Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 392789, 10 pages
http://dx.doi.org/10.1155/2013/392789

Research Article

A Hybrid Bat Algorithm with Path Relinking for Capacitated Vehicle Routing Problem

Yongquan Zhou,[1,2] Jian Xie,[1] and Hongqing Zheng[1]

[1] *College of Information Science and Engineering, Guangxi University for Nationalities, Nanning, Guangxi 530006, China*
[2] *Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning, Guangxi 530006, China*

Correspondence should be addressed to Yongquan Zhou; yongquanzhou@126.com

Received 8 May 2013; Accepted 24 July 2013

Academic Editor: Praveen Agarwal

Copyright © 2013 Yongquan Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The capacitated vehicle routing problem (CVRP) is an NP-hard problem with wide engineering and theoretical background. In this paper, a hybrid bat algorithm with path relinking (HBA-PR) is proposed to solve CVRP. The HBA-PR is constructed based on the framework of continuous bat algorithm; the greedy randomized adaptive search procedure (GRASP) and path relinking are effectively integrated into bat algorithm. Moreover, in order to further improve the performance, the random subsequences and single-point local search are operated with certain loudness (probability). In order to verify the validity of the method in this paper, and it's efficiency and with other existing methods, several classical CVRP instances from three classes of CVRP benchmarks are selected to tested. Experimental results and comparisons show that the HBA-PR is effective for CVRP.

1. Introduction

The vehicle routing problem (VRP) is a classical combinatorial optimization problem that was proposed in the late 1950s, and it is still a research hotspot in the field of Operations Research. The capacitated vehicle routing problem (CVRP) is introduced by Dantzig and Ramser in 1959 [1], which designs a set of customer demands that have to be served with a fleet of vehicles from a depot or central node, each vehicle has the uniform capacity, and each customer has a certain demand that must be satisfied at minimal cost. These costs usually represent distances, traveling times, number of vehicles employed, or a combination of these factors.

It is known that the CVRP is an NP-hard problem [2]. Various approaches have been presented to solve the CVRP during the last decades, such as linear programming [3], several metaheuristics [4–6], and many hybrid heuristics with variable neighborhood search or constructive heuristic methods [7–10]. The overview of methods presented in [6] shows that at least 29 different methods for the CVRP exist; all achieve more or less comparable performance. Although several methods can produce good solutions, the computational time is long when the scale of instances is large. Meanwhile, an abundance of methods for the CVRP is a population-based algorithm and the parameter setting of the algorithm, is pretty important; however, the parameter setting of many metaheuristics is not considered in the literature.

Bat algorithm (BA) is a fairly new metaheuristics proposed by Yang [11] in 2010, which inspired by the intelligent echolocation behavior of microbats when they forage. As we know, many new metaheuristics have been widely used and successfully applied to solve the CVRP notwithstanding BA has not yet been applied to solve the CVRP, and BA has been applied to solve other problems with great success. For example, Gandom et al. focus on solving constrained optimization tasks [12]. Yang and Gandom apply bat algorithm to solve many global engineering optimizations [13]. A classification mode is proposed by Mishra et al. using bat algorithm to update the weights of a functional link artificial neural network (FLANN) classifier [14]. Meanwhile, some researchers have improved bat algorithm and applied it to various optimization problems. Xie et al. proposed a DLBA bat algorithm based on differential operator and Lévy flights trajectory to solve function optimization and nonlinear equations [15]. Wang et al. proposed a new bat algorithm with mutation (BAM) to solve the uninhabited

combat air vehicle (UCAV) path planning problem [16]. In this paper, we propose a hybrid bat algorithm (HBA-PR) to solve capacitated vehicle routing problem.

The rest of this paper is organized as follows. The problem description of CVRP and original bat algorithm are described in Section 2. The hybrid bat algorithm (HBA-PR) for CVRP is described detailedly in Section 3. The experimental results of the HBA-PR and comparisons with other previous algorithms are shown in Section 4. In the last section, we conclude this paper and point out some future work in Section 5.

2. Problem Descriptions and Bat Algorithm

2.1. Capacitated Vehicle Routing Problem. The CVRP is considered to be the classical version of the VRP, which designs a set of customer demands that have to be served with a fleet of vehicles from a depot or central node, each vehicle has the uniform capacity, and each customer has a certain demand. The objective makes the expended cost as minimal as possible. Let $G = (V, E)$ be a complete graph with a set of vertices $V = \{0, 1, \ldots, k\}$, where the vertex $\{0\}$ represents the depot and the remaining ones represent the customers. Each edge $e_{ij} = \{i, j\} \in E$ has a nonnegative cost c_{ij}, and each customer $i \in V' = V \setminus \{0\}$ has a demand d_i. Let $S = (1, 2, \ldots, m)$ be the set of homogeneous vehicles with capacity Q. The CVRP consists in constructing a set of up to k routes in such a way that (1) every route starts and ends at the depot; (2) all demands are accomplished; (3) the vehicle's capacity is not exceeded; (4) a customer is visited by only a single vehicle; (5) the sum of costs is minimized. The mathematical formulas are defined as follows [10]:

$$\min \quad Z = \sum_{i=0}^{k} \sum_{j=0}^{k} \sum_{s=0}^{m} c_{ij} e_{ijs}, \tag{1}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i=0}^{k} d_i y_{is} \le Q, \quad s = 1, 2, \ldots, m, \\ & \sum_{s=1}^{m} y_{is} = 1, \quad i = 1, 2, \ldots, k, \\ & \sum_{i=0}^{k} e_{ijs} = y_{is}, \quad j = 1, 2, \ldots, k; \ s = 1, 2, \ldots, m, \\ & \sum_{j=0}^{k} e_{ijs} = y_{is}, \quad i = 1, 2, \ldots, k; \ s = 1, 2, \ldots, m, \end{aligned} \tag{2}$$

where s denotes the number of vehicles; $e_{ijs} = 1$ if vehicle s is from i to j, otherwise $e_{ijs} = 0$; the $y_{is} = 1$ if vehicle s is loading (active), otherwise $y_{is} = 0$.

2.2. Basic Bat Algorithm. The basic bat algorithm (BA) is a metaheuristic first introduced by Yang in 2010. In simulations of BA, under several under idealized rules, the updated rules of bat's positions x_i and velocities v_i in a D-dimensional search space are defined. The new solutions x_i^t and velocities v_i^t at generation t are given by

$$\begin{aligned} \mathrm{fr}_i &= \mathrm{fr}_{\min} + \left(\mathrm{fr}_{\max} - \mathrm{fr}_{\min}\right)\beta, \\ v_i^t &= v_i^{t-1} + \left(x_i^t - x_*\right)\mathrm{fr}_i, \\ x_i^t &= x_i^{t-1} + v_i^t, \end{aligned} \tag{3}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution and fr_i denotes frequency of each bat. Generally, the frequency $\mathrm{fr}_i \in [\mathrm{fr}_{\min}, \mathrm{fr}_{\max}]$. Here x_* is the current global best location (solution) which is located after comparing all the solutions among all the n bats.

After the position updating of bat, a random number is generated; if the random number is greater than the pulse emission rate r_i, a new position will be generated around the current best solutions; it can be represented by

$$x = x_* + \varepsilon \times \mathrm{Ld}_t, \tag{4}$$

where $\varepsilon \in [-1, 1]$ is a random number, while $\mathrm{Ld}_t = \langle \mathrm{Ld}_i^t \rangle$ is the average loudness of all the bats at current generation t.

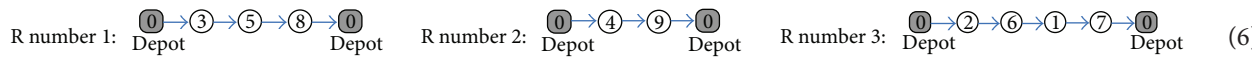
Furthermore, the loudness Ld_i and the pulse emission rate r_i will be updated, and a solution will be accepted if a random number is less than loudness Ld_i and $f(x_i) < f(x_*)$. Ld_i and r_i are updated by

$$\mathrm{Ld}_i^{t+1} = \alpha \times \mathrm{Ld}_i^t, \quad r_i^{t+1} = r_i^0 \times \left[1 - \exp\left(-\gamma \times t\right)\right], \tag{5}$$

where α, γ are constants and $f(\cdot)$ is fitness function. The algorithm repeat until the termination criterion is reached.

3. Hybrid Bat Algorithm with Path Relinking for CVRP

3.1. Solution Representation in HBA-PR. Since standard BA is a continuous optimization algorithm, the standard continuous encoding scheme of BA cannot be used to solve CVRP directly. In order to apply BA to solve CVRP, the first step is to devise a suitable representation for the candidate solutions in designing a hybrid bat algorithm for a particular problem. Each individual is a sequence with integer number which is the order of visiting these customers; the 0 represents the depot. For example, if we have the following three routes:



(6)

the coded individual with integer is $0 \to 3 \to 5 \to 8 \to 4 \to 9 \to 2 \to 6 \to 1 \to 7 \to 0$, and the bat individual is represented as $3 \to 5 \to 8 \to 4 \to 9 \to 2 \to 6 \to 1 \to 7$.

3.2. Hybrid Bat Algorithm. Aimed at the capacitated vehicle routing problem, based on the idea of bat algorithm, a hybrid bat algorithm is proposed, which integrates greedy randomized adaptive search procedure (GRASP) heuristic and bat algorithm, and the path relinking as an intensification strategy to explore local trajectories connecting elite solutions obtained by the proposed algorithm. The hybrid bat algorithm with path relinking is named as HBA-PR.

GRASP [17, 18] is a heuristic already applied to many optimization problems successfully [19–21]. GRASP consists of a two-phase iterative process: construction phase and local search phase. In the first phase, a greedy randomized solution is built. Due to this solution is not guaranteed to be locally optimal; a local search is performed in the second phase. The final result is simply the best solution found over all iterations.

The construction phase can be described as a process which stepwise adds one element at a time to a partial (incomplete) solution. According to a greedy function, all elements are sorted, and the Restricted Candidate List (RCL) is constructed based on the order, and then from the list randomly select a element. In the second phase, a local search is initialized from these points, this iterative process is repeated until a termination criterion is met, and the best solution found over all iterations is taken as the result.

RCL is created using a parameter α to restrict the size of this list. Candidate $e \in C$ is sorted according to their greedy function value $f(e)$. In a cardinality-based RCL, the latter is made up by the k top-ranked elements. In a value-based construction, the RCL consists of the elements in the set $\{e \in C : f_* \le f(e) \le f_* + \alpha \times (f^* - f_*)\}$, where $f_* = \min(f(e) : e \in C)$, $f^* = \max\{f(e) : e \in C\}$, and $\alpha \in [0, 1]$. Since the best value for α is often difficult to determine, a random value is often assigned. The values for α adopted in the constructive heuristics are set using reactive strategies, which usually leads to better performance than using fixed values [22].

In original bat algorithm, the bat individual randomly selects a certain range of frequency, its velocity is updated according to their selected frequency, and at last a new position is generated using its velocity and its own position. The idea is that the position of bat individual is updated by adjusting its frequency of sonic pulse. In this paper, the position updating of bat adopted GRASP to generate a new position, the frequency is used for restricting the size of CRL, frequency equivalent to parameter α in GRASP, and the frequency is variable value. In basic bat algorithm, the loudness Ld and the pulse emission rate r are updated according as the iterations proceed. As the loudness usually decreases, while the rate of pulse emission increases, it indicates that the bats approximate their prey (optimum solution). The pulse emission rate r is updated by

$$r(t) = \left(1 + \exp\left(\frac{-5}{t_{\max} \times (t - t_{\max}/2)}\right)^{-1}\right), \quad (7)$$

where t denotes the tth generation and $t_{\max}$ is the maximal generation. The rate r is similar to Sigmoid function, and the frequency fr is determined according to the pulse emission rate r, which is represented by

$$\text{fr} = \begin{cases} 1 - \max(0.2, \min(0.8, r)), & \text{rand} > r, \\ \max(0.2, \min(0.8, r)), & \text{rand} \le r, \end{cases} \quad (8)$$

where rand is a random number and 0.2 and 0.8 are experience parameters reference [18]. The frequency fr decreases gradually at firstly and then increases gradually, while the generation t increases. Figure 1 is the changing curve of rate r, Figure 2 is an example of frequency fr, and Algorithm 1 shows the pseudocode of greedy randomized construction with frequency fr.

The local search phase uses the 2-Opt heuristic for exchanging. Algorithm 2 shows the local search procedure. The input parameter is an initial solution S obtained by the Greedy_Randomized_Construction procedure. The current route need is divided into m subroute according to the load, set the start, and end of suroute said to 0; for example, $S = \{1, 2, 3, 4, 5, 6, 7\}$, subroute is $0 \to 1 \to 2 \to 3 \to 4 \to 0$, and $0 \to 5 \to 6 \to 7 \to 0$.

3.3. Hybrid Bat Algorithm with Path Relinking. Path relinking was originally proposed by Glover [23]; Laguna and Martí [24] were the first to use path relinking within a GRASP strategy. Path relinking generates new solutions by exploring trajectories connecting an initial solution x_s to an elite guiding solution x_t. The path relinking procedure consists in selecting moves that introduce attributes contained in the guiding solution x_t to the initial solution x_s until the initial solution is completely transformed in the guiding solution x_t. Path relinking may also be viewed as a constrained local search strategy applied to the initial solution x_s. Furthermore, there are several alternatives that have been considered, which involve tradeoffs between computation time and solution quality. These alternatives include *periodical relinking, forward relinking, backward relinking, back and forward relinking, mixed relinking, randomized relinking, and truncated relinking* [18].

One important issue in implementing a path relinking technique is the strategy to construct the elite set (ES). We adopted a fixed size elite set, and a solution x is inserted into the ES as follows.

A solution x is always inserted into ES if it is not full. Otherwise, the generated solution x is inserted in ES only if its cost is better than the worst cost solution found in ES, and the worst cost solution is replaced by the solution x. Algorithm 3 shows the pseudocode for the algorithm to construct and maintain the elite set ES.

Algorithm 4 shows the pseudocode of path relinking procedure; an instance is $x_s = \{1, 2, 3, 4, 5, 6, 7\}$, $x_t = \{1, 3, 4, 2, 5, 6, 7\}$, according to process of Algorithm 4, the first different element is position 2 in x_s; after executing replace operation, the $x_1 = \{1, 3, 4, 2, 5, 6, 7\}$; the second different element is position 3 in x_s; after executing replace operation, the $x_2 = \{1, 2, 4, 3, 5, 6, 7\}$; the third different element is position 4 in x_s; after executing replace operation, the $x_3 = \{1, 4, 3, 2, 5, 6, 7\}$; if the x_2 is better than the x_t, and the only, then return x_2.

$S \leftarrow R$ rand_Chosen_Vertex(v); // R is selected at random a vertex as initial solution
$C \leftarrow V \setminus v$; // The candidate set C is initialized
While $C \neq \varnothing$ **do**
  $IC \leftarrow$ Evaluate_Incremental_Costs(S); // the incremental costs are evaluated
  $ic_{\min} \leftarrow \min(\{ic \in IC\})$;
  $ic_{\max} \leftarrow \max(\{ic \in IC\})$;
  $RCL \leftarrow \{e \in C \mid ic(e) \leq ic_{\min} + fr \times (ic_{\max} - ic_{\min})\}$; // RCL is created
  $s \leftarrow$ Select_Element (RCL); // a vertex s is randomly selected from RCL
  $S \leftarrow$ Obtain_Min_Solution(S, s); // the partial tour is updated by inserting the vertex s
  $C \leftarrow C \setminus \{s\}$; // the candidate set C is updated
end
return S

Algorithm 1: *Greedy_Randomized_Construction* (fr).

$S' \leftarrow S$;
$\{sr\} \leftarrow$ Construct_Sub-route(S');
for each sub-route sr **do**
  $sr' \leftarrow$ 2-opt (sr); // carry out the 2-opt operation
end
$S' \leftarrow$ Construct_Indiviaual ($\{sr'\}$);
if $f(S') < f(S)$ **then**
  $S \leftarrow S'$
end
return S

Algorithm 2: *Local_Search_Phase* (S).

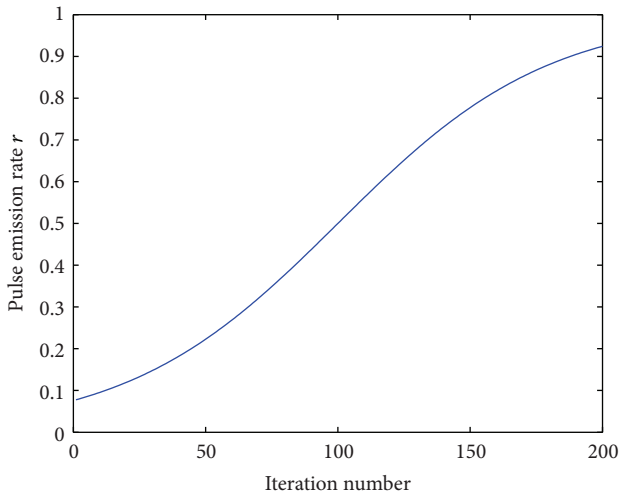


Figure 1: Changing curve of rate r.

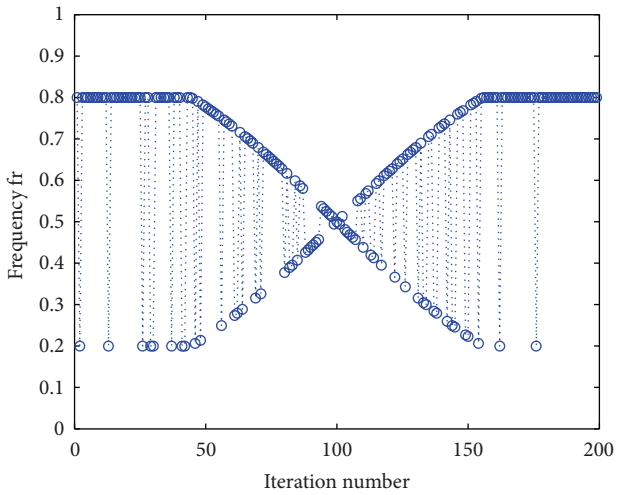


Figure 2: Changing curve of frequency fr.

if cardinality(ES) $\neq ps$ **then** // ps is the population size
  $ES \leftarrow ES \cup \{x\}$
else
  $ES_w \leftarrow$ the worst solution in ES
  if $f(x) < f(ES_w)$ **then**
    $ES \leftarrow ES \setminus \{ES_w\}$
    $ES \leftarrow ES \cup \{x\}$
  end
end
return ES

Algorithm 3: *Elite_Set* (x).

3.4. Subsequence and Single-Point Local Search. In this paper, the loudness Ld_i of bat individual i is relative to its own fitness fit_i, the better fitness, and the lower loudness. The loudness can be described by

$$\mathrm{Ld}_i = \frac{(\mathrm{fit}_i - \mathrm{fit}_* + 0.1)}{(\mathrm{fit}^* - \mathrm{fit}_* + 0.1)}, \tag{9}$$

where constant 0.1 is used for avoiding the denominator is zero, and fit_i is the fitness of individual i, fit_* and fit^* are the minimum and maximum fitness in current population, respectively. In HBA-PR, the loudness reflects the quality of individual. In this algorithm, there are two kinds of local search are embedded into HBA-PR to further improve the performance, random subsequence local search that and

$x_s \leftarrow x$; // initial solution
$x_t \leftarrow ES_*$; // guiding solution
$f(x_t) \leftarrow f(ES_*)$;
$\Delta \leftarrow Difference(x_s, x_t)$; // find out the difference position between x_s x_t
for $i = 1$: $cardinality(\Delta)$ **do**
  $j \leftarrow Find\_Position(x_s, x_t, \Delta_i)$; // find the position of the element of x_t in Δ_i which in x_s
  $x_i \leftarrow Replace(x_s, \Delta_i, j)$; // replace the j position of x_s with the element of x_s in Δ_i
  $x_i \leftarrow Replace(x_s, x_t, \Delta_i)$; //replace the Δ_i position of x_s with the corresponding element of x_t
  if $f(x_i) < f(x_t)$ **then**
    $x_* \leftarrow x_i$;
    $f(x_t) \leftarrow f(x_i)$;
  end
end
return x_*

Algorithm 4: *Path_Relinking* (ES_*, x).

single-point local search. The random subsequence local search includes random subsequence inverse and random subsequence insert, and the single-point local search includes single-point insert and single-point swap.

For random subsequence insert, firstly, randomly selected an origin of subsequence, and then randomly selected a length of subsequence which is less than length of individual S. Secondly, after determining the subsequence $S1$, randomly selected an insert point in remainder subsequence $S2$, $S = S1 \cup S2$; the $S1$ is inserted into $S2$ location in insert point. An example is shown in Figure 2. For random subsequence inverse, firstly, randomly selected a subsequence with random length, and then the inverse operation is performed. An example is shown in Figure 3.

For single-point swap, choose two different positions from a permutation randomly and swap them. For single-point insert, choose two different positions from a permutation randomly, and the element in first position is insert into the back of second element. Similarly, two instances are shown in Figures 4 and 5.

In local search part, the random subsequence local search is performed before random single-point local search. The random subsequence insert and random subsequence inverse are preformed according to loudness Ld. In other words, if a random number is greater than the loudness Ld_i, the random subsequence insert is performed; otherwise, the random subsequence inverse is performed. Similarly, the random single-point insert and random single-point swap are performed with the loudness Ld. If a random number is greater than the loudness Ld_i, the insert operation is performed (Figure 6); otherwise, the swap operation is performed. Note that where local search is operated on the current optimal individual ES_* in elite set ES, local search is performed for each individual. Algorithms 5 and 6 show the pseudocode of subsequence and single-point local search.

3.5. HBA-PR Framework for CVRP. We propose in this work, to incorporate greedy randomized adaptive search procedure, path-relinking strategies, subsequence, and single-point local search to the bat algorithm by defining distinct ways to solve capacitated vehicle routing problem. This iterative process is repeated until the termination criterion is met; Algorithm 7 shows the pseudocode of HBA-PR for CVRP.

4. Numerical Simulation Results and Comparisons

To test the performance of the proposed HBA-PR which is extensively investigated by a large number of experimental studies computational simulations are carried out with some well-studied problems taken from the web http://www.branchandcut.org/, a reference site which contains detailed information regarding a large number of benchmark instances. In this paper, 12 instances from three classes of benchmarks are selected. The first class is Augerat et al. Set A instances, the second class is Augerat et al. Set P instances, and the third class is Augerat et al. Set E instances. So far, these problems have been widely used as benchmarks to certify the performance of algorithms by many researchers.

All computational experiments are conducted with MATLAB 2012a, and in our simulation, numerical experiments are run on a PC with AMD Athlon (tm) II X4 640 Processor 3.0 GHz and 2.0 GB memory. In the experiment, the termination criterion is set as maximum generation $t_{\max} = 200$. Each instance independently run15 times for comparison.

4.1. Parameter Analysis. In the subsection, parameters of HBA-PR are determined by experiments, and the impact of each parameter is analyzed. In particular, the HBA-PR has few parameters; we only need to test population size (ps) in HBA-PR. A small ps which may lead to insufficient population information is provided, and the diversity cannot be guaranteed. On the other side, a large one which indicates diversity is sufficient, but the computing time will increase and the precision of optimal solution may have lesser improvement. In order to evaluate the sensitivity of parameters ps, three benchmarks selected from different benchmark set are chosen to run 10 times. These benchmarks are *A_n33_k5*, *E_n23_k3* and *P_n19_k2*, and the statistical result and convergence curves are shown in Figures 7, 8, 9, and 10.

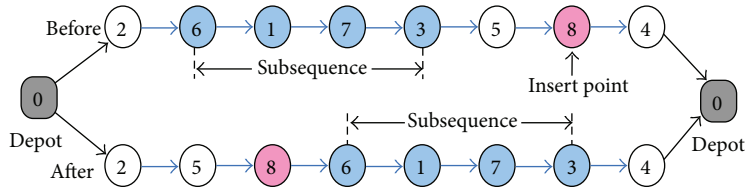


Figure 3: Random subsequence insert operation.

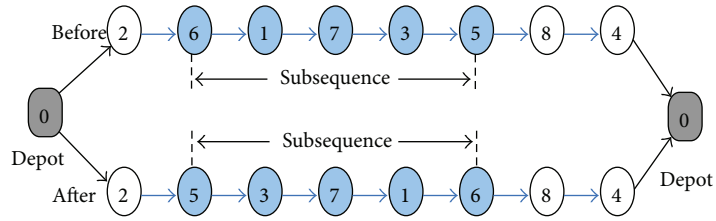


Figure 4: Random subsequence inverse operation.

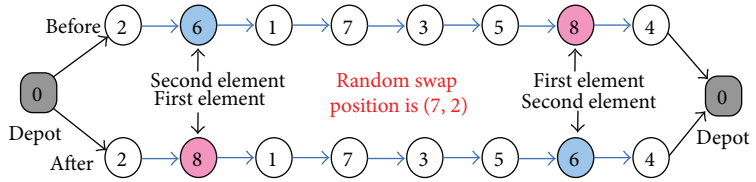


Figure 5: Single-point swap operation.

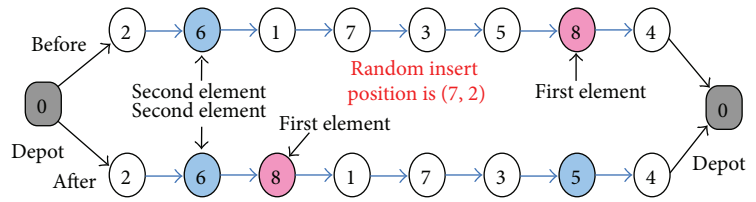


Figure 6: Single-point insert operation.

```
for i = 1: ps do
    if rand > Ld_i then
        x ← Sub-sequence_Insert(ES_*); // perform random sub-sequence insert operation
    else
        x ← Sub-sequence_Inverse(ES_*); // perform random sub-sequence inverse operation
    end
end
return x
```

Algorithm 5: *Sub-sequence_Local_Search* (ES_*).

```
for i = 1: ps do
    if rand > Ld_i then
        x ← Single-point_Insert(ES_*); // perform random single-point insert operation
    else
        x ← Single-point_Swap(ES_*); // perform random ingle point swap operation
    end
end
return x
```

Algorithm 6: *Single-point_Local_Search* (ES_*).

(1) *Initialize the ps, bat population and other parameters;*
(2) *Evaluate fitness for each individual;*
(3) $ES \leftarrow Elite\_Set(x)$;
(4) ***while*** $t \leq t_{\max}$ ***do***
(5) *Compute pulse emission rate by* (7);
(6) *Determine frequency* fr *by* (8);
(7) ***for*** $i = 1$: *ps* ***do***
(8) $x \leftarrow Greedy\_Randomized\_Construction(fr)$;
(9) $x \leftarrow Local\_Search\_Phase\ (x)$;
(10) ***end***
(11) *Evaluate fitness for each individual x;*
(12) $ES \leftarrow Elite\_Set(x)$;
(13) $ES_* \leftarrow Select\_Best\_Elite(ES)$; // *select the best individual in elite set ES*
(14) ***for*** $i = 1$: *ps* ***do***
(15) $x \leftarrow Path\_Relinking(ES_*, x)$;
(16) ***end***
(17) *Evaluate fitness for each individual x;*
(18) $ES \leftarrow Elite\_Set(x)$;
(19) $Ld_i \leftarrow Compute\_Loudness(f(x))$; //*Compute loudness of each individual by* (9);
(20) $ES_* \leftarrow Select\_Best\_Elite(ES)$;
(21) $x \leftarrow$ *Sub-sequence_Local_Search* (ES_*) //*carry out random sub-sequence local search*
(22) *Evaluate fitness for each individual x;*
(23) $ES \leftarrow Elite\_Set(x)$;
(24) $ES_* \leftarrow Select\_Best\_Elite(ES)$;
(25) $x \leftarrow$ *Single-point_Local_Search* (ES_*) //*carry out random single-point local search*
(26) *Evaluate fitness for each individual x;*
(27) $ES \leftarrow Elite\_Set(x)$;
(28) $t = t + 1$
(29) ***end***
(30) *Output result and plot*

Algorithm 7: HBA-PR.

Table 1: Comparison of results for Augerat et al. Set A, E, and P instances.

Instance	Capacity	Tightness	I_BKS	R_BSK	CWS	SR-GCWS	CS-GRASP	HBA-PR
A-n33-k5	100	0.89	661	662.76	712.05	662.11	662.1452	662.1101
A-n33-k6	100	0.9	742	742.83	776.26	742.69	743.5785	742.6933
A-n37-k5	100	0.81	669	672.59	707.26	672.47	672.4652	672.4652
A-n39-k6	100	0.88	831	833.20	863.08	833.20	—	835.2518
E-n23-k3	4500	0.75	569	—	—	—	569.7461	568.5625
E-n22-k4	6000	0.94	375	375.28	388.77	375.28	—	375.2798
E-n33-k4	8000	0.92	839	838.72	843.1	837.67	—	837.9253
E-n51-k5	160	0.97	521	524.94	584.64	524.61	—	524.6111
P-n19-k2	160	0.97	212	212.66	237.90	212.66	212.6569	212.6569
P-n20-k2	160	0.97	216	217.42	234.00	217.42	217.4156	217.4156
P-n22-k2	160	0.96	216	217.85	239.50	217.85	217.8522	217.8522
P-n51-k10	80	0.97	741	742.48	790.97	741.50	—	743.2648

The ordinate normalized fitness (log) in Figures 8–10 is logarithm of normalized fitness; the aim is to show the convergence curves clearly. The normalization formula of fitness is $(\text{fit} - \text{fit}_*)/(\text{fit}^* - \text{fit}_*)$, where fit_* is the best-known solution, and fit^* is the initial fitness.

Figure 7 represents the relative error of test case *A_n33_k5*, *E_n23_k3*, and *P_n19_k2* after 10 times independent running, which shows the sensitivity of parameter *ps*. From the three test cases, the parameter *ps* can be determined. For *A_n33_k5* and *P_n19_k2*, the performance of HBA-PR is better when $ps = 30$. For *E_n23_k3*, that *ps* is equal to 50 is best, but the performance is good as well, while $ps = 30$. From Figures 8–10, the information provided by population is sufficient when *ps* is greater than 20; however, the convergence rate is better when $ps = 30$ among the three instances. Considering the tradeoff between the stability of algorithm and the rate of

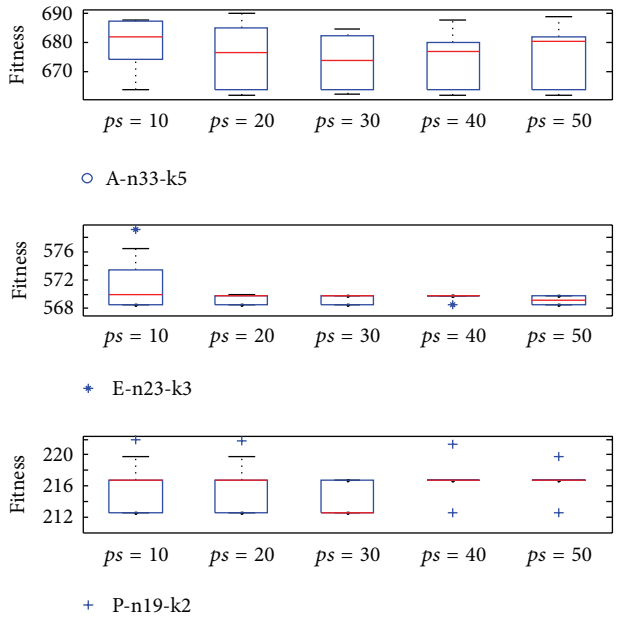


Figure 7: Box-and-whisker diagram of selected three instances.

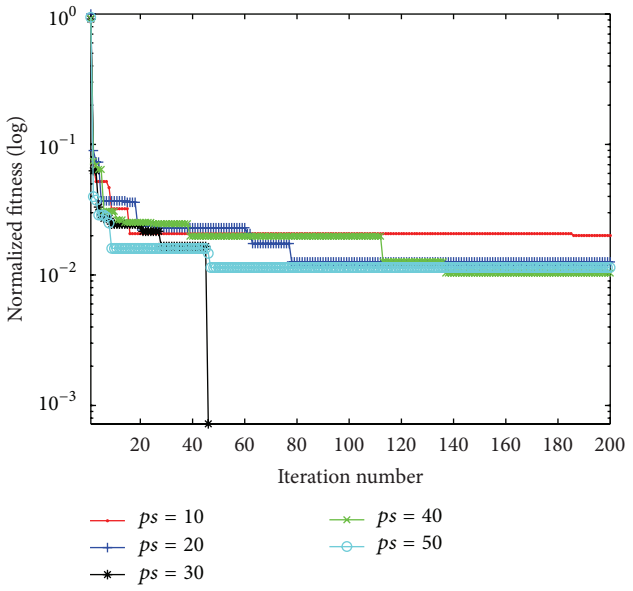


Figure 8: Convergence curve for *A_n33_k5*.

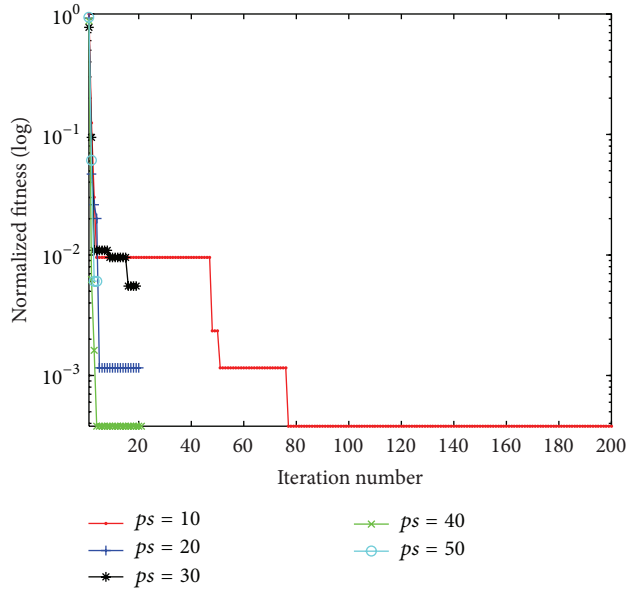


Figure 9: Convergence curve for *E_n23_k3*.

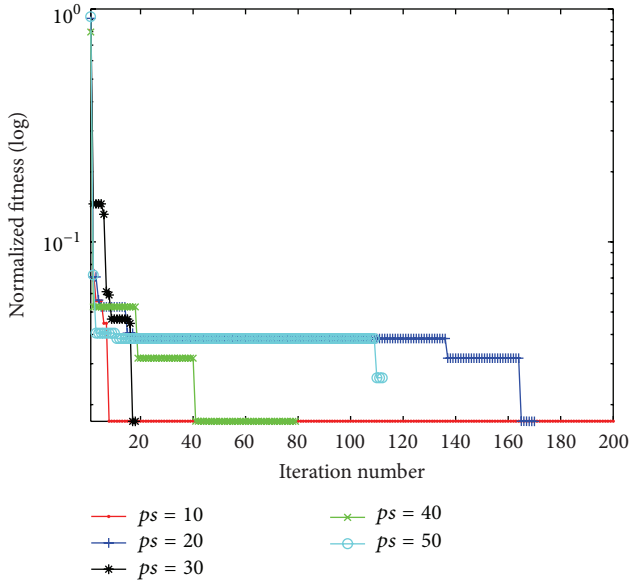


Figure 10: Convergence curve for *P_n19_k2*.

convergence, the parameter ps takes a compromise values, $ps = 30$.

4.2. Comparisons of Simulation Results. In order to show the effective of HBA-PR, we carry out a simulation to compare HBA-PR with other state-of-art algorithms, that is, a parallel version of the classical Clarke and Wright Savings (CWS) heuristic and SR-GCWS (Simulation in Routing via the Generalized Clarke and Wright Savings heuristic) proposed by Juan et al. [9], CS-GRASP proposed by Zheng et al. [10]. Results of these simulations are summarized in Table 1, which contain the following information of each instance: vehicle capacity; tightness (demand/capacity); I_BKS is integral best-known solution (BKS) or "optimal" value according to the web http://www.branchandcut.org/; R_BSK is verifies real costs for the best-known solution according to [9]; CWS is the costs associated with the solution given by the parallel version of the CWS heuristic; SR-GCWS is the best solution obtained by SR-GCWS method; HBA-PR is our best solution, where "—" represents no records in the literature.

From the simulationresults obtained by testing HBA-PR, it demonstrates that using the proposed HBA-PR to solve the CVRP is effective, and the performance of HBA-PR is prominent. From Table 1, all instances achieved a good quality

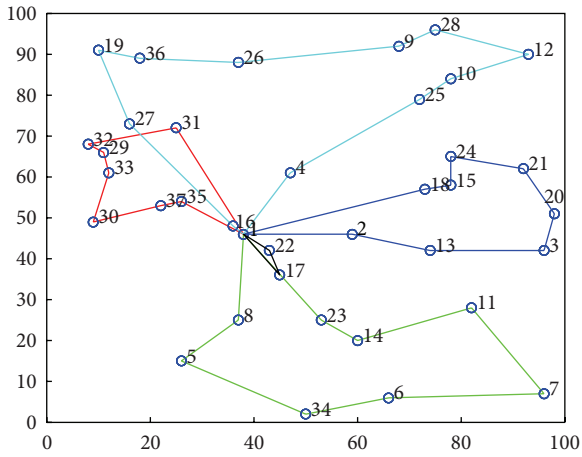


Figure 11: Optimal routes of *A_n37_k5*.

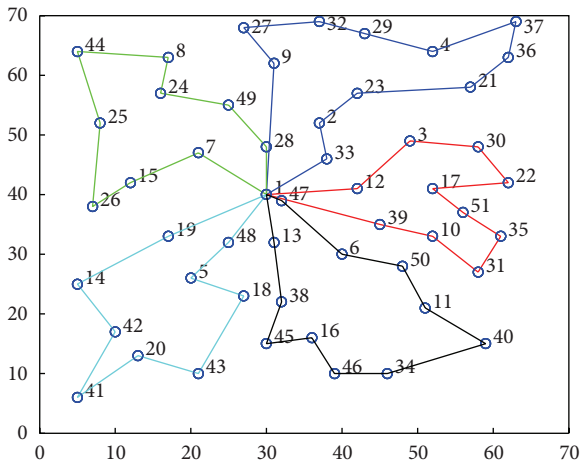


Figure 12: Optimal routes of *E_n51_k5*.

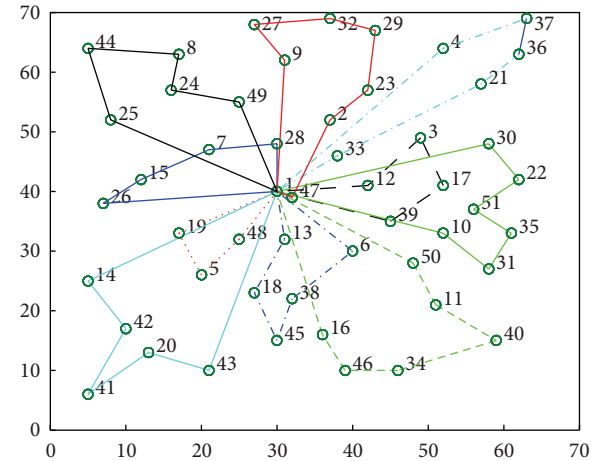


Figure 13: Optimal routes of *P_n51_k10*.

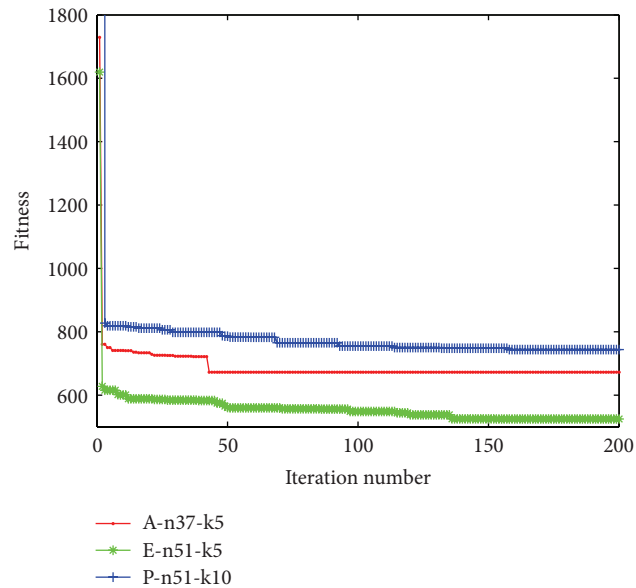


Figure 14: Convergence curve of three instances.

solution, and solutions of 6 instances are better than the best-known solution ("optimal" value). HBA-PR has matched 10 of the 12 best-known solutions except for *P_n51_k10* and *A_n39_k6*, and the average deviation from the real costs best-known solution is 0.057%. HBA-PR outperforms CWS for the 12 instances. Comparing with CS-GRASP and HBA-PR, HBA-PR outperforms CS-GRASP in terms of *A_n33_k5*, *A_n33_k6*, and *E_n23_k3*; the other instances have same results. The SR-GCWS and HBA-PR have similar results, and the gap is very small.

Furthermore, Figure 11 shows the best solution found so far by using our methodology for the *A_n37_k5*.vrp file, where the depot (using 1 instead of 0) is at the center. Analogously, Figures 12 and 13 show the best solution found by HBA-PR for the *E-n51-k5*.vrp file and *P_n51_k10*.vrp file. The convergence curve for some test problems has been shown in Figure 14. From Figure 14, for instance, *A_n37_k5*, it converges to an optimal solution after 43 generation, and it expends about 170 generations, for instance, *E-n51-k5* and *P_n51_k10* when algorithm converges to an optimal solution, which demonstrate that the HBA-PR has a faster convergence rate.

In general, the proposed HBA-PR can produce good solutions when compared with existing heuristics for solving the CVRP, and the convergence rate of HBA-PR is faster. These results seem to indicate that the hybrid bat algorithm with path relinking is an alternative to solve the capacitated vehicle routing problem.

5. Conclusions

The capacitated vehicle routing problem is important in the fields of Operations Research, which is an NP-hard problem. Bat algorithm is a continuous metaheuristics; it cannot be used to solve the CVRP directly. In this paper, a hybrid bat algorithm with path relinking (HBA-PR) for solving the CVRP has been presented. This methodology, which does not require any particular fine tuning of parameters or configuration process, combines the classical greedy randomized

adaptive search procedure (GRASP) with bat algorithm; the path relinking as an intensification strategy to explore local trajectories connecting elite solutions obtained by proposed algorithm; subsequence and single-point local search are effectively integrated into HBA-PR. Results show that our methodology is able to provide fine-quality solutions which can compete with the ones provided by some exact and heuristic approaches. Moreover, because of its simplicity and flexibility, we think that this methodology can easily be adapted to other variants of the vehicle routing problem and even to other combinatorial problems, for example, the vehicle routing problem with time window and traveling salesman problem, which is our further work.

Acknowledgments

This work is supported by National Science Foundation of China under Grant no. 61165015, Key Project of Guangxi Science Foundation under Grant no. 2012GXNSFDA053028, Key Project of Guangxi High School Science Foundation under Grant no. 20121ZD008, the Funded by Open Research Fund Program of Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China under Grant no. IPIU01201100, and the Innovation Project of Guangxi Graduate Education under Grant no. YCSZ2012063.

References

[1] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, pp. 80–91, 1959.

[2] M. Haimovich, A. H. G. Rinnooy Kan, and L. Stougie, "Analysis of heuristics for vehicle routing problems," in *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds., vol. 16, pp. 47–61, North-Holland, Amsterdam, The Netherlands, 1988.

[3] P. Toth and A. Tramontani, "An integer linear programming local search for capacitated vehicle routing problems," in *The Vehicle Routing Problem: Latest Advances and New Challenges*, pp. 275–295, Springer, New York, NY, USA, 2008.

[4] H. Nazif and L. S. Lee, "Optimised crossover genetic algorithm for capacitated vehicle routing problem," *Applied Mathematical Modelling*, vol. 36, no. 5, pp. 2110–2117, 2012.

[5] T. J. Ai and V. Kachitvichyanukul, "Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem," *Computers and Industrial Engineering*, vol. 56, no. 1, pp. 380–387, 2009.

[6] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126–135, 2011.

[7] P. Chen, H.-K. Huang, and X.-Y. Dong, "Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1620–1627, 2010.

[8] A. Yurtkuran and E. Emel, "A new hybrid electromagnetism-like algorithm for capacitated vehicle routing problems," *Expert Systems with Applications*, vol. 37, no. 4, pp. 3427–3433, 2010.

[9] A. A. Juan, J. Faulin, R. Ruiz, B. Barrios, and S. Caballé, "The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing problem," *Applied Soft Computing Journal*, vol. 10, no. 1, pp. 215–224, 2010.

[10] H. Q. Zheng, Y. Q. Zhou, and Q. F. Luo, "A hybrid cuckoo search algorithm-GRASP for vehicle routing problem," *Journal of Convergence Information Technology*, vol. 8, no. 3, 2013.

[11] X.-S. Yang, "A new metaheuristic Bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO)," *Studies in Computational Intelligence*, vol. 284, pp. 65–74, 2010.

[12] A. H. Gandom, X. S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1239–1255, 2013.

[13] X. S. Yang and A. H. Gandom, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, pp. 464–483, 2012.

[14] S. Mishra, K. Shaw, and D. Mishra, "A new meta-heuristic bat inspired classification approach for microarray data," *Procedia Technology*, vol. 4, pp. 802–806, 2012.

[15] J. Xie, Y. Q. Zhou, and H. Chen, "A novel bat algorithm based on differential operator and Lévy-flights trajectory," *Computational Intelligence and Neuroscience*, vol. 2013, Article ID 453812, 13 pages, 2013.

[16] G. Wang, L. Guo, H. Duan, L. Liu, and H. Wang, "A bat algorithm with mutation for UCAV path planning," *The Scientific World Journal*, vol. 2012, Article ID 418946, 15 pages, 2012.

[17] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[18] M. G. C. Resendel and C. C. Ribeiro, "GRASP with path-relinking: recent advances and applications," in *Metaheuristics: Progress as Real Problem Solvers*, pp. 29–63, Springer, 2005.

[19] P. Festa and M. G. C. Resende, "An annotated bibliography of GRASP. I. Algorithms," *International Transactions in Operational Research*, vol. 16, no. 1, pp. 1–24, 2009.

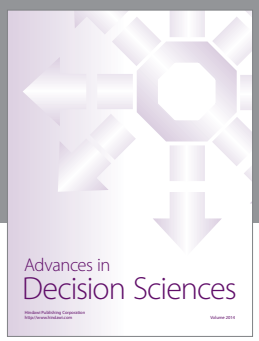
[20] P. Festa and M. G. C. Resende, "An annotated bibliography of GRASP. II. Applications," *International Transactions in Operational Research*, vol. 16, no. 2, pp. 131–172, 2009.

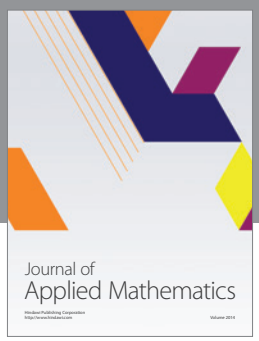
[21] M. Mestria, L. S. Ochi, and S. de Lima Martins, "GRASP with path relinking for the symmetric euclidean clustered traveling salesman problem," *Computers and Operations Research*, 2012.

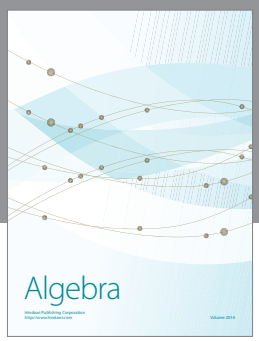
[22] M. G. C. Resende, C. C. Ribeiro, F. Glover, and R. Mart, "Scatter search and path-relinking: fundamentals, advances, and applications," in *Handbook of Metaheuristics*, M. Gendreau and J. Y. Potvin, Eds., pp. 87–107, Springer, Boston, Mass, USA, 2010.

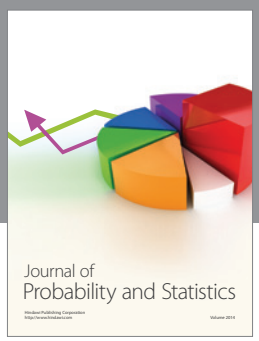
[23] F. Glover, "Tabu search and adaptive memory programming-advances, applications and challenges," in *Interfaces in Computer Science and Operations Research*, R. S. Barr, R. V. Helgason, and J. L. Kennington, Eds., pp. 1–75, Kluwer, 1996.

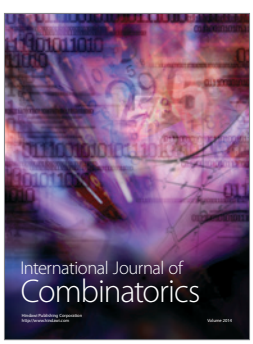
[24] M. Laguna and R. Martí, "GRASP and path relinking for 2-layer straight line crossing minimization," *Journal on Computing*, vol. 11, no. 1, pp. 44–52, 1999.

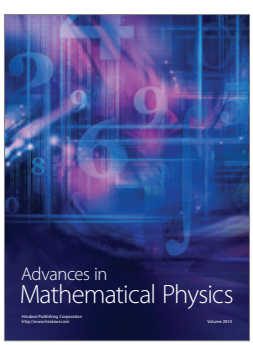


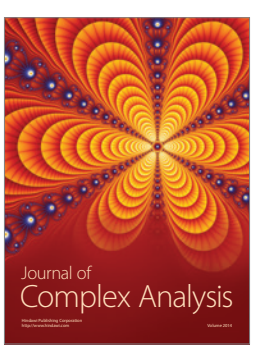


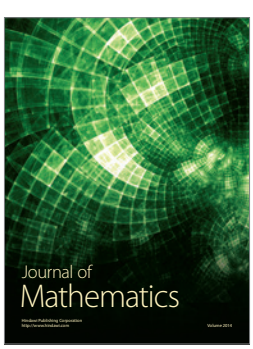


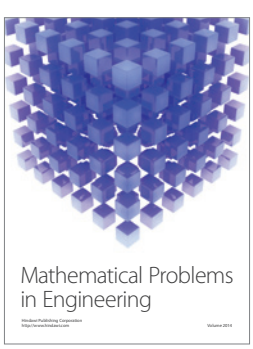


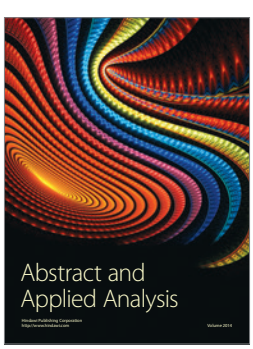


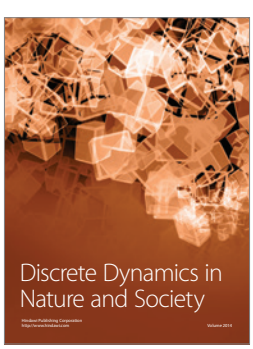


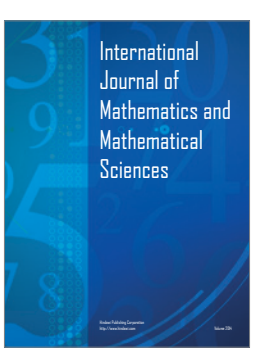












Hindawi

Submit your manuscripts at
http://www.hindawi.com