Algorithmic Operations Research

The AID Method for Global Optimization

Mahamed G.H. Omran and Fred Glover

Volume 6, Number 1, Spring 2011

URI: https://id.erudit.org/iderudit/aor6_1art02

See table of contents

Publisher(s)

Preeminent Academic Facets Inc.

ISSN

1718-3235 (digital)

Explore this journal

Cite this article

Omran, M. G. & Glover, F. (2011). The AID Method for Global Optimization. *Algorithmic Operations Research*, *6*(1), 20–28.

Article abstract

An Alternating Intensification/Diversification (AID) method is proposed to tackle global optimization problems, focusing here on global function minimization over continuous variables. Our method is a local search procedure that is particularly easy to implement, and can readily be embedded as a supporting strategy within more sophisticated methods that make use of population-based designs. We perform computational tests comparing the AID method to 20 other algorithms, many of them representing a similar or higher level of sophistication, on a total of 28 benchmark functions. The results show that the new approach generally obtains good quality solutions for unconstrained global optimization problems, suggesting the utility of its underlying notions and the potential value of exploiting its multiple avenues for generalization.

Algorithmic Operations Research Vol.6 (2011) 20–28

The AID Method for Global Optimization

Mahamed G. H. Omran [a] and Fred Glover [b]

[a]Gulf University for Science and Technology, Kuwait, Kuwait.
[b]OptTek Systems, Inc., 2241 17th Street, Boulder, CO 80302 USA.

Abstract

An Alternating Intensification/Diversification (AID) method is proposed to tackle global optimization problems, focusing here on global function minimization over continuous variables. Our method is a local search procedure that is particularly easy to implement, and can readily be embedded as a supporting strategy within more sophisticated methods that make use of population-based designs. We perform computational tests comparing the AID method to 20 other algorithms, many of them representing a similar or higher level of sophistication, on a total of 28 benchmark functions. The results show that the new approach generally obtains good quality solutions for unconstrained global optimization problems, suggesting the utility of its underlying notions and the potential value of exploiting its multiple avenues for generalization.

Key words: Intensification/Diversification, global optimization, local search, scatter search, path relinking

1. Introduction

Consider the nonlinear optimization problem

$$(P) Minimize f(x) : x \in X = \{x : U \geq x \geq L\} \tag{1}$$

where f(.) is a nonlinear function, x = ($x_1, \ldots, x_n$) is a vector of real-valued (continuous) variables, and L and U are n-vectors of constants whose entries are assumed finite.

Our method for (P) is an easy-to-implement approach from the local search class, which does not make use of solution pools as employed in evolutionary population-based search. Because of its straightforward character, and the fact that it requires (almost) no tuning, it can readily be embedded as a subroutine within evolutionary procedures. In fact, our method makes use of procedures for combining pairs of solutions using strategies derived from scatter search and path relinking, often considered as methods belonging to the evolutionary class, but makes no recourse to the population-handling mechanisms embodied in these approaches.

Various local search types of procedures previously applied to function optimization include, for example, Nelder and Mead simplex search [16], tabu search [5,6], iterated local search [22], simultaneous perturbation stochastic approximation [20], reactive affine shaker [2], among numerous others. Well-known methods belonging to the evolutionary category that have been applied to function optimization include genetic algorithms [9], evolution strategies [1], particle swarm optimization [15], differential evolution [21], and scatter search [7], to cite just a small sampling. A unidimensional approach that shows promise for solving some classes of high-dimensional problems by successive examination of single coordinate dimensions is given in [4].

Our present study performs computational tests on 28 benchmark problems comparing the AID method to 20 other algorithms for function optimization (including algorithms reported in the well-known IEEE CEC2005 competition). The outcomes disclose that AID performs well compared with these alternative methods. The outcomes invite the speculation that procedures based on generalizing the AID method or integrating it within a more advanced method that exploits population pools may prove more promising yet.

The reminder of the paper is organized as follows: Section 2 introduces the fundamental ideas of AID, accompanied by pseudo-code. Methods used for computational comparisons and results of the experiments are presented in Section 3, and Section 4 summarizes our conclusions.

Email: Mahamed G. H. Omran [omran.m@gust.edu.kw], Fred Glover [glover@opttek.com].

2. The Alternating Intensification/Diversification (AID) Method

Our proposed optimizer starts by generating a series of *initiating solutions*. The best initiating solution is subjected to an intensification (local search) method to produce a better initiating solution. This solution is submitted to a simple diversification step that complements the initiating solution relative to the bounds on the variables. The resulting pair of solutions is then processed by a method SSPR_Combine that consists of mechanisms for combining solutions derived from scatter search (SS) and path relinking (PR). Finally, the best solution obtained from SSPR_Combine is subjected to an intensification method to produce the solution that becomes the initiating solution for the next iteration, and the entire process is repeated until a stopping criterion is met. The framework of AID is shown in Alg. 1.

Algorithm 1 A general framework for the AID method

```
Generate an initiating solution x. (Each initiating
solution is a candidate for the best solution,
subsequently denoted x*.)
Improve x using a local search.
repeat
    Generate the complement y of x to create the
    diverse solution pair (x,y).
    Create z = SSPR_Combine(x,y)
    if $f(z) < f(x)$ then
        x = z {assuming a minimization problem}
    end if
    Improve x using a local search to yield a new
    initiating solution.
until a stopping criterion is met
```

The framework of Alg. 1 needs to be tailored for the problem at hand. A variety of methods can be used to produce a solution that takes the role of the "complement" solution y, depending on the problem context. For example, a diversifying mechanism that can produce such a solution y in the setting of the quadratic assignment problem, and which can be used in many other problem settings, appears in Kelly [14]. Similarly, alternative combination methods can be used in the role we have given to SSPR_Combine [1].

[1] Normally a diversification process would be designed to produce new solutions that are diverse relative to a collection of given solutions, but here we simplify the notion of diversification to consider a new solution that is "diverse" in the sense of being suitably different from a single current solution.

We give a detailed description of the version of AID we have produced for continuous function optimization in the following subsections.

3. Generating the first initiating solution

An initiating solution to launch the method can be generated randomly or by choosing a good starting point in a more strategic manner. In this paper, P solutions are randomly generated and the best one (defined in terms of f) is used as the first initiating solution.

Improvement method

The Enhanced Unidimensional Search (EUS) algorithm [4] is used in the intensification process. EUS is an enhanced variant of the Classical Unidimensional Search (CUS) for solving (P), and hence operates as a line search algorithm that runs dimension by dimension. The method is easy to implement and good in handling high dimensional problems. In this paper we use interval scanning to find the best value for the *ratio* parameter of EUS for each problem.

Generating the complement solution

We generate the complement solution by defining complementation relative to the bounds on the variables embodied in the vectors L and U. The rationale behind using this form of the complement is that it may be conceived as the "opposite" of the solution from which it derives (adopting a view often used in the context of binary optimization).

The inequality $L \leq x \leq U(i.e., x_i \in [L_i, U_i], i = 1, 2, \ldots, n)$, gives rise to the complement y of x given by

$$y_i = L_i + U_i - x_i \tag{2}$$

It is important to note in applying this definition that L and U may give exceedingly "loose" bounds on the problem variables, and hence our use of the complement could be improved by a process that successively revises L and U based on solution history. We do not undertake to do this in our present study.

Generating the combined solution z

Given two solutions x and y, the combination method SSPR_Combine(x,y) used to generate the solution z is inspired by the scatter search and path relinking template of [7] . Two solutions are generated, the first one based on scatter search and the second based on path relinking. The scatter search combination is given by:

C1: $z = x + 0.5r(y - x)$ where r is a random number in the range (0,1).

This solution was used in the study of [8] and adds some randomness to our method. The path relinking approach we employ may be described as follows.

C2: A randomized version of path relinking proposed by [24] is used as a diversification strategy that exhibits elements of intensification. To generate a path, the path relinking operation progressively introduces elements (here, values of variables) of x into y, where x and y are renamed if need be so that $f(x) \leq f(y)$. One element of x is randomly chosen at each relinking step and produces the current z by replacing the corresponding element of y, generating a new solution z that lies "between" x and y. This process is repeated until z = x. The path is then extended by introducing an element randomly into z. The best solution found over the whole path is returned as the result of the path relinking process.

Path relinking processes typically exclude solutions from consideration as candidates for a "best solution" on the path if they are less than a specified minimum distance from x and y. Although such a refinement has proved useful in a variety of settings (see, e.g., the survey of [17]), we did not attempt to make use of it here.

The two solutions produced by C1 and C2 are then compared and the best (in terms of f) is returned as the result of combining x and y.

It can be seen that on each AID iteration there is a balance between diversification (via the complement and path relinking) and intensification (via path relinking and EUS). Furthermore, AID requires no parameter tuning except for selecting a stopping criterion and choosing the number of iterations allotted as a limit for the improving method. This represents an important advantage for AID. A description of the version of AID implemented in this paper is shown in Alg. II.

Algorithm 2 Pseudo code of AID for continuous optimization

```
Generate P solutions randomly
Evaluate the solutions in P
Set the first initiating solution, x, to be the best
solution in P according to f
Apply an intensification process
$x = EUS(x)$
repeat
    Generate the complement
    $y = L + U - x$
    Generate two solutions
    $z_1 = x + 0.5r(y - x)$ where $r$ is a random number
    in the range $(0, 1)$
    $z_2 = path\_relinking(x, y)$
    Set the ultimate solution z to be the best of $z_1$ and
    $z_2$; i.e.,
    $z =$ arg max $(f(z_1), f(z_2))$.
    if $f(z) < f(x)$ then
        $x = z$ {assuming a minimization problem}
    endif
    Apply an intensification process
    $x = EUS(x)$
until a stopping criterion is met
```

Note that it is possible for x to go through this process without changing. We did not incorporate a device to exclude such a possibility. It would intuitively be beneficial to incorporate additional controls within Algorithm II to insure that x changes before applying the intensification process.

4. Experimental Results

In this section we first compare the performance of AID with that of the simultaneous perturbation stochastic approximation (SPSA) method [19] and the reactive affine shaker (RASH) method [2] when applied to 17 test problems with n ranging from 2 to 30. The initialization ranges for each problem are the same as in their " original sources":

http://www.cyberiad.net/realbench.html
http://solon.cma.univie.ac.at//glopt.html

For AID, the number of iterations assigned to EUS is set to 2000. For SPSA, $A = 0$, $\alpha = 1$, $\gamma = 1/6$, $a = 0.4$ and $c = 1$. For RASH, ρ_e = 1.2 and ρ_r = 0.8. The results reported in this section are averages and standard deviations over 30 simulations. Each simulation was allowed to run for 100,000 evaluations of the objective function if n $<$ 10 and for 500,000 function evaluations otherwise. The statistically significant best solutions have been shown in bold (using the non-parametric statistical test called Wilcoxon's rank sum test for independent samples [23] with $\alpha = 0.05$).

All the tests are run on an Apple MacBook computer with Intel Core Due 2 processor running at 2.0 GHz with 2GB of RAM. Mac OS X 10.5.6 is the operating system used. All programs are implemented using MATLAB version 7.6.0.324 (R2008a) environment.

Table I summarizes the results obtained by applying SPSA, RASH and AID to the benchmark functions. The table reports the average gap, Avg. GAP, where the optimality gap for a given solution x and the best solution x* is defined as:

$$GAP = |f(x) - f(x*)|. \qquad (3)$$

The results show that AID generally outperformed both SPSA and RASH on all the benchmark functions except for the Quartic function where SPSA performed better (the expected reason is the presence of noise).

Table II shows the count (as a percentage) of how often a given type of combination employed by our method (SS from C1 or PR from C2) finds the best solution for representative functions. The results show that PR is the combination method that often yields the best solution. Note that these outcomes do not truly disclose the importance of a given combination approach, since a combination that did not find the best solution still could have been essential for giving a solution at some stage that allowed another combination to find a best solution.

We next compared AID against seven more advanced algorithms, starting with the Dialectic Search method and Simulated Annealing (in two variants) as reported in [13].

In order to compare AID with these Dialectic search and SA methods, AID was run 250 times and the average fitness and average number of function evaluations were recorded in Table III for the three functions with dimensions 20 and 50. Table III shows that AID reached better solutions faster than both Dialectic search and SA in each of the test cases.

In addition, we tested the AID algorithm on the six function suite in [11] and compared the results to those of the five popular metaheuristics reported in detail in that paper: a Simple Genetic Algorithm (SGA) [12], Evolutionary Programming (EP) [3], Evolution Strategies (ES) [18], Particle Swarm Optimization (PSO) [15] and the more recently-proposed Group Search Optimizer (GSO) [11]. These algorithms are population-based, stochastic search approaches. Table IV summarizes AID's results (over 50 independent runs as in [11]). Results in the table for SGA, EP, ES, PSO and GSO are reproduced from [11]. AID, GSO, SGA and CPSO use 150,000 FEs, EP uses 300,000 FEs and ES uses 600,000 FEs. The results show that AID returned the best objective values on five functions and the fourth best objective value on one function (the Schwefel's Problem 2.26 function).

In our concluding set of experiments, we compare AID with the 11 methods reported on the CEC 2005 competition. Nine functions have been chosen from the CEC 2005 benchmark set. Functions F1, F2 and F6 are unimodal functions, F4 is a noisy unimodal function, F9, F10 and F12 are multimodal functions and F8 and F13 are never-solved multimodal functions. All methods use 100,000 function evaluations and are run for 25 independent runs as suggested by [10]. The effectiveness of the methods is measured based on the number of successful runs (a run is successful if GAP $\leq 1\,\mathrm{e}{-8}$). On the other hand, efficiency is measured in terms of the number of function evaluations (FEs) as defined by

$$\text{FEs} = \text{mean}(\textit{fevals}) \times (25 \div \#\text{successful runs}). \qquad (4)$$

where *fevals* includes only successful runs.

Table V shows the results of applying AID to the CEC functions. From the effectiveness point of view, AID ranks joint first on 3 functions (F1, F2 and F9). AID ranks 5^{th} on one (F12), it ranks 6^{th} on another (F6) and AID ranks 8^{th} on F4. On three functions (F8, F10 and F13) AID could not find the solution within the desired precision. However, two of these functions (i.e. F8 and F13) are never solved problems. If we compare AID with the 11 methods when applied to F8 and F13 based on the Avg. GAP value, we can see that AID ranks joints first on F8 and 3^{rd} on F13.

If we consider now the efficiency, AID was the fastest approach on F9, 2 nd fastest method on F12, 4^{th} fastest algorithm on F1, F2, F4 and F6.

Thus, in general, despite its simplicity, AID managed to perform very well compared to 11 well-known (generally more complicated and population-based) methods when applied to this set of difficult problems.

Table I

Function	*SPSA*	*RASH*	*AID*
Goldstein and Price (n = 2)	1.056917e+00 (5.384539e-01)	6.649372e-01 (1.491853e+00)	**3.557155e-14 (1.577324e-14)**
Shubert (n = 2)	1.495288e+01 (1.168565e+01)	7.549889e+00 (8.758797e+00)	**8.831024e-06 (2.955870e-14)**
Branin (n = 2)	1.121405e-03 (3.581681e-03)	4.550881e-02 (6.020034e-02)	**3.577297e-07 (0.000000e+00)**
Easom (n = 2)	7.333304e-01 (4.497734e-01)	7.438125e-01 (3.717542e-01)	**3.666369e-01 (4.900928e-01)**
Six Hump Camel back (n = 2)	2.965881e-02 (8.264845e-02)	3.954712e-02 (7.322557e-02)	**4.534899e-07 (1.580884e-16)**
Hartmann(3,4) (n = 3)	4.006554e-02 (3.732602e-02)	3.016177e-02 (2.849831e-02)	**2.126673e-07 (1.126690e-16)**
Shekel (n = 4)	**4.248490e+00 (3.441910e+00)**	7.864224e+00 (9.883646e-01)	**2.904562e+00 (3.468383e+00)**
Michalewicz (n = 10)	3.549294e+00 (6.828568e-01)	5.443982e+00 (2.901188e-01)	**5.861889e-02 (5.387001e-02)**
Rosenbrock(n = 30)	2.101477e+01 (1.065353e+01)	3.677846e+03 (8.032824e+02)	**1.239540e-09 (1.688920e-09)**
Levy (n = 30)	4.407136e+01 (8.614072e+00)	1.418438e+02 (1.653082e+01)	**1.499760e-32 (1.113480e-47)**
Rastrigin (n =30)	4.179089e+01 (1.152411e+01)	3.631780e+02 (2.361351e+01)	**3.429553e-13 (1.330866e-13)**
Normalized Schwefel (n =30)	2.951031e+02 (1.099462e+01)	9.562784e+01 (4.349717e+01)	**5.362229e-13 (9.163700e-14)**
Griewank (n =30)	4.585298e+02 (4.629519e+01)	4.510160e+02 (4.640301e+01)	**0(0)**
Salomon (n =30)	2.266324e+01 (1.058797e+00)	2.224619e+01 (1.530951e+00)	**3.329112e-03 (1.823429e-02)**
Step (n =30)	7.055667e+02 (2.166790e+02)	4.757237e+04 (5.694358e+03)	**0(0)**
Quartic function (n =30)	**7.508891e-04 (3.317812e-04)**	6.310150e+01 (1.393730e+01)	6.115969e-02 (5.304495e-02)
Sphere (n =30)	4.569992e-02 (2.029247e-02)	4.947104e+04 (4.268285e+03)	**3.557155e-14 (1.577324e-14)**

Mean and standard deviation (SD) of the function optimization results.

Table II

Function	*C1*	*C2*
Easom (n = 2)	13.3333	86.6667
Six Hump Camel back (n = 2)	0	100
Hartmann(3,4) (n = 3)	0	100
Shekel (n = 4)	0.4167	99.5833
Michalewicz (n = 10)	0	100
Rosenbrock(n = 30)	0	100
Salomon (n =30)	23.1871	76.8129
Quartic function (n =30)	31.6667	68.3333

This table shows the count of how often a given type of combination finds the best solution for AID when representative problems.

Table III

	AID	*Dialectic*	*SA-0.98*	*SA-0.99*
Function	Value FEs	Value FEs	Value FEs	Value FEs
Rastrigin (n=20)	$< 10^{-12}$ **10.745K**	$< 10^{-3}$ 208K	24.4 3.4M	22.4 6.8M
Rastrigin(n=50)	$< 10^{-12}$ **11.858K**	$< 10^{-3}$ 818K	87.3 8.3M	86.8 9.9M
DeJong (n=20)	= 0 **6K**	$< 10^{-3}$ 848	$< 10^{-3}$ 946	$< 10^{-3}$ 946
DeJong (n=50)	= 0 **6K**	$< 10^{-3}$ 3.7K	$< 10^{-3}$ 2.5K	$< 10^{-3}$ 2.5K
Alpine (n=20)	$< 10^{-14}$ **6K**	$< 10^{-3}$ 86K	$< 10^{-3}$ 1M	$< 10^{-3}$ 2M
Alpine (n=50)	$< 10^{-13}$ **6K**	$< 10^{-3}$ 458K	$< 10^{-3}$ 2.9M	$< 10^{-3}$ 5.8M

Average minimum value and average number of function evaluations (FEs) over 250 runs for continuous function minimization with dimensions 20 and 50. SA cooling factors are set to 0.98 and 0.99. The results of the Dialectic search and SA methods are reproduced from [13].

Table IV

Function (n=30)	*SGA*	*EP*	*ES*	*CPSO*	*GSO*	*AID*
Sphere	37.9109 (12.8125)	1.5752e-02 (9.6308e-03)	0.1541 (0.7470)	1.4638e-20 (5.2324e-20)	9.1120e-09 (1.5094e-08)	**0(0)**
Rosenbrock	1387.0214 (664.5327)	60.1078 (37.0862)	170.9490 (341.0239)	1958.190 (12720.2231)	49.1093 (31.217)	**1.903042e-09 (2.901381e-09)**
Schwefel's Problem 2.26	-12296.4843 (182.4633)	-7313.6252 (708.4178)	-7424.6424 (654.6766)	-9717.1033 (714.7694)	**-12569.4848 (0.021968)**	-9.634095e+03 (1.995549e+02)
Rastrigin	38.7142 (8.5158)	17.2683 (4.7685)	71.3950 (15.4787)	56.5377 (19.6147)	2.6736 (1.6404)	**4.024514e-13 (1.635588e-13)**
Ackley	2.6070 (0.2984)	0.1082 (2.8982e-02)	3.9424 (1.4460)	0.3260 (0.6127)	2.4871e-05 (2.1946e-02)	**3.925749e-14 (6.733153e-15)**
Griewank	1.3301 (0.1323)	0.1405 (0.2580)	2.5573e-02 (2.9625e-02)	1.5848e-02 (2.1320e-02)	2.5030e-02 (2.9514e-02)	**0(0)**

Mean and standard deviation (SD) of the function optimization results over 50 runs. The results of SGA, EP, ES, PSO and GSO are reproduced from [11].

Table V

CEC Function (n=10)	Avg. GAP	#successful runs	FEs
F1	5.456968e-14(1.996275e-14)	25	6.000000e+03
F2	1.705303e-13(1.333096e-13)	25	6.000000e+03
F4	3.723694e+00(1.767073e+01)	23	4.4594e+04
F6	1.594632e-01(7.973158e-01)	24	3.2605e+04
F8	2.000000e+01(1.192652e-11)	0	–
F9	1.136868e-13(4.019437e-14)	25	6.000000e+03
F10	3.414685e+01(2.046458e+01)	0	–
F12	6.382937e+01(2.995944e+02)	16	1.5246e+04
F13	4.446956e-01(1.722232e-01)	0	–

Performance of AID when applied to 9 CEC test problems with n=10.

4.1. *AID variants*

Several variants of AID are examined in this subsection, to get a better picture of how the method may work when different SS and PR options are selected for generating combinations:

a) AID-1

In this variant, only PR is used as a combination method (SS is removed). The rationale behind this modification is to see whether PR by itself is responsible for the good behavior of AID.

b) AID-2

AID-2 replaces C1 with the following combination methods:

C1A: $z_1 = (1/3)x + (2/3)y$

C1B: $z_2 = (2/3)x + (1/3)y$

The rationale behind examining C1A and C1B is that they give a small number of points to look at on the line between x and y that are evenly separated from the endpoints. (This separation is for diversification purposes.) Thus, we want to see whether C1 can be replaced by simple combination methods that do not rely on randomization. (C2 remains intact in this test.)

c) AID-3

In AID-3, PR is modified to alternate the choice of initiating and guiding solutions. A detailed algorithm for the alternating path relinking is shown in Alg. III.

d) AID-4

In this variant, C2 is replaced by the following combination methods:

C2A: path_relinking(y, x)

C2B: path_relinking(x, y)

C2C: alternating_path_relinking(x, y)

The path relinking method used in C2A and C2B is different from the one used in AID's C2. In C2, $f(x) \leq f(y)$ while C2A and C2B make no reference to objective values (i.e. the guiding solution could be better or worse than the initiating solution).

e) AID-5

AID-5 is the same as AID except that C2 is replaced by:

C2A: path_relinking(y, x)

C2A': path_relinking(y, x)

C2A": path_relinking(y, x)

Algorithm 3 Pseudo code of the alternating path relinking algorithm.

```
Notation:

x* = the best solution found during the alternating
path relinking approach, excluding the two
solutions x and y since we don't want to choose
one of these solutions.

$f* = f(x*)$

Alternating Path Relinking Algorithm:

Set $x^a = x; x^b = y$.

$f* =$ a large positive value
I = {i  $x^a(i) \neq x^b(i)$}.
GuideIs$X^b$ = True

while I is not empty do
    Select an index i from I (randomly or strategically)
    if $GuideIsX^b = True$ then
        (x$^b$ is the guiding solution)
        x$^a$(i) = x$^b$(i)
        if $f(x^a) < f*$ then
            $x* = x^a$
            $f* = f(x*)$
        end if
        GuideIsX$^b$ = False

    else
        x$^a$ is the guiding solution
        x$^b$(i) = x$^a$(i)
        if$f(x^b) < f*$ then
            $x* = x^b$
            $f* = f(x*)$
        end if
        GuideIsX$^b$ = True

    end if
    I = I - {i}
end while
```

C2A' and C2A" are just two repetitions of C2A, but will get different results because of the randomization within these methods.

The above variants performed nearly the same or slightly worse than AID and consequently we do not report their outcomes in detail.

5. Conclusion

We have proposed an Alternating Intensification/Diversification (AID) local search method that makes use of solution combination strategies from scatter search and path relinking. In contrast to the customary SS and PR procedures, however, these combination strategies are applied to a single pair of solutions at each iteration, rather than to a collection of pairs (or triples, etc.) drawn from a population of solutions. The pair that is combined consists of an initiating solution x and its complement y, where in the present setting we define complementation relative to the vectors L and U of lower and upper bounds. The best combination z takes the role of a new vector x submitted to a local improvement (intensification) method. (In this study, the EUS method was used as the improvement method.) The outcome of this intensification step finally becomes the new initiating solution x.

Our method was compared against 20 different methods from the literature. Specifically, we compared against the SPSA and RASH methods on 17 benchmark functions, against the recently proposed dialectic search and SA on six test cases, and against five other popular metaheuristic methods on six benchmark functions. AID in general outperformed all of the other methods on the benchmark functions. In addition, AID was compared with 11 methods when applied to 9 CEC benchmark functions. AID generally obtains good quality solutions on these functions. Finally, we introduced five other variants of AID, though we found that their performance was not significantly different than that of our basic approach.

Given the straightforward character of our procedure, we envision that it may advantageously be embedded as a subroutine to yield an enhancement for more complex methods. Appealing avenues for future work include investigating the use of other combination methods and intensification processes, and the incorporation of additional controls to avoid re-visiting regions that are exceedingly close to previously generated solutions. Additional opportunities for future research consist of generalizing AID to handle constrained problems, and applying these generalized procedures to engineering optimization problems from real-world settings.

References

[1] T. Bäck, F. Hoffmeister, and H. P. Schwefel. A survey of evolution strategies. *Proceedings of the Fourth International Conference on Genetic Algorithms and their Applications*, pages 2–9, 1991.

[2] M. Brunato and R. Battiti. R.: RASH: a self-adaptive random search method. *Adaptive and multilevel metaheuristics, studies in computational intelligence*, page 136, 2008.

[3] D. B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. Wiley-IEEE Press, 2006.

[4] V. Gardeux, R. Chelouah, P. Siarry, and F. Glover. Unidimensional search for solving continuous high-dimensional optimization problems. In *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*, pages 1096–1101. IEEE, 2009.

[5] F. Glover. Tabu Search–Part i. *Informs journal on computing*, 1(3):190–206, January 1989.

[6] F. Glover. Tabu Search–Part II. *Informs journal on computing*, 2(1):4–32, January 1990.

[7] F. Glover. A template for scatter search and path relinking. In *Artificial evolution*, pages 1–51. Springer, 1998.

[8] F. Glover, M. Laguna, and R. Marti. Scatter search. In *Advances in evolutionary computing*, pages 519–537. Springer-Verlag New York, Inc., 2003.

[9] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley, 1989.

[10] N. Hansen. Compilation of results on the 2005 CEC benchmark function set. *Computational Laboratory (CoLab), Institute of Computational Science, ETH Zurich, Tech. Rep*, 2006.

[11] S. He, Q. H. Wu, and J. R. Saunders. A novel group search optimizer inspired by animal behavioural ecology. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1272–1278. IEEE, 2006.

[12] J. H. Holland. *Adaptation in natural and artificial systems*. MIT Press Cambridge, MA, USA, 1992.

[13] S. Kadioglu and M. Sellmann. Dialectic search. In *Proceedings of the 15th international conference on Principles and practice of constraint programming*, pages 486–500. Springer-Verlag, 2009.

[14] J. P. Kelly, M. Laguna, and F. Glover. A study of diversification strategies for the quadratic assignment problem. *Computers & Operations Research*, 21(8):885–893, 1994.

[15] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Perth, Australia, 1995.

[16] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308, 1965.

[17] M. G. C. Resende, C. C. Ribeiro, F. Glover, and R. Martí. Scatter search and path-relinking: Fundamentals, advances, and applications. *Handbook of Metaheuristics*, pages 87–107, 2010.

[18] H. P. P. Schwefel. *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc. New York, NY, USA, 1993.

[19] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *Automatic Control, IEEE Transactions on*, 37(3):332–341, 2002.

[20] W. Spears, K. De Jong, T. Bäck, D. Fogel, and H. De Garis. An overview of evolutionary computation. In *Machine Learning: ECML-93*, pages 442–459. Springer, 1993.

[21] R. Storn and K. Price. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. *International computer science institute-publications-TR*, 1995.

[22] T. Stützle. Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539, 2006.

[23] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[24] P. Y. Yin, F. Glover, M. Laguna, and J. X. Zhu. Cyber swarm Algorithms-Improving particle swarm optimization using adaptive memory strategies. *European Journal of Operational Research*, 201(2):377–389, 2010.

Received 16-2-2010; revised 30-8-2010; accepted 20-11-2010