ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Binary Particle Swarm Optimisation With Improved Scaling Behaviour

Denise Gorse[1]

[1]Dept of Computer Science, University College London,
Gower Street, London WC1E 6BT, UK
D.Gorse@cs.ucl.ac.uk

Abstract. A boolean particle swarm optimisation (PSO) algorithm is presented that builds on the strengths of earlier proposals but which by introducing a wholly random element into the search process shows greatly improved performance in higher dimensional search spaces in comparison also to the binary PSO algorithm of Kennedy and Eberhart.

Keywords. Particle swarm, binary PSO, boolean PSO, discrete search.

1 Introduction

Since the introduction of particle swarm optimisation (PSO) [1], a population-based search algorithm based on observations of social behaviour in birds and other animals, there have been many proposed applications [2]. However some problems have a discrete character and the original PSO was not suited to searching in such spaces.

A binary form of PSO was first proposed also by Kennedy and Eberhart [3]. Like their continuous PSO this algorithm (here denoted KE BPSO) is simple in summary and has remained popular. But as will be demonstrated it has poor scaling behaviour. It is also possible to criticise the algorithm on the grounds that it changes the meaning of velocity from the usual one of a rate of change of position to the probability of a velocity component's taking on some specified value [4,5], and some of KE BPSO's difficulties with higher dimensional search spaces may be linked to these issues.

There have since been a large number of reformulations of BPSO, though mostly variants of the original KE proposal. Boolean PSO [5,6,7] is a more radical reformulation that works entirely with binary variables and bitwise operators (AND, OR, NOT, etc) and as such might be considered a more natural extension of the principles of PSO to discrete problems. However these boolean algorithms too are not always effective in higher dimensional search spaces.

It will be shown that a simple adaptation of one such boolean algorithm, presented initially by Afshinmanesh and Marandi in [5] and with some modification in [8,9] not only improves performance substantially, by five to eight orders of magnitude in some cases, but also makes the new algorithm considerably more effective than the Kennedy-Eberhart binary algorithm, still widely used in discrete search applications.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

2 Continuous PSO

PSO combines learning based on each particle's own past experience (*cognitive* contribution) and learning based on following the swarm's best performing member (*social* contribution). Each particle i=1..S has a velocity $\mathbf{v}_i$ and position $\mathbf{x}_i$, where the latter encodes a candidate solution as a parameter vector of length N. The equations used to update velocity and position in continuous PSO are

$$\mathbf{v}_{i,t+1} = w\mathbf{v}_{i,t} + c_1 r_1(\mathbf{p}_{i,t} - \mathbf{x}_{i,t}) + c_2 r_2(\mathbf{g}_t - \mathbf{x}_{i,t}) \quad , \tag{1a}$$

$$\mathbf{x}_{i,t+1} = \mathbf{x}_{i,t} + \mathbf{v}_{i,t+1} \quad , \tag{1b}$$

where $\mathbf{p}_{i,t}$ ('personal best' or *pbest*) is the best parameter position found at time t by particle i, $\mathbf{g}_t$ ('global best' or *gbest*) is the best position found at this time by any particle, c_1 and c_2 are constants most often set to 2.0, and r_1 and r_2 are random numbers generated uniformly from [0,1]. In addition w is a (usually time-decreasing) *inertia weight* that controls the degree to which new search directions are pursued, and it is also usually recommended to restrain the velocity to a maximum magnitude V_{max} (a value of 4 or 5 is typical) so that the search remains within useful bounds.

3 Binary PSO

In the form of discrete PSO proposed by Kennedy and Eberhart [3] the particle velocities $\mathbf{v}_i$ remain real-valued and are updated as in (1a). In order to generate an updated binary position, velocity components are transformed into [0,1] by a sigmoid limiting function *sig*() such that the probability the jth component of particle i's position vector is equal to 1 is given by Prob(x_{ij}=1)=*sig*(v_{ij}). As pointed out in [4] this update rule introduces a number of potential problems: (i) velocity is no longer linked to the likelihood of a change in position but to the probability of being in a particular position; (ii) the algorithm has no memory in that position updates now do not directly take into account past positions; (iii) the sigmoid function can saturate and causes bits to be stuck in either a 0 or 1 state; and (iv) the use of a time-decreasing inertia weight instead of promoting convergence causes a drift toward a condition of maximum uncertainty (equal likelihood of position components being in bit-state 0 or 1).

The effects of (iii) and (iv), since they act oppositely, may with luck cancel, and this may account for some of the KE algorithm's successes, but this does not stop these features from being potential problems, and alternative BPSOs sought.

4 Boolean PSO

In a boolean PSO all variables are binary and arithmetic operators are replaced by logical ones. In the following '.' will mean AND, '+' OR, and '⊕' the XOR operation. For convenience binary vectors will be combined with these operators, with the understanding that the operators are to be applied to corresponding vector components.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

One such proposal by Afshinamanesh and Marandi [5] has an especial simplicity, with equations (1a, 1b) being modified to the form

$$\mathbf{v}_{i,t+1} = w.\mathbf{v}_{i,t} + r_1.(\mathbf{p}_{i,t} \oplus \mathbf{x}_{i,t}) + r_2.(\mathbf{g}_t \oplus \mathbf{x}_{i,t}) \quad , \tag{2a}$$

$$\mathbf{x}_{j,t+1} = \mathbf{x}_{i,t} \oplus \mathbf{v}_{i,t} \tag{2b}$$

in which Prob(r_1=1)=ρ_1, Prob(r_2=1)=ρ_2, Prob(w=1)=ω. Note that a large velocity (a high probability of a $\mathbf{v}_i$-component being 1) is now associated with a high probability of change in the corresponding position bit, since according to (2b) a v_{ij}=1 will cause a corresponding position bit flip whether x_{ij}=0 or x_{ij}=1.

Velocity is restricted here as for conventional PSO and KE BPSO, though V_{max} is now the maximum number of randomly chosen bit positions allowed to change their value, set to 5 bits out of an overall dimension BN=40 in [8] and 15 bits out of 40 in [9]. However using these values may be inappropriate for a problem of significantly larger dimension; experiments by the present author showed that convergence is then both prohibitively slow and often to sub-optimal solutions. Without guidance from the original work as to how V_{max} should scale a linear increase in this parameter is used here such that a maximum of 1/4 of the total number of bits (halfway between the recommendations of [8] and [9]) are allowed to change at each iteration.

4.1 Boolean PSO With Noise

In the test problems considered by Afshinmanesh and Marandi and co-workers the search space was relatively small (30 or 40 dimensional). It will be seen later that this algorithm, like KE BPSO, does not scale well as the search space is enlarged, in the experiments to be described below up to 1000 binary dimensions. It is proposed here to add a 'pure noise' term to the velocity update rule (2a), which now becomes

$$\mathbf{v}_{i,t+1} = b + (1-b)w.\mathbf{v}_{i,t} + r_1.(\mathbf{p}_{i,t} \oplus \mathbf{x}_{i,t}) + r_2.(\mathbf{g}_t \oplus \mathbf{x}_{i,t}) \quad , \tag{3}$$

with the position update rule (2b) as before, and where Prob(b=1)=β. β is an exploration parameter which allows a small component of the position update rule to be unrelated either to a particle's current velocity or to the personal or global best positions.

5 Results

The work uses a swarm of 100 particles, the PSO algorithms run for 1000 iterations with the inertia weight w linearly decreasing from 1.0 to 0.2, Prob(r_1=1) = Prob(r_2=1) = 0.5, and with 10 runs performed under each set of experimental conditions.

5.1 Test Functions

The functional minimisation problems considered here are formulated most naturally in N real dimensions; however these functions (members of the de Jong test set) have

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

also been used frequently to test the effectiveness of binary optimisation algorithms by coding the inputs using *B* bits per variable such that overall the search space is now *BN*-dimensional. When converted into a binary search problem in this way these problems become challenging, especially if the number of bits per variable is large.

Table 1 lists the four functions used in this work. Of these the Sphere function f_1 is the easiest while the Rosenbrock function f_2 is considered most difficult.

Table 1. Test functions used

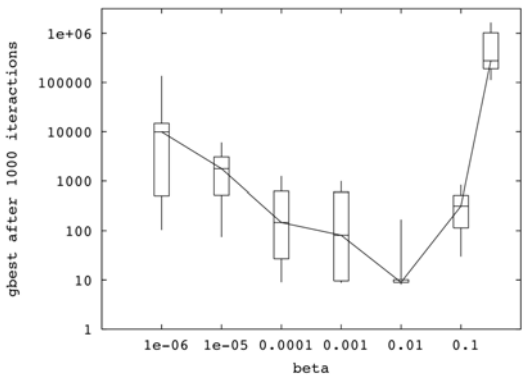
Sphere: $f_1(\mathbf{x}) = \sum_{j=1}^{N} x_j^2$	Rosenbrock: $f_2(\mathbf{x}) = \sum_{j=1}^{N-1} (100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2$
Griewangk: $f_3(\mathbf{x}) = \frac{1}{4000}\sum_{j=1}^{N} x_j^2 - \prod_{j=1}^{N}\left(\cos\frac{x_j}{\sqrt{j}}\right) + 1$	Rastrigin: $f_4(\mathbf{x}) = \sum_{j=1}^{N} (x_j^2 - 10\cos(2\pi x_j) + 10)$

A note on solution encoding. Binary optimisation algorithms applied to function minimisation problems of this kind do not normally recognise a difference between more and less significant bits, and hence the problems are made more difficult by using a larger number of bits per variable, *B*, as well as a larger number of variables, *N*, and a larger input domain. In order to create a challenging problem it was decided to follow [4] in using *B*=20 and a domain [-50,50] for each of the *N* variables.

5.2 Exploring the Role of the Noise Parameter β

The boxplot representation in Figure 1 below shows the effect of the parameter β on solution quality for the *N*=10 Rosenbrock function.

Fig. 1. Solution quality as a function of the noise parameter β for the *N*=10, *B*=20 (overall problem dimension 100) Rosenbrock discrete minimisation problem.



It can be seen that for small values of β (with the Afshinmanesh-Marandi algorithm recovered in the limit $\beta \to 0$) solution quality is significantly degraded. However it is seen also that it is undesirable to have too large a value for this parameter. This is

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

an instance of the control/exploration tradeoff familiar in any search or optimisation process that involves an element of pure randomness, the role here of a nonzero β.

Usng a value of β=0.01 the proposed algorithm was benchmarked for all four of the test functions against Kennedy-Eberhart BPSO (KE BPSO) and the Afshin-manesh-Marandi algorithm (equivalent here to using β=0). Input number N ranged from 3 (overall problem dimension 60) to 50 (dimension 1000), with results below.

Table 2. Comparison of Kennedy-Eberhart PSO (KE) with β=0.01 and β=0.00 boolean PSOs on the four test functions. Median gbest value after 1000 iterations is given, with the best performing algorithm for each problem dimension indicated by boldface type.

N	dim	f1: Sphere			f2: Rosenbrock		
	(BN)	KE BPSO	β=0.00	β=0.01	KE BPSO	β=0.00	β=0.01
3	60	6.139e-8	**6.821e-9**	**6.821e-9**	1.905e+0	2.000e+0	**1.881e+0**
5	100	2.568e-3	4.893e-4	**1.137e-8**	5.758e+0	4.272e+0	**3.767e+0**
10	200	8.915e+0	3.067e+0	**2.274e-8**	3.472e+3	9.944e+3	**8.956e+0**
20	400	6.547e+2	9.794e+1	**6.185e-7**	5.701e+6	3.682e+5	**1.694e+2**
50	1000	9.376e+3	1.678e+3	**2.241e+0**	6.823e+8	5.272e+7	**1.177e+3**

N	dim	f3: Griewangk			f4: Rastrigin		
	(BN)	KE BPSO	β=0.00	β=0.01	KE BPSO	β=0.00	β=0.01
3	60	1.081e-2	3.003e-2	**1.002e-2**	3.929e-4	9.950e-1	**1.353e-6**
5	100	3.984e-2	5.735e-2	**9.966e-3**	2.484e+0	3.109e+0	**1.062e+0**
10	200	4.350e-1	1.870e-1	**7.654e-2**	6.725e+1	2.554e+1	**4.214e+0**
20	400	1.166e+0	7.269e-1	**1.079e-1**	9.161e+2	2.628e+2	**2.919e+1**
50	1000	3.344e+0	1.400e+0	**8.369e-2**	9.282e+3	1.963e+3	**1.655e+2**

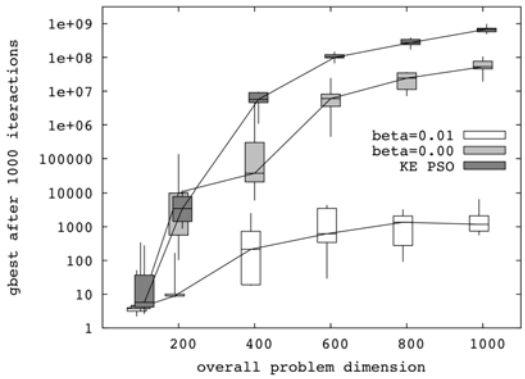
It can be seen that while the β=0.01 algorithm outperforms the others only marginally for $N \leq 5$ (problem dimension ≤ 100) for higher dimensions the improvement is considerably more marked, and Figure 2 illustrates this trend for the most difficult of the test functions, the Rosenbrock function, displaying boxplot data as a function of increasing problem dimension (100 to 1000 binary dimensions).

6 Discussion

It has been demonstrated that poor scaling behaviour can be ameliorated for a boolean PSO by a modification of the search process that introduces a 'pure noise' component. This is somewhat similar to the use of a mutation operator in a genetic algorithm, and the use of a mutation-like source of randomness was found to play an important role in [10] in which KE BPSO was compared favourably with a GA for the design of logic circuits. However the relative ease with which such a random search element can be inserted into a boolean PSO suggests this framework may be better suited than the Kennedy-Eberhart algorithm to such high dimensional applications.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Fig. 2. Scaling behaviours for variable *N*, *B*=20 Rosenbrock problem. Boxes have been offset slightly to improve readability; experiments were carried out for *N*=5,10,20,30,40,50.



References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, pp. 1942—1948. IEEE Press, New York (1995)
2. Banks, A., Vincent, J., Anyakoha, C.: An Review of Particle Swarm Optimization. Part II: Hybridisation, Combinatorial, Multicriteria and Constrained Optimization, and Indicative Applications. Natural Computing 7, 109—124 (2008)
3. Kennedy, J., Eberhart, R.C.: A Discrete Binary Version of the Particle Swarm Algorithm. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 4101—4108. IEEE Press, New York (1997)
4. Khaneser, M.A., Teshnehlab, M., Shoorehdeli, M.A.: A Novel Binary Particle Swarm Optimization. In: 15th Mediterranean Conf. on Control and Automation, pp. 1—6 (2007)
5. Afshinmanesh, F., Marandi, A., Rahimi-Khan, A.: A Novel Binary Particle Swarm Optimization Using Artificial Immune System. In: EUROCON 2005, pp. 217—220. IEEE Press, New York (2005)
6. Xing-Chen, H., Zheng, Q., Lei, T., Li-Ping, S.: Learning Bayesian Network Structures with Discrete Particle Swarm Optimization Algorithm. In: FOCI 2007, pp. 47—52. IEEE Press, New York (2007)
7. Zaharis, Z.D., Goudos, S.K., Yioultsis, T.V.: Application of Boolean PSO with Adaptive Velocity Mutation to the Design of Optimal Linear Antenna Arrays Excited by Uniform-Amplitude Current Distribution. J. Electromag. Waves and Appl. 25, 1422—1436 (2011)
8. Marandi, A., Afshinmanesh, F., Shahabadi, M., Bahrami, F.: Boolean Particle Swarm Optimization and Its Application to the Design of a Dual-Band Dual-Polarized Planar Antenna. In: 2006 IEEE Congress on Evolutionary Computation, pp. 3212—3218. IEEE Press, New York (2006)
9. Afshinmanesh, F., Marandi, A., Shahabadi, M.: Design of a Single-Feed Dual-Band Dual-Polarized Printed Microstrip Antenna Using a Boolean Particle Swarm Optimization. IEEE Trans. Antennas and Propagation 56, 1845—1852 (2008)
10. Coello Coello, C.A., Hernández Luna, E., Hernández Aguirre, A.: Use of Particle Swarm Optimization to Design Combinational Logic Circuits. In: Tirrell, A.M., Haddow, P.C., Torresen, J. (eds.) ICES 2003. LNCS vol. 2606, pp.398—409. Springer, Heidelberg (2003)