VOL. 3, NO. 5, May 2013 ISSN 2225-7217
ARPN Journal of Science and Technology

http://www.ejournalofscience.org

A Hybrıd Partıcle Swarm Genetıc Algorıthm (Psoga) For N-Queen Problem

[1]A.A. Ojugo., [2] E.R. Yoro., [3] A.O. Eboka., [4] I.J.B. Iyawa, [5] M.O. Yerokun
[1] Department of Mathematics/Computer Sci, Federal University of Petroleum Resources Effurun, Delta State
[2] Department of Computer Science, Delta State Polytechnic Ogwashi-Uku
[3,4,5] Department of Computer Science, Federal College of Education (Technical), Asaba, Delta State.
[1] ojugo_arnold@yahoo.com,[1] ojugoarnold@hotmail.com, [2] rumerisky@yahoo.com,[3] agapenexus@hotmail.co.uk,[4] iyawaben@hotmail.com

ABSTRACT

Evolutionary algorithms have proven to be robust tools in data processing for modeling dynamic, non-linear and complex processes due to their flexible mathematical structure to yield optimal results even with imprecise, ambiguity and noise at its input. The study is a soft-computing heuristic-based, hybrid approach that aims to provide computational intelligence via solving optimization task. The hybrid will become a veritable algorithm for computing dynamic and discrete states for multipoint search in CSPs tasks with application areas to include image and video analysis, network design and construction, communication, simulation, multiprocessor load balancing, OS task scheduling/resource allocation, parallel processing, power generation, medicine, economics, security/military, fault diagnosis and recovery, forecasting and predictions, data mining, signal processing, cloud and clustering computing to mention a few. The hybrid algorithm: (a) via particle swarm optimization, places a number of simple agents or particles in space so that each can evaluate the objective, fitness function. Thus, each particle determines its movement around the space by combining some aspects of its own current and best locations with those of other members of the swarm, along with some random perturbation. The next iteration occurs after all the particles have moved to their new locations with updated velocities and the entire swarm will tend to move closer to optimal, and then (b) via genetic algorithm (GA), it defines the swarm of particle via three (3) operators (selection, crossover and mutation) to result in unique values. Crossover and mutation factors will prevent the particle best (pbest) from entrapment at local minima and study contributes in its design of a hybrid PSOGA model for implementation of an N-Queen classical CSP task to provide computational intelligence.

Keywords: *swarms, evolutionary algorithms, soft-computing, constraints, objective function, fitness function*

1. INTRODUCTION

Soft Computing (SC) aims to harness Artificial Intelligence as a synergy of other fields, dedicated to solve problems by exploiting numeric data and human knowledge simultaneously via mathematical models and symbolic reasoning – to yield a technique that is tolerant to imprecision, uncertainty, partial truth and noise in data via optimization – so as to make such models as soft as human brain (Abarghouei, Ghanizadeh and Shamsuddin, 2009).

Real world optimization requires tuning to be robust, so that even with noise employed at its input, its output value and solution is guaranteed of high quality. Studies show that tuning search method to find robust optima has furthered the field to yield advances termed Evolutionary Programming Algorithms – capable of performing both quantitative (numeric) and qualitative data processing that ensures qualitative statements of knowledge and experience in form of natural languages. SC spans across several branches as inspired by evolution, natural laws and behavioural patterns in biological populations. Examples include GA, ANN etc (Parsopoulos and Vrahatis, 2004) – all metaheuristic optimization in constraint satisfaction problems in vector space made up of intelligent agents that searches a space for its optimal fitness. SC ideas mimics natural agents seeking food and have proven efficient in complex optimization. Kennedy and Eberhart (1997) notes robust optimization three (3) feats in their attempt to explore dynamic processes:

a. **Robustness** helps to estimate system's effectiveness even with noise implementation.
b. **Continuous adaptation** yield agents void of local minima, balances exploitation versus exploration, and introduce random agents of high diversity to slow convergence in space so that, learning feats of change and yields accordingly, a biased result,
c. **Flexibility** aids effective decision making even with uncertainty at the model's input that impacts future. Thus, model predicts future with an algorithm that focuses on both objective function and facilitate adaptation with blackbox integration.

PSO or GA is derived from translating into mathematical models, principles of biological processing in the fastest time to yield implicit and predictive evolution of a model that stems from experience in its ability to recognize data feats and behaviours. And in turn, yield an optimal fitness of high quality and void of overfitting, irrespective of modification via other approximations that uses multiple agents. These cannot be ignored as they constantly affect any solution's quality (Coello, Pulido and Lechuga, 2002, Heppner and Grenander, 1990).

1.1 Particle Swarm Optimizaton (PSO)

PSO is a heuristic method inspired by the stochastic process of swarm and collaborative behavior of biological population and uses cooperative swarming where each particle represents a candidate solution in the swarm and tends to explore its solution space to a CSP task. It is modeled on swarm intelligence that finds solution via predicting social behavior in the presence of an objective function with a number of agents/particles in space. Each, evaluate a fitness or objective function at its current location, and determines its movement by combining aspects of its own best current location (pbest) with those of other swarm members of random perturbation. With each iteration, particles move to their new locations with updated feats – so the whole swarm approaches the objective function (Eberhart and Kennedy, 1995). Each particle determines its movement by combining its own current and best locations with those, of other members – to yield a new swarm that tends to move closer to the optimal solution.

Its application is in multi- dimensional and objective tasks (Parsopoulos and Vrahatis, 2002). Particles are randomly optimized with a space to generate particles or solutions. With each iteration – each particle evaluates its own fitness (best location based on objective function) cum the fitness of other particles following the current best performing particle (Cello et al., 2002; Hu et al., 2005). Each particle keeps track of its own solution that resulted in the best fitness and also sees the candidate or particle solution for the best performing particle in the neighborhood (Heppner and Grenander, 1990).

1.2 Genetic Algorithm (GA)

Holland (1975) proposed GA as search method inspired by same process of evolution cum genetics and uses hill climbing method from an arbitrary selected number of genes to find an approximate solution in a search space and it has four operators: **initialization**, **selection**, **crossover** and **mutation** (Hassan and Crosswley, 2004 and Hassan, Cohanin, De Wec and Venter, 2006). Model starts with randomly initialized population and evolve to represent a solution in space, each of which comprise of a set of candidates, whose quality is the measure of its fitness function and quantitative representation of such candidates (Homaifar, Turner and Ali, 1992). Thus, **selection**, **crossover** and **mutation** operators are applied to each pool of solutions so as to gradually improve their quality. Competition for food and space allows the swarm to evolve – where stronger offspring dominate weaker ones – a desired improvement over various moves. Solutions with better fitness value are more likely to reproduce offspring – so that only the fittest agents survive and reproduce (Dozier and Carlisle, 2001).

The reproductive process creates diversity in the pool via the process of evolution (combination of two or more individual solutions) as new candidates are created from previous ones in order to generate new pool. Genetic exchange between two parents results in **crossover** to create a better, fitter solution or candidate. Repeated selection (survival of fittest) and crossover causes continuous evolution of a generation that will better survive in a competitive state. Mutation causes candidates' sporadic or random alteration, and helps in regeneration of lost genetic materials (Rudolph, 1996). If newly generated pool contains an output close enough to desired value, the solution has been found; Else, new pool continues in the same process until a solution is reached, or till a number of generations is produced.

GA's strength resides in parallel traversing of a system by proposing solutions whose initial population is randomly generated and are continuously evaluated via a fitness function and GA and its variants, have been successfully applied to areas, depending on the nature of the task to yield new generations (via crossover and mutation) with its iterations repeated until a termination state is reached (Poli, Wright, McPhee and Langdon, 2006b).

1.3 N-Queen Overview

This is a classical CSP task with same number of variables as values that allows permutation of such variables to yield solution that is assigned unique value – so that if a permutation satisfies all feats or constraints, yields a feasible (one or more) solution. The task simply places N-number of queens on an N X N chessboard so that on every row, diagonal and column – there exists only one queen and no queen attacks another. This study is used as benchmark to for heuristic search such as artificial intelligence areas in backtracking, divide-and-conquer and other CSP tasks (Hassan and Crossley, 2003; Kennedy et al., 2001). N-Queen has twelve unique solutions from three variants: (a) finding one solution, (b) finding a family of solutions and (c) finding all solutions. This work is a hybrid algorithm for N-Queen, deals with finding family of solutions.

2. PROBLEM STATEMENT

Various studies have shown that most of the evolutionary algorithms used in local, systemic and global search in discrete as well as continuous spaces employ hill-climbing technique that sometimes makes them get stuck at local minima, a function of their speed. Thus, hybrids are designed to cub and help avert such defects.

3. OBJECTİVES OF THE STUDY

The study explores an N-Queen, CSP task – solved via heuristic search hill-climbing method to yield a solution that sometimes, is stuck at local minima, and speed shrinks as solution nears a global optima. Thus, PSO in this hybrid – will yield an initial solution as data with known velocities and position – as GA via its

selection, **crossover** and **mutation**, yields a robust flexible solution with continuous adaptation to provide needed computational intelligence.

4. SIGNIFICANCE OF THE STUDY

The application of this hybrid will become a veritable SC algorithm for computing dynamic and discrete states and consequently, for multipoint search in CSPs optimization tasks. As used in N-Queen, application areas includes image and video analysis, communication, antenna designs, VLSI, simulation, data routing and compression, electrical network design/reconstruction, control, multiprocessor load balancing, OS task scheduling and resource allocation, parallel processing, power generation, medical and pharmaceutical, finance and economics, security and military, engine design/automation, system fault diagnosis and recovery, forecasting and predictions, data mining, signal processing, cloud and clustering computing etc.

5. LIMITATIONS OF STUDY

For tasks that employ particle swarm optimization, it provides randomized, dependent data – in that each candidate solution depends on the velocities, acceleration and positions of other particles in the swarm. Thus, the study employs GA to provide a means for another selection of data and /or particles cum candidate solutions via its structured learning (that addresses the general problem of determining the existence of statistical dependencies amongst the data variables) to yield better generation with its crossover and mutation.

6. PSOGA FRAMEWORK

The framework is divided into two namely:

6.1 PSO Framework

PSO in attempting to simulate motion via its investigating collective intelligence cum socio-cognitive of swarms (bird/fish), specifies a model of randomly initialized solutions propagated in space towards an optimal result, over number of moves based on large amount of data about the domain, assimilated and shared by the swarm. In a bid to generate and select particles (solution) that is adapted to their environment (objectives and constraints) – so that in a number of moves, desirable traits (feat) evolves and remains in the swarms' composition (result set generated in that move) over traits with weaker undesirable characteristics (Hassan et al., 2005). PSO is continuous and thus, is modified to handle discrete design variables. Kennedy, Eberhart and Shi (2001) notes PSO's three (3) basic steps as:

a. **Particle Position/Velocity:**

A particle refers to a point in space that changes its position from one iteration to another based on velocities updates. The positions Xi and velocities Vi of the initial particles swarm are randomly generated using lower and upper bounds of the design variables values (X_{min} and X_{max}) as in Eq. 1 with an initialization process that allows the swarms to be randomly distributed across the space:

$$X_o^1 = X_{min} + rand(X_{max} - X_{min}) \text{ and}$$

$$V_o^1 = \frac{X_{min} + rand(X_{max} - X_{min})}{\nabla t} = \frac{\text{Position}}{\text{Time}} \quad (1)$$

b. **Velocities update:**

Particle velocity is updated in time t+1 via particle's fitness values (a function of particle's current position) in space at time t. The fitness value of the particles, determine particles with best global value in current swarm Pgi and the best position of each particle in time Pi (current and previous moves). Velocity update uses the effect of the current motion V^i_t – to yield search direction V^it+1 for the next iteration. To ensure good coverage and avoid local minima entrapment, the formula uses uniformly distributed rand() along with three (3) values that effects new search direction, incorporated as three weight factors: (a) current motion/inertia factor ϖ, (b) particle's own memory or self confidence factor ϕ1 and (c) swarm influence or confidence factor ϕ2 as in Eq. 2:

$$V_{t+1}^1 = \omega + V_{(t)}^1 + \varphi_1 \, rand() \frac{(P^1 - X_t^1)}{\nabla t} + \varphi_2 \, rand() \frac{(P_t^2 - X_t^2)}{\nabla t} \quad (2)$$

V^it+1 = The particle Velocity i at time t+1
ϖ+V^it = The current motion
X^it = Particle position in time t and t is time
ϕ1 = self-confidence factor with value range 1.5 - 2
ϕ2 = swarm confidence with value range 2 - 2.5
ϕ1*rand() [(Pi-Xit)/∇t] = Particle's influence
ϕ2*rand() [(Pgt-Xit)/∇t] = Swarm's influence

c. **Position update:**

Position update is in 3-steps: (i) velocity update, (ii) position update and (iii) fitness calculation – all repeated until a desired convergence criterion is reached – for which, the stop criterion is set (i.e., maximum change in best fitness is smaller than specified tolerance for a specified number of moves as in Eq. 3, which describes particle position update:

$$X_{t+1}^1 = X_t^1 + V_{t+1}^1 * \nabla t \left| f(P_t^g) - f(P_{t-g}^g) \right| \in \varepsilon \quad (3)$$

$$f(x) = \varphi(X) + \sum_{i=1}^{N_{max}} X_j * \max[0, g_i(x)] \quad (4)$$

Design variables in PSO can take values, even outside their lower/upper bounds constraint – due to their

current position and computed velocity (function of rapid growing vector velocity), causing particles to diverge instead of converge. To avoid this, such variables that violate bounds are artificially brought back to its nearest side constraint via Eq. 4 (helps avoid particle velocity explosion and handles functional constraints using a linear exterior penalty).

Traditional PSO notes each particle is a solution in the space and encoded as a string of positions that represents a multidimensional space. All the dimensions are independent of each other. Thus, updates of velocities and particles are performed independently in each dimension (a merit of PSO). However, this is not applicable for permutation task as the N-Queen – since all elements are not independent of each other. Thus, it is possible that two or more positions can have same value after such an update, which breaks permutation rule – and implies that all conflict must be resolved. Particle velocity is added on each dimension to update the particle – a distinct measure. If velocity is larger, particle explores more distant areas and is more likely to change to a new permutation series – and new velocity in such a permutation scenario signifies possibility of particle change (velocity update formula remains same). However, velocity is limited to absolute values – as it only represents the difference between particles.

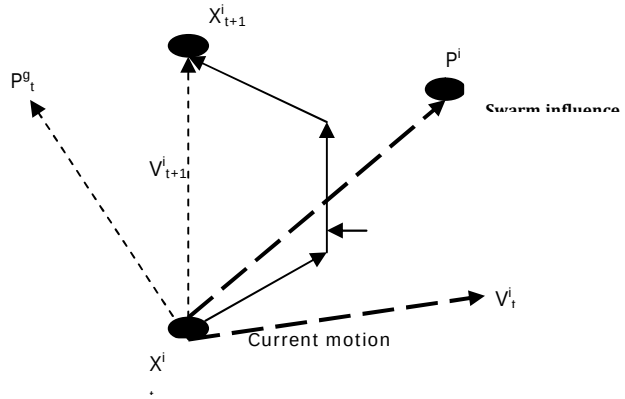

Fig 1: PSO Velocity/Particles update

6.2 GA Framework

When the PSO framework is modified, it allows for some shortcomings. For example, if particles tries to follow same sequence as the nbest, it will stay in the current position forever when the identical to nbest. Only then, is a new mutation factor introduced so that the particle will randomly swap one pair of positions in the permutation if it is identical to nbest; otherwise, mutation factor is ignored. The n-queens problem is used to test the performance and validity of the new velocity and particle update technique (Abarghouei et al, 2009; Kilic and Kaya, 2001) noted the steps under these headings as thus:

6.2.1 Encoding and Fitness

The **encoding** scheme, encodes data (N-Queen) for an 8 X 8 chessboard – so that each class of solution consist of a set of candidate solutions encoded via fixed length vector, and each feat encoded as one or more pool of different types.

The **fitness** function sees the solution set from the various candidates as evaluated at training to determine its goodness of fit. If the solution is reached correctly with appropriate particle position and velocities update – then the algorithm is considered good; else, it is bad and not be selected for crossover to produce offspring. Thus, the more candidate solutions are detected, the higher its fitness value. A merit of fitness function is that in changing weights w1 and w2, it helps identify network flaws. Unlike other fitness functions, selection criteria w1/w2 is not crucial factor to its performance. Thus, we adopt **support/confidence** fitness model – so that in each particle solution:

If A then B,

Support = |A and B| / N and
Confidence = |A and B| / |A|
Fitness = w1 * support + w2 * confidence

6.2.2 Selection Scheme

The adopted selection is the **tournament** in which, solutions are randomly chosen from the current pool or generation – so that with each next move or iteration, a lesser number is chosen. This continues until one is chosen from the last two or three solutions remains. This becomes the selected parents that create the new **offspring**. This scheme is selected based on its reputation of maintaining population diversity and we recall, the goal of this study is not to create **best solution (global optima)**, but to create set of solutions good enough to form a family of solution not stuck at **local optimums**. Thus, population diversity needs to be maintained.
Algorithm: Tournament Selection {}

1. Input: Population of chromosome
2. Output: Selected Chromosome for crossover
3. Randomly select 3-chromosomes from pool
4. Pick best 2-solution based on fitness value
5. Return the selected two solution
6. Apply Crossover | Select best solution as parent

6.2.3 Crossover

The model chooses two random cross points from the solution as in fig. 2, with exchanges in the midsection between the parents to form new children.

6.2.4 Mutation

Each gene in a chromosome may or may not change depending on the probability of mutation rate. Mutation improves population diversity needed in this work, and its algorithm is as thus:

Algorithm: Mutation{}

1. Input: A chromosome rule
2. Output: Mutated solution, a fns of mutation rate
3. Set mutation threshold (between 0 and 1)
4. For each network attribute in chromosome
5. Generate a random number between 0 and 1
6. If random number > mutation threshold then
7. Generate Random value for N-Queen
8. Set solution attribute value with
9. Generated attribute value
10. End if
11. End For Each

Solutions are updated via crossover and mutation:

1. Velocity is normalized between [0-1], dividing it by maximum range of the particles.
2. Each position randomly determines if crossover is needed as determined by the velocity.
3. If required, the position will set to the value of same position in nbest by swapping the values.

Thus, the complete GA algorithm is as thus:
Algorithm PSOGA{}

1. Input: Generations and population size.
2. Output: A set of permuted solutions.
3. Randomly Initial created solution population.
4. Set w1 = 0.2, w2 = 0.8, MaxGen = 400 (epoch)
5. Set N = total number of record in set
6. Set generationCounter = 0
7. For each solution in population
8. Set A = 0, AB = 0
9. For Each record in training set
10. If record matches chromosome
11. AB = AB + 1
12. End If
13. If record matches only condition part
14. A = A+1
15. End if
16. End For Each record and Each Solution
17. Select 30-best fitted solutions into new pool
18. For each solution in new pool/population
19. Select chromosome for breeding
20. Do crossover and mutation to new offspring
21. Place newly created offspring into population
22. End For each
23. Kill old pool, new pool now current pool
24. Increment generationCounter by 1
25. If generationCounter < MaxGen then
26. Goto line 5
27. Else goto line 27
28. End

7. METHODS AND MATERIALS

Parameters are as thus: n = 25, local version of PSO and neighborhood size of 7 with the maximum velocity for mutation. Inertia weight [0.5+rand/2.0], swarm- and self-confidence factors equals 1.5 each. Table 1 is results for 20-500 queens with each parameter combined at 100-runs.

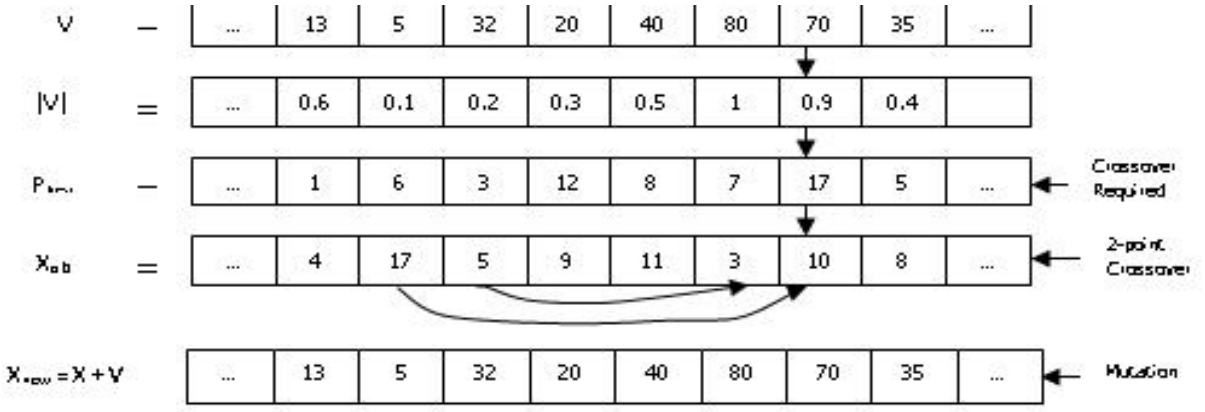

Fig 2: Crossover and mutation of velocities and particle positions

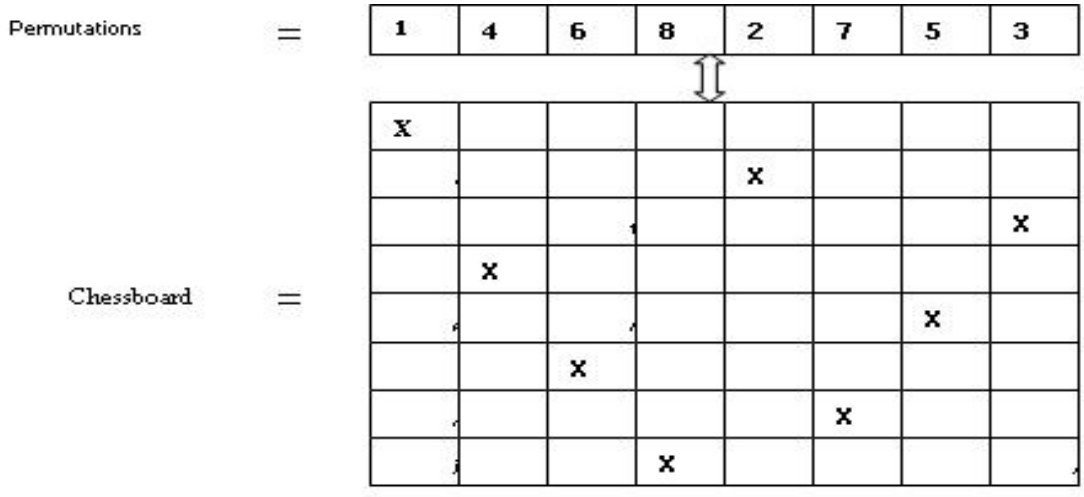

Fig 3: Chessboard Permutation for N = 8 particles

8. RESULT DISCUSSION

From the results, it is seen that PSOGA successfully finds a solution of the n-queens task or problem in a short time better than PSO, GA and GAPSO hybrids (Hu et al, 2005a; Ojugo, 2012). Also, its number of functions evaluation increases almost near linearly as number of queens increases when plotted.

From fig. 3, 1st queen is in row 1 in the 1st column, the 2nd queen is at row 4 in the 2nd column, the 3rd queen is at row 6 in 3rd column; the 4th queen is at the 8th row in the 4th column, the 5th queen is on the 2nd row in the 5th column, the 6th queen is on the 7th row in the 6th column, the 7th queen is on the 5th row in the 3rd column; while the last and 8th queen is at the 3rd row in the 8th column as in Fig. 3. Thus, Homaifar et al. (1992) and Clerc (1999) notes with permutation, horizontal and vertical conflicts are resolved and eliminated.

The fitness function is redefined as number of conflicts that appears along the diagonals of a chessboard or it can be modified to minimize the number of conflicts or collision-where the fitness value of an ideal solution (finding a family of solution for the n-queens problem) should be equal to zero.

PSO starts with randomly initialized swarm space that evolves into successive iterations – to eliminate need for user-supplied start value. Thus, to perform this optimization, it is mixed with GA operators to propagate its population from one move (iteration) to the next. First, **selection** mimics survival of the fittest concept; while **crossover** attempts to propagate feats of a good surviving design from the current pool into the future pool so as to yield better fitness value on the average. Lastly, **mutation** promotes diversity in population properties and allows for global search as well as prevents the algorithm from being trapped in the search for a local minima.

Table 1: Fitness Evaluation for Optimal Function

Queen(s)	This study	Hu Et al	Homaifar	Kilic*
20	5,964.5	5,669.7	2,043	6,024
50	25,789.1	14,991.4	59,227	19,879
100	43,786.3	36,799.4	224,208	44,578
200	99,032.8	93,439.9	340,991	86,747
500	195,643	195,657.6	423,765	132,987

* These number are digitized from basic GA results of Kilic's work. Thus, they are not accurate (as culled from Hu et al., 2007)

9. RECOMMENDATIONS

For the GA model, we adopted w1 and w2 as 0.2 and 0.8 respectively; while PSO adopts a suggested upper and lower bound with $\varpi = 0.5$, $\phi 1 = 1.5$ and $\phi 2 = 1.5$ yields better convergence rate for all test. Other value combinations led to non-convergence or slower convergence. In addition to applying penalty function to handle tasks with violated functional constraints, it is recommended that particle vector velocity that violates the side constraint be reset to 0 via velocity update of Eq. 4 – since, if a particle is infeasible, there is a big chance that the last search direction (velocity) was not feasible. Thus, in design of complex model with discrete variables (such as technology choices), Eq. 1 is a simple, effective way to implement with PSO, that will round particle position coordinates to the nearest integer if the variables being investigates are discrete (Kennedy and Eberhart, 1997; Kennedy and Mendes, 2002).

REFERENCES

[1] Abarghouei, A., Ghanizadeh, A and Shamsuddin, S., (2009). Advances in soft computing methods in edge detection, J. Advance Soft Comp. Apps., ISSN: 2074-8523, 1(2).

[2] Coello, C. A., Pulido, G. T and Lechuga, M.S.,

(2002): Handling multiple objectives with PSO Cong. On Evo. Comp., Vol. 8, 256–279.

[3] Clerc, M., (1999): Swarm and the queen: towards a deterministic and adaptive particle swarm optimization, Cong. Evo. Comp. Vol. 5, 123-132.

[4] Dozier, G and Carlisle, A. (2001). Tracking changing extrema with particle swarm optimizer, Auburn University: Technical Report CSSE01-08.

[5] Eberhart, R and Kennedy, J., (1995): New optimizer using particle swarm theory, Proc 6th Int symp. Micro machine and human science, Japan: Na-goya.

[6] Hassan, R and Crosswley, W., (2004): Variable population-based sampling for probabilistic design optimization and with a genetic algorithm, AIAA-2004-0452), 42nd Aerospace Science meeting, Reno, NV.

[7] Hassan, R., Cohanin, B., De Wec and Venter, G., (2006): Comparism of PSO and GA, American Institute of Aeronautic and Astronautics (AIAA-2006), 44th Aerospace Science meeting Washington–DC.

[8] Heppner, H and Grenander, U (1990): A stochastic non-linear model for coordinated bird flocks, In Krasner, S (Ed.), The ubiquity of chaos, (pp. 233–238). Washington: AAAS.

[9] Homaifar, A.A., Turner, J and Ali, S., (1992): The n-queens problem and genetic algorithms, In Proceedings of the IEEE Southeast conference, pp 262-267.

[10] Hu, X., Eberhart, R.C and Kennedy, J., (2005a): Solving constrained nonlinear optimization problems with particle swarm optimization. In Proc. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2005), USA: Orlando.

[11] Hu, X., Eberhart, R.C and Shi, Y. (2005b): Swarm intelligence for permutation optimization: case study of n-queens, Genetic and Evo. Comput. Conf. (GECCO-2005), Sixth workshop on Memetic algorithms: pp 243 – 246.

[12] Kennedy, J and Eberhart, R. C. (1995). Particle swarm optimization. Proc. Int Conf on Neural networks, pp. 1942–1948, Honolulu, HI, Pis-cataway: IEEE.

[13] Kennedy, J and Eberhart, R. C. (1997): Discrete binary version of the particle swarm algorithm. Proc. Conf. on systems, man, and cybernetics, pp. 4104–4109. Piscataway: IEEE.

[14] Kennedy, J., Eberhart, R. C and Shi, Y., (2001): Swarm intelligence. San Francisco: Kaufmann.

[15] Kennedy, J and Mendes, R. (2002). Population structure and particle swarm performance. In Proc. Cong. Evol. Comp. (pp.1671–1676), Honolulu, HI. Piscataway: IEEE.

[16] Kilic, A. and Kaya, M.A (2001): A new local search algorithm based on genetic algorithms for the n-queens problem, Proc. Genetic and Evol. comp. conf. (GECCO-2001), 158 – 161

[17] Ojugo, A.A., (2012): Gravitational search neural network algorithm for rainfall runoff modeling, unpublished dissertation, Abakiliki: Ebonyi State University.

[18] Parsopoulos, K. E and Vrahatis, M. (2004): On computation of all global minimizers via particle swarm optimization, IEEE Transactions on Evolutionary Computation, 8, 211–224.

[19] Poli, R., Wright A.h., McPhee, N.F and Langdon, W.B., (2006b): Emergent behaviour, population-based search and low-pass filtering, Proc. IEEE world Congress on Comp. Intelligence, IEEE congress on Evo.Comp., pp395-402, Vancouver: Piscataway.