# A Hybrid Swarm Intelligence Technique for Solving Integer Multi-objective Problems

Ibrahim M. El-henawy
Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Egypt

Mahmoud M. Ismail
Department of Operations Research, Faculty of Computers and Informatics, Zagazig University, Egypt

## ABSTRACT

The multi-objective integer programming problems are considered time consuming. In the past, mathematical structures were used that can get benefits of high processing powers and parallel processing. A general approach to generate all non-dominated solutions of the multi-objective integer programming (MOIP) Problem is developed. In this paper, a hybridization of two different swarm intelligent approaches, stochastic diffusion search, and particle swarm optimization techniques is presented for solving integer multi-objective problems. The hybrid implementation allows us to avoid certain drawbacks and weaknesses of each algorithm, which means that we are able to find an optimal solution in an acceptable computational time. Our hybrid implementation allows the MOIP algorithm to reach the optimal solution in a considerably shorter time than is needed to solve the model using the entire dataset directly within the model. Our hybrid approach outperforms the results obtained by each technique separately. It is able to find the optimal solution in a shorter time than each technique on its own, and the results are highly competitive with the state-of-the-art in large-scale optimization. Furthermore, according to our results, combining the PSO with SDS approach for solving IP problems appears to be an interesting research area in combinatorial optimization.

## Keywords

Swarm Intelligence, Integer programming, Multi-objective, Stochastic Diffusion Search and Particle Swarm Optimization

## 1. INTRODUCTION

Optimization can be viewed as one of the major quantitative tools in network of decision making, in which decisions have to be taken to optimize one or more objectives in some prescribed set of circumstances. In view of the practical utility of optimization problems there is a need for efficient and robust computational algorithms, which can numerically solve on computers the mathematical models of medium as well as large size optimization problem arising in different fields. Heuristics and bioinspired techniques have become efficient and effective alternatives for researchers in solving several complex optimization problems. These techniques are not able to reach the optimal solution for large-scale combinatorial optimization problems in spite of their effectiveness. But these techniques are able to provide satisfactory solutions for most of the applied problems within acceptable computational times. In contrast, mathematical programming techniques, particularly the Integer multi-objective problems, have been studied and developed by scholars over several decades with the main goal of obtaining optimal solutions to difficult problems using as little CPU time as possible. For these reasons, a hybrid swarm intelligence technique has been suggested. In recent years, swarm intelligence, which can be considered as a branch of Artificial Intelligence techniques, has attracted much attention of researchers, and has been applied successfully to solve a variety of problems. A swarm can be viewed as a group of agents cooperating with certain behavioural pattern to achieve some goal [12]. There are a number of different models of swarm intelligence that have been proposed and investigated, and among the most commonly used swarm intelligence models include ant colony optimization [5, 9], particle swarm optimization [7, 14], honey bee swarming [25, 26], stochastic diffusion search, and bacterial foraging [22, 23]. These algorithms have proved their mettle in solving complex and intricate optimization problems arising in various fields. The paper is organized such that the next section provides a brief overview of integer programming and some definitions. Section 3 describes Particle Swarm Optimization technique. Section 4 describes Stochastic Diffusion Search technique. Section 5 describes the method of the proposed hybrid swarm intelligence technique used. Section 6 discusses the computational results. In section 7, a conclusion is introduced.

## 2. MULTIOBJECTIVE PROBLEMS

[6, 11, 17, 24] Most real world optimization problems involve the optimization of more than one function, which in turn can require a significant computational time to be evaluated. That is, decision makers are not able to base their decisions on a single criterion in order to identify one or more attractive courses of action. Moreover, real decision problems are such that conflicting criteria are often used to evaluate alternative solutions. Multiobjective Programming is the branch of Mathematical Programming that deals with problems for which more than one objective function is required to evaluate the merit of alternative decisions. Formally, the problem is stated as follows:

$$MAX \quad (f_1(x), f_2(x) \dots f_n(x))$$

$$S.t. \quad x \in X$$

$$\text{Where} \quad x = (x_{1,} x_2, \dots\dots\dots x_n)$$

$X$ is the set of feasible solutions

$F_i$ is the i $^{th}$ objective function (or decision criterion).

Then the problem doesn't have a single solution that could optimize all objectives simultaneously, but it has efficient (Pareto-optimal) solutions that can best attain the objectives as greatly as possible. In this context, deterministic techniques are difficult to apply to obtain the set of Pareto optimal solutions of many multiobjective optimization problems (MOPs), so stochastic methods have been widely used and applied. Among them, the use of evolutionary algorithms for solving MOPs has significantly grown in the last years, giving raise to a wide variety of algorithms, such as NSGA-II, SPEA2, PAES, and many others. Most multiobjective programming techniques focus on finding the set of efficient points (E) for a given problem or, in the case of heuristic procedures, an approximation of the efficient set ($\hat{E}$).

To select a suitable compromise solution from all efficient alternatives, a decision process is necessary. Depending on how the computation and the decision processes are combined in search for compromise solution, there exist three broad classes of multiobjective optimization methods:

1. A prior articulation of preferences (a prior methods)
2. A posterior articulation of preferences (posterior method).
3. progressive articulation of preferences (Interactive methods)

The DM provides global preference information during the optimization process.

## 2.1 Definition 1 (complete optimal solution)

$x^*$ Is said to be a complete optimal solution, *if and only if there exists* $x^* \in X$ such that $z_i(x^*) \le z_i(x), i = 1,\ldots., k,$ for all $x \in X$ .

## 2.2 Definition 2 (Pareto optimal solution)

$x^*$ Is said to be a Pareto optimal solution, if and only if there doesn't exist another $x \in X$ such that $z_i(x) \le z_i(x^*)$ for all $i$ and for all $z_j(x) \ne z_j(x^*)$ for at least one $j$.

## 2.3 Definition 3 (Weak Pareto optimal solution)

$x^*$ Is said to be a weak Pareto optimal solution, if and only if there doesn't exist another $x \in X$ such that $z_i(x) \prec z_i(x^*), i = 1,\ldots.k$ .

## 2.4 Definition 4 (Ideal Vector)

Is the Vector which containing the decision variables corresponding to the optima of the objective functions of the problems considering each objective separately.

## 2.5 Definition 2.5 (Pareto Front)

For a given MOP $\vec{f}(x)$ and Pareto optimal set P*, the Pareto front (PF*) is defined as:

$$PF^* := \{ \vec{u} = \vec{f} = f_1, f_2, \ldots. f_n \mid x \in P^* \}.$$

## 2.6 Definition 6 Pareto Dominance

A vector $\vec{u} = (u_1, \ldots.., u_n)$ is said to dominate

$\vec{v} = (v_1, \ldots.., v_n)$ (denoted by $\vec{u} \le \vec{v}$ ) if and only if *u* is partially less than *v*.

## 3. PARTICLE SWARM OPTIMIZATION

The particle swarm optimization (PSO) was inspired by the observations of birds flocking and fish schooling. It differs from other well-known Evolutionary Algorithms (EA) [2, 10, 13]. As in EA a population of potential solutions is used to probe the search space; but, no operators, inspired by evolution procedure, are applied on the population to generate a new promising solution. Instead, in PSO, each individual (named particle) of the population (called swarm), adjusts its trajectory towards its own previous best solution (called pbest) and the previous best solution attained by any member of its topological neighborhood. There are different kinds of sharing information between particles. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and the particles benefit from the discoveries and the previous experiences of all other companions during the search for promising regions of the landscape [16]. Alternatively, there are some local variants of PSO wherein particles only make use of their own information and that of the best of their adjacent neighbors. Each particle in PSO has two main characteristics: its position and its velocity. Assume that the current position and velocity vector of the i-th particle in the d-dimensional search space are denoted as $X_i = (x_{i1}, x_{i2}, \ldots., x_{id})$ and $V_i = (v_{i1}, v_{i2}, \ldots., v_{id})$, respectively. The best earlier position of the *i-th* particle is represented as $pbest_i = (pbest_{i1}, pbest_{i2}, \ldots., pbest_{id})$.There are different kinds of PSO including global vision of PSO with inertia weight (GWPSO), local vision of PSO with inertia weight (LWPSO), global vision of PSO with constriction factor (GCPSO), and local vision of PSO with constriction factor (LCPSO) [27]. In GWPSO, which is very popular among researchers, there are two methods for updating position and velocity of each particle. The best position of entire group at *k-th* iteration is used in the first method while in the second method; the best position of entire group up to the current search is employed. In the first method, the position $\mathbf{x}_{id}^{k}$and velocity $\mathbf{v}_{id}^{k}$ of particle *i* in the *k-th* iteration are updated as follow:

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}$$

$$v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1 (pbest_{id}^{k} - x_{id}^{k}) + c_2 r_2 (gbest - x_{id}^{k})$$

In the second equation $w$ is the inertia weight, $c_1$ and $c_2$ are positive constants called cognitive and social parameters, respectively, and $r_1$ and $r_2$ are random numbers selected in the interval [0 1]. The constants $c_1$ and $c_2$ represent the weighting of the stochastic acceleration terms that pull each particle towards pbest and gbest positions and usually are set $c_1 = c_2 = 2$ . In the second method gbest is replaced by gbestk. As will be shown later, in the numerical examples of mixed-variables or in the problems that only have discrete variables, usage of gbestk is more suitable compared to the use of gbest. In other words, the success rate of gbestk is higher than that of gbest. The reason is firstly due to the fast convergence of gbest and secondly, the inability of particles to escape from local minima in gbest method. In other words, since the discrete variables are rapidly converged the continuous variables will be obliged to search in a limited specific area which might not be the optimum area. The role that inertia weight w plays in the convergence behavior of PSO is very important. The inertia weight is employed to control the effect of the previous velocities on the current velocity. This way, the parameter *w* makes a compromise between global and local exploration abilities of the swarm. In PSO, when the search continues, the inertia term decreases linearly as follows:

$$w = w_{max} - \left( \frac{w_{max} - w_{min}}{k_{max}} \right) k$$

Where $w_{max}$ and $w_{min}$ are the maximum and minimum values of the inertia term, respectively, and $k_{max}$is the maximum number of iterations. These parameters are assumed to be:

$$w_{max} = 1, \qquad w_{min} = 0$$

Sometimes as particle oscillations become wider, the system will gain tendency to explode [13]. The usual means of preventing explosion is simply to define a parameter $\mathrm{w_{max}}$ and curb the velocity of every individual i from exceeding that velocity on each dimension d. In the case that velocity violates, it will be modified as follows:

$$\text{If } v_{id} > v_{max} \text{ then } v_{id} = v_{max}$$

$$\text{If } v_{id} < -v_{max} \text{ then } v_{id} = -v_{max}$$

The effect of this is to allow particles to oscillate within the bounds [13].

```
Initialise particles
While ( stopping condition is not met )
    For all particles
        Evaluate fitness value of each particle
        If (current fitness <pbest )
pbest = current fitness
        If (pbest<global (or local ) best)
global (or local ) best = pbest
        Update particle velocity
        Update par ticle position
    End
End
```

**Fig 1: The pseudocode of PSO algorithm**

Although PSO has been used mainly to solve unconstrained, single-objective optimization problems, PSO algorithms have been developed to solve constrained problems, multi-objective optimization problems, problems with dynamically changing landscapes, and to find multiple solutions. There are some disadvantages of PSO Algorithm as follow:

- The method easily suffers from the partial optimism, which causes the less exact at the regulation of its speed and the direction.
- The method cannot work out the problems of scattering and optimization.
- The method cannot work out the problems of non-coordinate system, such as the solution to the energy field and the moving rules of the particles in the energy field.

## 4. STOCHASTIC DIFFUSION SEARCH

This section introduces Stochastic Diffusion Search (SDS) [3], a multi-agent global search and optimization algorithm, which is based on simple interaction of agents. A high-level description of SDS is presented in the form of a social metaphor demonstrating the procedures through which SDS allocates resources.SDS introduced a new probabilistic approach for solving best-fit pattern recognition and matching problems. SDS, as a multi-agent population-based global search and optimization algorithm, is a distributed mode of computation utilizing interaction between simple agents [8]. Unlike many nature inspired search algorithms, SDS has a strong mathematical framework, which describes the behavior of the algorithm by investigating its resource allocation [19], convergence to global optimum [20], robustness and minimal convergence criteria [18] and linear time complexity [21]. The SDS algorithm commences a search or optimization by initializing its population. In any SDS search, each agent maintains a hypothesis, h, defining a possible problem solution. After initialization two phases are followed, test Phase, and diffusion phase. In the test phase, SDS checks whether the agent hypothesis is successful or not by performing a partial hypothesis evaluation which returns a boolean value. Later in the iteration, contingent on the precise recruitment strategy employed, successful hypotheses diffuse across the population and in this way information on potentially good solutions spreads throughout the entire population of agents. In the Test phase, each agent performs partial function evaluation, pFE, which is some function of the agent's hypothesis; pFE = f(h). In the diffusion phase, each agent recruits another agent for interaction and potential communication of hypothesis.

```
Initialisingagents()
While ( stopping condition is not met)
    Testing hypotheses ()
    Diffusion hypotheses ()
End
```

**Fig 2: The pseudocode of SDS algorithm**

Passive recruitment mode is employed In SDS algorithm. In this mode, if the agent is inactive, a second agent is randomly selected for diffusion; if the second agent is active, its hypothesis is communicated to the inactive one. Otherwise a completely new hypothesis is generated for the first inactive agent at random. The main disadvantage of the SDS is in the case of search spaces distorted heavily by noise, diffusion of activity due to disturbances will decrease an average number of inactive agents taking part in random search and in effect will increase the time needed to reach the steady state.

```
For ag = 1 to No of agents
    If (ag. activity () == false )
        rag = pick a random agent ()
        If (r ag. activity () == true)
            ag. setHypothesis ( r ag .getHypothesis())
        Else
            ag. setHypothesis (randomHypothsis())
End
```

**Fig 3: Passive Recruitment Mode**

## 5. PROPOSED SDS-PSO TECHNIQUE

Because of the drawbacks of PSO algorithm, a hybrid swarm intelligence technique called SDS-PSO technique has been proposed for solving integer programming problems to produce better solution by improving the effectiveness and reducing the limitations. The motivating thesis justifying the merging SDS and PSO is the partial function evaluation deployed in SDS, which may mitigate the high computational overheads entailed when deploying a PSO onto a problem with a costly fitness function. In the hybrid algorithm, each PSO particle has a position, and a velocity; each SDS agent, on the other hand, has hypothesis and status. Every PSO particle is an SDS agent too together termed psAgents. In the psAgent, SDS hypotheses are defined by the PSO particle positions and a status which determines whether the psAgent is active or inactive (see Figure 4).

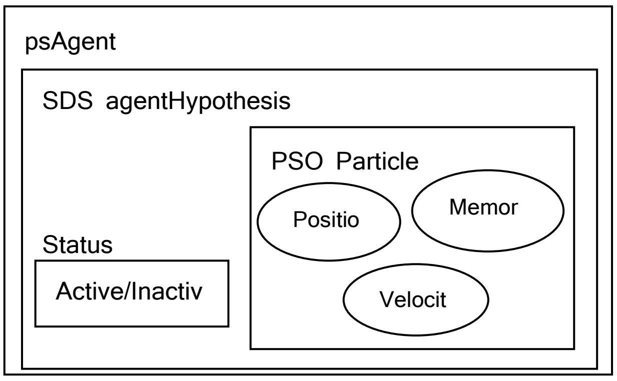

**Fig 4. psAgent**

Figure 5 shows the pseudocode of the proposed SDS-PSO technique. In the test-phase of a stochastic diffusion search, each agent has to partially evaluate its hypothesis. The fitness of each psAgent's particle's personal best is compared against that of a random psAgent; if the selecting psAgent has a better fitness value, it will become active, otherwise it is flagged inactive. On average, this mechanism will ensure 50% of psAgents remain active from one iteration to another. In the Diffusion Phase, each inactive psAgent picks another psAgent randomly, if the selected psAgent is active, the selected psAgent communicates its hypothesis to the inactive one; if the selected psAgent is inactive too, the selecting psAgent generates a new hypothesis at random from the search space. In the proposed technique, after each *n* number of PSO function evaluations, one full SDS cycle is executed.

## 5.1 Different Performance Measures

The use of performance measures (or metrics) allows a researcher or computational scientist to assess in a quantitative way the performance of their algorithms. The SDS-PSO field has no different. SDS-PSO performance measures tend to focus on the phenotype or objective domain as to the accuracy of the results. This is different to what most operations researchers do. They tend to use metrics in the genotype domain. But since there is an explicit mapping between the two, it doesn't really matter in which domain that define that metrics. There are many techniques for measuring the performance of Multiobjective problems [15] such as Hyperarea and Ratio, and Generational Distance.

```
Solving problem using LP
Initialize psAgents
While (stopping condition is not met)
      For all psAgents
            Evaluate fitness value of each particle
            If (evaluation counter MOD n == 0)
            //START SDS
            // TEST PHASE
           For ag = 1 to No of psAgents
              rag = pick−random−psAgent ()
              If ( ag. pbestFitness() <=r ag . pbestFitness() )
                 ag. setActivity (true)
              Else
                 ag. setActivity ( false )
              End i f
           End for
           // DIFFUSION PHASE
          For ag = 1 to No of psAgents
          If ( ag. activity () == false )
             rag = pick−random−psAgent()
          If ( r ag . activity () == true )
ag. set-psAgentHypothesis(r ag.get-psAgentHypothesis())
          Else
           ag. set-psAgentHypothesis( randomHypothesis())
          End if
          End for
          End if
           // END SDS
          If (current fitness <pbest)
            pbest = current fitness
            If (pbest< global (or local ) best)
               global (or local) best = pbest
           Update particle velocity
           Update particle position
    End
End
```

**Fig 5. Pseudocode of the proposed technique**

### 5.1.1 Hyperarea and Ratio (H, HR)

These metrics define the area of coverage $PF_{KNOWN}$ has with respect to the objective space. This would equate to the summation of all the areas of rectangles, bounded by the origin and $(f_1(\vec{x}), f_2(\vec{x}))$, for a two-objective SDS-PSO.

$$H = \{\bigcup_i a_i \mid v_i \in PF_{KNOWN}\}$$

Where $v_i$ is a nondominated vector in $PF_{KNOWN}$ and $a_i$ is the hyperarea calculated between the origin and vector $v_i$. But if $PF_{KNOWN}$ is not convex, the results can be misleading. It is also assumed in this model that the origin is (0, 0) [4].

### 5.1.2 Generational Distance (GD)

It reports how far, on average, $PF_{KNOWN}$ is from $PF_{TRUE}$. This metric requires that the researcher knows $PF_{TRUE}$. It is mathematically defined as in the following equation

$$GD = \frac{\left(\sum_{i=1}^{n} d_i^p\right)^{1/p}}{n}$$

Where $n$ is the number of vectors in $PF_{KNOWN}$, $p$=2 and $d_i$ is the Euclidean distance between each member and the closest member of $PF_{TRUE}$, in the phenotype space. When GD = 0, $PF_{KNOWN} = PF_{TRUE}$.

## 6. COMPUTATIONAL RESULTS

This section presents computational results of the proposed SDS-PSO technique to solve integer multi-objective problems. The computational results show performance comparison between the Non-Dominated Sorting Genetic Algorithm (NSGAII) and our proposed SDS-PSO technique. The proposed SDS-PSO technique runs on computer (Core 2 Due CPU, 2.2GHz, 2048MB RAM.). The technique has been compared to NGSAII with different measures. The first measure used is: Hyperarea and Ratio, the results are shown in table 1.

**Table 1. Hypervolume area according the two techniques**

| Technique / Problem size | Hyperarea | |
|---|---|---|
| | NSGAII | SDS-PSO |
| 10 | 149674 | 149294 |
| 50 | 309855 | 335271 |
| 100 | 263880 | 263475 |
| 500 | 844427 | 852897 |
| 1000 | 15647268 | 17156790 |

It's appears from the previous table that the hyperarea covered by SDS-PSO is larger than the hyperarea covered by NGSAII. Another comparative measure have been used is the average of the nearest neighbor, the results of this measure are shown in table 2.

**Table 2. Average distance to the nearest neighbor**

| Average Distance for Nearest Neighbor | | | | | | |
|---|---|---|---|---|---|---|
| size / Technique | | 10 | 50 | 100 | 500 | 1000 |
| NSGAII | Min | 40.6 | 33.0 | 32.3 | 14.0 | 22.2 |
| | Max | 59.4 | 42.2 | 65.3 | 23.1 | 126.4 |
| | **Avg.** | **48.5** | **36.1** | **43.3** | **18.1** | **50.5** |
| SDS-PSO | Min | 28.6 | 63.5 | 34 | 47.5 | 22.1 |
| | Max | 71.8 | 142.5 | 62.3 | 161.4 | 246.8 |
| | **Avg.** | **42.7** | **89.8** | **43.3** | **50.6** | **129.8** |

For more analysis, in 1000 variables problem size we made parametric analysis to see the appropriate technique in defined range of weights.

**Table 3. Parametric analysis for 1000 variables problem**

| Parametric / Technique | $0 \le w_1 \le 0.3$, $0.7 \le w_2 \le 1$ | $0.3 < w_1 \le 0.7$, $0.3 < w_2 \le 0.7$ | $0.7 < w_1 \le 1.0$, $0.3 < w_2 \le 1.0$ |
|---|---|---|---|
| **NGSAII** | **28.076** | 44.237447 | 43.9664 |
| **SDS-PSO** | 36.03377 | **41.8739** | **31.6227** |

The result was: for $0 \le w_1 \le 0.3$ *and* $1.0 \ge w_2 \ge 0.7$ NSGAII produces the smallest average distance to the nearest neighbor. For $0.7 < w_1 \le 1.0$ *and* $0.3 > w_2 \ge 1.0$ SDS-PSO produces the smallest average distance to the nearest neighbor. For $0.3 < w_1 \le 0.7$ *and* $0.7 > w_2 \ge 0.3$ the SDS-PSO produces the smallest average distance to the nearest neighbor. The results of parametric analysis for 1000 variables problem are shown in table 3.

## 7. CONCLUSION

This paper proposed a hybrid swarm intelligence technique called SDS-PSO to solve integer multi-objective problems, and compares its performance with the Non-Dominated Sorting Genetic Algorithm (NSGAII). The proposed technique effectively overcomes the drawbacks of PSO technique, such as the partial optimism, which causes the less exact at the regulation of its speed and the direction. It also, overcomes the drawbacks of SDS technique. SDS-PSO technique also, increases the efficiency of the solution process, improves the performance scalability, and increases the diversification of solutions at the same time, reducing the execution time in comparison with NSGAII technique. The proposed technique has been tested by solving a set of different knapsack problems. It is capable to provide a considerable reduction of time compared with other techniques most obviously at lower scale problems. In general, the proposed SDS-PSO technique seems an efficient alternative for solving Integer multi-objective problems, when deterministic approaches fail, or it could be considered as an algorithm for providing good initial points to deterministic methods, as the Branch and Bound technique, and thus, help them converge to the global minimizer of the integer problem.

## 8. REFERENCES

[1] Arbel, A., and Korhonen, P.J. 2001. Using objective values to start multiple objective linear programming algorithms. European Journal of Operational Research, 128:587–596.

[2] Banzhaf, W., Nordin, P., Keller, R.E., and Francone, F.D. 1998. Genetic Programming-An Introduction, Morgan Kaufmann. San Francisco.Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[3] Bishop, J. 1989. Stochastic searching networks, London, UK, Proc. 1st IEE Conf. on Artificial Neural Networks. 329–331.

[4] Carlos, A. CoelloCoellol, and Gary, B., Lamont, A. 2004. Applications of Multi-objective Evolutionary Algorithms. ISBN 978-981-256-106-0, 981-256-106-4.

[5] Colorni, A., Dorigo, M., Maniezzo, V. 1992. Distributed Optimization by Ant Colonies. In: Varela, F., Bourgine, P. (eds.) Proceedings of the First European Conference on Artifical Life, MIT Press, Cambridge. 134–142.

[6] David, A., Van Veldhuizen, and Lamont, G.B. 1998. Evolutionary Computation and Convergence to a Pareto Front. In John R. Koza, editor, Late Breaking Papers at the Genetic Programming 1998 Conference, Stanford University, California. Stanford University Bookstore. 221-228.

[7] del Valle, Y., Venayagamoorthy, G.K., Mohaghenghi, S., Hernandez, J.C., Harley, R.G. 2008. Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. IEEE Transactions on Evolutionary Computation 12 .171–195.

[8] DeMeyer, K., Bishop, J.M., and Nasuto, S.J. 2003. Stochastic diffusion: Using recruitment for search. Evolvability and interaction: evolutionary substrates of communication, signalling, and perception in the dynamics of social complexity (ed. P. McOwan, K. Dautenhahn& CL Nehaniv) Technical Report. 60–65.

[9] Dorigo, M., Maniezzo, V., Colorni, A. 1996. The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics 26. 29–41.

[10] Eberhart, R.C., Simpsonand, P.K. Dobbins, R.W. 1996. Computational Intelligence PC Tools, Academic Press Professional. Boston.

[11] Eckart Zitzler and Lothar Thiele. 1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. Evolutionary Computation, IEEE Transactions on Volume 3, Issue 4. 257- 271.

[12] Grosan, C., Abraham, A., Monica, C. 2006. Swarm Intelligence in Data Mining. In: Abraham, A., Grosan, C., Ramos, V. (eds.) Swarm Intelligence in Data Mining. SCI, vol. 34, Springer, Heidelberg. 1–16.

[13] Kennedy, J. and Eberhart, R.C. 2001. Swarm Intelligence, Morgan Kaufmann Publishers.

[14] Kennedy, J., Eberhart, R. 1995. Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4. 1942–1948.

[15] Lai, T. H., Sahni, S. 1984. Anomalies in parallel B&B algorithms, Res. Contrib. 27 (6) 594-602.

[16] Laskari, E.C., Parsopoulos, K.C. and Vrahatis, M.N. 2002. Particle Swarm Optimization for Integer Programming, Proceedings IEEE Congress on Evolutionary Computation, Hawaii, U.S.A. 1576-1581.

[17] Mezmaz,M., Melab, N., and Talbi,E.G. 2007. An efficient load balancing strategy for grid-based branch and bound algorithm. Parallel Computing, volume: 33, number: 4-5, 2007, ISSN: 0167-8191. Elsevier Science Publishers B. V., Amsterdam, The Netherlands. 302-313.

[18] Myatt, D.R., Bishop, J. M., and Nasuto, S.J. 2004. Minimum stable convergence criteria for stochastic diffusion search. Electronics Letters, 40(2). 112–113.

[19] Nasuto, S.J. 1999. Resource Allocation Analysis of the Stochastic Diffusion Search.PhDthesis, University of Reading, Reading, UK.

[20] Nasuto, S.J. and Bishop, J.M. 1999. Convergence analysis of stochastic diffusion search. Parallel Algorithms and Applications, 14(2).

[21] Nasuto, S.J., Bishop, J.M., and Lauria, S. 1998. Time complexity of stochastic diffusion search. Neural Computation, NC98.

[22] Passino, K.M. 2000. Distributed Optimization and Control Using Only a Germ of Intelligence. In: Proceedings of the 2000 IEEE International Symposium on Intelligent Control. 5–13.

[23] Passino, K.M. 2002 Biomimicry of Bacteria Foraging for Distributed Optimization and Control. IEEE Control Systems Magazine 22. 52–67.

[24] Santana-Quintero, L.V. and Coello, C.A. 2005. an Algorithm Based on Differential Evolution for Multi-Objective Problems, International Journal of Computational Intelligence Research, ISSN 0973-1873 Vol.1, No.2. 151–169.

[25] Seeley, T.D. 1996. The Wisdom of the Hive. Harward University Press.

[26] Teodorovic, D., Dell'orco, M. 2005. Bee Colony Optimization-A Cooperative Learning Approach to Complex Transportation Problems, Advanced OR and AI Methods in Transportation. 51–60.

[27] Yu, B., Yuan, X. and Wang, J. 2000. Short-Term Hydro-Thermal Scheduling using Particle Swarm Optimization Method. Energy Conversion and Management, Vol. 48. 1902-1908.