Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 861765, 15 pages
http://dx.doi.org/10.1155/2014/861765

Research Article

Combining Interval Branch and Bound and Stochastic Search

Dhiranuch Bunnag

Department of Mathematics, Chiang Mai University, Chiang Mai 50200, Thailand

Correspondence should be addressed to Dhiranuch Bunnag; dhiranuch@yahoo.com

Received 19 August 2014; Revised 17 October 2014; Accepted 17 October 2014; Published 24 November 2014

Academic Editor: Douglas R. Anderson

This paper presents global optimization algorithms that incorporate the idea of an interval branch and bound and the stochastic search algorithms. Two algorithms for unconstrained problems are proposed, the hybrid interval simulated annealing and the combined interval branch and bound and genetic algorithm. The numerical experiment shows better results compared to Hansen's algorithm and simulated annealing in terms of the storage, speed, and number of function evaluations. The convergence proof is described. Moreover, the idea of both algorithms suggests a structure for an integrated interval branch and bound and genetic algorithm for constrained problems in which the algorithm is described and tested. The aim is to capture one of the solutions with higher accuracy and lower cost. The results show better quality of the solutions with less number of function evaluations compared with the traditional GA.

1. Introduction

Many problems in economics, business, sciences, and engineering are modeled as constrained optimization problems:

$$\begin{aligned} &\min_{x\in\Omega} \quad f(x) \\ &\text{subject to} \quad g_i(x) \le 0 \quad i = 1,\ldots,n_g \\ &\qquad\qquad h_i(x) = 0 \quad i = 1,\ldots,n_h. \end{aligned} \tag{1}$$

There are various approaches to the problems. Interval algorithms use branch and bound techniques to capture all solutions. One drawback is that they often require more memory and CPU time [1]. Stochastic search algorithms are usually easy to implement and no assumption about the continuity and differentiability is required. They are commonly used in numerous fields. Even though there is no guarantee on the solution reported when the algorithm is terminated in a finite time, the theoretical support is only on the convergence in probability. We have an interest in combining the interval branch and bound technique with stochastic search. We first study the combined algorithms for unconstrained problems and then modify them to handle the constrained problems.

The paper is organized as follows. In the next section, the related algorithms are introduced, namely, the interval branch and bound, simulated annealing, and the continuous genetic algorithms. The studies of the improvement of the described algorithms are also discussed. Our proposed algorithms are presented in Section 3. We demonstrate two algorithms for unconstrained and one for constrained problems. In Section 4, the numerical experiments and the discussion are given, followed by the conclusions in Section 5.

2. Interval Branch and Bound and Stochastic Algorithms

2.1. Interval Branch and Bound

2.1.1. Unconstrained Optimization. An interval algorithm is a tool using interval arithmetics for finding all solutions of the optimization problems. Before the discussion of the algorithms, let us first introduce the notations that we will use in this paper.

For an unconstrained problem, we study $\min_{x\in\Omega} f(x)$.

Ω: A search domain

$f : \mathbb{R}^n \to \mathbb{R}$

$\square f(Y) = \{f(x) : x \in Y\}$: The range of f over Y

$\mathbf{I}$: a set of real compact interval $[a, b]$, $a, b \in \mathbb{R}$

$\mathbf{I^n}$: a set of n-dimensional column vectors

$F : \mathbf{I^n} \rightarrow \mathbf{I}$ an inclusion function of f

$V \in \mathbf{I^n}$ will be called "box"

n_b: A number of subregions

n_s: A number of sample points from a given box

`listx`: A list containing the boxes and their information $\{(X_k, \text{lb}(F(X_k)))\}$

n_l: A number of boxes in `listx` or the length of `listx`

T: Temperature

T_{init}: An initial temperature for cooling schedule of simulated annealing

maxiter: Maximum number of iterations

$mid(A)$: Midpoint of an interval A, $mid(A) = (a+b)/2$

$w(A)$: The width of an interval $A = [a, b]$, $w(A) = b-a$

If $V \in \mathbf{I^m}$, $w(V)$ is the width of the box $V = \max_{i=1}^{m}\{w(V_i)\}$.

ϵ_x: A tolerance for the width of the box

ϵ_F: A tolerance for the width of the interval F

$fbest$: The best value of all f the algorithm has been encountered

xbest: A vector corresponding to $fbest$

$fbest(i)$: A value of $fbest$ at iteration i

$\text{lb}(A)$: A lower bound of interval A

$\text{ub}(A)$: An upper bound of interval A.

An interval branch and bound have the procedures given in Algorithm 1.

A working box could be selected to be the one that has been on the list the longest or the one with the least lower bound of its inclusion function F. The bisection direction usually is the direction with the maximum width. Box V will be discarded if $\text{lb}(F(V)) > fbest$. The termination conditions can be set by using a prescribed maximum number of iterations, the width of the box, or the width of the interval F. A detailed discussion can be found in [2]. When the algorithm ended, all optima are contained in the boxes of the list.

The algorithm shows difficulty when the dimension is high or the function is complicated. The width of the boxes in the list decreases very slowly so that the improvement of the best is found.

We describe here two versions of the interval algorithms. The first is proposed by Ichida and Fujii, in which the working box is the box with the least lower bound of F. The second is Hansen's algorithm, in which the working box is the oldest box in the list.

2.1.2. Constrained Optimization. For constrained problems, a box V will be deleted from the list with the condition $fbest < \text{lb}(F(V))$ only if all points in V are feasible. The feasibility of a box is considered by using a flag vector $r = (r_1, \ldots, r_m)$ where $m = n_g + n_h$. The element r_i is assigned by the following rules.

For an inequality constraint, if $\text{ub}(G_i(V)) \leq 0$ then $r_i = 0$. If $\text{lb}(G_i(V)) > 0$,$r_i = 2$. Otherwise, it is indeterminate, $r_i = 1$.

For an equality constraint, we will consider the relaxed problem $|h_i(x)| \leq \epsilon_h$ and r_i is set according to the width of ϵ_h and the bound of H as follows:

If $(\text{lb}(H(V)) \leq 0$ and $\text{ub}(H(V)) \geq 0)$

  if $(w(H) < \epsilon_h)$

    $r_i = 0$

  else

    $r_i = 1$

else

  $r_i = 2$.

The status of a box is taken through the flag vector r. If at least one element of r is 2, the status is labeled as 2 (this box will be deleted from the list). If all elements of r are 0, the box is feasible and the status is set to be 0. Other than that the status is 1. Usually, a box with status 1 will be bisected.

2.2. Simulated Annealing. Simulated annealing (SA) is a stochastic search technique, analogy with thermodynamics. There is a mechanism in avoiding entrapment in local optima by allowing an occasional uphill move. It also incorporates a temperature parameter into the procedure, explore more at high temperature, and restrict it when the temperature is low. The basic idea of the search is that, for a given current state i with an energy level E_i, generate a subsequent state j randomly. If $E_j - E_i \leq 0$ then accept state j as the current state. Otherwise accept state j with the probability $e^{-(E_j - E_i)/T}$ where T is the temperature. For the minimization problem, the solution corresponds to a state of the system and the objective function corresponds to the energy level. Prior to the process, the cooling schedule $\alpha(T)$ and the neighborhood structure must be defined. SA is described in Algorithm 4. A thoroughly discussion can be found in [4].

2.3. Population Based Methods. Population based methods use a population of points in each iteration. One advantage of populations is that if it has multiple optimums it will be captured in its final population. The examples of such techniques are genetic algorithm (GA), ant colony, particle swarm, and differential evolution. They are widely used in business, sciences, and engineering. The simple GA will be described next.

GA imitates the natural evolution involving three processes, creation of the offsprings, selection, and mutation. The creation of offsprings offers the diversification for the search, but the selection process is narrowing it. The mutation prevents the entrapping in the local minimum. GA is outlined is Algorithm 5.

The main idea of the selection process is to choose better individuals for the next generations by considering the fitness of each one. We are now concentrating on minimization problem; thus the better fit individual is the one with the lower value of f. Good selection schemes must allow the convergence to the optimal solution without getting caught

(1) Put the domain into list.
(2) **repeat**
(3) Choose a working box, called V, and bisect it into two subboxes V_1 and V_2 and put them on the list.
(4) Delete V from the list.
(5) Discard the box in the list if it has no solution.
(6) **until** (the termination criteria hold)

Algorithm 1: Interval branch and bound algorithm.

in a local minimum. There are many proposed selection techniques and the study of their convergence.

Some of the methods are, for example, elitist selection (the best individuals of each generation must be selected), the proportional selection (the better fit individuals have higher probability to be selected), the ranking selection (the individuals are ranked according to their fitness and the selection is based on this ranking), and the tournament selection (individuals are divided into subgroups and the members of each group compete against each other, and then only one is selected to be in the new generation). The mutation rate is set up such that $\mu(t)$ converges to 0 when t is increased. The termination condition can be set to be the maximum number of generations or the unchanged of the best value over a given number of generations. The comparison of performance of the selection schemes including the convergence can be seen in, for example, [5–7].

The disadvantage of GA is that there is no guarantee for finding optimal solutions within a finite amount of time and the tuning of parameters such as population size or mutation rate are sometimes based on trial and error. However, there are no requirements on differentiability and continuity.

2.4. The Modifications. The previous described algorithms can be modified for an improvement. Some of them that related to our works are given next.

For the interval branch and bound algorithms, the adjustment can be made to the following aspects for better solutions:

(i) an inclusion function: the kite inclusion function [8],

(ii) the subdivision of domain: multisection [8–10],

(iii) the selection of a box for further process: the box with the largest rejection index such as the one defined in Casado et al. [11],

$$pf^*(Y) = \frac{f^* - \mathrm{lb}(F(Y))}{\mathrm{ub}(F(Y)) - \mathrm{lb}(F(Y))}, \quad (2)$$

where $pf^*(Y)$ represents the rejection index of a box Y.

The studies for the better accuracy of solutions and the speed of algorithm when using an interval branch and bound to solve constrained problems are, for example, as follows.

(i) Casado et al. [11] define a rejection index of a box Y as in (2) to identify a good candidate box to contain a global minimum.

(ii) Lagouanelle and Soubry [8] use a new inclusion function called kite enclosure.

(iii) Sun and Johnson [12] introduce local sampling strategies to the working box. The convergence proof is presented along with the numerical results.

(iv) Karmakar and Bhunia [13] demonstrate how to obtain one of the solutions by partitioning the accepted box (start with the search domain) into 2^m subboxes where m is a number that each edge is divided. Calculate the function values of each subbox and then use interval order relations to choose a new accepted box.

There are many approaches in handling constraints for GA. The classifications can be seen in [3, 14]. We describe those that combine interval branch and bound and genetic algorithm.

Alander [15] suggests two ways in combining the two algorithms. The first is replacing, at least partly, the function to be optimized by some of its interval extensions. The other is using GA in some internal problems of the interval algorithm.

Sotiropoulos et al. [16] use an interval branch and bound technique to create subregions and choose the midpoint of each subregion to be individuals in an initial population for GA.

Zhang and Liu [17] use rejection index (2) as a fitness function for a box Y where f^* is the best found. The population is taken from $N > 1$ highest fitness boxes. Mutation is performed on 1/3 of the population with some probability. The best fit box is split into N subboxes.

An attempt to combine the interval branch and bound and simulated annealing for unconstrained problem can be seen in Shary [18]. The lower bound of F is used in calculating the probability of accepting a new box. If this box is accepted, it will be bisected and the half with smaller lower bound of F is chosen to be the next leading box. Otherwise, choose a different box. It is tested on a six-hump camel function and Rastrigin's function with the domain $[-10, 10]^2$.

3. The Proposed Algorithms

We first study the performance of Ichida-Fujii (A2) and Hansen (A3) for a design of our algorithm. A set of unconstrained problems in Appendix A is used. Tables 1 and 2 show that Ichida-Fujii is more effective than Hansen in terms of the speed (number of iterations), storage (the list length), and cost (number of function evaluations). It appears that

Table 1: The value of the best found, the maximum list length, the number of function evaluations, and the number of iterations from Ichida-Fujii A2 and Hansen A3 when $n = 10$.

fn	*fbest*		Max length		# f eval		Number of iterations	
	A2	A3	A2	A3	A2	A3	A2	A3
1	$4.80E-07$	9	188	43754	2734	1,600,002	1522	400000
2	$8.90E-07$	24.918	38	73340	618	1,600,002	311	400000
3	$9.30E-07$	$5.60E-14$	22	8970	642	1,482,934	355	370733
4	$6.00E-07$	0.156	36	40659	962	1,600,002	527	400000
5	$6.00E-07$	$7.50E-08$	3	3	962	1,122	479	89588
6	$9.20E-07$	$1.10E-13$	23	8682	714	916,702	357	229175
7	$8.00E-07$	44.761	52	318948	670	1,600,002	336	400000
8	$7.70E-07$	1.063	50	174631	710	1,600,002	352	400000
9	$1.00E-06$	1.061	44	174631	690	1,600,002	344	400000
10	$1.00E-06$	6.731	49	181938	690	1,600,002	344	400000
11	$8.50E-07$	0.532	112	189216	842	1,600,002	473	400000

Table 2: The value of the best found, the maximum list length, the number of function evaluations, and the number of iterations from Ichida-Fujii A2 and Hansen A3 when $n = 20$.

fn	*fbest*		Max length		# f eval		Number of iterations	
	A2	A3	A2	A3	A2	A3	A2	A3
1	$9.80E-07$	19	408	3,536	6094	202,965	682	600000
2	$3.40E-07$	220.313	113	709	1250	22,065	153	600000
3	$9.30E-07$	500	56	547	1426	45,860	159	600000
4	$9.50E-07$	21	96	274	2114	12,730	239	600000
5	$5.90E-07$	$3.80E-05$	3	3	1932	1442	239	600000
6	$9.90E-07$	5105	56	465	1434	28,205	177	600000
7	$9.60E-07$	381.58	102	384	1350	18,365	166	600000
8	$9.60E-07$	1.125	107	867	1414	20,880	176	600000
9	$9.30E-07$	1.1249	92	905	1382	25,450	171	600000
10	$9.30E-07$	9.5609	106	519	1382	26,165	171	600000
11	$9.20E-07$	3.8742	234	2,138	1898	54,270	209	600000

searching and bisecting the box with the least lower bound F is a fastest way to reach the optimal solutions.

For unconstrained problems if we assume that f is continuous, we know for certain that the box with the least lower bound of F must contain a minimum. The earlier a quality *fbest* can be discovered, the faster the unwanted boxes can be deleted.

To obtain a quality *fbest*, we consider combining SA and GA with an interval branch and bound. SA and GA act as search engine while the interval branch and bound are responsible for keeping all solutions.

When dealing with constrained problem the deletion condition will be in effect only if the box is feasible or infeasible. For most problems, the list will contain a high percentage of the boxes with status 1 which is indeterminate. That means the width of the box must be small enough in order to split feasible from the infeasible region. The situation is even worse when the dimension of the problem is high.

There are two concerns in our algorithm as follows:

(i) choosing a potential box to search: as a result, a quality *fbest* can be obtained to promote the deletion of unwanted boxes,

(ii) bisecting the box to isolate a feasible region.

For unconstrained problem, we can search in a box with the least lower bound of F. However, for constrained problem we need a function that provides information about the value of f and the lower bound of F in making a decision about which box to search for a better *fbest*. Let us first define a function $\mathtt{fit}(X)$ to be the minimum value of f among the feasible points from a box X. If the box contains x^*, $\mathtt{fit}(X)$ is getting close to f^* when the width of X approaches 0. The information of f and $\mathrm{lb}(F)$ is combined by using a function called `fun` defined by

$$\mathtt{fun}(X) = (1 - \beta_k) * \mathtt{fit}(X) + \beta_k * \mathrm{lb}(F(X)). \quad (3)$$

(1) Set $A = \Omega$.
(2) Calculate $F(A)$ and set $fbest = f(\text{mid}(A))$.
(3) Initialize `listx` $= \{(A, \text{lb}(F(A)))\}$.
(4) **repeat**
(5) Bisect A in the direction of the maximum length of the edge such that $A = V_1 \cup V_2$.
(6) Calculate $F(V_1)$ and $F(V_2)$.
(7) Remove $(A, \text{lb}(F(A)))$ from `listx`.
(8) Enter the pairs $(V_1, \text{lb}(F(V_1)))$ and $(V_2, \text{lb}(F(V_2)))$ into `listx` in a way that $\text{lb}(F(X))$ in the list do not decrease.
(9) Denote the first box in `listx` as A.
(10) Let $x_c = \text{mid}(A)$ and calculate $f(x_c)$.
(11) $fbest = \min\{fbest, f(x_c)\}$
(12) Delete a pair $(V, \text{lb}(F(V)))$ from `listx` if $fbest < \text{lb}(F(V))$.
(13) **until** (the stopping condition becomes true)

Algorithm 2: Ichida-Fujii.

(1) Set $A = \Omega$.
(2) Calculate $F(A)$ and set $fbest = f(\text{mid}(A))$.
(3) Initialize `listx` $= \{(A, \text{lb}(F(A)))\}$.
(4) **repeat**
(5) Bisect A in the direction of the maximum width such that $A = V_1 \cup V_2$.
(6) Calculate $F(V_1)$ and $F(V_2)$.
(7) Remove $(A, \text{lb}(F(A)))$ from `listx`.
(8) Enter the pairs $(V_1, \text{lb}(F(V_1)))$ and $(V_2, \text{lb}(F(V_2)))$ at the end of `listx`.
(9) Discard a pair $(V, \text{lb}(F(V)))$ from `listx` if $fbest < \text{lb}(F(V))$.
(10) Denote the first pair of `listx` by $(A, \text{lb}(F(A)))$.
(11) $fbest = \min\{fbest, f(\text{mid}(A))\}$
(12) **until** (the stopping condition becomes true)

Algorithm 3: Hansen.

$\{\beta_k\}$ is a nonincreasing sequence and $\beta_k \in [0, 1]$. Note that if $\beta_k = 1$ for all k, the algorithm uses $\text{lb}(F(X))$ in making decision for a searched box. And when $\beta_k = 0$ for all k, only function `fit` which related to f will be considered.

We will first present two algorithms for unconstrained problems to study their efficiency in terms of the speed, storage, and cost. Then the algorithm for constrained problems is described. Our goal is to get information of the search region from the interval branch and bound and then provide it for the continuous genetic algorithm to improve its efficiency.

3.1. Hybrid Interval Branch and Bound and Simulated Annealing for Unconstrained Problems. Our algorithm uses simulated annealing as a mechanism that encourages the search in a promising box, at the same time avoiding entrapment in a local minimum. It is described in Algorithm 6. We choose the box with maximum difference of the best value found so far and the least lower bound of F, $\max_i\{fbest - \text{lb}(F(V_i))\}$, to be bisected. Since $\text{lb}(F(V)) < f^* < fbest$, it might result in being able to discard half of the box.

Algorithm 6 is different from [18] in the selection of a working box. We also evaluate more than one point to update the value of the best found so far, $fbest$.

The stopping criteria are $w(F(V)) < \epsilon_F$ or $w(V) < \epsilon_x$ for all boxes V in `listx`, or the maximum number of iterations has exceeded a prescribed value *maxiter*.

We use a linear cooling schedule by prescribing the number of iterations at each temperature. The temperature is changed by using the given cooling rate; that is, new temperature = cooling rate $*$ temperature.

For Algorithm 6, the parameters that can be adjusted are the number of the initial boxes n_b, an initial temperature T_{init}, and the annealing schedule.

Note that the value of `fit`(X) can be obtained by performing some iterations of your choice of search algorithm in box X. In each iteration, `fit` value of two boxes, A and C, is calculated. Thus, $fbest$ can also be updated.

The Convergence. The following behaviors can be observed from the mechanism of Algorithm 6.

(i) At an initialization stage, the probability that the box in `listx` is selected to be a working box A is $1/n_b$. In each iteration every box has a probability of $1/(n_l - 1)$ to be a box C. Both A and C will be searched by randomly choosing n_s points and recording the best found in $fbest$ (Algorithm 7). Thus, $fbest(i)$ is

(1) Set up an initial state x_0, initial temperature T, the number of iterations for a fixed temperature N_T and $fbest = f(x_0)$.
(2) **repeat**
(3) **for** $i = 0$ to N_T **do**
(4) Randomly select a new state x from its neighborhood.
(5) $fbest = \min\{fbest, f(x)\}$
(6) Calculate $\delta = f(x) - f(x_0)$.
(7) Generate a random number r uniformly in the range $[0, 1]$.
(8) **if** $(r < \exp(-\delta/T))$ **then**
(9) $x_0 = x$.
(10) **end if**
(11) **end for**
(12) Update $T = \alpha(T)$.
(13) **until** (the stopping condition becomes true)

Algorithm 4: Simulated annealing.

(1) Set $t = 0$.
(2) Set up an initial population of individuals P_t.
(3) **repeat**
(4) Create offsprings from population P_t and put them in a set C.
(5) Select a new generation P_{t+1} from P_t and C.
(6) Perform mutation to individuals in P_{t+1} with probability $\mu(t)$.
(7) Set $t = t + 1$.
(8) **until** (the termination condition is met)

Algorithm 5: Genetic algorithm.

a decreasing sequence. This process can be viewed as a random search in the domain that shrink over time.

(ii) For each box V in `listx`, the following inequality holds:

$$\mathrm{lb}\,(F\,(V)) \le fbest\,(i) \le \mathrm{ub}\,(F\,(V))\,. \tag{4}$$

(iii) At iteration i, A or C will be bisected depending on whom has a higher value of $fbest - \mathrm{lb}(F)$.

(iv) It is possible that there is a box V in which the width is not small and survives through iterations. This means $fbest(i) - \mathrm{lb}(F(V)) < fbest(i) - \mathrm{lb}(F(A(i)))$ for some i where $1 \le i \le n_l$. Since $fbest(i) - \mathrm{lb}(F(V))$ decreases with i, this V will be selected to be A sometimes later. Therefore, a sequence of $w(A)$ is not decreasing.

(v) Every box has a nonzero probability to be bisected to make $w(F(V))$ smaller, although those boxes in `listx` have different size. We can conclude that $w(F(V)) \to 0$ as $n \to \infty$ for $V \in$ `listx`.

We can show that all solutions will be in `listx` after the algorithm successfully terminates.

Let us assume the properties of an inclusion function $w(F(Y)) \to 0$ as $w(Y) \to 0$. Denote U_n to be a union of all boxes in `listx` at iteration n; that is, $U_n = \cap_{i=1}^{n_l} U_{ni}$:

$$X^* \subseteq \bigcap_{i=1}^{\infty} U_n. \tag{5}$$

From the discarding rule the box U_{nj} will be discarded if $fbest(n) < \mathrm{lb}(F(U_{nj}))$. Since $f^* \le fbest(n)$ and $\square f(U_{nj}) \subseteq F(U_{nj})$, the solution is still in the list. Therefore, all boxes in the list contain solutions; that is, $X^* \subseteq \cap_{i=1}^{\infty} U_n$.

Suppose $x \in U_n$ for all n. There exist a box X_n where $x \in X_n$ and $\mathrm{lb}(F(X_n)) \le fbest(n)$. The properties $w(F(X_n)) \to 0$ as $n \to \infty$ and $\square f(X_n) \to f(x) \in F(X_n)$ imply that $F(X_n) \to f(x)$ as $n \to \infty$.

Now suppose that Y_n is a box containing the minimum of f; then $\mathrm{lb}(F(Y_n)) \le \mathrm{lb}(F(X_n)) \le fbest(n) \le \mathtt{fit}(Y_n) \in F(Y_n)$. With the properties $F(Y_n) \to f^*$ and $\mathrm{lb}(F(Y_n)) \to f^*$, we can conclude that $x \in X^*$.

3.2. Integrated Interval Branch and Bound and GA for Unconstrained Problems. Algorithm 8 will combine a population technique with the interval branch and bound to give a higher probability in obtaining a better value of the best found, $fbest$. Thus, more boxes will be deleted from the list. The population is selected from n_w boxes with the least lower bound of F to create a new set of candidates using the linear crossover from Michalewicz's book [3] (Algorithm 10). The information of two points is combined and the two outputs are controlled to be in the search domain. They are not necessary in a set of working boxes. Only the best n_k points are carried to the next generation.

The parameters that can be adjusted for the performance of Algorithm 8 are the following: the number of boxes n_b, the number of individuals per generation n_p, the number of working boxes n_w, the number of individuals to include

(1) Set $T = T_{\text{init}}$.
(2) Subdivide the domain Ω into n_b boxes and put them in `listx`.
(3) Randomly select an active box from `listx`, called A.
(4) Calculate `fit`(A) using Algorithm 7.
(5) $fbest$ = `fit`(A).
(6) **repeat**
(7) Pick a new box at random from `listx` excluding A, called C.
(8) Calculate `fit`(C).
(9) Calculate $\lambda = \exp(-(\text{lb}(F(C)) - \text{lb}(F(A)))/T)$.
(10) Calculate $d_a = fbest - \text{lb}(F(A))$, $d_c = fbest - \text{lb}(F(C))$.
(11) Randomly select a number $r \in \text{Unif}(0, 1)$.
(12) **if** $r < \lambda$ **then**
(13) **if** $d_a < d_c$ **then**
(14) Bisect C in the direction of the maximum length of the edge such that $C = V_1 \cup V_2$.
(15) Delete C from `listx`. Put V_1 and V_2 into `listx`.
(16) Set $A = V_1$.
(17) **else**
(18) Bisect $A = V_1 \cup V_2$. Delete A from `listx`. Put V_1 and V_2 into `listx`.
(19) Set $A = C$.
(20) **end if**
(21) **else**
(22) **if** $d_a > d_c$ **then**
(23) Bisect $A = V_1 \cup V_2$. Delete A from `listx`. Put V_1 and V_2 into `listx`.
(24) Set $A = V_1$.
(25) **else**
(26) Bisect $C = V_1 \cup V_2$. Delete C from `listx`. Put V_1 and V_2 into `listx`.
(27) **end if**
(28) **end if**
(29) Calculate `fit`(A) using Algorithm 7.
(30) $fbest = \min\{\text{fit}(A), fbest\}$.
(31) Update T using annealing schedule.
(32) Check deletion criteria to remove some boxes from `listx`.
(33) **until** (stopping criteria are satisfied)

Algorithm 6: Hybrid interval SA.

(1) Randomly select n_s points from a given box X and evaluate f of each point.
(2) Let y be the minimum value of f among the n_s sample points.

Algorithm 7: Calculate y = `fit`(X).

in the new generation n_k, and the maximum number of iterations. There are also two procedures that can be changed, the process of creating a new set of points and the rule for selecting a new generation (Algorithm 9).

3.3. Integrated Interval Algorithm and GA for Constrained Problems. The major disadvantage of the interval methods is that they require more memory and CPU time than the noninterval algorithms. We propose an algorithm that integrates a known bound from an interval algorithm and the quickness of a genetic algorithm. Of course, a certainty of the solutions is lost. However, an improvement of the quality of the solution and a reduction of the cost are gained.

Let us first introduce additional functions which will be used in the algorithm. `ifit_box`(X) is 0 if at least one feasible point has been found from box X. Otherwise, it is 1. Elements of the flag vector r corresponding to box X are 0 or 1. Therefore, we define `nviol_box`(X) to be the sum of the elements of a flag vector r. It roughly indicates the amount of constraint violation for a box X. The status of a box in the list is either 0 or 1, since a box with status 2 is discarded right after becoming known. For constrained problem `fun`(X) is modified. The term (`ifit_box`(X) + 1) is added, taking care of feasibility. Consider

$$\begin{aligned}\texttt{fun}(X) = {} & \left(1 - \beta_k\right) * \texttt{fit}(X) * (\texttt{ifit\_box}(X) + 1) \\ & + \beta_k * \text{lb}(F(X)).\end{aligned} \tag{6}$$

In the case that a feasible point is not discovered in Step 3 of Algorithm 11, the upper bound of $F(X)$ will be assigned to `fit`(X).

(1) Subdivide the domain Ω into n_b boxes $\{X_1, \ldots, X_{n_b}\}$ and calculate $F(X_i)$.
(2) Put the box X_k along with its lower bound of F in `listx` in nondecreasing order of $\text{lb}(F(X_k))$ that is, `listx` $= \{(X_k, \text{lb}(F(X_k)))\}$.
(3) Set $t = 0$.
(4) Let $\mathscr{W}_t$ be a set of the first n_w boxes on `listx`.
(5) Randomly select n_p points from $\mathscr{W}_t$ and put them in a set $\mathscr{P}_t$. Evaluate f value for each point in $\mathscr{W}_t$.
(6) Set $fbest$ to be the minimum value of f from $\mathscr{W}_t$.
(7) **repeat**
(8) Generate a new set of n_p points, $\mathscr{C}_t$, from $\mathscr{P}_t$ using Algorithm 9.
(9) Evaluate f value of each point in $\mathscr{C}_t$. Update $fbest$.
(10) Let A be a box with the maximum value of $fbest - \text{lb}(F(X_i))$ where $X_i \in \mathscr{W}_t$.
(11) Bisect A in the direction of the maximum length of the edge such that $A = V_1 \cup V_2$.
(12) Calculate $F(V_1)$ and $F(V_2)$.
(13) Remove $(A, \text{lb}(F(A)))$ from `listx`.
(14) Enter the pairs $(V_1, \text{lb}(F(V_1)))$ and $(V_2, \text{lb}(F(V_2)))$ into `listx` in a way that $\text{lb}(F(V_i))$ do not decrease.
(15) Let $\mathscr{W}_{t+1}$ be a set of the first n_w boxes on `listx`.
(16) Prepare $\mathscr{P}_{t+1}$ by choosing n_k points with the lowest value of f from the two sets $\mathscr{P}_t$ and $\mathscr{C}_t$. The other $n_p - n_k$ points from $\mathscr{W}_{t+1}$.
(17) Delete a pair $(V, \text{lb}(F(V)))$ from `listx` if $fbest < \text{lb}(F(V))$.
(18) Set $t = t + 1$.
(19) **until** (the stopping condition becomes true)

Algorithm 8: Integrated interval algorithm and GA for unconstrained problem.

Reqiure: a set P, a number of points m
(1) **repeat**
(2) Randomly select two points from P.
(3) Perform linear crossover using Algorithm 10 and put the output points in C
(4) **until** (the number of points in $\mathscr{C}$ is m)
(5) **return** a set C

Algorithm 9: Create a new set of points from a given set P.

We use a simple GA without mutation in Algorithm 12. The mutation is omitted because GA will be invoked in every iteration. The parameters that related to GA are a number of individuals in each generation and the maximum number of generations. The difference of Algorithm 11 and GA is pointed out next.

In GA, an initial population is randomly chosen from a search domain. Then this population is evolved through the three operators crossover, selection, and mutation. The change of individuals in a population is through the creation of children and mutation process.

In Algorithm 11, GA is performed in every iteration with the assigned value of maximum generations, *max_gen*. An initial population consists of the best individual, *xbest*, and the individuals randomly chosen from a given box considered as a potential region for a better solution. In a big picture, it is similar to performing mutation with the rate of one at every *max_gen* iteration. The mutation is biased because it is restricted to those in the promising region, which is `listx[idsearch]`. After this, the offsprings and populations are allowed to be in Ω. For Algorithm 11, the convergence is achieved when no box can be discarded. Thus those boxes left in the list have $w(F) < \epsilon_F$ or $w(X) < \epsilon_x$.

4. Numerical Results

4.1. Unconstrained Problems. Tables 3 and 6 show the value of $fbest$ found by Algorithms 2, 3, 4, 6, and 8 for $n = 10$ and $n = 20$, respectively. The maximum number of iterations is set to be 400,000 and 600,000. In the tables, A1 stands for Algorithm 1 and similarly for other algorithms. In Algorithms 6 and 8, the maximum length of `listx`, the number of iterations, and the number of function evaluations are the maximum number taken over ten runs. The tolerance $\epsilon_F = 10^{-6}$ is set. The algorithm is successfully terminated with $w(F(V)) < 10^{-6}$ for all ten runs. Since $fbest$ is in the range of 10^{-7}, the table presents only $1E-07$.

Table 4 displays the maximum length of `listx` and the number of iterations used in algorithm of Ichida-Fujii (A2). Since A2 uses the least storage, the ratio of the amount of the storage used by the other algorithms and A2 is presented in the tables. For example, in problem 1 the maximum list length

Require: $v = (v_1, \ldots, v_n)$, $w = (w_1, \ldots, w_n)$ where $l_i \le v_i$, $w_i \le u_i, i = 1, \ldots, n$
(1) Randomly select an integer $k \in \{0, 1, \ldots, n-1\}$.
(2) Randomly select a from the following given interval.
(3) if $v_k > w_k$, $a \in [\max((l_k - w_k)/(v_k - w_k), (u_k - v_k)/(w_k - v_k)), \min((l_k - v_k)/(w_k - v_k), (u_k - w_k)/(v_k - w_k))]$.
(4) if $v_k = w_k$, $a = 0$.
(5) if $v_k < w_k$, $a \in [\max((l_k - v_k)/(w_k - v_k), (u_k - w_k)/(v_k - w_k)), \min((l_k - w_k)/(w_k - v_k), (u_k - v_k)/(w_k - v_k))]$.
(6) $v' = (v_0, \ldots, v_{k-1}, aw_k + (1-a)v_k, \ldots, aw_{n-1} + (1-a)v_{n-1})$.
(7) $w' = (w_0, \ldots, w_{k-1}, av_k + (1-a)w_k, \ldots, av_{n-1} + (1-a)w_{n-1})$.
(8) **return** v', w'

Algorithm 10: Linear crossover (Michalewicz [3]).

(1) Set $T = T_{\text{init}}$.
(2) Subdivide Ω into n_b boxes and put them in `listx`.
(3) Calculate `fit`(X) using Algorithm 7 for every box in `listx` and set the value of `ifit_box`(X).
(4) $fbest = \min\{$`fit`$(X)\}$ where $X \in$ `listx`.
(5) Randomly choose an integer `idactive` $\in \{1, \ldots, n_l\}$. Denote the box in `listx[idactive]` by A.
(6) Calculate `nviol_box`(A) and $F(A)$.
(7) **repeat**
(8) Randomly choose an integer `idnew` $\in \{1, \ldots, n_l\} \setminus \{$`idactive`$\}$. Let C be the box in `listx[idnew]`.
(9) Calculate `nviol_box`(C) and $F(C)$.
(10) Calculate $\lambda = \exp(-($`fun`$(C) -$ `fun`$(A))/T)$ where `fun` is defined in (6).
(11) Randomly select a number $r \in \text{Unif}(0, 1)$.
(12) **If** $r < \lambda$ **then**
(13) `idsearch = idactive`
(14) **else**
(15) `idsearch = idnew`
(16) **end if**
(17) Perform GA by using Algorithm 12.
(18) Let `idbisect` be an integer corresponding to the box with max{`nviol_box`(C), `nviol_box`(A)}.
(19) Bisect the box from `listx[idbisect]` as V_1 and V_2.
(20) Delete `listx[idbisect]`.
(21) Check status of V_1 and V_2. Discard the box with status 2.
(22) Put V_1 and V_2 at the bottom of `listx`.
(23) **if** (`idsearch == idbisect`) **then**
(24) `idactive` $= n_l$
(25) **else**
(26) `idactive = idsearch`
(27) **end if**
(28) Check deletion criteria to remove some boxes from `listx`.
(29) Update T using annealing schedule.
(30) Update β_t.
(31) **until** (stopping criteria are satisfied)

Algorithm 11: Integrated interval algorithm and GA for constrained problems.

(1) Set up an initial population P_0. It consists of *xbest* and the other *n_pop* − 1 points from the box corresponding to `idsearch`.
(2) **for** $i = 1$ to max *_gen* **do**
(3) Use linear crossover, Algorithm 10, produces a set of individuals C where $C \subset \Omega$.
(4) Choose the *n_pop* best from C and P_i and put them in P_{i+1}. For feasible, the less f value is the better. For infeasible, the less violation, measured by $\sum_{j=1}^{n_g} \max\{g_j(x), 0\} + \sum_{j=1}^{n_h} |h_j(x)|$, is the better.
(5) Update *fbest* and `fit` value of the box if it applies.
(6) **end for**

Algorithm 12: GA (`idsearch`, max *_gen*, *xbest*, *n_pop*, Ω).

Table 3: The value of $fbest$ from A2, A3, A4, A6, and A8 for $n = 10$.

fn	A2 Ichida-Fujii	A3 Hansen	A4 SA	A6 Hybrid ISA	A8 Intv and GA
1	$4.8E-07$	9	$1.8E-06$	$1.E-07$	$1.E-07$
2	$8.9E-07$	24.918	$7.8E-09$	$1.E-07$	$1.E-07$
3	$9.3E-07$	$5.6E-14$	$7.4E-09$	$1.E-07$	$1.E-07$
4	$6.0E-07$	0.156	0	$1.E-07$	$1.E-07$
5	$6.0E-07$	$7.5E-08$	0	$1.E-07$	$1.E-07$
6	$9.2E-07$	$1.1E-13$	$9.5E-09$	$1.E-07$	$1.E-07$
7	$8.0E-07$	44.761	16.0625	$1.E-07$	$1.E-07$
8	$7.7E-07$	1.063	0.1156	$1.E-07$	$1.E-07$
9	$1.0E-06$	1.061	0.2249	$1.E-07$	$1.E-07$
10	$1.0E-06$	6.731	1.3388	$1.E-07$	$1.E-07$
11	$8.5E-07$	0.532	0.7077	$1.E-07$	$1.E-07$

Table 4: The maximum list length from A2 and the ratio of the maximum list length from A3, A6, and A8 to the maximum list length from A2 when $n = 10$. Also the result of the number of iterations.

fn	Max length	Ratio			Number of iterations	Ratio		
	A2	A3	A6	A8	A2	A3	A6	A8
1	188	232.7	4.2	1.1	682	586.5	5.1	1.1
2	38	1930	6.2	1.2	153	2614.4	10	1
3	22	407.7	13.6	1.6	159	2331.7	14.3	1.1
4	36	1129.4	5.3	1.3	239	1673.6	8.5	1.1
5	3	1	69	87	239	1.17	0.10	0.09
6	23	377.5	13.9	1.7	177	1294.8	10.3	1
7	52	6133.6	3.8	1.4	166	2409.6	5.4	1
8	50	3492.6	4.4	1.3	176	2272.7	5.9	1.1
9	44	3968.9	6.5	1.4	171	2339.2	6.3	1
10	49	3713	4.8	1.4	171	2339.2	7.1	1
11	112	1689.4	5.4	1	209	1913.9	7	1.1

of A4 is 4.2 times of the maximum list length of A1, which is about 1714. Table 5 shows the number of function evaluations of A2 and the ratio of the number of function evaluations used by the other algorithms and A2 for $n = 10$. Tables 6, 7, 8, and 9 present similar results for $n = 20, 40$ and 100, respectively.

The observations from the numerical results are the following.

(1) Ichida-Fujii (A2) works best.

(2) Our proposed algorithms (A6 and A8) are faster than SA (A4) and Hansen (A3).

(3) The hybrid interval SA (A6) can handle a higher dimensional domain better than SA (A4).

(4) Maximum list length used by A6 and A8 is mostly about 1-2 times of the used by A2 for $n = 20, 40, 100$ even though A8 uses a lot more of function evaluations. It implies that an effort on function evaluations does not contribute much to the reduction of the boxes. However, it shows that the structure of the algorithm can keep the storage under control. It suggests using a small number of sample points or number of individuals in the population.

(5) Algorithms 6 and 8 use a lot more storage than A2 in problem 5 but works better when the dimension is higher.

(6) Even if the algorithm found a high quality of $fbest$ at an early iteration, it may not be able to discard some boxes right away because those boxes are not small enough that the condition on the value of the lower bound of F will be satisfied. At each iteration only one box is bisected; the removing process is put on hold.

(7) When $n = 100$, Algorithm 6 does not work for problem 11. The termination is due to the memory before the reasonable result is obtained.

(8) Algorithm 6 still works quite well when n is higher, but the population based method, A8, shows the trouble with the memory.

The result suggests that the structure of the algorithm as in A6 seems to handle the length and the number of function evaluations quite well. However, using population based method captures the best faster. Therefore, the number of populations and the maximum generation must be adjusted for not having to waste too much of the number of function

Table 5: The total number of function evaluations both F and f from A2 and the ratio of the number of function evaluations from A3, A6, and A8 to the number of function evaluations from A2 when $n = 10$.

fn	# f eval	Ratio			
	A2	A3	A4	A6	A8
1	2734	585.2	1406	15.4	4.7
2	618	2589	6262.1	26.9	4.2
3	642	2309.9	4135.5	42.7	4.7
4	962	1663.2	3355.5	784.1	4.5
5	962	1.17	1024.95	4.92	8.45
6	714	1283.9	3721.3	30.8	4.2
7	670	2388.1	3567.2	16	4.3
8	710	2253.5	2925.4	17.7	4.5
9	690	2318.8	2995.7	20.2	4.3
10	690	2318.8	3042	21.1	4.4
11	842	1900.2	2663.9	20.8	4.6

Table 6: The value of $fbest$ of A2, A3, A4, A6, and A8 for $n = 20$.

fn	A2	A3	A4	A6	A8
	Ichida-Fujii	Hansen	SA	Hybrid ISA	Intv and GA
1	$9.8E-07$	19	$3.23E-06$	$1.E-07$	$1.E-07$
2	$3.4E-07$	220.313	$6.07E-08$	$1.E-07$	$1.E-07$
3	$9.3E-07$	500	$2.05E-08$	$1.E-07$	$1.E-07$
4	$9.5E-07$	21	0	$1.E-07$	$1.E-07$
5	$5.9E-07$	$3.8E-05$	1	$1.E-07$	$1.E-07$
6	$9.9E-07$	5105	$1.40E-08$	$1.E-07$	$1.E-07$
7	$9.6E-07$	381.58	46.7629	$1.E-07$	$1.E-07$
8	$9.6E-07$	1.1250	0.0172361	$1.E-07$	$1.E-07$
9	$9.3E-07$	1.1249	0.0394193	$1.E-07$	$1.E-07$
10	$9.3E-07$	9.5609	1.72853	$1.E-07$	$1.E-07$
11	$9.2E-07$	3.8742	4.49503	$1.E-07$	$1.E-07$

evaluations. This information influences the development of A11 for constrained problems.

4.2. Constrained Problems. The parameters setting for Algorithm 11 is described next. The maximum generation for GA in each iteration is set to be in the range of 15 and 25. The population size is 8–16 depending on the size of the domain. An initial β is 1. Both cooling temperature and a sequence of β are linearly decreasing. A parameter n_b, line 2 of A11, is set to be 2.

In problems 2, 3, and 5, Algorithm 11 is terminated with the condition that no box in the list can be processed ($w(X) \leq 10^{-6}$ or $w(F(X)) \leq 10^{-6}$). The other problems are terminated with the maximum iterations 10,000.

Table 10 shows information about the problem and the experimental results: the dimension, the number of constraints, the maximum width of the search domain, the error of $fbest$ found by Algorithm 11, and the error from regular GA. The seventh column is the ratio of the number of function evaluations used by GA to the number used by Algorithm 11. GA usually uses more of the number of function evaluations except for the last two problems that it uses about the same amount of number of function evaluations but Algorithm 11 discovers better solutions. The last column shows the percentage of the reduction of the number of function evaluations when Algorithm 11 is used.

Algorithm 11 successfully found optimal solutions with the condition of no box to be processed in problems 2, 3, and 5. For the other problems, A11 reports better solutions with less number of function evaluations compared with the traditional GA.

Notice that the test problems only consist of inequality constraints. For the equality constraints or the mixed one, the results are not impressive. The branch and bound process does not provide good information about solutions at an early stage. All boxes in the list still have status 1 when the algorithm reaches the maximum number of iterations.

5. Conclusions

We proposed two hybrid algorithms for unconstrained problems, Algorithms 6 and 8. Metropolis criterion is used for choosing a search box. Algorithm 6 performs the search by random sampling and Algorithm 8 by GA. The box to be bisected is considered by the maximum difference of the best value found by the algorithm and the lower bound of

Table 7: The maximum list length from A2 and the ratio of the maximum list length from A3, A6, and A8 to the maximum list length from A2 when $n = 20$. Also the result of the total number of function evaluations (F and f).

fn	Length		Ratio		# fn eval		Ratio		
	A2	A3	A6	A8	A2	A3	A4	A6	A8
1	408	8.67	1.12	2.1	6094	33.31	454.71	2.81	31.17
2	113	6.27	0.74	0.99	1250	17.65	3483.2	2.91	18.68
3	56	9.77	1.29	2.38	1426	32.16	1922.86	3.01	30.79
4	96	2.85	1.06	1.25	2114	6.02	470.67	2.97	19.13
5	3	1	139.67	175.67	1932	0.75	735.51	4.79	8.47
6	56	8.3	1.46	1.57	1434	19.67	1794.98	3.17	20.15
7	102	3.76	1.32	1.44	1350	13.6	1758.52	3.18	20.2
8	107	8.10	1.21	1.26	1414	14.77	1412.31	3.14	20.27
9	92	9.84	1.34	1.35	1382	18.42	1422.58	3.04	19.34
10	106	4.9	1.15	1.21	1382	18.93	1470.33	3.06	19.57
11	234	9.14	1.07	1.04	1898	28.59	1190.73	2.98	19.2

Table 8: The maximum list length from A2 and the ratio of the maximum list length from A6 and A8 to the maximum list length from A2 when $n = 40$. Also the result of the total number of function evaluations (F and f).

fn	Max length	Ratio		# fn eval	Ratio	
	A2	A6	A8	A2	A6	A8
1	858	1.16	1.23	12494	2.66	20.05
2	187	0.44	1.21	2502	2.82	18.94
3	147	1.06	1.07	2870	2.84	19.75
4	222	1.20	1.20	4282	2.80	19.53
5	3	27.00	27.67	3842	3.07	4.48
6	143	1.34	1.33	2878	2.96	20.80
7	278	1.01	1.05	2718	2.97	20.92
8	213	1.28	1.28	2830	2.96	20.22
9	178	1.39	1.40	2762	2.83	19.80
10	210	1.21	1.25	2762	2.85	19.72
11	541	3.00	35.80	3834	6.85	401.70

an inclusion function of the box. Both of them perform better than the traditional SA and Hansen's algorithm, where the bisected box is considered by age. The structure of Algorithm 6 is modified to handle constrained problem, Algorithm 11.

The contribution of our proposed Algorithm 11 is a good structure of the algorithm which gives the following advantages.

(1) It reduces the number of function evaluations of the regular genetic algorithm.

(2) If the problem is not so complicated, the solution is ensured by the branch and bound process. This is the advantage of our hybrid algorithm over GA. If the storage is limited, the quality of the reported solution is still acceptable. Moreover, the region that might contain optimum is still in the list. If required, the local search can be applied to that region.

(3) The branch and bound process, which is easy to implement, is responsible for providing the potential individuals for GA. It can be viewed as acceleration for GA.

One weak point of our algorithms is that the deletion process is not activated until the box is small enough, although a high quality $fbest$ is found in an early iteration.

A further study includes the division of a box and the technique for handling equality constraints.

Appendices

A. Unconstrained Problems [1]

(1) Modified Rosenbrock function over $[-100, 100]^n$

$$f(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(1 - x_i \right)^2 \right]. \quad \text{(A.1)}$$

(2) Zakharov function over $[-9, 11]^n$

$$f(x) = \sum_{i=1}^{n} x_i^2 + \left(0.5 \sum_{i=1}^{n} i x_i \right)^2 + \left(0.5 \sum_{i=1}^{n} i x_i \right)^4. \quad \text{(A.2)}$$

Table 9: The maximum list length from A2 and the ratio of the maximum list length from Algorithms A6 and A8 to the maximum list length from A2 when $n = 100$. Also the result of the total number of function evaluations (F and f).

fn	Max length	Ratio		# fn eval	Ratio	
	A2	A6	A8	A2	A6	A8
1	2238	1.24	1.32	2238	2.96	3.27
2	465	1.41	1.63	465	3.36	6.33
3	389	1.18	2.29	389	3.26	5.24
4	742	1.03	1.04	742	3.22	3
5	3	69.33	69.67	3	3.37	3
6	383	1.42	1.43	383	3.33	3
7	710	1.1	1.11	710	3.03	3
8	655	1.05	1.05	655	3.26	3.02
9	465	1.3	1.3	465	3.12	3.01
10	524	1.24	1.25	524	3.12	3
11	1586	1.02	—	1586	2.98	—

Table 10: The result of constrained problems. The ratio in the seventh column is the ratio of number of functions evaluations used by GA to the number used by A11. The last column is the percentage of the number of function evaluations that can be reduced when using A11.

fn	Dim	# constr	Max width	Error A11	Error GA	Ratio	% reduction
c1	2	2	100	$5.1E-04$	$4.9E-03$	1.84	45.72
c2	2	2	10	0	$6.8E-03$	659.76	99.85
c3	2	1	1	0	$6.6E-02$	178.97	99.44
c4	2	2	6	$5.5E-05$	$2.3E-03$	1.93	99.44
c5	5	6	24	0	$1.4E-04$	1.42	29.74
c6	6	4	0.31	$2.0E-04$	$4.1E+01$	1.98	49.6
c7	6	6	30	$9.6E-03$	$9.0E-02$	1.93	48.29
c8	7	4	100	$6.9E-03$	$1.7E-02$	11.87	48.29
c9	8	6	100	$9.1E-03$	$8.2E-02$	1.00	0.19
c10	13	9	100	$4.0E-05$	$1.3E-03$	1.00	0.45

(3) Sphere function over $[-95, 105]^n$

$$f(x) = \sum_{i=1}^{n} x_i^2. \tag{A.3}$$

(4) Schwefel function 2.22 over $[-10, 8]^n$

$$f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|. \tag{A.4}$$

(5) Schwefel function 2.21 over $[-100, 80]^n$

$$f(x) = \max_{i=1}^{n} \{|x_i|\}. \tag{A.5}$$

(6) Step function over $[-100, 200]^n$

$$f(x) = \sum_{i=1}^{n} (x_i + 0.5)^2. \tag{A.6}$$

(7) Generalized Rastrigin function over $[-6.12, 5.12]^n$

$$f(x) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right). \tag{A.7}$$

(8) Modified Griewank function over $[-590, 600]^n$

$$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right). \tag{A.8}$$

(9) Another modified Griewank function over $[-590, 600]^n$

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \frac{\left[2 + \cos\left(x_i/\sqrt{i}\right)\right]}{3} + 1. \tag{A.9}$$

(10) Locatelli's modification #2 of Griewank function over $[-590, 600]^n$

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \sum_{i=1}^{n} \ln\left[2 + \cos\left(\frac{x_i}{\sqrt{i}}\right)\right] + n\ln 3. \tag{A.10}$$

(11) Locatelli's modification #3 of Griewank function over $[-590, 600]^n$

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \sum_{i=1}^{n} \ln\left[2 + \cos\left(\sum_{j=1}^{n} A_{ij} x_j\right)\right] + n\ln 3, \tag{A.11}$$

where $A_{ij} = 1$ if $i \neq j$ and $A_{ii} = n + 1$.

B. Constrained Problems [3, 19]

(c1) min $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$ subject to

$g_1(x) = (x_1 - 5)^2 + (x_2 - 5)^2 + 100 \le 0$

$g_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 - 82.81 \le 0$

The bounds: $L = (13, 0)$ and $U = (100, 100)$

$x^* = (14.095, 0.84296)$, $f(x^*) = -6961.81388$

(c2) max $f(x) = \sin^3(2\pi x_1)\sin(2\pi x_2)/x_1^3(x_1 + x_2)$ subject to

$g_1(x) = x_1^2 - x_2 + 1 \le 0$

$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0$

The bounds: $L = (0, 0)$ and $U = (10, 10)$

$x^* = (1.2279713, 4.2453733)$, $f(x^*) = 0.095825$

(c3) min $f(x) = -x_1 - x_2$ subject to

$g_1 = -(3/32)[(x_1-1)^2+(x_2-1)^2]-(3/16)(x_1-1)(x_2-1) \le 0$

The bounds: $L = (0, 0)$ and $U = (1, 1)$

$f(x^*) = -1.8729$

(c4) min $f(x) = (x_1^2 + x_2 - 11)^2 + (x_2^2 + x_1 - 7)^2$ subject to

$g_1 = -4.84 + (x_1 - 0.05)^2 + (x_2 - 2.5)^2 \le 0$

$g_2 = 4.84 - x_1^2 - (x_2 - 2.5)^2 \le 0$

The bounds: $L = (0, 0)$ and $U = (6, 6)$

$f(x^*) = 13.59085$

(c5) min $f(x) = 5.3578547x23 + 0.8356891x1x5 + 37.293239x1 - 40792.141$ subject to

$g_1(x) = u(x) - 92 \le 0$

$g_2(x) = -u(x) \le 0$

$g_3(x) = v(x) - 110 \le 0$

$g_4(x) = -v(x) + 90 \le 0$

$g_5(x) = w(x) - 25 \le 0$

$g_6(x) = -w(x) + 20 \le 0$ where

$u(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5$

$v(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2$

$w(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4$

The bounds: $L = (78, 33, 27, 27, 27)$ and $U = (102, 45, 45, 45, 45)$

$x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$, $f(x^*) = -30665.539$

(c6) min $f(x) = 4.3x_1 + 31.8x_2 + 63.3x_3 + 15.8x_4 + 68.5x_5 + 4.7x_6$ subject to

$g_1 = 4.97 - 17.1x_1 - 38.2x_2 - 204.2x_3 - 212.3x_4 - 623.4x_5 - 1495.5x_6 + 169x_1x_3 + 3580x_3x_5 + 3810x_4x_5 + 18500x_4x_6 + 24300x_6x_5 \le 0$

$g_2 = -1.88 - 17.9x_1 - 36.8x_2 - 113.9x_3 - 169.7x_4 - 337.8x_5 - 1385.2x_6 + 139.0x_1x_3 + 2450x_4x_5 + 600x_4x_6 + 17200x_6x_5 \le 0$

$g_3 = -429.08 + 273x_2 + 70x_4 + 819x_5 - 26000x_4x_5 \le 0$

$g_4 = -78.02 - 159.9x_1 + 311x_2 - 587x_4 - 391x_5 - 2198x_6 + 14000x_1x_6 \le 0$

The bounds: $L = (0, 0, 0, 0, 0, 0)$ and $U = (0.31, 0.046, 0.068, 0.042, 0.028, 0.0134)$

$f(x^*) = 0.0156$

(c7) min $f(x) = -25(x_1 - 2.0)^2 - (x_2 - 2)^2 - (x_3 - 1.0)^2 - (x_4 - 4.0, 2)^2 - (x_5 - 1.0)^2 - (x_6 - 4.0)^2$ subject to

$g_1 = -(x_3 - 3)^2 - x_4 + 4 \le 0$

$g_2 = -(x_5 - 3)^2 - x_6 + 4 \le 0$

$g_3 = x_1 - 3x_2 - 2 \le 0$

$g_4 = -x_1 + x_2 - 2 \le 0$

$g_5 = x_1 + x_2 - 6 \le 0$

$g_6 = -x_1 - x_2 + 2 \le 0$

The bounds: $L = (0, 0, 1, 0, 1)$ and $U = (30, 30, 5, 6, 5)$

$f(x^*) = -310$

(c8) min $f(x) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$ subject to

$g_1(x) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127 \le 0$

$g_2(x) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \le 0$

$g_3(x) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196 \le 0$

$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0$

The bounds: $L = (-10, \ldots, -10)$ and $U = (10, \ldots, 10)$

$x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$,

$f(x^*) = 680.6300573$

(c9) min $f(x) = x_1 + x_2 + x_3$ subject to

$g_1(x) = -1 + 0.0025(x_4 + x_6) \le 0$

$g_2(x) = -1 + 0.0025(-x_4 + x_5 + x_7) \le 0$

$g_3(x) = -1 + 0.01(-x_5 + x_8) \le 0$

$g_4(x) = 100x_1 - x_1x_6 + 833.33252x_4 - 83333.333 \le 0$

$g_5(x) = x_2x_4 - x_2x_7 - 1250x_4 + 1250x_5 \le 0$

$g_6(x) = x_3x_5 - x_3x_8 - 2500x_5 + 1250000 \le 0$

The bounds: $L = (100, 1000, 1000, 10, \ldots, 10)$ and $U = (10000, 10000, 10000, 1000, \ldots, 1000)$

$x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$,

$f(x^*) = 7049.3307$

(c10) min $f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$ subject to

$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0$

$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0$

$$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$
$$g_4(x) = -8x_1 + x_{10} \leq 0$$
$$g_5(x) = -8x_2 + x_{11} \leq 0$$
$$g_6(x) = -8x_3 + x_{12} \leq 0$$
$$g_7(x) = -2x_4 - x_5 + x_{10} \leq 0$$
$$g_8(x) = -2x_6 - x_7 + x_{11} \leq 0$$
$$g_9(x) = -2x_8 - x_9 + x_{12} \leq 0$$

The bounds: $L = (0, \ldots, 0)$ and $U = (1, \ldots, 1, 100, 100, 100, 1)$

$x^* = (1, \ldots, 1, 3, 3, 3, 1)$, $f(x^*) = -15$.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The author would like to thank Professor Min Sun for his advice and encouragement on the interval algorithm. His kindness is gratefully acknowledged. This work was supported by Chiang Mai University, Thailand.

References

[1] M. Sun, "A fast memoryless interval-based algorithm for global optimization," *Journal of Global Optimization*, vol. 47, no. 2, pp. 247–271, 2010.

[2] H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization*, Halstad Press, New York, NY, USA, 1988.

[3] Z. Michalewicz, *Genetic algorithms + Data Structures = Evolution Programs*, Springer, New York, NY, USA, 3rd edition, 1994.

[4] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, New York, NY, USA, 1989.

[5] T. Blickle and L. Thiele, "A comparison of selection schemes used in genetic algorithms," Tech. Rep., Swiss Federal Institute of Technology (ETH), Computer Engineering and Communications Networks Lab (TIK), Zurich, Switzerland, 1995.

[6] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, pp. 69–93, Morgan Kaufmann, 1991.

[7] N. M. Razali and J. Geraghty, "Genetic algorithm performance with different selection strategiesin solving TSP," in *Proceedings of the World Congress on Engineering*, pp. 1134–1139, IAENG Publications, July 2011.

[8] J.-L. Lagouanelle and G. Soubry, "Optimal multisections in interval branch-and-bound methods of global optimization," *Journal of Global Optimization*, vol. 30, no. 1, pp. 23–38, 2004.

[9] A. E. Csallner, T. Csendes, and M. C. Markót, "Multisection in interval branch-and-bound methods for global optimization—I. Theoretical results," *Journal of Global Optimization*, vol. 16, no. 4, pp. 371–392, 2000.

[10] M. C. Markót, T. Csendes, and A. E. Csallner, "Multisection in interval branch-and-bound methods for global optimization. II. Numerical tests," *Journal of Global Optimization*, vol. 16, no. 3, pp. 219–228, 2000.

[11] L. G. Casado, I. Garcia, and T. Csendes, "A heuristic rejection criterion in interval global optimization algorithms," *BIT. Numerical Mathematics*, vol. 41, no. 4, pp. 683–692, 2001.

[12] M. Sun and A. W. Johnson, "Interval branch and bound with local sampling for constrained global optimization," *Journal of Global Optimization*, vol. 33, no. 1, pp. 61–82, 2005.

[13] S. Karmakar and A. K. Bhunia, "On constrained optimization by interval arithmetic and interval order relations," *OPSEARCH*, vol. 49, no. 1, pp. 22–38, 2012.

[14] C. A. Coello Coello, "A survey of cons traint handling techniques used with evolutionary algorithms," Tech. Rep., Laboratorio Nacional de Informtica Avanzada, 1999.

[15] J. T. Alander, "Interval arithmetic and genetic algorithms in global optimization," in *Artificial Neural Nets and Genetic Algorithms*, pp. 388–392, Springer, Vienna, Austria, 1995.

[16] D. G. Sotiropoulos, E. C. Stavropoulos, and M. N. Vrahatis, "A new hybrid genetic algorithm for global optimization," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 30, no. 7, pp. 4529–4538, 1997.

[17] X. Zhang and S. Liu, "A new interval-genetic algorithm," in *Proceedings of the 3rd International Conference on Natural Computation (ICNC '07)*, pp. 193–197, August 2007.

[18] S. P. Shary, "Randomized algorithms in interval global optimization," *Numerical Analysis and Applications*, vol. 1, no. 4, pp. 376–389, 2008.

[19] C. A. Floudas and P. M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Springer, New York, NY, USA, 1990.