Use of Kaczmarz's Method in Intelligent-Particle Swarm Optimization

O. Tolga Altinoz[1], A. Egemen Yilmaz[2], Gabriela Ciuprina[3]

[1] TED University, Turkey
tolga.altinoz@tedu.edu.tr
[2] Ankara University, Turkey
aeyilmaz@eng.ankara.edu.tr
[3] Politehnica University of Bucharest, Romania
gabriela@lmn.pub.ro

Abstract

Intelligent-particle swarm optimization algorithm (IPSO) is an improved heuristic method estimating the quadratic model of the problem in the hand. The conventional way to find the coefficients of the second order equation is to solve the nonhomogeneous linear equation ($Ax = b$), where A is the Vandermonde matrix. Although this method seems to be quite simple to execute, matrix inversion must be performed at every iteration. Another disadvantage is that the Vandermonde matrix must be square, which means all data from the members in population cannot be used. Therefore, in this study, the iterative Kaczmarz's Method is integrated to IPSO. The obtained results are compared with the conventional algorithm.

1. Introduction

Over the last decade, optimization algorithms have been increasingly attracting attention of researchers from different disciplines. The heuristic optimization methods have successfully been applied for solution of the complex problems. As one of these algorithms, particle swarm optimization (PSO) [1] takes huge attention (more than 24000 citations according to Scopus as of the day this paper is written) due to its simple yet improvable structure. As one of the PSO variants, the intelligent-PSO (IPSO) [2] algorithm proposes more intelligent particles, which have the ability of observation, communication, reasoning, and learning.

In the IPSO algorithm, each particle can observe the position of others, and use their experiences to decide its own movement strategy by extracting the landscape of the problem, which can be relatively high-order but down-converted into a second order quadratic function. By this approximation, the unknown possible best position and its cost value can be obtained. Even though IPSO is composed of simple computations, the approximation stage is the most crucial and time-consuming part in the algorithm.

The approximated second order equation is extracted from the current cost values and position vectors of all particles. By using this data, the linear equation ($Ax = b$) is formed. The matrix A is called the Vandermonde matrix. The composition of the Vandermonde matrix is nothing but of the position vectors, which are the solution candidates. However, in order to obtain a solution (i.e. to find x), the inverse of the Vandermonde matrix must be computed. Therefore, the Vandermonde matrix must be square; where the numbers of its rows (and columns) must be equal to the number of unknown parameters in the second order polynomial. For example; if the problem dimension is equal to 3, then Vandermonde matrix must be 12-by-12. That means, if the population size of the problem is 100, then the positions of the 88 members remain unused. This has an impact on the performance. To investigate the influence of unused data on performance, Kaczmarz's Method (Algebraic Reconstruction Technique - ART) was applied into the algorithm and compared with the conventional form.

ART is an iterative algorithm, and it is preferred to find direct solutions for $Ax = b$ typr linear equations. Instead of the conventional solution ($x = A^{-1}b$ or $x = A^T(AA^T)^{-1}b$) with a square A matrix, the equation can be solved with this recursive method for a non-square matrix A. Although ART is a good candidate for solutions, it may become useless when compared to conventional method. For example, for a 10 dimensional problem, the Vandermonde matrix becomes 66-by-66, which is a large matrix, and almost all members of the population could be used.

In this study, Vandermonde matrix method (conventional way) and ART are compared for some benchmark problems. The solution quality (along with the iteration number of ART, and the problem dimension) is also investigated. The aim of this study is to find trade-off between two methods, and to propose a proper and parallelizable implementation of IPSO, which can be used in many engineering problems of various disciplines.

2. Intelligent Particle Swarm Optimization

The particle swarm optimization algorithm [1] (PSO) is one of the most popular optimization algorithms, and its origins rely on modelling of the behaviors of animals in a swarm. There are many improvements proposed since the original version was introduced in 1995 [3]. These improvements are mostly based on the systematic re-organization and hybridization of the original algorithm with ideas from other search/optimization algorithms.

In 2002, a novel method called Intelligent Particle Swarm Optimization (IPSO) for optimization algorithm was proposed by Ciuprina et al. [2]. Instead of making an improvement on PSO, this algorithm was based on hybridization of PSO with the Random Search Algorithm (RSA) [4] (Algorithm 1).

This study was supported by a joint grant from TUBITAK (with Grant Nr. 112E168) and ANCS-UEFISCDI (with Grant Nr. 605/01.01.2013)

Initialization;
Select random dx;
if $f(x+b+dx) < f(x)$ **then**
  $x = x + b + dx$;
  $b = 0.2b + 0.4dx$;
else
  if $f(x+b-dx) < f(x)$ **then**
    $x = x + b + dx$;
    $b = 0.2b - 0.4dx$;
  else
    $b = 0.5b$;
  end
end

Algorithm 1: Random Search Algorithm

for $j = 1$ *to* N **do**
  Randomly decide initial position x_j and temperament coefficient $U_j \epsilon [0, 1]$ Evaluate cost function f_j for particle x_j;
  Initial polarization is zero $p_j = 0$;
end
Find x_{best}, U_{best}, and f_{best};
for $j = 1$ *to* N **do**
  for *repeat until the stopping consitions(s) are met* **do**
    Generate dx_j randomly so that $x_j + p_j + dx_j$ in the search space;
    Update $dx_j = 2randU_j dx_j + 2rand(1 - U_j)(x_{best} - x_j)$;
    Calculate $x_{test,j} = x_j + p_j + dx_j$;
    Calculate $f_{test,j} = f(x_{test,j})$;
    if $f_{test,j} < f_j$ **then**
      $x_j = x_{test,j}$;
      $f_j = f_{test,j}$;
      $p_j = 0.2p_j + 0.4dx_j$;
      if $f_j < f_{best}$ **then**
        Improve best by quadratic modelling;
        Update $U_{best} = U_{best} + \alpha(U_j - U_{best})$;
      end
    else
      Calculate $x_{test,j} = x_j + p_j - dx_j$;
      Calculate $f_{test,j} = f(x_{test,j})$;
      if $f_{test,j} < f_j$ **then**
        $x_j = x_{test,j}$;
        $f_j = f_{test,j}$;
        $p_j = 0.2p_j - 0.4dx_j$;
        if $f_j < f_{best}$ **then**
          Improve best by quadratic modelling;
          Update $U_{best} = U_{best} + \alpha(U_j - U_{best})$;
        end
      else
        $p_j = 0.5p_j$;
      end
    end
  end
end

Algorithm 2: Intelligent Particle Swarm Optimization

All particles in PSO present the same decision behavior, which is indicated as Algorithm 1. In IPSO, same rules for RSA is implemented, except that; a) the new velocity ($x \pm dx$) update rule ($dx_j = 2randU_j dx_j + 2rand(1 - U_j)(x_{best} - x_j)$) b) temperament coefficient of the particle (U_j), which is a parameter expressing learning. It determines the attraction of a particle to the best position. c) obtaining best position and cost value from quadratic interpolated problem equation. The pseudo code of IPSO is indicated as Algorithm 2.

In Algorithm 2, the key section is Quadratic modelling, which is the estimation of the problem function with a second order equation. Then, the derivative of the estimated problem is computed in order to obtain the minimum of the function; this yields the new best position and corresponding best cost value. In the next section, two quadratic modelling techniques will be discussed.

2.1. Polynomial Interpolation

The best position and its corresponding cost function are obtained via the quadratic function. This function is estimated from the particle's position and cost values. In each iteration, the obtained data has the size equal to the number of the particles, where $(x_k, f(x_k))|k = 0, .., N-1$. It is assumed that the data is ordered with respect to the particle position $x_0 < x_1 < ... < x_{k-1}$. It is desired to find a polynomial function such that the curve of this function fits the given data. The n degree polynomial function (in IPSO the second order polynomial is taken in to account) is given in (1).

$$P(x) = a0 + a_1 x + a_2 x^2 + ... + a_n x^n = \sum_{k=0}^{n} a_k x^k \quad (1)$$

Therefore this problem can be formulated as a linear system ((**??**)) so that all given data must satisfy this equation.

$$\begin{bmatrix} x_0^n & x_0^{n-1} & ... & x_0 & 1 \\ x_1^n & x_1^{n-1} & ... & x_1 & 1 \\ ... & ... & ... & ... & ... \\ x_n^n & x_n^{n-1} & ... & x_n & 1 \end{bmatrix} \begin{bmatrix} a_n \\ a_{n-1} \\ ... \\ a_0 \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ ... \\ f_n \end{bmatrix} \quad (2)$$

The simplest way to solve this linear equation ($Ax = b$) is the inversion of the matrix A, and multiplication with the b vector. $x = A^{-1}b$. There are three main disadvantages of this approach: a) the matrix inverse operation. b) The Vandermonde matrix used throughout interpolation must be a square matrix. That means, some of the particles' position and cost values cannot be integrated into the matrix. c) The data couples must be sorted in such a manner that only the smallest cost valued particles will be part of the Vandermonde matrix. Therefore, in this study, in order to get rid of these problems, Kaczmarz's method (also known as Arithmetic Reconstruction Technique - ART) is applied inside the IPSO algorithm, and its performance is compared to the traditional Vandermonde matrix solution approach.

2.1.1. Kaczmarz's Method

This method is an iterative algorithm for solving the linear equation $Ax = b$, which is proposed by Kaczmarz in 1937 [5]. The method solves the linear system without inverting the A matrix, which has many rows. In this study, this method is compared with the traditional matrix inversion method. The ART has the advantages: a) matrix inversion operation becomes

an iterative operation, b) the matrix is not necessarily square; hence, it is possible to use all data obtained throughout the optimization process. On the other hand, despite these advantages, the algorithm has a big drawback related to the iteration size. As given in the proof of the Kaczmarz's method indicated as Algorithm 3, it is shown that the result of ART exactly converges to the real matrix inverse only if the number of iterations goes to infinity. But it should be noted that even a bad estimation of matrix inverse can yield a sufficiently-successful stochastic search in the optimization algorithm. In the implementations, the impact of the ART iterations will be investigated via the test problems.

for $i = 1$ *to selected maximum iteration* **do**
  for $j = 1$ *to* m*, the length of the row of* A **do**
    $k = im + j - 1$;
    $x^{(k+1)} = x^{(k)} + \mu \left(b_j - a_j^T x^{(k)} \right) \frac{a_j}{a_j^T a_j}$;
  end
end

Algorithm 3: Kaczmarz's Method

Proof. (1) Show nonincreasing sequence ([6] and [7])

$$\|x^{(k+1)} - x^*\|^2 < \|x^{(k)} - x^*\|^2$$

$$x^{(k+1)} - x^* = x^{(k)} - x^* + \mu \left(b_j - a_j^T x^{(k)} \right) \frac{a_j}{a_j^T a_j}$$

Note that

$$\|x + y\|^2 = \|x\|^2 + \|y\|^2 + 2x^T y$$

$$\|x^{(k+1)} - x^*\|^2 = \|x^{(k)} - x^* + \mu \left(b_j - a_j^T x^{(k)} \right) \frac{a_j}{a_j^T a_j}\|^2$$

$$= \|x^{(k)} - x^*\|^2 + \|\mu \left(b_j - a_j^T x^{(k)} \right) \frac{a_j}{a_j^T a_j}\|^2$$

$$+2\mu \left(b_j - a_j^T x^{(k)} \right)^T \frac{a_j^T}{a_j^T a_j} \left(x^{(k)} - x^* \right)$$

$$= \|x^{(k)} - x^*\|^2 + \|\mu \left(b_j - a_j^T x^{(k)} \right) \frac{a_j}{a_j^T a_j}\|^2$$

$$+2\mu \left(b_j - a_j^T x^{(k)} \right)^T \frac{1}{a_j^T a_j} \left(a_j^T x^{(k)} - b_j \right)$$

$$= \|x^{(k)} - x^*\|^2 + \mu^2 \left(b_j - a_j^T x^{(k)} \right)^2 \frac{a_j a_j^T}{\left(a_j^T a_j \right)^2}$$

$$-2\frac{\mu}{a_j^T a_j} \left(b_j - a_j^T x^{(k)} \right)^2$$

$$= \|x^{(k)} - x^*\|^2 - \frac{\mu}{a_j^T a_j} (2 - \mu) \left(b_j - a_j^T x^{(k)} \right)^2$$

Hence $0 < \mu < 2$ and $x^{(k)}$ is a non increasing sequence

□

Proof. (2) $x^{(k+1)}$ converges to x^* ([6] and [7])

$$\|x^{(k+1)} - x^*\|^2 - \|x^{(k)} - x^*\|^2 = -\frac{\mu}{a_j^T a_j} (2 - \mu) \left(b_j - a_j^T x^{(k)} \right)^2$$

$$= -\frac{\mu}{a_j^T a_j} (2 - \mu) \left(a_j^T \left(x^* - x^{(k)} \right) \right)^2$$

$$a_j \left(x^* - x^{(k)} \right) \to 0$$

Assume $x^{(k)} \to z^*$

$$A (x^* - z^*) = 0$$

There exists y^* such that $z^* = A^T y^*$ and $x^* = A^T \left(AA^T \right)^{-1} b$

$$A \left(A^T \left(AA^T \right)^{-1} b - A^T y^* \right) = 0$$

$$AA^T \left(AA^T \right)^{-1} b - AA^T y^* = 0$$

$$b - AA^T y^* = 0$$

$$b = AA^T y^*$$

$$b = Az^*$$

Hence $z^* = x^*$ □

3. Implementation on Benchmark Functions

The performance comparisons of the following two methods: a) conventional IPSO and b) IPSO with Kaczmarz's Method (KIPSO) are performed as regards the obtained cost function values and the execution times. For this purpose, five benchmark functions are selected from the literature. Table 1 presents these benchmark functions.

The performance of the KIPSO algorithm is highly dependent to the number of iterations. Therefore, the number of iterations for the Kaczmarz's method is varied, and its impact on the solutions are analyzed. The number of iterations is varied 3, 5, 10, 50 and 100. Same analysis is performed on benchmark functions with dimensions 2, 5 and 10. Table 2 presents the time and cost values of afore-cited methods for two-dimensional problems.

Table 2. The cost value and execution time (in sec.) of the benchmark functions with 2 dimensions. ART(x) stands for KIPSO with x number of iterations.

Func.	IPSO		ART(3)	
	f(x)	Time	f(x)	Time
f_1	$2.01e^{-28}$	0.53	$1.5e^{-4}$	0.57
f_2	$1.7e^{-2}$	0.51	$3.9e^{-2}$	0.83
f_3	$5.86e^{-29}$	0.52	$3.65e^{-4}$	0.54
f_4	$1.1e^{-3}$	0.52	$2.7e^{-2}$	0.81
f_5	$7.56e^{-4}$	0.52	$2.6e^{-2}$	0.73
	ART(5)		ART(10)	
f_1	$2.5e^{-3}$	0.62	$1.3e^{-3}$	0.62
f_2	$5.4e^{-2}$	1.01	$4.4e^{-2}$	1.48
f_3	$7.05e^{-4}$	0.54	$2e^{-3}$	0.59
f_4	$2.7e^{-2}$	0.97	$3.3e^{-3}$	1.4
f_5	$6.7e^{-2}$	1.02	$1.7e^{-2}$	1.51
	ART(50)		ART(100)	
f_1	0	0.58	0	2.3
f_2	$5.4e^{-2}$	5.4	0	0.55
f_3	0	0.53	0	1.78
f_4	$1.7e^{-3}$	5.22	$9.48e^{-5}$	3.68
f_5	$6.7e^{-2}$	5.41	$8.8e^{-16}$	0.55

The results presented in Table 2 indicates that best possible cost values can be obtained with the ART(100) and ART(50).

Table 1. Selected benchmark functions, where d is the dimension

Function	Searching Domain
$f_1(x) = \sum_{i=1}^{d} x_i^2$	$[-100, 100]^d$
$f_2(x) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	$[-10, 10]^d$
$f_3(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^d$
$f_4(x) = \sum_{i=1}^{d} 100 \left(x_i^2 - x_{i+1} \right)^2 + (x_i - 1)^2$	$[-5, 10]^d$
$f_5(x) = e - 20exp\left(-0.2\sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - exp\left(\frac{1}{10} \sum_{i=1}^{d} cos\left(2\pi x_i\right) \right) + 20$	$[-5.12, 5.12]^d$

As the iteration number increases for low dimensions, ART approaches to IPSO. All values become worseafor ART(10). Table 3 presents the results for 5-dimensional benchmark problems.

Table 3. The cost value and execution time (in sec.) of the benchmark functions with 5 dimensions. ART(x) stands for KIPSO with x number of iterations.

Func.	IPSO		ART(3)	
	f(x)	Time	f(x)	Time
f_1	$4.35e^{-26}$	0.58	$6.25e^{-5}$	0.61
f_2	$1.7e^{-1}$	0.54	$1.9e^{-1}$	1.01
f_3	$3.33e^{-23}$	0.58	$2.8e^{-3}$	0.56
f_4	3.52	0.58	3.81	0.86
f_5	$1.1e^{-1}$	0.57	1.67	1.7
	ART(5)		ART(10)	
f_1	$7e^{-5}$	0.6	$9.9e^{-10}$	0.59
f_2	$3.4e^{-1}$	1.5	$2.6e^{-1}$	3.41
f_3	$2e^{-3}$	0.6	$2.7e^{-3}$	0.6
f_4	7.15	1.47	6.99	2.89
f_5	$7.3e^{-1}$	2.9	$1.8e^{-1}$	4.24
	ART(50)		ART(100)	
f_1	$9.9e^{-12}$	0.71	$3.85e^{-11}$	0.85
f_2	$4.3e^{-1}$	16.6	$6.6e^{-1}$	43.45
f_3	$2.2e^{-3}$	0.83	$4.1e^{-3}$	0.83
f_4	7.03	17.71	9.02	37.69
f_5	$1.8e^{-1}$	19.15	$1.8e^{-1}$	37.39

The results in Table 3 clearly demonstrates the performance of IPSO against ART variations. ART presents relatively worse performance regardless of the number of iterations. The performance of ART increases with the low number of iterations. Table 4 is given for 10-dimensional problems.

Higher dimensional problems are always more difficult for the optimization algorithms. In Table 4, the results of IPSO and KIPSO are presented for ten dimensional versions of the same benchmark problems. The results are on behalf of the ART(3). It can be concluded from the results in Table 4 that ART(3) provides better solutions in almost the same time. In order to achieve a more general comment, same analysis is executed for higher dimension, i.e. 30.

Table 5 demonstrates the comparison of IPSO and ART(3) for 30-dimensional problems. It is clear that for higher dimensions ART(3) presents better performance against IPSO with lower number of iterations. In summary, for two dimensional problems, iteration between 50 and 100 can present better performance. For dimension 5, it is shown that ART(10) outperforms to other ART variations. As the dimension increases to ten, ARTs with iteration 3-5 present the best performance. Table 5 gives the results for problems with 30 dimensions. It is shown in the table that, the solution accuracy of the ART(3) is better for most of the problems in exchange for the time consumption.

Table 4. The cost value and execution time (in sec.) of the benchmark functions with 10 dimensions. ART(x) stands for KIPSO with x number of iterations.

Func.	IPSO		ART(3)	
	f(x)	Time	f(x)	Time
f_1	$9.32e^{-26}$	0.6	$1.3e^{-4}$	0.69
f_2	2.58	0.67	$5.8e^{-1}$	1.42
f_3	$1.37e^{-21}$	0.59	$3.65e^{-4}$	0.66
f_4	112.56	0.7	3.81	0.82
f_5	1.25	0.94	0.35	1.31
	ART(5)		ART(10)	
f_1	$1.27e^{-4}$	0.72	$4e^{-6}$	0.84
f_2	1.06	2.16	1.04	4.15
f_3	$3.3e^{-4}$	0.74	$1.1e^{-3}$	0.83
f_4	11.22	1.35	17.07	3.93
f_5	$8.4e^{-1}$	3.05	2.53	22.5
	ART(50)		ART(100)	
f_1	$4.96e^{-8}$	1.86	$1.12e^{-9}$	2.95
f_2	2.77	67.49	2.88	189.2
f_3	$2.6e^{-3}$	1.75	$3e^{-3}$	3.13
f_4	141.9	58.53	71.75	152.5
f_5	2.92	352.81	2.92	698.96

4. Conclusion

In this study, an investigation on the intelligent particle swarm optimization is carried out via modelling of the problem with a quadratic function. For this comparison, Kaczmarz's method (a.k.a ART) is implemented, and simulation / test results are compared as regards the problem dimension and the number of ART iterations. According to the obtained results, ART requires longer execution time compared to IPSO; however, as the problem dimension increases, the performance of ART also increases for smaller iterations. In lower dimensions, larger number of iterations will be required to obtain same or

Table 5. The cost value and execution time (in sec.) of the benchmark functions with 30 dimensions. Size of the population is 500; number of iterations is 50. ART(x) stands for KIPSO with x number of iterations.

Func.	IPSO		ART(3)	
	f(x)	Time	f(x)	Time
f_1	$8.37e^{-24}$	4.43	$7.15e^{-5}$	22.14
f_2	15.77	19.48	$9.5e^{-1}$	366
f_3	$4.9e^{-18}$	5.07	$1.8e^{-3}$	23.27
f_4	$1.36e^{4}$	24.11	29.61	203.92
f_5	2.53	15.24	0.79	862.68

better performance. In addition, it shold be noted that ART is an iterative method for which the implementation consists of nested loops for vector multiplication. Therefore, its structure is appropriate for parallelization. In the future studies, parallel-implementations of ART will be carried out on GPU boards in order to degrade the execution time for the $Ax = b$ solution.

5. References

[1] J. Kennedy and R. Eberhart " Particle swarm optimization", *Int. conf. on Neural Networks,*, pp: 1942-1948, 1995.

[2] G. Ciuprina, D. Ioan, and I. Munteanu " Use of intelligent particle swarm optimization in electromagnetics", *IEEE Trans. on Magnetics,*, vol. 38, no.2, pp: 1037-1040, 2002.

[3] A. Khare and S. Rangnekar " Particle swarm optimization: a review", *Applied Soft Computing,*, vol. 13, no.5, pp: 2997-3006, 2013.

[4] F.J. Solis, J.B. Wets " Minimization by random search techniques", *Mathematics of Operational Research,*, vol. 6, no.1, pp: 19-30, 1981.

[5] S. Kaczmarz " Approximate solution of systems of linear equations", *Int. J. Control (reprint of original paper in 1937),*, vol. 57, no.6, pp: 1269-1271, 1993.

[6] P.C. Parks "S. Kaczmarz (1895-1939)", *Int. J. Control,*, vol. 57, no.6, pp: 1263-1267, 1993.

[7] E.K.P. Chong, S.H. Zak, "An introduction to optimization", *John Wiley and Sons,* Chicago, USA, 2001.