MACIS

Mathematical Aspects of Computer and Information Sciences

Proceedings of the First International Conference on Mathematical Aspects of Computer and Information Sciences

EDITED BY
Dongming Wang and Zhiming Zheng

July 24–26
2006

Beijing, China

Conference Themes

Modeling and Analysis of Complex Systems
Symbolic and Numeric Computation
Algorithms and Complexity

www.cc4cm.org/macis2006 cm.buaa.edu.cn/macis2006

Unified Particle Swarm Optimization for Hadamard Matrices of Williamson Type

Ilias S. Kotsireas, Christos Koukouvinos, Konstantinos E. Parsopoulos and Michael N. Vrahatis

Abstract. In this work we apply the recently proposed Unified Particle Swarm Optimization (UPSO) method to the search for Hadamard matrices of the Williamson type. The objective functions that arise from the classical Williamson construction, are ideally suited for UPSO algorithms. This is the first time that swarm intelligence methods are applied to this problem.

Mathematics Subject Classification (2000). 05B20, 13P10.

Keywords. Hadamard Matrices, Computational Algebra, Unified Particle Swarm Optimization, metaheuristics, Hadamard equivalence.

1. Introduction

Hadamard matrices arise in Statistics and Combinatorics and have many applications in Engineering, Optical Communications, Cryptography and other areas. The book [3] is a very readable and self-contained introduction to Hadamard matrices.

There are several well-known constructions for Hadamard matrices. Hadamard matrices of Williamson type are typically made up of four square matrices satisfying certain algebraic conditions.

The Computational Algebra formalism developed in [7] allows us to apply UPSO methods to the search for Hadamard matrices of the Williamson type. The objective functions (OFs) that arise from the Williamson construction are directly usable in UPSO algorithms.

This work is supported in part by an NSERC grant.

2. Hadamard Matrices of Williamson Type

The classical Williamson construction for Hadamard matrices is based on the 4×4 array

$$W = \begin{pmatrix} A & B & C & D \\ -B & A & -D & C \\ -C & D & A & -B \\ -D & -C & B & A \end{pmatrix} \tag{2.1}$$

which has the property

$$WW^T = (A^2 + B^2 + C^2 + D^2) \otimes I_4.$$

(here A, B, C, D are square matrices of order n and the symbol $\otimes$ denotes the Kronecker product). When A, B, C, D are square circulant and symmetric $(1, -1)$ matrices of order n, then W turns out to be a Hadamard matrix of order $4n$, i.e. we have $WW^T = 4nI_{4n}$. See [17] or [4] for all the details.

2.1. Systems of polynomial equations arising from the four Williamson array

In this section we detail the four and eight Williamson arrays constructions for Hadamard matrices and define the systems of polynomial equations arising from these constructions. Let n be an odd positive integer with $n \geq 3$ and let U be the matrix of order n

$$U = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix}$$

which has the property $U^n = I_n$. Following Williamson, [4], we will use the matrix U to define the block matrices of order n in the four and eight Williamson arrays, as polynomials in U with ± 1 coefficients. Then the block matrices will commute with each other. Moreover, by imposing symmetry condition on the coefficients, the block matrices will be symmetric, in view of the fact that $U^T = U^{-1}$. In the next two paragraphs we detail these ideas for the four and eight Williamson arrays separately.

2.2. Four Williamson array construction

In the four Williamson array (2.1), define the four matrices A, B, C, D by polynomials in U as follows:

$$\begin{array}{rcccccccc} A & = & a_0 I_n & + & a_1 U & + & \cdots & + & a_{n-1} U^{n-1} \\ B & = & b_0 I_n & + & b_1 U & + & \cdots & + & b_{n-1} U^{n-1} \\ C & = & c_0 I_n & + & c_1 U & + & \cdots & + & c_{n-1} U^{n-1} \\ D & = & d_0 I_n & + & d_1 U & + & \cdots & + & d_{n-1} U^{n-1} \end{array} \tag{2.2}$$

where the $4n$ coefficients $a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}, c_0, \ldots, c_{n-1}, d_0, \ldots, d_{n-1}$ satisfy the additional symmetry conditions

$$a_{n-i} = a_i, b_{n-i} = b_i, c_{n-i} = c_i, d_{n-i} = d_i, \ i = 1, \ldots, n-1. \tag{2.3}$$

Let $m = \frac{n-1}{2}$. Then we see that we actually need only the $4(m+1) = 2n+2$ coefficients $a_0, \ldots, a_m, b_0, \ldots, b_m, c_0, \ldots, c_m, d_0, \ldots, d_m$ to define the polynomials in U, because the symmetry conditions (2.3) imply

$$a_1 = a_{n-1}, a_2 = a_{n-2}, \ \ldots, \ a_{m-1} = a_{n-m+1}, a_m = a_{n-m}$$

and the analogous conditions for the coefficients b_i, c_i and d_i. Now the matrix identity $WW^T = 4nI_{4n}$ can be stated as a system of m (resp. $\frac{n-1}{2}$) polynomial quadratic equations in the $4(m+1)$ (resp. $2n+2$) unknowns

$$w_1 = 0, \ \ldots, \ w_m = 0,$$

supplemented by the $4(m+1)$ (resp. $2n+2$) quadratic equations

$$\underbrace{a_0^2 = 1, \ldots, a_m^2 = 1}_{a_i's}, \underbrace{b_0^2 = 1, \ldots, b_m^2 = 1}_{b_i's}, \underbrace{c_0^2 = 1, \ldots, c_m^2 = 1}_{c_i's}, \underbrace{d_0^2 = 1, \ldots, d_m^2 = 1}_{d_i's},$$

to account for the fact that the polynomials (2.2) are defined with ± 1 coefficients. Each of the m equations $w_1, \ldots, w_m$ contains the constant factor 2, m quadratic monomials in the variables a_i, m quadratic monomials in the variables b_i, m quadratic monomials in the variables c_i and m quadratic monomials in the variables d_i. For each i ranging from 1 to m, the equation w_i contains the quadratic monomials $a_0a_i, b_0b_i, c_0c_i, d_0, d_i$. In particular, this means that the sum $w_1 + \cdots + w_m$ contains the factors $a_0(a_1 + \cdots + a_m)$, $b_0(b_1 + \cdots + b_m)$, $c_0(c_1 + \cdots + c_m)$ and $d_0(d_1 + \cdots + d_m)$. For each i ranging from 1 to m, the equation w_i contains $m-1$ quadratic monomials of the second elementary symmetric function in the m variables $a_1, \ldots, a_m$ (and the corresponding quadratic monomials of elementary symmetric functions for the b_i, c_i and d_i variables). The structure of the indices of the quadratic monomials for the a, b, c, d variables in each equation w_i, is the same. For illustration, we mention the general form of the first equation w_1:

$$w_1 = 2 + \left(\sum_{i=1}^{m} a_{i-1}a_i\right) + \left(\sum_{i=1}^{m} b_{i-1}b_i\right) + \left(\sum_{i=1}^{m} c_{i-1}c_i\right) + \left(\sum_{i=1}^{m} d_{i-1}d_i\right) = 0.$$

3. Unified Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population–based, stochastic optimization algorithm [5]. Its dynamic is governed by fundamental laws encountered in swarms in nature hence it is categorized as a swarm intelligence algorithm [6].

Similarly to other population–based algorithms, PSO exploits a population of search points to probe the search space. In the context of PSO, the population is called a *swarm*, while the search points are called *particles*. Each particle moves in the search space with an adaptable velocity, recording the best position it has

ever visited in the search space. In minimization problems, such positions have the lowest function values.

The adaptation of the velocity is based on information coming from the particle itself, as well as, from the rest of the particles. More specifically, each particle has a "neighborhood" that consists of some prespecified particles and the best position ever attained by any member of the neighborhood is communicated to the particle and influences its movement.

Assume the problem of minimizing an n–dimensional function,

$$\min_{X \in S} f(X), \qquad S \subset \mathbb{R}^n.$$

Then, a swarm to tackle this problem consists of N particles,

$$\mathbb{S} = \{X_1, X_2, \ldots, X_N\},$$

which are n–dimensional vectors, $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in S$, $i = 1, \ldots, N$. The *velocity*, $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top$, of the ith particle, as well as its *best position*, $P_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in S$, are also n–dimensional vectors.

The neighborhoods are usually defined based on the particles' indices. The most common neighborhood topology is the "ring" topology, where the neighborhood of a particle consists of particles with neighboring indices. Thus, a neighborhood of radius m of X_i is the set

$$\mathbb{X}_i = \{X_{i-m}, \ldots, X_i, \ldots, X_{i+m}\},$$

where the particle X_1 is assumed to follow immediately after X_N.

Let g_i denote the index of the particle that attained the best previous position among all the particles in the neighborhood of X_i, i.e.,

$$f(P_{g_i}) \leqslant f(P_j), \quad \forall\ j \in \{i-m, \ldots, i+m\},$$

and let t be the iteration counter. Then, the velocity and position of X_i are updated according to the equations [1, 16],

$$V_i(t+1) = \chi\Big[V_i(t) + c_1 R_1\big(P_i(t) - X_i(t)\big) + c_2 R_2\big(P_{g_i}(t) - X_i(t)\big)\Big], \quad (3.1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1), \quad (3.2)$$

where χ is a parameter called the *constriction coefficient*; c_1, c_2 are positive acceleration parameters called *cognitive* and *social* parameter, respectively; and R_1, R_2 are vectors with components uniformly distributed in the range $[0, 1]$. All vector operations in Eqs. (3.1) and (3.2) are performed componentwise. The best positions are updated at each iteration according to,

$$P_i(t+1) = \begin{cases} X_i(t+1), & \text{if } f\big(X_i(t+1)\big) < f\big(P_i(t)\big), \\ P_i(t), & \text{otherwise.} \end{cases}$$

Clerc and Kennedy studied the stability of PSO, proposing values of its parameters that promote convergence of the algorithm towards the most promising solutions in the search space [1, 16].

The search procedure of a population–based algorithm such as PSO consists of two main phases, *exploration* and *exploitation*. The former is responsible for the detection of the most promising regions in the search space, while the latter promotes convergence of the particles towards the best solution detected so far. These two phases can take place either once or successively during the execution of the algorithm.

There are two main variants of PSO, with respect to the number of particles that comprise the neighborhoods. In the *global* variant, the whole swarm is considered as the neighborhood for every particle. On the other hand, in the *local* variant, the neighborhood size is strictly smaller than the size of the swarm. The global variant converges faster than the local one, since all particles are attracted by the same best position. Therefore, it is distinguished for its exploitation ability. On the other hand, the local variant has better exploration properties, since the information regarding the best position of each neighborhood is gradually communicated to the rest of the particles through their neighbors in the ring topology. Thus, the attraction towards a specific point is weaker, preventing the swarm from getting trapped in suboptimal solutions. Proper selection of the neighborhood size affects PSO's trade–off between exploration and exploitation, albeit there is no formal procedure to determine the optimal size.

Unified Particle Swarm Optimization (UPSO) was recently proposed as a scheme that harnesses the local and global PSO variants, combining their exploration and exploitation properties [11, 12, 13]. Let X_i be the ith particle of the swarm, g be the index of the best particle in the whole swarm and g_i be the index of the best particle in the neighborhood of X_i, as described in the previous section. Also, let $\mathcal{G}_i(t+1)$ be the velocity update of X_i for the global PSO variant, let $\mathcal{L}_i(t+1)$ be the velocity update of X_i for the local PSO variant, and t denote the iteration counter. Then, from Eq. (3.1), it holds that,

$$\mathcal{G}_i(t+1) = \chi\left[V_i(t) + c_1 r_1\big(P_i(t) - X_i(t)\big) + c_2 r_2\big(P_g(t) - X_i(t)\big)\right], \quad (3.3)$$

$$\mathcal{L}_i(t+1) = \chi\left[V_i(t) + c_1 r_1'\big(P_i(t) - X_i(t)\big) + c_2 r_2'\big(P_{g_i}(t) - X_i(t)\big)\right]. \quad (3.4)$$

The aggregation of the search directions defined by Eqs. (3.3) and (3.4) results in the main UPSO scheme [11],

$$\mathcal{U}_i(t+1) = u\,\mathcal{G}_i(t+1) + (1-u)\,\mathcal{L}_i(t+1), \quad u \in [0,1], \quad (3.5)$$

$$X_i(t+1) = X_i(t) + \mathcal{U}_i(t+1). \quad (3.6)$$

The parameter u is called the *unification factor* and it balances the influence of the global and local search directions. The standard global PSO variant is obtained by setting $u = 1$ in Eq. (3.5), while $u = 0$ results in the standard local PSO variant. All intermediate values of $u \in (0,1)$ define composite UPSO variants that combine the exploration and exploitation properties of the global and local PSO variant.

Besides the basic UPSO scheme, a stochastic parameter can also be incorporated in Eq. (3.5) to enhance UPSO's exploration capabilities [11]. Thus, depending

on which variant UPSO is mostly based, Eq. (3.5) becomes,

$$\mathcal{U}_i(t+1) = r_3\, u\, \mathcal{G}_i(t+1) + (1-u)\, \mathcal{L}_i(t+1), \tag{3.7}$$

which is mostly based on the local variant, or,

$$\mathcal{U}_i(t+1) = u\, \mathcal{G}_i(t+1) + r_3\, (1-u)\, \mathcal{L}_i(t+1), \tag{3.8}$$

which is mostly based on the global variant. The parameter $r_3 \sim \mathcal{N}(M, \Sigma)$ is a normally distributed parameter with mean vector M and covariance matrix Σ. The use of r_3 imitates mutation in evolutionary algorithms. However, the mutation in UPSO is biased towards directions that are consistent with the PSO dynamic, in contrast to the pure random mutation used in evolutionary algorithms. Following the assumptions of Matyas [8], a proof of convergence in probability was derived for the UPSO variants of Eqs. (3.7) and (3.8) [11].

4. Sample Results

In this section we report on some results we obtained using UPSO for the following 36-variable OF

```
OF =
|a0*a2+a1*a3+a2*a4+a3*a5+a4*a6+a5*a7+a6*a8+a7*a8
+b0*b2+b1*b3+b2*b4+b3*b5+b4*b6+b5*b7+b6*b8+b7*b8
+c0*c2+c1*c3+c2*c4+c3*c5+c4*c6+c5*c7+c6*c8+c7*c8
+d0*d2+d1*d3+d2*d4+d3*d5+d4*d6+d5*d7+d6*d8+d7*d8+2|
+
|a0*a1+a1*a2+a2*a3+a3*a4+a4*a5+a5*a6+a6*a7+a7*a8
+b0*b1+b1*b2+b2*b3+b3*b4+b4*b5+b5*b6+b6*b7+b7*b8
+c0*c1+c1*c2+c2*c3+c3*c4+c4*c5+c5*c6+c6*c7+c7*c8
+d0*d1+d1*d2+d2*d3+d3*d4+d4*d5+d5*d6+d6*d7+d7*d8+2|
+
|a0*a3+a1*a2+a1*a4+a2*a5+a3*a6+a4*a7+a5*a8+a6*a8
+b0*b3+b1*b2+b1*b4+b2*b5+b3*b6+b4*b7+b5*b8+b6*b8
+c0*c3+c1*c2+c1*c4+c2*c5+c3*c6+c4*c7+c5*c8+c6*c8
+d0*d3+d1*d2+d1*d4+d2*d5+d3*d6+d4*d7+d5*d8+d6*d8+2|
+
|a0*a4+a1*a3+a1*a5+a2*a6+a3*a7+a4*a8+a5*a8+a6*a7
+b0*b4+b1*b3+b1*b5+b2*b6+b3*b7+b4*b8+b5*b8+b6*b7
+c0*c4+c1*c3+c1*c5+c2*c6+c3*c7+c4*c8+c5*c8+c6*c7
+d0*d4+d1*d3+d1*d5+d2*d6+d3*d7+d4*d8+d5*d8+d6*d7+2|
+
|a0*a7+a1*a6+a1*a8+a2*a5+a2*a8+a3*a4+a3*a7+a4*a6
+b0*b7+b1*b6+b1*b8+b2*b5+b2*b8+b3*b4+b3*b7+b4*b6
+c0*c7+c1*c6+c1*c8+c2*c5+c2*c8+c3*c4+c3*c7+c4*c6
+d0*d7+d1*d6+d1*d8+d2*d5+d2*d8+d3*d4+d3*d7+d4*d6+2|
+
```

```
|a0*a8+a1*a7+a1*a8+a2*a6+a2*a7+a3*a5+a3*a6+a4*a5
+b0*b8+b1*b7+b1*b8+b2*b6+b2*b7+b3*b5+b3*b6+b4*b5
+c0*c8+c1*c7+c1*c8+c2*c6+c2*c7+c3*c5+c3*c6+c4*c5
+d0*d8+d1*d7+d1*d8+d2*d6+d2*d7+d3*d5+d3*d6+d4*d5+2|
+
|a0*a5+a1*a4+a1*a6+a2*a3+a2*a7+a3*a8+a4*a8+a5*a7
+b0*b5+b1*b4+b1*b6+b2*b3+b2*b7+b3*b8+b4*b8+b5*b7
+c0*c5+c1*c4+c1*c6+c2*c3+c2*c7+c3*c8+c4*c8+c5*c7
+d0*d5+d1*d4+d1*d6+d2*d3+d2*d7+d3*d8+d4*d8+d5*d7+2|
+
|a0*a6+a1*a5+a1*a7+a2*a4+a2*a8+a3*a8+a4*a7+a5*a6
+b0*b6+b1*b5+b1*b7+b2*b4+b2*b8+b3*b8+b4*b7+b5*b6
+c0*c6+c1*c5+c1*c7+c2*c4+c2*c8+c3*c8+c4*c7+c5*c6
+d0*d6+d1*d5+d1*d7+d2*d4+d2*d8+d3*d8+d4*d7+d5*d6+2|.
```

The OF contains the 36 binary variables

```
a0,...,a8,b0,...,b8,c0,...,c8,d0,...,d8
```

The seven solutions below were obtained.

```
[ 1  1 -1  1 -1  1  1  1 -1  1 -1 -1 -1 -1  1 -1  1  1
 -1  1  1 -1 -1 -1 -1 -1  1 -1  1 -1 -1  1  1 -1  1  1]

[-1  1  1  1 -1  1  1 -1 -1  1 -1  1 -1 -1 -1 -1 -1  1
  1 -1  1  1 -1  1 -1 -1  1  1  1  1 -1 -1 -1  1 -1 -1]

[1 -1 -1  1 -1 -1 -1  1  1  1  1 -1  1  1  1 -1 -1 -1
 1 -1 -1  1 -1  1 -1 -1 -1 -1  1 -1  1  1 -1 -1  1  1]

[1 -1 -1 -1 -1 -1  1  1 -1 -1 -1  1 -1  1  1 -1  1  1
 1 -1  1 -1 -1 -1  1  1  1 -1  1  1  1 -1 -1  1 -1  1]

[1 -1  1 -1 -1  1  1 -1 -1 -1  1  1 -1  1  1  1 -1 -1
-1 -1  1 -1 -1 -1  1  1  1  1 -1 -1  1 -1  1 -1 -1 -1]

[1 -1  1  1 -1  1 -1 -1  1 -1  1 -1  1  1  1  1  1 -1
 1 -1 -1 -1  1 -1 -1  1  1  1  1  1 -1 -1 -1  1 -1 -1]

[-1  1  1 -1  1 -1  1  1  1 -1 -1  1 -1 -1 -1  1  1  1
  1 -1 -1  1 -1 -1 -1  1  1 -1  1 -1  1  1 -1 -1  1  1]
```

5. Conclusion

In this work we use UPSO algorithms to tackle hard discrete optimization problems arising in the search for Hadamard matrices of Williamson type. The results are quite encouraging and we firmly believe that these algorithms constitute a very

promising avenue to explore, in connection with these problems. We anticipate that we will report many more results in the final version of the paper.

References

[1] M. Clerc, J. Kennedy, The particle swarm–explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.

[2] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization, in: V. W. Porto, N. Saravanan, D. Waagen, A. E. Eiben (Eds.), Evolutionary Programming, Vol. VII, Springer, 1998, pp. 611–616.

[3] Geramita, A. V. and Seberry, J., Orthogonal designs. Quadratic forms and Hadamard matrices, Lecture Notes in Pure and Applied Mathematics, 45, Marcel Dekker Inc., New York, 1979.

[4] Hall, Jr., M., Combinatorial theory, 2nd ed. John Wiley & Sons Inc., New York, 1998.

[5] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings IEEE International Conference on Neural Networks, Vol. IV, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.

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

[7] I. Kotsireas and C. Koukouvinos, Constructions for Hadamard matrices of Williamson type Journal of Combinatorial Mathematics and Combinatorial Computing, accepted.

[8] J. Matyas, Random Optimization, Automatization and Remote Control 26 (1965) 244–251.

[9] K.E. Parsopoulos, M.N. Vrahatis, Initializing the particle swarm optimizer using the nonlinear simplex method, in: A. Grmela, N. Mastorakis (Eds.), Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, WSEAS Press, 2002, pp. 216–221.

[10] K.E. Parsopoulos, M.N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization, Natural Computing 1 (2–3) (2002) 235–306.

[11] K.E. Parsopoulos, M.N. Vrahatis, UPSO: A Unified Particle Swarm Optimization Scheme, Lecture Series on Computer and Computational Sciences, Vol. 1, Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004), VSP International Science Publishers, 2004, pp. 868–873.

[12] K.E. Parsopoulos, M.N. Vrahatis, Unified Particle Swarm Optimization in Dynamic Environments, Lecture Notes in Computer Science (LNCS), Vol. 3449, Springer Verlag, 2005, pp. 590–599.

[13] K.E. Parsopoulos, M.N. Vrahatis, Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems, Lecture Notes in Computer Science (LNCS), Vol. 3612, Springer Verlag, 2005, pp. 582–591.

[14] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings IEEE Conference on Evolutionary Computation, IEEE Service Center, Anchorage, AK, 1998, pp. 69–73.

[15] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: V.W. Porto, N. Saravanan, D. Waagen, A.E. Eiben (Eds.), Evolutionary Programming, Vol. VII, Springer, 1998, pp. 591–600.

[16] I.C. Trelea, The particle swarm optimization algorithm: Convergence analysis and parameter selection, Information Processing Letters 85 (2003) 317–325.

[17] Wallis, W. D. and Street, A. P. and Seberry Wallis, J., Combinatorics: Room squares, sum-free sets, Hadamard matrices, Lecture Notes in Mathematics, Vol. 292, Springer-Verlag, Berlin, 1972.

Ilias S. Kotsireas
Wilfrid Laurier University
Department of Physics and Computer Science
75 University Avenue West
Waterloo, Ontario N2L 3C5
Canada
e-mail: ikotsire@wlu.ca

Christos Koukouvinos
Department of Mathematics
National Technical University of Athens
Zografou 15773, Athens
Greece
e-mail: ckoukouv@math.ntua.gr

Konstantinos E. Parsopoulos
Computational Intelligence Laboratory (CILAB)
Department of Mathematics, University of Patras and
University of Patras Artificial Intelligence Research Center (UPAIRC)
GR-26110 Patras
Greece
e-mail: kostasp@math.upatras.gr

Michael N. Vrahatis
Computational Intelligence Laboratory (CILAB)
Department of Mathematics, University of Patras and
University of Patras Artificial Intelligence Research Center (UPAIRC)
GR-26110 Patras
Greece
e-mail: vrahatis@math.upatras.gr