25. Proceeding IBICA - A Fast Bee Colony Optimization

by Arta Moro Sundjaja

Submission date: 07-Mar-2018 01:59PM (UTC+0700)
Submission ID: 623588408
File name: 25._Proceeding_IBICA_-_A_Fast_Bee_Colony_Optimization.pdf (277.18K)
Word count: 5504
Character count: 26006

2012 Third International Conference on Innovations in Bio-Inspired Computing and Applications

A Fast Bee Colony Optimization For Travelling Salesman Problem

Abba Suganda Girsang
Institute of Computer
and Communication Engineering,
Department of Electrical Engineering,
National Cheng Kung University,
Tainan 70101, Taiwan, R.O.C
Email: gandagirsang@yahoo.com

Chun-Wei Tsai
Department Geoinformatics,
Chia Nan University of Pharmacy Science,
Tainan 71710, Taiwan, R.O.C
Email:cwtsai87@gmail.com

Chu-Sing Yang
Institute of Computer
and Communication Engineering,
Department of Electrical Engineering,
National Cheng Kung University,
Tainan 70101, Taiwan, R.O.C
Email:csyang@ee.ncku.edu.tw

***Abstract*—This paper presents a modified bee colony optimization (BCO) by pattern reduction to reduce the computation time, called BCOPR. Although BCO was robustness optimization, but likes the other algorithm for solving optimization problem, BCO has many reduncation computations on its convergence process, as a consequence, it will more computation time. Two operators are developed to BCOPR in this paper. The first one BCOPR1, is used to cut down the computation time by avoiding performing the same process on the preferred edge. On the second operator, BCOPR2, a bee is possible to duplicate the best previous-iteration tour if her first stage tour is the same with part of the best previous-iteration tour. In addition, likes BCO original, BCOPR is also use local search to enhance the quality solution. To evaluate the performance of the proposed algorithm, BCOPR uses some various benchmarks of TSP. Our experimental results show BCOPR reduce computation time as well as achieve good solution.**

***Keywords*-Bee colony optimization, Pattern Reduction, Travelling Salesman Problem, Computation Time**

I. Introduction

Since a few decades until now, research of travelling salesman problem (TSP) is still interesting for researches. It can be seen that many methods are still developed to solve that problem. Some methods might be new, modified, or embedded with the other methods. The aim of TSP is to find the shortest path of a sales who visits every city exactly once and finally return to start one. In single-solution-based metaheuristic algorithms, there are such as simulated annealing [1], tabu search [2][3][4]. In swarm or population intelligence, there are genetic algorithm [5], ant colony optimization [6][7][8], particle swarm optimization [9][10], bee colony optimization [12][13][14][15][16].

Now days, two research groups are inspired by the behavior of bee, and use bee colony optimization (BCO) for their proposed methods. One BCO was introduced by Lucic and Teodrovic [12][13][14], the other one was introduced by Wong et al. [15][16]. Though they have the different algorithm to describe BCO, but both of their algorithms were naturally inspired by the intelligent foraging behavior of honey bees. This paper follows the first BCO model which is introduced by Lucic and Teodrovic. Swarm behavior bee also encourages generate the other algorithm such as bee system, artificial bee colony (ABC), marriage in honey bee optimization (MBO), etc. Besides solving TSP, swarm bee also uses to solve vehicle routing problem [17], numerical optimization [18], data mining [19], job shop scheduling [20], clustering [21], and so forth.

According to our observation, there are some redundant computation on BCO before converging the result. BCOPR offers an algorithm to cut the redundant computation time in order to get efficient bee colony optimization. In BCO, each bee moves from one node (nectar) to other nodes with a certain probability. The probability for a bee to move from one node i to another node j, is affected by number of bees that visited the edge. The higher number bee visits one edge, the higher probability for a bee to choose that edge. So, it is possible one edge to be a preferred edge, because in most iterations, the edge always be visited. However, a bee still considers the other edges by determine all possibility. In fact, at last, the preferred edge is highly probably chosen.

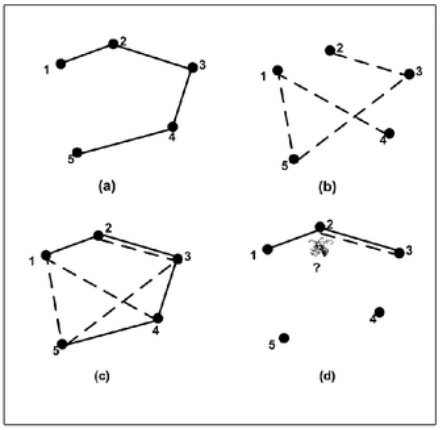

Fig. 1. (a) The path constructed by bee B1, (b) The path constructed by bee B2, (c) The path that constructed by bee B1 and B2, and (d) Edge *e23* is indicated as preferred edge.

As illustrated on Fig.1 (a) and Fig.1 (b), bee B1 and B2 travel on nodes 1-2-3-4-5 , 2-3-5-1-4 respectively. It shows on Fig.1 (c) that edge *e23* is travelled both of bee. For edge

978-0-7695-4837-1/12 $26.00 © 2012 IEEE
DOI 10.1109/IBICA.2012.44

e23 is travelled more than the other, edge *e23* is indicated as preferred edge that might be part of solution. Nevertheless, in next step, if one bee is on position node 2, a bee has to repeat to compute the probability of all nodes. Though at last, it can be predicted that a bee chooses node 3. So, it will be efficient if an edge, indicated as a preferred edge, is chosen automatically without computing the other probability edge. On Fig.1 (d), when a bee on position node 2, she is expected to move automatically to node 3, because edge *e23* is indicated as preferred edge than the others (*e21*, *e24*, and *e25*).

The redundant process also occurs on performing stage-tour. In each iterations, the best of bee's tour will be kept. In our investigation, in many times, if the bee performs one stage sequence (sub-tour) is same with part sequence of the best previous-iteration tour, bee tends to follow the retain of the best previous-iteration tour. In other words, the bee copies the sequence of the best previous-iteration tour. However, the bee has to travel and also takes the process. It can be indicated as a redundant process that also takes computation time.

The remainder of the paper is organized as follow. The Related work is described on section II. Section III provides a detailed description of the proposed algorithm. Experiment results is presented in Section IV. Conclusion is given in Section V.

II. Related Works

Bee colony optimization (BCO) [12][13][14][15][16] is designed to create the multi agent system (colony of artificial bees) to solve the combinatorial optimization problems. Application of BCO for TSP aims to iteratively find the shortest tour. On BCO method, a bee starts as an unemployed forager that has no knowledge about the food (nectar). Bees start explore from one nectar to the others nectars, visit one stage, before back to their hive. This step is called *forward pass (FP)*. One tour consists number of stages which is a collection of a certain number of nectars. On TSP model, a nectar can be represented as a city, while stage is a part of complete tour or sub-solution. In [13] Teodrovic described, probability of bees chosen city *j* from *i* to be visited by the *k*-th bee during stage *u+1* and iteration *z* defined Eq.1 as follow :

$$P_{ij}^{k}(u+1,z)=\begin{cases}\dfrac{e^{-ad_{ij}\sum_{h}^{z-1}\tilde{n}_{ij}(h)}}{\sum_{l\in N_k(u,z)} e^{-ad_{il}\sum_{h}^{z-1}\tilde{n}_{il}(h)}}, \\ \text{i=g}_k(u,z), j\in N_k(u,z), \forall k,u,z \\ 0 \qquad \text{Otherwise,}\end{cases} \tag{1}$$

where :
h=max(*z-b,1*),
i,j = node indexes (*i,j=1,2,..,|N|*),
d_{ij} = length of link (*i,j*),
k = bee index (k = 1,2,..., B),
B = the total number of bees in the hive,
z = iteration index (z = 1, 2,..., M),
M = maximum number of iteration,
s = number of nodes visited by every artificial bee during one stage,
u = stage index (*u*=1,2,...,$\lceil$ |(*N*-1)| /*s* $\rceil$),
n_{il}(h) = total number of bees that visited link (*i*,l) in *h*-th iteration,
b = memory length,
$g_k(u,z)$ = last node that bee *k* visits at the end of stage *u* in iteration *z*,
$N_k(u,z)$ = set of unvisited nodes for bee *k* at stage *u* in iteration *z* (in one stage bee will visits *s* nodes; we have $|N_k(u, z)| = |N| - us$),
a = input parameter given by analyst.

From Eq.1, the greater distance between city *i* and *j* leads to the lower probability to move from city *i* to *j*. In other hand, the greater number of iterations *z* is, the higher the influence of distance. It explains the addition of iteration makes a bee has less freedom in the exploration solution. As a notion, not all bees start foraging in the first stage. Some of them start on the second stage, third stage, etc. Once a bee is on, she will be active until the end of iteration. During the tour, bees choose each cities is based on probability.

After performing one stage, the bees return to the hive in order to unload the nectar and store it then. It is called backward pass (BP). On BP, the bees are divided into 3 types :

Scout : retain the previous stage and continue next stage without interacting the others.
Follower : abandon the food source (previous stage) and will follow the others by see the recruiter's waggle dance.
Recruiter : retain her previous stage and will recruit the follower bee to her way.

To categorize these three of type bees, BCO uses Eq.2 that determines the probability that the bee *k* in next stage (*u+1*) use the same partial tour in stage *u* in iteration *z* as described below :

$$P_k(u+1,z)=e^{-\frac{L_k(u,z)-min_{r\in w(u,z)}(L_r(u,z))}{uz}}, \tag{2}$$

where $L_k(u,z)$ is the length of partial route that is discovered by bee k in stage u in iteration z.

Eq.2 shows that the greater probability is, the more likely for a bee to be a recruiter. Since the probability of best path bee will be 1 ($P_k = 1$), she will be a absolutely recruiter. During a stage, if a bee fails to be a recruiter, she will be a follower. Clearly, by using Eq.2, the shorter tour drives to be a recruiter bee. Follower bee will change her previous tour and follow the recruiter bee's tour. A recruiter bee whose tour is shorter, tends to be followed by one follower bee.

To illustrate the concept of BCO more clearly, Fig.2, Fig.3, and Table I, show a sample sequence step in one iteration. For instance, number of bee *B=6*, number of city *N=10*, number of stage *s=3* with first stage, second stage, third stage whose number cities 4, 4, 3 respectively. Fig.2, Fig.3 show only three bee performance until FP2 on second stage, while Table I

presents all of bee performance as well as the completed tour.

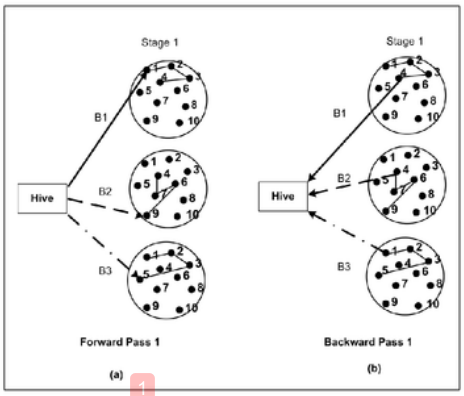

Fig. 2. (a) Forward Pass(FP1),(b) Backward Pass (BP1) on first stage (stage1).

TABLE I
SAMPLE PROCESS BCO ON TSP

Bee	Fw Pass (FP1) stage1	Bc Pass BP1	Fw Pass (FP2) stage2	Bc Pass BP2	Fw Pass (FP3) stage3
B1	1,2,3,4	S	1,2,3,4–5,6,7,8	F	2,8,7,10,6,5,3,9–4,1,2
B2	9,6,7,4	F	5,3,2,1–7,9,6,4	S	5,3,2,1,7,9,6,4–10,8,5
B3	**5,3,2,1**	R	5,3,2,1–8,10,4,7	R	5,3,2,1,8,10,4,7–9,4,5
B4	2,8,7,10	R	2,8,7,10–1,3,9,5	F	**5,3,2,1,8,7,6,10–9,4,5**
B5	8,1,3,7	F	**5,3,2,1–8,7,6,10**	R	5,3,2,1,8,7,6,10–4,9,5
B6	0,0,0,0	F	2,8,7,10–6,5,3,9	R	2,8,7,10,6,5,3,9–4,1,2

Bold means the best tour of all bees in each stage.

At first on first stage (stage1) as shown on Fig.2 (a), each bees explores 4 cities (stage1) in Forward Pass (FP1). Bee B1 travels 1-2-3-4, B2 travels 9-6-7-4, B3 travels 5-3-2-1. A bee chooses one node to other nodes by using Eq.1. The exploration other bees can be seen completely on Table I.

In each backward pass, bees divide into 2 main types scout and the others (recruiter and follower). For instance, probability to be a scout is 10 %. On backward pass 1 (BP1), B1 is chosen as a scout bee (S). Using Eq.2, since B3 gets the best on first stage tour, she will be a absolutely recruiter (R). Using also Eq.2, suppose B4 also chosen as a recruiter (R). B2, B5, are categorized as follower bees (F). B6 is not active on first stage, and start be active on second stage as a follower (F).

On BP1 process, as recruiter bees, B3, B4 perform waggle dance to give information about their tour. The length of waggle dance is determined by the quality of food source. Longer dance means better food source. On TSP problem, quality of food source represents length of tour which the shorter tour is better. The shorter tour has a great probability to choose by the follower bee on the next stage.

In that sample on second stage (stage2), Forward Pass (FP2) can be shown on Fig.3.

B1 : as a scout, she continues her previous stage without considering her friends tour. B1 continues on second stage from node 4 and moves to node 5-6-7-8 in a row by using Eq1.

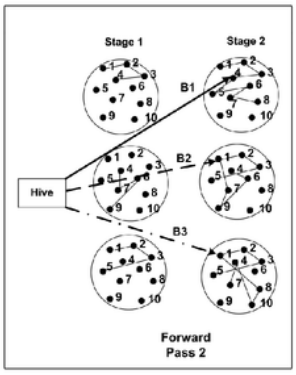

Fig. 3. Forward Pass (FP2) on second stage (stage2)

B2 : as a follower, she changes her previous stage tour and continues by follow the B3's previous stage-tour. Therefore, she continues on second stage from node 1 and moves to nodes 7-9-6-4 in a row by using Eq1.

B3 : as a recruiter, she retains her previous stage and continues on second stage from node 1 and moves to nodes 8-10-4-7 in a row by using Eq1.

B4 : as a recruiter, she retains her previous stage and continues on second stage from node 10 and moves to nodes 1-3-9-5 in a row by using Eq1.

B5 : as a follower, she changes her previous stage tour and continues by follow the B3's previous stage-tour. Therefore, she continues on second stage from node 1 and moves to nodes 8-7-6-10 in a row by using Eq1.

B6 : starts to be active and follower, she follows B4's previous stage-tour and starts on second stage from node 10 and moves to nodes 6-5-3-9 in a row by using Eq1.

After travelling 4 nodes on second stage (totally 8 nodes from first stage), all bees return back to hive to interact each other. Processing of BP1 that has been mentioned above are repeated on BP2 and continue performing FP3 on third stage (or until finish the tour). At last in this sample the best tour is found on bee B4.

III. PROPOSED ALGORITHM

A. The Concept

There are two things that the possibility of having redundant process BCO.

First, solution is built on the path preferred by bee. The edge that is often visited by bee may become part of solution. It is possible after perform some iterations, bees on node i tends to choose same node in next iteration. However, in choosing that node, a bee performs the same calculation process that takes longer time. In more detail, the distance between nodes i and j is computed repeatedly. Process to select destination each nodes with their probability must be done whereas possibility get node j as a destination is very high or limit 1. Therefore, this algorithm offers one role. If edge eij is identified as edge preferred, then once a bee on position node i, she decides

acclamation move to node j without considering the other nodes.

Second, solution (the best tour) of each iteration will be stored. In each iteration, one bee, in the first stage, travel on some cities. It is possible that her first stage-tour will be same as a part tour of the best previous-iteration tour. Furthermore, if the bee continues her next stage-tour until the end, she tends to follow the best previous-iteration tour. However, the bee has to perform the same process such as choosing cities in next stage.

In this paper, we will employed a fast method to enhance the performance of BCO, called pattern reduction (PR). The basic idea of PR is to eliminate redundant computation for making efficient process. PR technique was first implemented to reduce computation time for k-means clustering [22]. PR was also successfully implemented on some optimization problems, such as Genetic Algorithm on TSP, ACO on TSP [23][24]. The main method PR is detect the redundant process and compress it to single process in order to reduce computation time. In this research, PR is embedded into BCO to reduce computation time.

B. The Algorithm BCOPR

As mentioned concept above, BCOPR involves two operator BCOPR1 and BCOPR2 to solve the redundant process on BCO. Both of operators have 2 important steps:

1) Detection Step: On operator 1, BCOPR1 detects the high frequently edge visited. This detection is implemented by setting the number of threshold values δ_0 of each edge. Each bee travels one edge *eij* will increase number visiting of edge *eij*, namely δ_{ij}. If one edge was visited at least number of threshold value ($\delta_{ij} \geq \delta_0$) then the edge will compressed. As a notion, values of δ_0 should be given as fair values. If δ_0 is too small, BCOPR1 tends to generate local optimal, contrary the large δ_0 makes the computation time still be high.

On operator2, BCOPR2 will record bee's tour in the first stage. BCOPR2 will check whether her first stage-tour equals with part of the best previous-iteration tour. If they are equal, a bee will not continue her tour, but only copies the best previous-iteration tour. For instance, if there is one of the best tour until z iteration when the solution is 5-3-2-1-8-7-6-10-9-4-5. In iteration *z+1*, if a bee in the first stage was 2-1-8-7, she need not continue to next stage process. It is because her first stage-tour equals with part of the best previous-iteration tour. BCOPR only detects forward sequence. So, for above example, if a bee in her stage-tour was 7-8-1-2, it is not indicated as a part of of the best previous-iteration tour. In addition, this method only detects the first stage tour. Detecting on the second, third, fourth, etc stage-tour takes longer time, whereas benefit to skip some nodes will be less. Therefore, it is more efficient, if this method only detects the first stage-tour.

2) Compression Step: On operator1, if BCOPR1 detects edge *eij* that already visited several certain times, then one time a bee in node *i*, BCOPR1 automatically chooses node *j* without consider the other node. Therefore, equation of 1 will be modified as follows :

$$P_{ij}^{k}(u+1,z) = \begin{cases} 1 & \text{if } \delta_{ij} \geq \delta_0 \\ \dfrac{e^{-ad_{ij}\frac{z}{\sum_h^{z-1} n_{ij}(h)}}}{\sum_{l \in N_k(u,z)} e^{-ad_{il}\frac{z}{\sum_h^{z-1} n_{il}(h)}}}, & \\ \text{i=g}_k(u,z), j \in N_k(u,z), \forall k,u,z & \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

Based on this equation, if $\delta_{ij} \geq \delta_0$, a bee on node *i*, BCOPR1 forces to choose node *j* and ignores considering the other nodes. δ_0 can be set as constant*number bee. In this research constant will be 5. By choosing automatically one node, BCOPR1 eliminates some process such as determine distance edges as well as probability computation.

```
1. For each iteration
2.    For each stage st
3.       For each bee
            //Perform operator BCOPR1
4.          Bee chooses some cities on stage st by using Eq 3
            //Perform operator BCOPR2
5.          If st=1, check tour of stage 1, whether match with part of
            the best each previous iteration tour.
6.          If match, PR2←Yes, otherwise PR2←No.
7.          If (PR2=Yes) copy best tour previous iteration
8.          If (PR2=No) do
9.             Categorized bee to be a Scout or (Recruiter and Follower)
10.            Determine the length of bees stage tour
11.         End if (PR2=No)
12.      End for bee
13.      On each bees, If (PR2=No)
14.         Categorized bee to be a Rec or Fol by Eq 2
15.         All bees Recruiter and Follower will exchange information,
            bee Follower tends to follow the Recruiter bee tour.
16.      End if (PR2=No)
17.   End for stage
18.   Get the best tour iteration by comparing the all bees tour.
19. End for iteration
20. Get the best tour by comparing the best of all iteration.
21. The best tour performs 2-opt
22. Find the best tour
```

Fig. 4. Algorithm BCOPR

On operator 2, BCOPR2, a bee that her first stage-tour is same with part of the best previous-iteration tour, the bee will stop her process in next stage-tour. Her total tour duplicates the best previous-iteration tour. In other words, she skips the second stage, third stage, etc until last stage tour. So, using this operator, a bee probably produces one complete tour though she only takes one stage-tour. For above example, a bee which travels first stage-tour 2-1-8-7 has a complete tour 5-3-2-1-8-7-6-10-9-4-5 without performing all stages. Algorithm of BCOPR is described on Fig.4.

IV. Experiment Results

Experiments are conducted to evaluate the performance of BCOPR. The performance is compared with original BCO or without PR. To know the performance of each operators, BCOPR1 and BCOPR2 also are tried to compare BCO. All algorithms are executed 10 times on each seven dataset benchmark TSP [11]. The experiments focus 2 aspects : quality solution (Q) and duration computation time (T).

The quality solution is given in terms of best solution (Qbest), and average solution of 10 experiments (Qavg). QBCO, Q1, Q2, Q12 represent the result for quality solution for BCO without PR, only operator BCOPR1, only operator BCOPR2,combining operator BCOPR1 and BCOPR2 respectively either for best case (best) or average case (avg). Term ΔQ for quality solution shows the difference/error to best known solution (BKS).

The duration of computations time is determined on seconds (s) and given in terms of average of 10 experiments (Tavg). TBCO, T1, T2, T12 represent the duration of computation time for BCO without PR, only operator BCOPR1, only operator BCOPR2, combining operator BCOPR1 and BCOPR2 respectively for case (avg). Term ΔT for the duration of computation shows the difference /error to TBCO (avg).

Furthermore, Table II shows setting of various parameters of proposed algorithm. Number stage (s) on table II is set such that number node of each stage is 15 to 25. That number is considered as a fair number. If number stage is too small, frequency of interaction becomes higher and takes more computation time. Contrary if number stage is too high, frequency of interaction becomes lower and makes opportunity to discard the bad stage-tour be lower.

TABLE II
Parameter Setting of BCOPR algorithm

Parameter	Symbol	Set value
number of bee	B	20
paramater	a	1 or 1.25 or 1.5
iteration	M	100
memory length	b	20-50
number stage	s	eil51=3 ; st70=4 ; eil101=7 a280=19; pcb441=25; u724=30 p1002=40
possiblity to be scout		10 %
threshold preferred edge	δ_0	5 *number bee

Table III summarizes the experiment result of BCO original that was introduced by P.Lucic and friends [12][13][14]. P.Lucic and friends use the real type to determine the length tour.

TABLE III
Results by obtained BCO original (with 2-opt heuristic)

Dataset	Optimal	Q-best	ΔQ-best(%)	Q-avg	ΔQ-avg(%)
eil51	428.9	431.1	0.53	433.8	1.14
st70	677.1	678.6	0.22	684.3	1.06
eil101	640.2	642	0.35	665.6	3.97
a280	2586.8	2740.6	5.95	2784.8	7.66

Table IV shows the summarize result BCO that is built by author. It doesn't show the significant difference with the original author. On small dataset (eil51, st70), BCO is able to find the optimal value (best known solution/BKS) while the average achievement worse 1.1-1.8% than the BKS. In other datasets, although not achieving optimal quality, BCO differs only 2.1-7.6% of BKS.

TABLE IV
Results by obtained BCO for 10 different run

Dataset	BKS	QBCO-best	ΔQBCO best(%)	QBCO avg	ΔQBCO-avg(%)	TBCO-avg (s)
eil51	426	426	0	430.7	1.1	12.67
st70	675	675	0	687.2	1.8	29.67
eil101	629	642	2.07	654.9	4.1	74.10
a280	2579	2643	2.5	2690.8	4.3	1265
pcb442	50778	53524	5.4	54069	6.5	5131
u724	41910	43633	4.11	44847	7.0	22081
p1002	259045	273501	5.6	277637	7.2	41169

TABLE V
Results obtained by operator BCOPR1 for 10 different run

Dataset	Q1-best	ΔQ1 best(%)	Q1 avg	ΔQ1 avg(%)	T1 avg (s)	ΔT1 avg (%)
eil51	426	0	431.5	1.29	11.19	13
st70	678	0.44	689.8	2.19	24.9	19
eil101	642	2.07	654.1	3.99	60.36	23
a280	2661	3.17	2723.6	5.61	971	30
pcb442	53045	4.46	54012	6.36	3940	29
u724	44196	5.45	45182	7.8	16907	31
p1002	275362	6.29	278796	7.62	30963	33

TABLE VI
Results obtained by operator BCOPR2 for 10 different run

Dataset	Q2-best	ΔQ2 best(%)	Q2 avg	ΔQ2 avg(%)	T2 avg (s)	ΔT2 avg (%)
eil51	426	0	430.1	0.96	11.13	14
st70	679	0.59	687.6	1.87	25.5	16
eil101	644	2.38	649.4	3.24	64.2	15
a280	2646	2.6	2682.7	4.01	1085	17
pcb442	53731	5.81	54264	6.86	4119	23
u724	43994	4.97	44346	5.81	18854	17
p1002	277420	7.09	278584	7.54	36115	14

TABLE VII
Results obtained by BCOPR (combine operator BCOPR1+BCOPR2) for 10 different run on seven dataset benchmark

Dataset	Q12-best	ΔQ12 best(%)	Q12 avg	ΔQ12 avg(%)	T12 avg (s)	ΔT12 avg (%)
eil51	426	0	430.7	1.1	9.92	28
st70	680	0.74	691.6	2.46	20.9	42
eil101	637	1.27	649.8	3.31	54.2	37
a280	2681	3.95	2715.8	5.30	861	47
pcb442	53532	5.42	54512	7.35	3515	44
u724	42954	2.49	44970	7.30	14800	49
p1002	278694	7.15	280404	8.02	27402	50

Table V shows the performance only operator BCOPR1. It shows that best solution of BCOPR1 (Q1best) is able to achieve nearly the results of BCO. In fact, on eil51, eil101 can match BCO exactly with the difference 0%, 2.07% respectively from BKS. However on that datasets, operator BCOPR1 can reduce computation time 13%, 23%. In general, the quality of degraded BCO between 0-7.2% of the BKS, while the operator BCOPR1 degraded between 0-7.62% of the BKS. Yet, BCOPR1 can reduce computation time 13-33%. The greatest difference in quality occurs in benchmark u724 : Q1best which

differs 5.45% of the BKS, whereas QBCObest differs 4.11%. It only takes different 1.34%. However in this case, BCOPR1 achieves the benefit time until 31%.

Table VI shows the performance only operator BCOPR2. On eil51, BCOPR2 can achieve BKS value as well as BCO's performance. Quality of BCOPR2 is worse 0-7.54% than BKS. In other hand, quality of BCO is worse 0-7.2% than BKS. So, it can be said, BCOPR2 gets a little loss quality from BCO, says 0-1.7%. On average case, operator BCOPR2 outperforms BCO on eil51, eil101, a280, and u724, although differs only below 1%. However, for all of experiments, BCOPR2 can reduce computation time range 14-23%.

Table VII shows the performance BCOPR. BCOPR combines two operators, BCOPR1 and BCOPR2 as shown Fig.4. In some cases, quality of BCOPR outperforms BCOPR1 and BCOPR2, but in other cases, quality of BCOPR1 or BCOPR2 outperforms BCOPR. Nevertheless, BCOPR leads significantly on duration computation time than BCOPR1 and BCOPR2. Compared with the BCO, the quality of BCOPR differs with BKS on range 0-8.02%, while BCO produces a different quality between 0-7.2% of the BKS. BCOPR losses quality on range 0-1.5% compared to BCO where the difference 1.5% occurs on p1002 in best case. On this case, quality of BCOPR, BCO gets the difference 7.15%, 5.6% of the BKS respectively. Yet, BCOPR reduces compuatation time until 50%. On some cases,qaulity of BCOPR equals BCO, such as eil51 on best and average. Even, BCOPR leads BCO on eil101, u724 (best case), and eil101 (average case). From the above results, BCOPR reduces computation time significantly on range 28-50% from duration process of BCO.

V. Conclusion

BCOPR was successful in achieving the goal of decreasing computing time. BCOPR involves two operator BCOPR1 and BCOPR2 to detail process two method pattern reduction. Based on the result of the experiment, each of operator BCOPR were successfully applied on some variants dataset of benchmark TSP. BCOPR1, emphasizes removing the redundant process on the basis of path frequented. It can reduce time on range 13 to 33% . BCOPR2, emphasizes removing the process of the next stage tour. It can reduce time on range 14 to 23%. Merging BCOPR1 and BCOPR2, called BCOPR, reduces time significantly on range 28 up to 50%. Most of the algorithm with pattern reduction decreases a little quality solution, says about 0.1 to 2%, nevertheless reduces computation time significantly, says until 50%. In fact, on some cases, algorithm with pattern reduction has a better solution than the original BCO without pattern reduction. The result shows clearly that many of the computation are redundant on the converge process of BCO. BCOPR succeeses in cutting redundant process thereby reducing computation time significantly.

Acknowledgment

This work was supported in part by the National Science Council, Taiwan, ROC, under Contract No. NSC100-2218-E-041-001-MY2, NSC100-2218-E-041-001-MY2, and NSC101-2221-E-041-012.

References

[1] S. Kirkpatrick; C. D. Gelatt; M. P. "Vecchi Optimization by Simulated Annealing", Science, 220 (4598), pp :671-680, 1983.
[2] F. Glover, Tabu search-Part I, ORSA J. Comput.1, 190-206, 1989.
[3] F. Glover, Tabu search-Part II, ORSA J. Comput. 2, 4-32, 1990.
[4] J. Knox, "Tabu Search Performance on the Symmetric Travelling Salesman Problem", Computers and Ops. Res. 21, pag. 867-876, 1994.
[5] J. V. Potvin, "Genetic Algorithms for the Traveling Salesman Problem", Annals of Operations Research, vol. 63, pp. 339-370, 1996.
[6] M. Dorigo, V. Maniezzo, and A.Colomi "The ant system: Optimization by a colony of cooperating agents", IEEE Trans. Syst, Man, Cybern. B,vol. 26, no. 2, pp. 2941, 1996.
[7] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the travelling salesman problem", IEEE Transactionson Evolutionary Computation, vol. 1, no. 1, pp. 5366, 1997.
[8] T. Stutzle, and H.H. Hoos, "MAX-MIN Ant System", In Proceedings of Future Generation Computer System, 889-914, 2000.
[9] M. Clerc, "Discrete particle swarm optimization illustrated by the traveling salesman problem", http://www.mauriceclerc.net, 2000.
[10] K.P. Wang, L. Huang, C.G. Zhou, W. Pang, "Particle swarm optimization for traveling salesman problem", International Conference on Machine Learning and Cybernetics 3 (2003) 15831585.
[11] TSPLIB (2012) http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/
[12] P. Lucic, D. Teodorovic, "Transportation Modeling: An Artificial Life Approach",Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence, 2002.
[13] P. Lucic, "Modeling Transportation Problems Using Concepts of Swarm Intelligence and Soft Computing, PhD Thesis, Civil Engineering, Faculty of the Virginia Polytechnic Institute and State University, 2002.
[14] D. Teodorovic, P.Lucic, G. Markovic, M. DellOrco, "Bee Colony Optimization: Principles and Applications", 8th Seminar on Neural Network Applications in Electrical Engineering, NEUREL, 2006.
[15] L.P. Wong, M.Y.H. Low, C. S. Chong, "A bee colony optimization algorithm for travelling salesman problem", in Proceedings of Second Asia International Conference on Modelling and Simulation (AMS), pp. 818-823, 2008.
[16] L. P. Wong, M. Y. H. Low, and C. S. Chong, "Bee colony optimization with local search for travelling salesman problem", in Proc. of 6th IEEE International Conference on Industrial Informatics (INDIN).IEEE, pp. 10191025, 2008
[17] P. Lucic, D. Teodorovic,"Vehicle Routing Problem with Uncertain Demand at Nodes: The Bee System and Fuzzy Logic Approach", Fuzzy Sets in Optimization, Editor J.L. Verdegay, Springer-Verlag, Berlin Heidelbelg, pp.67-82, 2003.
[18] B. Basturk, D. Karaboga, "An Artificial Bee Colony (ABC) Algorithm for Numeric Function Optimization", IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA, 2006.
[19] K. Benatchba, L. Admane, M. Koudil, "Using Bees to Solve a Data Mining Problem Expressed as a Max-Sat One", In Proceedings of IWINAC'2005, International Work Conference on the Interplay between Natural and Artificial Computation, Canary Islands, Spain, pp. 212-220, 2005.
[20] C.S. Chong, M.Y.H. Low, A.I. Sivakumar, K.L. Gay, "A Bee Colony Optimization Algorithm to Job Shop Scheduling", Proceedings of the 37th Winter Simulation, Monterey, California, pp. 1954-1961, 2006.
[21] M. Fathian, B. Amiri, A. Maroosi, "Application of Honey-Bee Mating Optimization Algorithm on Clustering", Applied Mathematics and Computation, 190(2), pp.1502-1513, 2007.
[22] Chun-Wei Tsai, Chu-Sing Yang, Ming-Chao Chiang, "A Novel Pattern Reduction Algorithm for k-means Based Clustering," 2007 IEEE International Conference on Systems, Man and Cybernetics, pp.504-509, 2007.
[23] Shing-Pang Tseng, C.W. Tsai, M.C. Chiang, C. S. Yang, "Fast genetic algorithm based on pattern reduction",in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp.214219, 2008.
[24] Shing-Pang Tseng, C.W. Tsai, M.C. Chiang, C. S. Yang, "A fast ant colony optimization for traveling salesman problem", 2010 IEEE Congress on Evolutionary Computation, 2010.

25. Proceeding IBICA - A Fast Bee Colony Optimization

ORIGINALITY REPORT

97%	11%	97%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Abba Suganda Girsang, Chun-Wei Tsai, Chu-Sing Yang. "A Fast Bee Colony Optimization for Traveling Salesman Problem", 2012 Third International Conference on Innovations in Bio-Inspired Computing and Applications, 2012 Publication	97%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		