ATLANTIS PRESS

Advances in Intelligent Systems Research (AISR), volume 145

2017 International Conference on Electronic Industry and Automation (EIA 2017)

Efficient Query for Historical Data in Evolutionary Algorithm

Jie TIAN*
College of Information Technology
Shandong Women's University
Jinan, 250300 China
E-mail: tianjie918@163.com
+* Corresponding author

Pan YAN
Department of Computer Science and Technology
Taiyuan University of Science and Technology
Taiyuan, 030024 China
E-mail: sxycyyx@163.com

Huiwen HUANG
College of Information Technology
Shandong Women's University
Jinan, 250300 China
E-mail: huiwen__huang@163.com

***Abstract*—For the time-consuming problem in calculating the fitness value, this paper proposes a hash bucket with precision mechanism for a quick query of the data in the neighborhood of a particle. In order to establish a balance between the calculation accuracy and utility, it uses the hash tables with precision mechanism to solve the problem in the storage and query of historical calculation data so that the neighborhood of a to-be-evaluated individual can be determined more accurately to reduce the error in estimating the fitness value. Moreover, it uses the typical reference functions to separately test the effectiveness and accuracy of the algorithms based on the values obtained in different dimensions. The test result proves that compared with the other algorithms described in this paper, our algorithm can provide a better solution in the context of the same number of times for fitness calculation.**

Keywords—fitness inheritance; fitness estimation; computationally expensive optimization

I. INTRODUCTION

Due to the insensitivity to the characteristics of the optimizing problem itself without any requirement on the continuity or differentiability of the objective function, the swarm intelligence algorithm that can be implemented easily has more advantages with its better global searching ability compared with the traditional single-point optimization algorithm. Therefore, it can provide a new idea and approach for the optimization design with great progress and successful application having been achieved in the field of engineering application. However, in practical engineering applications, it always takes several minutes, hours or a couple of days [1-4] each time to acquire the value of an objective function, such as the fluid mechanics [5-7] applied in wing design, the parameter settings in engine controller and the finite element analysis [18] adopted in mechanical structure design. Hence, for the swarm intelligence algorithm with the operating time decided by the computing times of a fitness function, it has gradually aroused the attention of many people about how to reduce the computing time of the objective function to reduce the time spent on optimization [8].

In order to solve the time-consuming problem in calculating the fitness value, fitness estimation methods are usually adopted in the practical calculation [9-10] to replace the objective function that consumes much time on computation. Currently, the common fitness estimation methods include the fitness inheritance and the application of surrogate model [11-17]. However, which method will perform better in fitness estimation? It can hardly be answered in face of the various problems and applications in practice. But we can make such an assumption that if the accuracy of fitness estimation can be secured, it is a priority to adopt such a method that requires less amount of calculation, such as fitness inheritance. However, during the fitness estimation, the main problem still lies in the determination on which individuals will be calculated actually and which individuals will be estimated based on fitness value without a big error [18-19]. In this paper, we start from a simple fitness inheritance approach adopted in estimation strategies to discuss this problem.

II. INTRODUCTION OF FITNESS ESTIMATION STRATEGY

Fitness estimation strategy means the use of some given information to make an estimation on the unknown fitness value so as to replace the actual calculation based on a complex objective function. It's an estimation strategy adopted to save the time spent on algorithm optimization by reducing the number of times in actual fitness calculation. So far, there are many research papers and achievements about the fitness estimation strategies, which can be roughly grouped into two types: the surrogate model-based estimation strategy and the fitness inheritance-based estimation strategy. Fitness inheritance is a common estimation strategy by estimating the data in the neighborhood of a target particle to acquire the fitness value. However, as it's quite time consuming in acquiring the neighborhood data; it becomes extremely important about

how to make a quick acquisition of the data in the neighborhood.

A. Fitness Estimation Strategy Based on Fitness Inheritance

The idea of fitness inheritance was originally proposed by Smith et al. (1995)[20]. It is an intuitive concept stating that the fitness of an individual is directly derived from that of its parents, and it discussed two types of inheritance: averaged inheritance and weighted inheritance. Salami and Hendtlass (2003)[21] introduced a "fast evolutionary algorithm". With this method, a fitness value and associated reliability value are assigned to each new individual, and the individual is only evaluated if its reliability value is below a threshold. As Ducheyne et al (2003) [22]mentioned, in many real-world applications of evolutionary algorithms, the characteristics of the parents cannot be used as a good indicator of their offspring. The reason could be that although the vast majority of the offspring have some connections with their parents, but for each individual offspring, it is not necessarily similar to its parents, but instead being similar with the individuals in vicinity. Cui and Sun (Cui et al. 2006; Sun et al. 2013) [23-24]proposed the fitness inheritance model based on this similarity or credibility, using the particle swarm optimization evolutionary formula as well as the relationship between the individual locations, respectively. Due to the similarities among individuals as a result of the differences in the individuals' positions, Kim (Kim and Cho 2001) [25]used clustering for grouping the similar characteristics of the individuals. In every generation, to divide the current population cluster, only the individuals that are the closest to the cluster center are computed, and in each cluster, other individuals inherited the fitness of the calculated individuals in a weighted manner according to the Euclidean distances among the individuals. Coello (Reyes-Sierra and Coello 2005)[26] used a fitness inheritance model that also relied on weighted inheritance according to the Euclidean distances among individuals in multi-objective particle swarm optimization (PSO), while reducing the time required for calculation, and improved the performance in solving the classical test functions. Gomide (2006)[27] utilized a fuzzy clustering method that is based on fuzzy adaptive clustering of c-means, in which the individuals will be similar to the others in the same cluster. In this method, the most representative individuals in each cluster were calculated using the Euclidean distance weighting method and the membership degree weighting method to fitness inheritance. Fonseca et al (2009) [28] introduced similarity-based surrogate model based upon the k-nearest neighbors method (KNN). For a new individual generated by the evolutionary operators as the clustering center, neighbors are selected according to a decreasing order of similarity to the new individual, and then the new individual inherits the fitness of its neighbors using similarity weights. The similarity proposed by Fonseca is also based on the Euclidean distance among individuals. Subsequently, Fonseca et al (2012) [29] applied three different types of adaptive value inheritance models, including the averaged inheritance model, weighted inheritance model, and parent inheritance model, to a real-coded genetic algorithm for comparison.

In the above researches, fitness inheritance can effectively reduce the number of times in fitness calculation. However, the adoption of fitness inheritance firstly requires the selection of a proper training sample. Therefore, how to effectively choose a proper training sample has become an urgent problem that must be solved immediately.

III. THE APPLICATION OF HASH TABLE TO QUERY THE NEIGHBORHOOD

In order to solve the time-consuming problem in calculating the fitness value based on a complex algorithm, one solution has been proposed with the adoption of fitness estimation strategy. In other words, the calculated data around the target particle are used to estimate the fitness of the target particle. In most cases, fitness estimation strategy requires the query for the neighborhood data of a particle, which always demands a one-to-one comparison between the target particle and the other particles in terms of the position and distance. However, it will consume lots of time and resources regarding the comparison of distance between the particles.

To solve the problem about the continuity and the wide range of the historical calculating data among the continuous problems in an evolutionary algorithm, we've designed a hash table with precision mechanism. The application of precision mechanism in the hash table can effectively group the data into different scopes according to the precision. For example, if the position of a two-dimensional particle is "1.23456, 2.34567", then it can be stored in the hash table with a precision of 2 as "1.23, 2.35", or "1.235, 2.346" in the hash table with a precision of 3. Also it's "1.2346, 2.3457" in the hash table with a precision of 4. With this method, although it's the same number, it can correspond to multiple Particle Positions in a hash table with different precision. Therefore, the position of a particle can be grouped into different scopes according to the different precision.

In a hash bucket with precision mechanism, the position of different particles (key value) might have the same numerical value after the processing by the precision mechanism. However, the fitness value corresponding to the position of each individual particle is different. In view of this, we change the way that the data are stored in the hash table. By doing this, not only the fitness result but also the complete information of the Particle Position can be preserved in the value. For example, if the position of Particle a is "3.123456" with a fitness of "0.888888" and a precision of 3, the key will be "3.123" with a value of "0.888888" after the precision processing. In the hash table with a precision of 3, the key will still be stored as "3.123", but the value will be stored as "3.123456; 0.888888". If the position of Particle b is "3.123123" with a fitness of "0.111111", the key of Particle b is also "3.123" after the processing in the hash table with a precision of 3, but the fitness will be different from that of Particle a. In this case, if the data should be put into the hash table directly, the new fitness value (the fitness of Particle b) will overlay the fitness value (the fitness of Particle a) stored previously, which is

adverse to the diversity of the particle data that have been stored. With our new value storage method, all of the fitness values will be preserved, although the key is the same. Therefore, with the same key in the hash table with a precision of 3, the value of Particle a and b can be saved as "3.123456; 0.888888, 3.123123; 0.111111". In this way, multiple particles will have the key of "3.123" after the precision processing, and there will be multiple neighborhood data around "3.123".

Particle fitness can be estimated through the following processes in a hash bucket with precision mechanism: Assume that we need to estimate the fitness of Particle P through the following four hash buckets, which separately include the hash bucket used to store the original data and the hash buckets with a precision of 2, 3 and 4 respectively. Firstly, make a query in the hash table used to store the initial precision. If the query succeeds, it means that this particle has been calculated and the data stored in the hash table is also the fitness of Particle P. However, if the query should fail, estimate the fitness of Particle P based on the data in the neighborhood of Particle P. Start the query from the hash bucket with a lower precision. In this case, search the Key of Particle P in the hash bucket with a precision of 2 after the precision processing. If the query should succeed, check whether the estimation condition has been met. If yes, make an estimation on the fitness of Particle P. But if the query should fail (the neighborhood data of Particle P haven't been calculated in the hash bucket with this precision) or the estimation condition fails to be met (insufficient particles are searched), search the hash bucket with greater precision. In this case, search the hash bucket with a precision of 3, where the scope for the neighborhood data of Particle P becomes larger. Although it would be prone to the detection of the calculated neighborhood data of Particle P, it also leads to a decrease in data accuracy. Should the query fail or the estimation condition fail to be met in the hash bucket with a precision of 3, extend the precision range of hash bucket and make a search in the hash bucket with a precision of 4. However, if the estimation on the fitness of Particle P should fail again, calculate the fitness value according to the fitness function and then save the calculation result into every hash bucket.

Key1:[1.213648,3.264432,6.721916] Value1:3.2
Key2:[1.212136,3.263258,6.722416] Value2:6.8
Key3:[1.214209,3.264481,6.721832] Value3:7.5
In the hash bucket with a precision of 2:
Key[1.21,3.26,6.72] Value: [Key1:3.2] [Key2:6.8] [Key3:7.5]
In the hash bucket with a precision of 3:
Key[1.214,3.264,6.722]
Value: [Key1:3.2] [Key3:7.5]
Key[1.212,3.263,6.722] Value: [Key2:6.8]
In the hash bucket with a precision of 4:
Key[1.2136,3.2644,6.7219] Value: [Key1:3.2]
Key[1.2121,3.2633,6.7224] Value: [Key2;6.8]
Key[1.2142,3.2645,6.7218] Value: [Key3;7.5]

IV. ALGORITHM DESCRIPTION

For the calculation of fitness value, firstly take the current position of Particle Xi as the key and make a query in the hash table with the initial precision. If the Key can be found in the hash table, it means that Particle X_i has been calculated and the fitness value $F(X_i)$ has been stored in the table, where the corresponding value is also the target fitness value $F(X_i)$ that will be calculated. However, if the key is not available in the table, then calculate the distance D_{ij} between the existing Particle X_i and the Particle X_j that has been stored in the hash table with a precision mechanism. Assume that there are m ranges for the neighbor radius R of a particle. If the distance D_{ij} between two particles meets $D_{ij}<R$, then choose the fitness of n particles with the best fitness values among all particles that meet the condition to estimate the fitness of the existing Particle X_j according to a certain calculation method (the average or weighted calculation). However, in the case that there are less than n particles that meet the above condition, search the particles in the neighborhood with a larger radius. If the fitness that meets the condition still cannot be found within a preset radius of m ranges, calculate the actual fitness value of this particle and store it in the various hash tables with different precision mechanisms. Mark every fitness value if it has been obtained through the estimation strategies. If the final optimal value is acquired through the estimation, re-calculate the fitness value of this particle to get a more accurate result. The complete algorithm description is provided as below:

Algorithm 1:
Step1:The initialization of multiple hash tables (with one used to save the data with complete precision and the others separately representing the hash tables with different precision);
Step2:The initialization of population clusters and the calculation of every individual fitness;
Step3:After the processing of an individual position (namely Key value) based on different precision, separately save the Key and the corresponding fitness value into the relevant hash tables;
Step4:Make an evolution operation on the individuals to generate the offspring population according to a certain evolution rule;
Step5:Calculate the fitness of an individual in the population cluster according to the Algorithm 2;
Step6:Judge whether the algorithm termination condition has been met. If yes, the algorithm will come to an end. Otherwise, return to Step4.

Algorithm 2:
Step1:Firstly, query the Key in the hash table with complete precision;
Step2:If the Key already exists in the table, it indicates that the fitness of this individual has been calculated. Then extract the Value corresponding to this Key and take it as the fitness of this individual. However, if the Key is not available in the table, query the data in the neighborhood and estimate the target fitness according to the neighborhood data;

Step3: If the neighborhood data do not conform to the preset fitness estimation condition, the target particle is calculated according to the fitness function. After the processing based on different precision, the calculated fitness results are separately saved into different tables.

V. Simulation Experiment and Analysis

In order to verify the validity of the neighborhood query strategy through the hash buckets with precision mechanism, this paper adopts four commonly-used test functions (more details on Table VI), including two unimodal functions, which are the Sum of Different Power function (F1) and the Rosenbrock function (F2) and two multimodal functions, which are the Rastrigin function (F3) and the Griewank function (F4) to test the algorithm. In the simulation experiment, we've applied the mechanism of hash bucket into the pso algorithm (HTPSO) with the adoption of fitness-based neighborhood estimation strategy. Besides this, we've also adopted the hash table algorithm (SHPSO) with precision mechanism and the standard particle swarm optimization (PSO) algorithm to optimize the above four test functions and make a comparison with the optimization result obtained through the HTPSO algorithm.

The parameter settings of the algorithm are provided as below: The number of particles in the population is 300 with the cognitive coefficient c1=0.9 and the social coefficient c2=1.2 in 10 dimensions. Run the algorithm for 20 times with the number of iterations set to be 150. Table I shows the optimization result of the four test functions through the above three algorithms and the fewer computing times by the particle swarm optimization algorithm with the application of hash table than that through the standard particle swarm optimization algorithm in the context of similar computing results having been obtained. With an approximate number of times in the estimation, the HTPSO algorithm can provide a better fitness value than the SHPSO algorithm. It proves that the HTPSO algorithm can effectively reduce the number of times for fitness calculation when a similar fitness result is guaranteed compared with the original algorithm. In order to verify that HTPSO is an efficient neighborhood estimation strategy, we compare it with a common neighborhood estimation strategy (hereafter referred to as ESPSO), which also means the adoption of the fitness values of 10 particles that are nearest to the target particle to make an estimation. However, as the number of times for the estimation made based on the HTPSO is unknown in advance, then set an equal number of times for the estimation based on the ESPSO. Under the same experimental conditions, compare the mean fitness values obtained through both methods to verify the validity of the strategy proposed in this paper.

The experimental operating conditions are set as below: apply both of the fitness estimation strategies into the standard PSO algorithm with the number of particles in the population set to be 30, the cognitive coefficient c1=0.9 and the social coefficient c2=1.2 in 10 dimensions. Run the algorithm for 30 times with the number of iterations set to be 100.

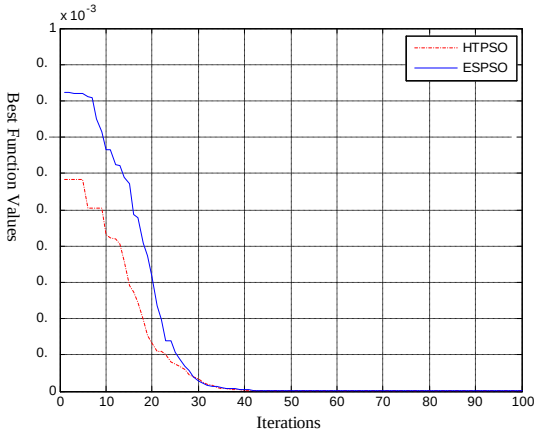

Figure 1. The precision of hashtables are 3, 4, 5

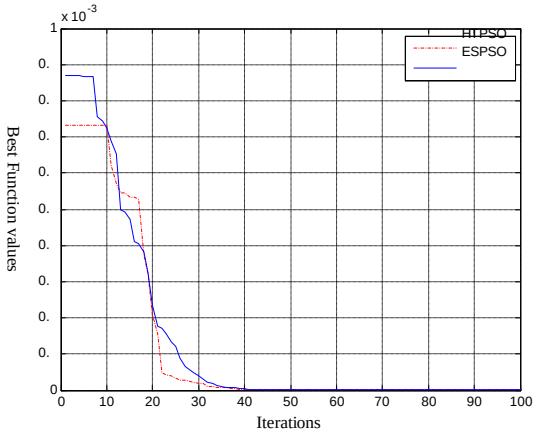

Figure 2. The precision of hashtables are 2, 3, 4

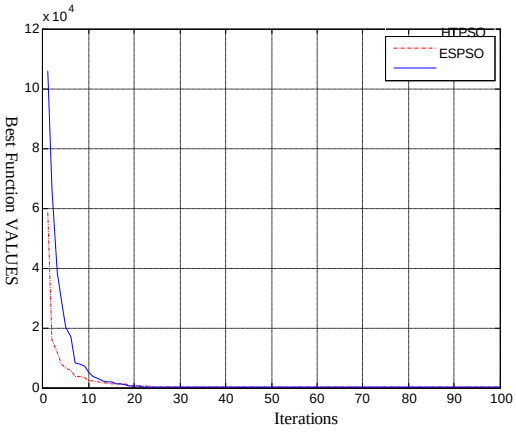

Figure 3. The precision of hashtables are 1, 2, 3

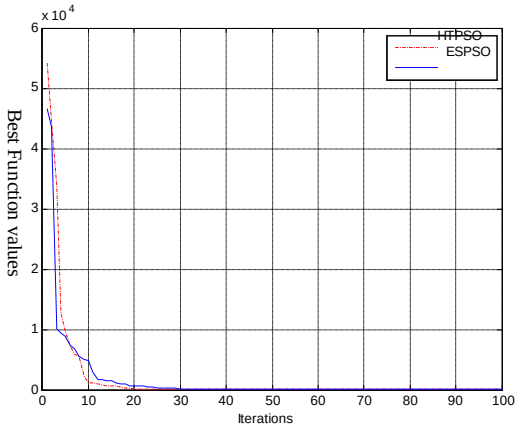

Figure 4. The precision of hashtables are 0, 1, 2

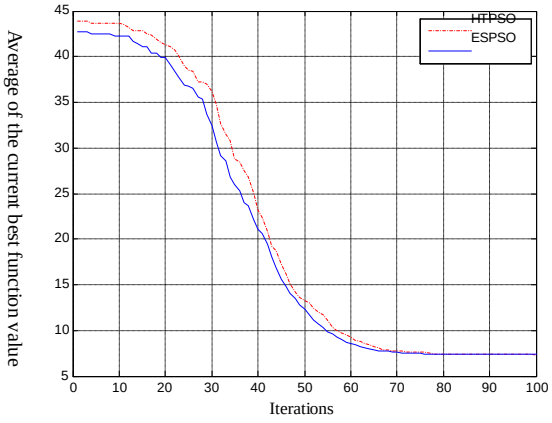

Figure 5. The precision of hashtables are 2, 3, 4

In the tables from Table II to Table V, both of the fitness estimation strategies based on the neighborhood data have been adopted to separately calculate the fitness results. The figures from Fig.1to Fig.8 show the trend of fitness results obtained in different neighborhood radius according to both of the estimation strategies. Both of the tables and figures reveal that the mean fitness values are quite similar through both of the strategies with little difference in the number of times for fitness calculation. Therefore, it proves that this new neighborhood data-based fitness estimation strategy works quite effectively.

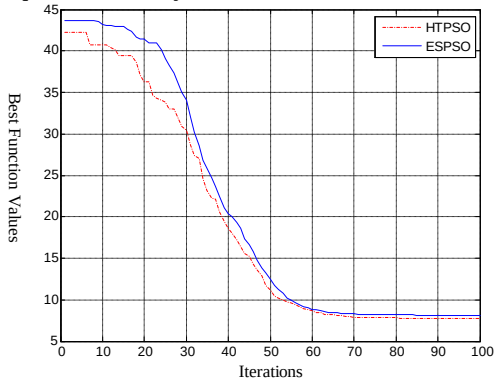

Figure 6. The precision of hashtables are 1, 2, 3

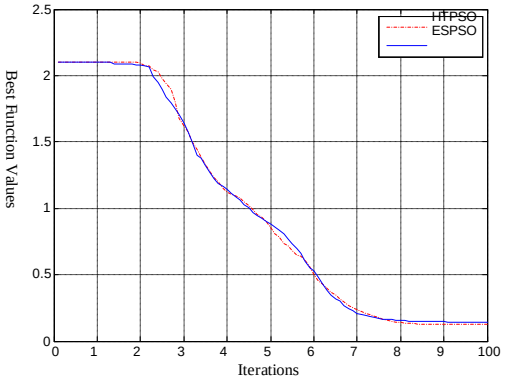

Figure 7. The precision of hashtables are 1,2,3

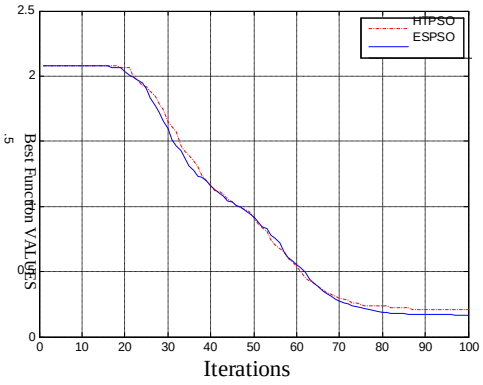

Figure 8. The precision of hashtables are 0,1,2

VI. CONCLUSION AND FUTURE WORKS

Our neighborhood data-based fitness estimation strategy is a feasible approach to solve the time-consuming problem in calculating fitness. In most cases, the common estimation strategies always require the calculation on the neighborhood data of a particle, which usually requires a one-to-one comparison between the target particle and the other particles in terms of the position and distance. In this paper, we utilize the special characteristics of hash table with fast storage and query to reduce the time spent on comparison during the query of neighborhood data, which to some extent improves the algorithmic efficiency.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China (Grant No. 61472269 and 61403272), College of Information Technology, Shandong Women's University, China.

REFERENCES

[1] Lim, D., et al., Generalizing surrogate-assisted evolutionary computation. Evolutionary Computation, IEEE Transactions on, 2010. 14(3): p. 329-355.

[2] Lim, D., et al. Trusted evolutionary algorithm. in Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. 2006. IEEE.

[3] Ong, Y.S., et al., Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems, in Knowledge Incorporation in Evolutionary Computation. 2005, Springer. p. 307-331.

[4] Ong, Y.S., P.B. Nair, and A.J. Keane, Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA journal, 2003. 41(4): p. 687-696.

[5] Le, M.N., et al., Evolution by adapting surrogates. Evolutionary computation, 2013. 21(2): p. 313-340.

[6] Won, K.S. and T. Ray, A framework for design optimization using surrogates. Engineering optimization, 2005. 37(7): p. 685-703.

[7] Karakasis, M.K., A.P. Giotis, and K.C. Giannakoglou, Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. International Journal for Numerical Methods in Fluids, 2003. 43(10-11): p. 1149-1166.

[8] Jin, Y., A comprehensive survey of fitness approximation in evolutionary computation. Soft computing, 2005. 9(1): p. 3-12.

[9] Shi, L. and K. Rasheed, A survey of fitness approximation methods applied in evolutionary algorithms, in Computational intelligence in expensive optimization problems. 2010, Springer. p. 3-28.

[10] Jin, Y., Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation, 2011. 1(2): p. 61-70.

[11] Sanchez, E., S. Pintos, and N.V. Queipo, Toward an optimal ensemble of kernel-based approximations with engineering applications. Structural and multidisciplinary optimization, 2008. 36(3): p. 247-261.

[12] Lim, D., et al. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. in Proceedings of the 9th annual conference on Genetic and evolutionary computation. 2007. ACM.

[13] Samad, A., et al., Multiple surrogate modeling for axial compressor blade shape optimization. Journal of Propulsion and Power, 2008. 24(2): p. 301-310.

[14] Sun, C., et al., A two-layer surrogate-assisted particle swarm optimization algorithm. Soft Computing, 2015. 19(6): p. 1461-1475.

[15] Paenke, I., J. Branke, and Y. Jin, Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. Evolutionary Computation, IEEE Transactions on, 2006. 10(4): p. 405-420.

[16] Liu, B., Q. Zhang, and G.G. Gielen, A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. Evolutionary Computation, IEEE Transactions on, 2014. 18(2): p. 180-192.

[17] Ferrari, S. and R.F. Stengel, Smooth function approximation using neural networks. Neural Networks, IEEE Transactions on, 2005. 16(1): p. 24-38.

[18] Jin, Y., M. Olhofer, and B. Sendhoff, A framework for evolutionary optimization with approximate fitness functions. Evolutionary Computation, IEEE Transactions on, 2002. 6(5): p. 481-494.

[19] Lu, X., K. Tang, and X. Yao. Classification-assisted differential evolution for computationally expensive problems. in Evolutionary Computation (CEC), 2011 IEEE Congress on. 2011. IEEE.

[20] Smith, R.E., B.A. Dike, and S. Stegmann. Fitness inheritance in genetic algorithms. in Proceedings of the 1995 ACM symposium on Applied computing. 1995. ACM.

[21] Salami, M. and T. Hendtlass, A fast evaluation strategy for evolutionary algorithms. Applied Soft Computing, 2003. 2(3): p. 156-173.

[22] Ducheyne, E., B. De Baets, and R. De Wulf. Is fitness inheritance useful for real-world applications? in Evolutionary Multi-Criterion Optimization. 2003. Springer.

[23] Sun, C., et al., A new fitness estimation strategy for particle swarm optimization. Information Sciences, 2013. 221: p. 355-370.

[24] Cui, Z., J. Zeng, and G. Sun, A fast particle swarm optimization. International Journal of Innovative Computing, Information and Control, 2006. 2(6): p. 1365-1380.

[25] Kim, H.-S. and S.-B. Cho. An efficient genetic algorithm with less fitness evaluation by clustering. in Evolutionary Computation, 2001. Proceedings of the 2001 Congress on. 2001. IEEE.

[26] Reyes-Sierra, M. and C.A.C. Coello. A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. in Evolutionary Computation, 2005. The 2005 IEEE Congress on. 2005. IEEE.

[27] Gomide, F. Fuzzy clustering in fitness estimation models for genetic algorithms and applications. in Fuzzy Systems, 2006 IEEE International Conference on. 2006. IEEE.

[28] Fonseca, L., H. Barbosa, and A. Lemonge, A similarity-based surrogate model for enhanced performance in genetic algorithms. Opsearch, 2009. 46(1): p. 89-107.

[29] Fonseca, L.G., A.C. Lemonge, and H.J. Barbosa. A study on fitness inheritance for enhanced efficiency in real-coded genetic algorithms. in Evolutionary Computation (CEC), 2012 IEEE Congress on. 2012. IEEE.

Appendix

TABLE I. Comparative Experimental Results of PSO, SHPSO and HTPSO on the Four Test Functions

function	Algorithm	Precision	Mean	Variance	Best	Worst	fitness estimation	fitness calculation
F1	PSO	--	6.3650e-18	7.8913e-18	2.4582e-21	1.2374e-17	--	45000
	SHPSO	4	1.7603e-18	4.2541e-18	3.7971e-22	1.9772e-17	9855	35145
	HTPSO	3,4,5	2.4635e-19	8.9203e-19	4.0814e-22	2.8309e-17	9739	35261
F2	PSO	--	2.0400e-18	7.6312e-18	1.7047e-21	3.2828e-17	--	45000
	SHPSO	4	9.4785e-18	1.0981e-18	1.9198e-21	3.7831e-17	1709	43291
	HTPSO	3,4,5	2.6030e-18	6.2789e-18	8.4212e-21	2.4169e-17	1616	43384
F3	PSO	--	0	0	0	0	--	45000
	SHPSO	9	0	0	0	0	1216.2	43784
	HTPSO	8,9,10	0	0	0	0	3375	41625
F4	PSO	--	0.0843	0.0485	0.0246	0.1550	--	45000
	SHPSO	2	0.0790	0.0380	0.0173	0.1894	11387	33319
	HTPSO	1,2,3	0.0671	0.0338	0.0099	0.1353	12334	32660

TABLE II. COMPARATIVE EXPERIMENTAL RESULTS OF ESPSO AND HTPSO ON F1

Algorithm	Precision of hash	Mean	Variance	Best fitness	fitness estim	fitness calculation
ESPSO	–	1.1543e-09	2.8124e-09	8.3649e-11	347	2653
HTPSO	3,4,5	2.2505e-09	4.7060e-09	4.9788e-13	362.6	2637.4
ESPO	–	31394e-09	8.1047e-09	3.4915e-12	864.3	213.7
HTPSO	2,3,4	3.2750e-09	5.496e-09	7.6299e-12	867.9	2132.1

TABLE III. COMPARATIVE EXPERIMENTAL RESULTS OF ESPSO AND HTPSO ON F2

Algorithm	Precision of hash bucke	Mean	Variance	Best fitness	fitness estimation	fitness calculation
ESPSO	–	3.2404e-04	0.0014		273.2	2726.8
HTPSO	1,2,3	8.0316e-05	2.1237e-04	6.9649e-09	244.3	275 7
ESPSO	–	0.0025	0.0126		900.4	2400.6
HTPSO	0,1,2	0.0039	0.007	2.8887e-07	605	2395

TABLE IV. COMPARATIVE EXPERIMENTAL RESULTS OF ESPSO AND HTPSO ON F3

Algorithm	Precision of hash bucket	Mean	Variance	Best fitness	Times of fitness estimation	Actual times of fitness calculation
ESPSO		7.3493	3.8184	0.0951	600	2400
HTPSO	2,3,4	7.3391	3.2771	0.0119	626.2	2373.8
ESPSO		8.1455	4.2946	0.9950	1196.3	1803.7
HTPSO	1,2,3	7.6935	3.4109	0.6030	1159.9	1840.1

TABLE V. COMPARATIVE EXPERIMENTAL RESULTS OF ESPSO AND HTPSO ON F4

Algorithm	Precision of hash	Mean	Variance	Best fitness	Times of fitness estimation	Actual times of fitness calculati
ESPSO	–	0.1429	0.0947		301	2699
HTPSO	1,2,3	0.1264	0.1092	0.0273	432.6	2567.4
ESPSO	–	0.1672	0.0749		870.4	2129.6
HTPSO	0,1,2	0.2112	0.1107	0.1097	831.8	2168.2

TABLE VI. TEST FUNCTIONS USED IN THE EXPERIMENTS

Function number	name	Function	Search range
F1	Sum of Different Power	$f_1(x) = \sum_{i=1}^{30} \lvert x_i \rvert^{i+1},$	[-1,1]
F2	Rosenbrock	$f_2(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2].$	[-30,30]
F3	Rastrigin	$f_3(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10],$	[-5.12,5.12]
F4	Griewank	$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$	[-600,600]