ISSN 22773061

AN EFFICIENT Hybrid Swarm Intelligence Technique for Solving Integer Programming

Ibrahim El-henawy

Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Egypt.

henawy2000@yahoo.com

Mahmoud M. Ismail

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

analyst_mohamed@yahoo.com *Corresponding author

ABSTRACT

In this paper, a hybridization of two different swarm intelligent approaches, stochastic diffusion search, and particle swarm optimization techniques is presented for solving integer programming 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 IP 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.

Indexing terms/Keywords

Swarm Intelligence, Integer programming, Stochastic Diffusion Search and Particle Swarm Optimization.

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

Vol 10, No 4

editor@cirworld.com
www.cirworld.com, member.cirworld.com

ISSN 22773061

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 Programming, 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 [10]. 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 [3], [6], particle swarm optimization [15], [4], honey bee swarming [29], [30], stochastic diffusion search, and bacterial foraging [24], [25]. 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 2 provides a brief overview of integer programming. 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.

INTEGER PROGRAMMING

It is often impossible to represent certain features of many real-world problems using only linear constraints and continuous variables. In modeling a real world problem, it is often necessary to represent discrete activities by variables which are restricted to take only integer values. The general mathematical form of integer programming problems is:

Maximize $z = \sum_{i=1}^{n} c_i x_i$

Subject to $\sum_{i=1}^{n} a_{ji} x_i \le b_j, (j = 1,2,...,m),$

$x_i \ge 0,$ x_i integers and $(i = 1,2,...,n).$ (1)

This problem is called the linear integer-programming problem. It is said to be a mixed integer program when some, but not all, variables are restricted to be integer, and is called a pure integer program when all decision variables must be integers. If the constraints are of a network nature, then an integer solution can be obtained by ignoring the integrality restrictions and solving the resulting linear program. In general, though, variables will be fractional in the linear-programming solution, and further measures must be taken to determine the integer- programming solution. We consider the 0-1 integer programming problem.

Maximize $z = \sum_{i=1}^{n} c_i x_i$

Subject to $\sum_{i=1}^{n} a_{ji} x_i \le b_j, (j = 1,2,...,m),$

$x_j = 0$ or 1 $(i = 1,2,...,n).$ (2)

We will restrict our attention to the case where *a* is 0-1, and where *b* is integer. Also with these restrictions, the problem is NP-hard, and several well known NP-hard problems, such as the set partitioning, covering and packing problems, are conveniently stated in this way. Common solution methods for ILP are based on solving LP, which can usually be done efficiently. If LP happens to give a 0-1 solution, this is also an optimal solution to ILP, and for certain common and nontrivial subclasses of 0-1 problems, LP always have an integer solution. For more difficult problems particular instances may also be easy in this sense. If however the LP solution has many noninteger values, very little information about the solution to the 0-1 problem is obtained in this way, and typically techniques such as branch and bound have to be used to resolve the solution to integrality. This can work very well for small problems and also for larger problems with special structure, but nevertheless strongly limits the range and size of problems that can be solved. For a full presentation of existing methods, see for example [12], [27]. Optimization techniques developed for real search spaces can be applied on Integer Programming problems and determine the optimum solution by rounding off the real optimum values to the nearest integer [23], [26]. One of the most common deterministic approaches for tackling

Integer Programming problems is the Branch and Bound (BB) technique [11], [22], [26]. The technique is based on the observation that the enumeration of integer solutions has a tree structure. The main idea of the branch and bound algorithm is to find an optimal solution and to prove its optimality by successively partitioning the feasible set of the solution, or the original problem, into subproblems of smaller size. These subproblems are investigated by computing lower/upper bounds of the objective function. These lower/upper bounds are used to avoid exhaustive search of the solution space. Evolutionary and Swarm Intelligence algorithms are stochastic optimization methods that involve algorithmic mechanisms similar to natural evolution and social behavior respectively. They can cope with problems that involve discontinuous objective functions and disjoint search spaces [8], [14], [28]. Genetic Algorithms (GA), Stochastic Diffusion Search (SDS), and the Particle Swarm Optimization (PSO) are the most common paradigms of such methods. Early approaches in the direction of Evolutionary Algorithms for Integer Programming are reported in [9], [13].

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) [1], [7], [8], [14], [28]. 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 [17]. 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) [31]. 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 x_{id}^k and velocity v_{id}^k of particle i in the k-th iteration are updated as follow:

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

$$v_{id}^{k+1} = wv_{id}^k + c_1 r_1 \left(pbest_{id}^k - x_{id}^k\right) + c_2 r_2 (gbest - x_{id}^k) \qquad (3)$$

In the second equation w is the inertia weight, $c1$ and $c2$ are positive constants called cognitive and social parameters, respectively, and $r1$ and $r2$ are random numbers selected in the interval [0 1]. The constants $c1$ and $c2$ represent the weighting of the stochastic acceleration terms that pull each particle towards pbest and gbest positions and usually are set $c1=c2=2$. 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:

$$w = w_{max} - \left(\frac{w_{max} - w_{min}}{k_{max}}\right) k \qquad (4)$$

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:

$$w_{max} = 1,\ w_{min} = 0 \qquad (5)$$

Sometimes as particle oscillations become wider, the system will gain tendency to explode [14]. The usual means of preventing explosion is simply to define a parameter 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:

$$\text{If } v_{id} > v_{max} \text{ then } v_{id} = v_{max} \qquad (6)$$

$$\text{If } v_{id} < -v_{max} \text{ then } v_{id} = -v_{max} \qquad (7)$$

The effect of this is to allow particles to oscillate within the bounds [14].

ISSN 22773061

```
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 t icle position
```

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.

STOCHASTIC DIFFUSION SEARCH

This section introduces Stochastic Diffusion Search (SDS) [2], 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 [5]. 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.

```
Initialising agents()
    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 )
        r ag = pick a random agent ()
        if (r ag. activity () == true)
            ag. setHypothesis ( r ag
.getHypothesis())
        else
            ag. setHypothesis
(randomHypothsis())
end
```

ISSN 22773061

Fig 3. Passive Recruitment Mode

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).

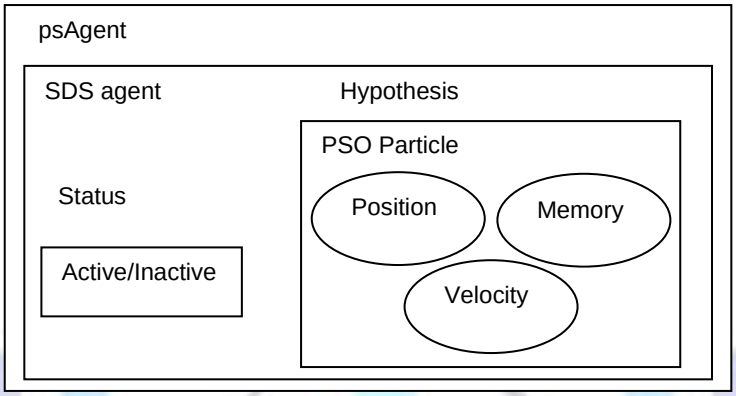

Fig 4. psAgent

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.

```
Solving problem using LP
Initialise 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
                r ag = 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 )
                    r ag = pick−random−psAgent()
                    if ( r ag . activity () == true )
                        ag. set-psAgentHypothesis(r ag. get-psAgentHypothesis())
                    else
                        ag. set-psAgentHypothesis ( randomHypothesis())
                    end i f
            end for
        end i f
        // 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
```

ISSN 22773061

Fig 5. Pseudocode of the proposed technique

COMPUTATIONAL RESULTS

This section presents computational results of the proposed SDS-PSO technique to solve Knapsack Problems. The computational results show performance comparison between the particle swarm optimization algorithm (PSO) and the proposed SDS-PSO technique. Table 1, and Figure 6 present execution time to solve knapsack problems by PSO and SDS-PSO technique. A proposed SDS-PSO technique run on computer (Core 2 Due CPU, 2.2GHz, 2048MB RAM.).

No. of Variables	SDS-PSO		PSO	
	Z	Time (ms)	Z	Time (ms)
10	241	98	241	360
20	425	279	425	751
50	569	621	569	1350
100	460	842	460	4687
200	658	1154	658	8547
500	965	2614	965	10115

Table 1. Execution time of different instances of knapsack problems

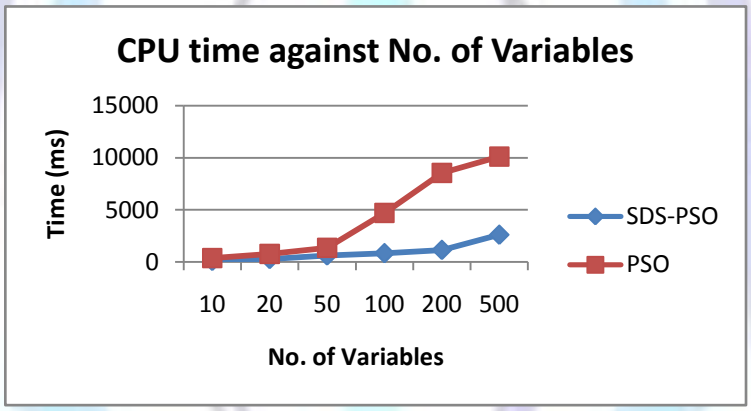

Fig 4. Execution time of different instances of knapsack problems

Computed results show that in small problems; with 10, and 20 variables, execution times were approximately equal in both algorithms (SDS-PSO, and PSO). However moving to largest scale problems, the execution time of SDS-PSO was obviously smaller than that in PSO. It may be included from the results that as the number of variables increase the difference between execution time of the two algorithms assures the better performance of the SDS-PSO algorithm with a clear reduction of execution time dealing with large problems. From the previous results we can conclude from the author's view, the efficiency improvement between SDS-PSO and PSO as shown in the following equation

$$\frac{T_k}{T_j} \times 100\%$$

Where the Tk is the execution time of SDS-PSO technique, and Tj is the execution time of PSO technique. The results are shown in table 2.

No. of Variables	Efficiency Improvement in Time
10	326.1%
20	537.9%
50	466.1%
100	188.6%
200	415.8%
500	728.5%

Table 2. Efficiency improvement in time between SDS-PSO and PSO

It appears that the efficiency in time of SDS-PSO is much greater than PSO. Also, it appears that in large problem sizes the time efficiency in SDS-PSO gets better than PSO.

CONCLUSION

This paper proposed a hybrid swarm intelligence technique called SDS-PSO to solve integer programming problems, and compares its performance with PSO technique. 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 PSO 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 Programming problems, when deterministic approaches fail, or it could b e considered as an algorithm for providing good initial points to deterministic methods, as the BB technique, and thus, help them converge to the global minimizer of the integer problem.

REFERENCES

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] Fogel, D.B. 1995. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press. New York.

[9] Gall, D.A. 1966. A Practical Multifactor Optimization Criterion, A. Levi, T.P.Vogl (Eds.), RecentAdvances in Optimization Techniques, 369-386.

[10] 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.

[11] Horst, R. and Tuy, H. 1996. Global Optimization, Deterministic Ap-proaches, Springer.

[12] Hu T.C., 1969. Integer programming and network flows, Addison Wesley.

[13] Kelahan, R.C. and Gaddy, J.L. 1978. Application of the Adaptive Random Search to Discrete and Mixed Integer Optimization", International Journal for Numerical Methods in Enginnering, Vol. 12, 289-298.

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

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

[16] Kennedy, J., Eberhart, R.C. and Shi, Y. (Eds.). 2001. Swarm Intelligence, Morgan Kaufmann, an Francisco, U.S.A.

[17] 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.

[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] Nemhauser, G.L. and Wolsey, L.A. 1988. Integer and Combinatorial Optimization, John Wiley and Sons.

[23] Nemhauser, G.L., Rinnooy Kan, A.H.G. and Todd, M.J. (Eds.). 1989. Handbooks in OR & MS, Vol. 1. Optimization, Elsevier.

[24] 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.

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

[26] Rao, S.S. 1996. Engineering Optimization-Theory and Practice, Wiley Eastern: New Delhi.

[27] Schrijver A. 1995. Theory of Linear and Integer Programming, Wiley.

[28] Schwefel, H.-P. 1995. Evolution and Optimum Seeking, Wiley.

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

[30] 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.

[31] 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.

Ibrahim El-henawy received the M.S. and Ph.D. degrees in computer science from State University of New York, USA in 1980 and 1983, respectively. Currently, he is a professor in computer science and mathematics department, Zagazig University. His current research interests are mathematics, networks, artificial intelligence, optimization, digital image processing, and pattern recognition

Mahmoud M. Ismail received the B.Sc. degree in information system and technology from Zagazig University in 2004, and he obtained his M.S. degree in operations research and decision support systems from monofia University, in 2008. Currently, he is a teaching assistant in operations research department, Zagazig University. His current research interests are optimization, artificial intelligence and decision support systems.