on 17 functions as reported Table 8. The functions in Table 8 are shifted, rotated, expanded and combined variants of classical functions such as sphere's, Rastrigin's, Rosenbrock's, Ackley's function. In the next subsection, we study the performance of SNMRVU on some hard functions, before comparing the results of SNMRVU algorithm with the results of two algorithms.

A. The performance of SNMRVU algorithm with the hard functions.

Figure 4 represents general performance of SNMRVU algorithm on three functions h_1, h_2, h_6, with different properties by plotting the number of iterations versus the function values. Figure 4 shows that the function values are rapidly decreases as the number of iterations increases, and the performance of SNMRVU algorithm is promising and can obtain good solutions with some of hard functions.

We compared our SNMRVU algorithm with the following methods:

Table 8: Benchmark Hard functions

H	Function name	Bounds	Global minimum
H1	Shifted Sphere	[-100,100]	-450
H2	Shifted Schwefel's 1.2	[-100,100]	-450
H3	Shifted rotated high conditioned elliptic	[-100,100]	-450
H4	Shifted Schwefel's 1.2 with noise in fitness	[-100,100]	-450
H5	Schwefel's 2.6 with global optimum on bounds	[-100,100]	-310
H6	Shifted Rosenbrock's	[-100,100]	390
H7	Shifted rotated Griewank's without bounds]	[0,600]	-180
H8	Shifted rotated Ackley's with global optimum	[-32,32]	-140
H9	Shifted Rastrigin's	[-5,5]	-330
H10	Shifted rotated Rastrigin's	[-5,5]	-330
H11	Shifted rotated Weierstrass	[-0.5,0.5]	90
H12	Schwefel's 2.13	[-100,100]	-460
H13	Expanded extended Griewank's + Rosenbrock's	[-3,1]	-130
H14	Expanded rotated extended Sca_e's	[-100,100]	-300
H15	Hybrid composition 1	[-5,5]	120
H16	Rotated hybrid comp.	[-5,5]	120
H17	Rotated hybrid comp. Fn 1 with noise in fitness	[-5,5]	120

1. *DECC-G(Deferential Evolution with cooperative co-evolution with a group-based problem) [37].*

DECC-G designed to increase the chance of optimizing interacting variable together by changing grouping structure dramatically. It splits an objective vector into smaller subcomponents and evolves each component with an evolutionary algorithm EA. After each recycle the DECC-G applied a weight to each subcomponent and evolves the weight vector with a certain optimizer. DECC-G uses one termination criterion, which is the number of evaluation number is 2.5 e^6, 5 e^6, for n = 500, 1000, respectively.

2. *AMALGAM-SO (A multi-algorithm genetically adaptive method for single objective optimization) [35].*

AMALGAM-SO is a hybrid evolutionary search algorithm. It considers the covariance matrix adaptation (CMA), genetic algorithm (GA), evolutionary strategy, deferential evolution(DE), parental-centric recombination operator (PCX), and particle swarm optimizer (PSO) for population evolution. It applies a self-adaptive learning strategy to favor individual algorithm that demonstrate the highest reproductive success during the search. The termination criterion of AMALGAM-SO method is to reach the global minimum of some functions within the tolerance limit 10^{-6} and 10^{-6} for the others.

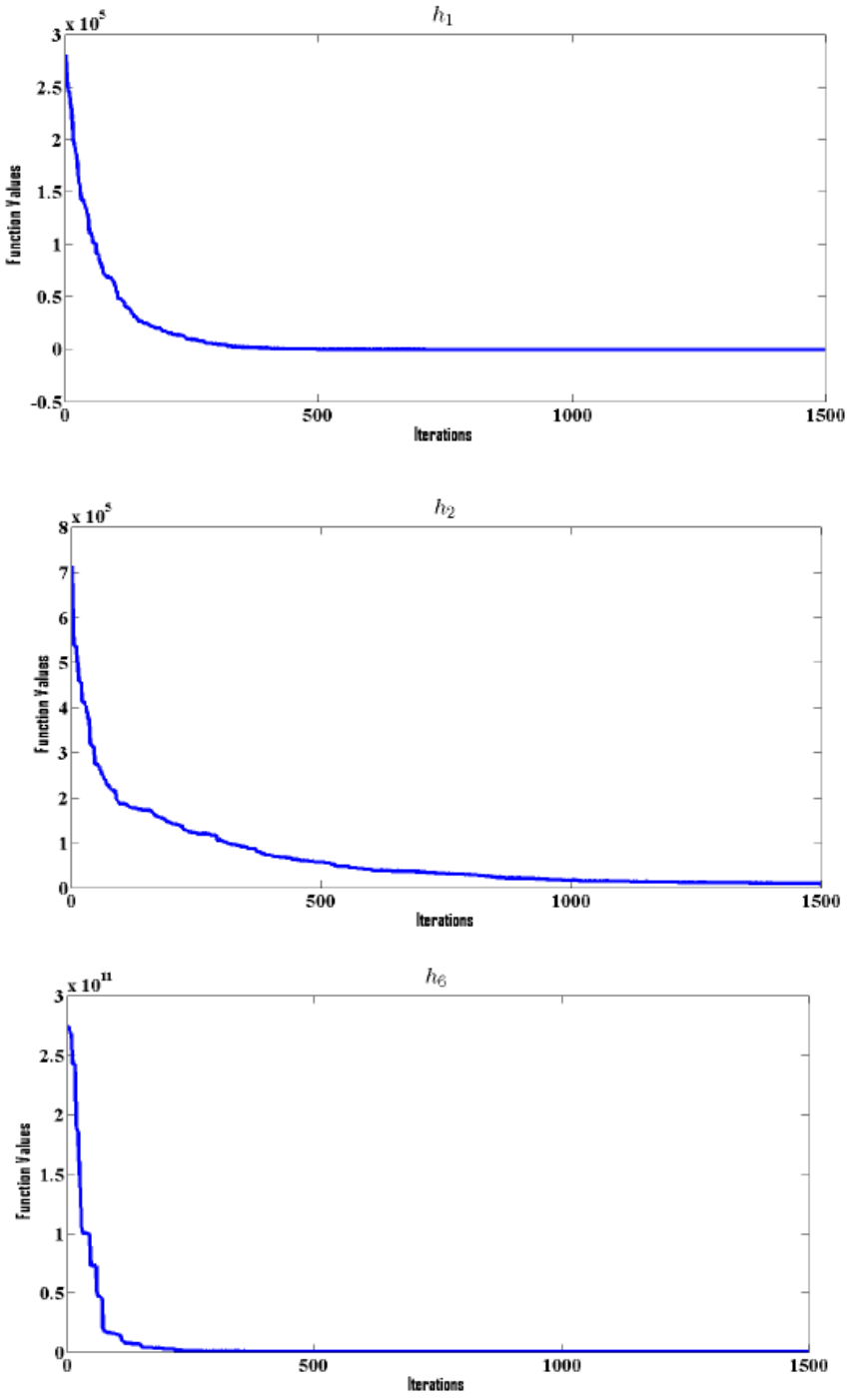


Figure 4 : The Performance of SNMRVU Algorithm With the Hard Functions.

B. Comparison between AMALGAM-SO and SNMRVU.

We compare the SNMRVU algorithm with AMALGAM-SO algorithm. The two algorithms are tested on 17 benchmark functions which listed in Table 8. The average means and standard deviations are listed in Tables 9. These results for 50 dimensions. The results between parentheses represent the average of the final objective function values, when none of the optimization runs reached to the recorded tolerance limit in Tol column in Table 9. The reported results in Table 9 show that the SNMRVU algorithm is outperforms the other algorithm for functions h_1, h_2, h_3,h_6, h_7, h_9, h_{10}, h_{12}, h_{13}. The AMALGAM-SO results for functions h_{10}, h_{12} are not reported in its original paper. The two algorithms are unable to locate solutions within the tolerance limit for function h_8, h_{13}, h_{14}, h_{16}, h_{17}.

Table 9: Functions 1-17 in Dimension n = 50: Mean Number of Function Evaluations.

		AMALGAM-SO		SNMRVU	
H	Tol	Mean	Std	Mean	Std
H_1	1e-6	7907	209	**4653**	9.12
H_2	1e-6	3.41e4	681	**14496**	792.13
H_3	1e-6	1.29e5	18.3	**29732**	195.14
H_4	1e-6	**3.48e5**	1.67e5	(3.116e3)	(495.19)
H_5	1e-6	**4.92e5**	2.68e4	(1.127e4)	(1532.4)
H_6	1e-2	2.11e5	1.2e5	**39546.24**	3245.1
H_7	1e-2	9596	463	**7214**	1.15
H_8	1e-2	(21.13)	(0.03)	**(19.99)**	(8.4e-06)
H_9	1e-2	4.76e5	4.09e4	**10342.11**	95.2
H_{10}	1e-2	-	-	**(198.371)**	(114.5)
H_{11}	1e-2	**2.65e5**	1.16e5	(14.23)	(3.425)
H_{12}	1e-2	-	-	**24132.21**	1854.4
H13	1e-2	(3.62)	(1.08)	**(1.945)**	(2.241)
H_{14}	1e-2	(19.69)	(0.92)	**(19.5)**	(0.45)
H15	1e-2	**4.93e5**	2.81e4	(9.24)	(8.14)
H16	1e-2	**(12.9)**	(3.73)	(22.63)	(66.4)
H17	1e-2	**(65.63)**	(59.58)	(74.5)	(61.44)

C. Comparison between DECC-G and SNMRVU.

In order to test the performance of SNMRVU algorithm with dimensions ≥ 500, SNMRVU is compared with DECC-G algorithm for 500, 1000 dimensions. The average means of objective function values and function evaluations are recorded in Table 10. The results in Table 10 shows that the SNMRVU performed significantly better on functions h_1, h_5, h_6, h_8, h_9 and with cheaper cost than the other algorithm.

Table 10 : Mean Number of Function Values and Function Evaluation with 500, 1000 Dimensions.

		Function Values	
H	n	DECC-G	SNMRVU
H_1	500	3.71e-13	0
	1000	6.84e-13	0
H_3	500	**3.06e8**	9.84e8
	1000	**8.11e8**	3.45e9
H_5	500	1.15e5	**1.09e5**
	1000	2.20e5	**2.14e5**
H_6	500	1.56e3	**4.15e5**
	1000	2.22e3	**8.15e2**
H_8	500	2.16e1	**2.0e1**
	1000	2.16e1	**2.0e1**
H_9	500	4.5e2	**1.20e2**
	1000	6.32e2	**2.11e2**
H_{10}	500	**5.33e3**	1.12e4
	1000	**9.73e3**	2.21e4
H13	500	**2.09e2**	1.15e3
	1000	**3.56e2**	3.86e3
		Function Evaluations	
H	n	DECC-G	SNMRVU
H_1	500	2.5e6	**29455**
	1000	5.00e6	**56324**
H_3	500	2.5e6	**69345**
	1000	5.00e6	**169521**
H_5	500	2.5e6	**65301**
	1000	5.00e6	**132547**
H_6	500	2.5e6	**134520**
	1000	5.00e6	**233548**
H_8	500	2.5e6	**63124**
	1000	5.00e6	**139587**
H_9	500	2.5e6	**63548**
	1000	5.00e6	**91247**
H_{10}	500	2.5e6	**74501**
	1000	5.00e6	**166487**
H13	500	2.5e6	**121458**
	1000	5.00e6	**211485**

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a SNMRVU algorithm in order to solve large scale global optimization problems. SNMRVU uses three strategies, the first strategy is variable partitioning process, which help our algorithm to achieve high performance with high dimensional problems, the second strategy is the effective neighborhood structure strategy, which help the SNMRVU algorithm to explore region around a current iterate solution by updating variables in the random selected partitions and the last strategy is the final intensification by applying the Nelder-Mead algorithm in the final stage. Invoking these three strategies together in SNMRVU algorithm represents the main difference between it and the other related

methods existing in the literature. SNMRVU algorithm is evaluated on a set of 27 benchmark functions with various properties, 10 classical functions and 17 hard functions. The obtained results give evidence that the SNMRVU algorithm is promising algorithm for solving large scale optimization problems, and in most cases its present significantly better results and cheaper than the other algorithms. In future, we intend to develop new operators to apply our SNMRVU algorithm in higher dimensional problems till 10,000 dimension.

VII. REFERENCES

[1] K. Bouleimen, H. Lecocq. "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version". European Journal of Operational Research, 149:268-281, 2003. doi: 10.1016/s0377-2217(02)00761-0.

[2] J. Brest, M. Maucec. "Population size reduction for the differential evolution algorithm". Appl Intell 29:228-247, 2008. doi: 10.1007/s10489-007-0091-x.

[3] V. Cerny. "A thermodynamical approach to the traveling salesman problem". An efficient simulation algorithm, Journal of Optimization Theory and Applications,45:41-51, 1985. doi: 10.1007/bf00940812.

[4] H. Cho, Y. Kim. "A simulated annealing algorithm for resource constrained project scheduling problems".Journal of the Operational Research Society, 48: 736-744, 1997. doi: 10.2307/3010062.

[5] P. Damodaran, M. Vlez-Gallego. "A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times" .Expert Systems with Applications, 39:1451-1458, 2012. doi: 10.1016/j.eswa.2011.08.029.

[6] S. Das, A. Abraham, U. Chakraborty, A. Konar. "Differential evolution using a neighborhood-based mutation operator" . IEEE Trans Evol Comput, 13(3):526-553, 2009. 10.1109/tevc.2008.2009457.

[7] E. Dolan. " Pattern Search Behaviour in Nonlinear Optimization". Thesis, 1999.

[8] S. Garcia, M. Lozano, F. Herrera, D. Molina, A. Snchez. "Global and local real-coded genetic algorithms based on parentcentric crossover operators". Eur J Oper Res 185:1088-1113, 2008. doi: 10.1016/j.ejor.2006.06.043.

[9] F. Glover, E. Taillard, D. Werra. "A user's guide to tabu search". Annals of operations research. 41 (1), pp 1-28. 1993. doi: 10.1007/BF02078647.

[10] M. Hadi, M. Orguna, W. Pedryczb. "Hybrid optimization with improved tabu search". Appl Soft Comput, 11(2):1993-2006, 2011. doi: 10.1016/j.asoc.2010.06.015.

[11] P. Hansen, N. Mladenovic, M. Prezreno. "Variable neighborhood search : methods and applications". Ann Oper Res, 175(1):367-407, 2010. doi: 10.1007/s10479-009-0657-6.

[12] Hedar, AF. Ali. "Tabu search with multi-level neighborhood structures for high dimensional problems". Appl Intell, 37:189-206, 2012. doi: 110.1007/s10489-011-0321-0.

[13] A. Hedar, A. F. Ali. "Genetic algorithm with population partitioning and space reduction for high dimensional problems", in: Proceeding of the 2009, International Conference on Computer Engineering and Systems, (ICCES09), PP.151-156 Cairo, Egypt 2009. doi: 10.1109/icces.2009.5383293.

[14] F. Herrera, M. Lozano, J. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioral analysis". Artif Intell Rev, 12:265-319, 1998. doi: 10.1023/a:1006504901164.

[15] F. Herrera, M. Lozano, D. Molina. "Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies". Eur J Oper Res, 169(2):450-476, 2006. doi: 10.1016/j.ejor.2004.08.009.

[16] L. Hvattum, F. Glover, "Finding local optima of high-dimensional functions using direct search methods", European Journal of Operational Research, 195:31-45, 2009. doi: 10.1016/j.ejor.2008.01.039.

[17] D. Kim, K. Kim, F. Chen, " Unrelated parallel machine scheduling with setup times using simulated annealing", Robotics and Computer-Integrated Manufacturing, 18:223-231, 2002. doi: 10.1016/s0736-5845(02)00013-3.

[18] S. Kirkpatrick, C. Gelatt, M. Vecchi. "Optimization by simulated annealing", Science,220(4598):671-680, 1983. doi: 10.1016/s0736-5845(02)00013-3.

[19] T. Kolda, R. Lewies, V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods", SIAM Review,45:385-482, 2003. doi: 10.1137/s003614450242889.

[20] M. Laguna, R. Mart, "Experimental testing of advanced scatter search designs for global optimization of multimodal functions" , J Glob Optim;33(2):235-255, 2005. doi: 10.1007/s10898-004-1936-z.

[21] W. Lee, W. Ch, P. Chen. "A simulated annealing approach to make span minimization on identical parallel machines". International Journal of Advanced Manufacturing Technology, 31:328-334, 2006. doi: 10.1007/s00170-005-0188-5.

[22] J. Liang, P. Suganthan, K. Deb, " Novel composition test functions for numerical global optimization, in: Proceedings of 2005 IEEE Swarm Intelligence Symposium, 8-10 June, pp.68-75, 2005. doi: 10.1109/sis.2005.1501604.

[23] J. Liang, A. Qin, P. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput, 10(3):281-295, 2006. doi: 10.1109/tevc.2005.857610.

[24] Y. Li, X. Zeng, "Multi-population co-genetic algorithm with double chain-like agents structure for parallel global numerical optimization", Appl Intell, 32: 0, 292-310, 2010. doi: 10.1007/s10489-008-0146-7.

[25] M. Locatelli, " Simulated annealing algorithms for continuous global optimization: Convergence conditions", Journal of Optimization Theory and Applications, 29(1):87-102, 2000. doi: 10.1023/A:1004680806815

[26] L. MacQueen, " Some methods for classification and analysis of multivariate observations", In:L.M. LeCam, N. Neyman (Eds.), Proceedings of 5th Berkeley

Symposium on Mathematical Statistics and Probabilty, University of California press, 281-297, 1967.

[27] N. Mladenovic, M. Drazic, V. Kovac, M. Angalovic, " General variable neighborhood search for the continuous optimization", Eur J Oper Res, 191:753-770, 2008. doi: 10.1016/j.ejor.2006.12.064.

[28] M. Montes, T. Stutzle, M. Birattari, M. Dorigo, "PSO: a composite particle swarm optimization algorithm", IEEE Trans Evol Comput, 13(5):1120-1132, 2009. doi: 10.1109/TEVC.2009.2021465

[29] J. Nelder, R. Mead, " A simplex method for function minimization", Computer journal, 7:308-313, 1965. doi: 10.1093/comjnl/7.4.308.

[30] C. Rosin, R. Scott, W. Hart, R. Belew, "A comparison of global and local search methods in drug docking". In: thomas Bck(Ed.), Proceeding of the Seventh international Conference on Genetic Algorithms, (ICGA97), Morgan Kaufmann, San Francisco, PP.221-228, 1997.

[31] K. Socha, M. Dorigo, "Ant colony optimization for continuous domains", Eur J Oper Res, 185(3):1155-1173 2008.doi: 10.1016/j.ejor.2006.06.046.

[32] F. Solis, R. Wets, "Minimization by random search techniques", Mathematical Operations Research 6:19-30, 1981. doi: 10.1287/moor.6.1.19.

[33] J. Spall, "Multivariate stochastic approximation using a simulation perturbation gradient approximation", IEEE Transactions on Automatic Control, 37:332-341, 1992. doi: 10.1109/9.119632.

[34] V. Torezon, "Multi-Directional Search, A Direct search algorithm for parallel machines", Phd thesis, Rice University 1989.

[35] J. Vrugt, A. Bruce, J. Hyman, Self-adaptive multimethode serach for global optimization in real parameter spaces. IEEE Transactions on Evolutionary Computation, 13, 2009. doi: 10.1109/tevc.2008.924428.

[36] M. Wright, D. Griffhs, G. Watson, "Direct search methods, Numerical Analysis 1995. Addison wesey longman, Harlow, United Kingdom 191-208, 1996.

[37] Z. Yang, K. Tang, X. Yao, " Large scale evolutionary optimization using cooperative coevolution. Information Sciences, 178:2986-2999, 2008. doi: 10.1016/j.ins.2008.02.017

How to cite

Ahmed Fouad Ali, "Hybrid Simulated Annealing and Nelder-Mead Algorithm for Solving Large-Scale Global Optimization Problems". *International Journal of Research in Computer Science*, 4 (3): pp. 1-11, May 2014. doi: 10.7815/ijorcs.43.2014.084