International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:7, 2012

A Fast Block-based Evolutional Algorithm for Combinatorial Problems

Huang, Wei-Hsiu Chang, Pei-Chann, Wang, Lien-Chun

***Abstract*—**The problems with high complexity had been the challenge in combinatorial problems. Due to the none-determined and polynomial characteristics, these problems usually face to unreasonable searching budget. Hence combinatorial optimizations attracted numerous researchers to develop better algorithms. In recent academic researches, most focus on developing to enhance the conventional evolutional algorithms and facilitate the local heuristics, such as VNS, 2-opt and 3-opt. Despite the performances of the introduction of the local strategies are significant, however, these improvement cannot improve the performance for solving the different problems. Therefore, this research proposes a meta-heuristic evolutional algorithm which can be applied to solve several types of problems. The performance validates BBEA has the ability to solve the problems even without the design of local strategies.

***Keywords*—**Combinatorial problems, Artificial Chromosomes, Blocks Mining, Block Recombination

I. Introduction

IN real world, discrete optimization is regarded as combinatorial optimization for decreasing the cost by the optimal solution from numerous feasible solutions. Discrete optimization is a branch of optimization in applied mathematics and computer science. Discrete optimization contains two main fields. One is combinatorial optimization, another one is integer programming. Combinatorial optimization is a topic in theoretical computer science and applied mathematics that consists of finding the least-cost solution to a mathematical problem in which each solution is associated with a numerical cost. For decades, numerous algorithms were proposed in solving combinatorial optimization problems (COPs), for one reason more concerned on this domain is hard to reach a reasonable solution, which with steady and better quality.

In our earlier researches (Chang et al. 2008 [1], 2008 [2], 2010 [3]), ACGA has been very successful in injecting ACs into the evolutionary process of GA to speed up the convergence. However, the solution quality still can be further improved when compared with other approaches. Here we proposed a block-based AC generation approach which is for better combination and competitive AC. However, due to the computation of the probability matrix, we adopt the Estimation of Distribution Algorithms (EDAs) for the conditional probabilities.

Huang, Wei-Hsiu is with the Department of Information Management, Yuan Ze University Taoyuan 32026, Taiwan, R.O.C.
Chang, Pei-Chann is with the Department of Information Management, Yuan Ze University Taoyuan 32026, Taiwan, R.O.C.
(Corresponding Author's E-mail: iepchang@saturn.yzu.edu.tw).
Wang, Lien-Chun is with the Department of Information Management, Yuan Ze University Taoyuan 32026, Taiwan, R.O.C.

That means, all of the connections of pairs of cities are considered with better probabilities to link. Therefore, the composed probability matrix represents the dominance matrix of the whole cities. This idea of the AC injection is used to improve the slow convergence and escape trapped in local optima.

II. Literature Review

A. Meta-heuristic algorithms

In the research domain of the combinatorial problems, the Traveling Salesman Problem is a classical problem in the area of Operations Research. There are several practical uses for this problem, such as Vehicle Routing [4] and Drilling Problems [5]. TSP has been extensively used as a comparison basis in order to improve different optimization techniques, such as Genetic Algorithms [6], Simulated Annealing [7], Tabu Search [8], Local Search [9], Ant Colony [10], and Neural Networks [11].

On the other hand, common, problem-independent heuristics like simulated annealing (SA) [12] and genetic algorithms (GAs) [13]-[15] deliver poor performance on large TSP instances [16]. They require high execution times for solutions whose quality is often not comparable with those achieved in much less time by their domain-specific local search counterparts. Therefore, a large number of approaches have been developed for solving TSPs. A very promising direction is the genetic algorithm (GA) combining with problem specific operators which is named as a Memetic Algorithm (MA) [17]. MA adopts the strategy of encoding the population and the genetic operations, so as to direct the individuals' heuristic study and searching direction. The technique does not ensure an optimal solution, however it usually gives good approximations in a reasonable amount of time. This paper aims at developing a meta-heuristic approach to improve the efficiency of the conventional algorithm via the injection of AC and recombine the blocks to speed up the process of the evolution; meanwhile, the efficiency can be improved during the later experimental results.

B. Estimation of Distribution Algorithms

In EDAs, the problem specific interactions among the variables of individuals are taken into consideration. In Evolutionary Computations the interactions are kept implicitly in mind whereas in EDAs the interrelations are expressed explicitly through the joint probability distribution associated with the individuals of variables selected at each generation. The probability distribution is calculated from a database of

selected individuals of previous generation. Then sampling this probability distribution generates offspring. Neither crossover nor mutation has been applied in EDAs. But the estimation of the joint probability distribution associated with the database containing the selected individuals is not an easy task. The flowchart of EDA is shown in the Figure 1.

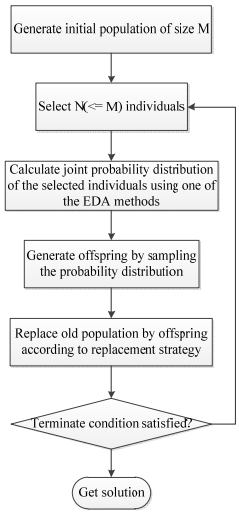

Fig. 1 EDA flowchart

III. Methodology

This paper aims at to develop an effective meta-heuristic algorithm which contains two phases. The first phase is to consider the mining approach which is adopted to find out the effective blocks. Meanwhile, these blocks will recombine via the competition and keep the blocks with high advantage. The second phase is to develop the approach of the AC composition. In this paper, we take advantage of NN and the designed approach in this paper to combine the competitive blocks and those remaining cities to combine the new AC with higher competition. At the same time, BBEA will supervise the AC and the solutions last iteration to select a proper partial linkage to reassemble. Then the reassembled solutions will be considered to select the solutions with satisfactory efficiency to update the probability matrix. The process will be repeated until the defined iteration number is met. The systematic diagram is given as Figure 2.

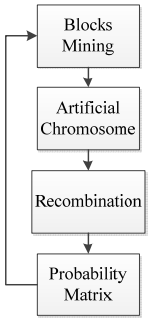

Fig. 2 Research systematic diagram

A. BBEA Framework

The process of this research is as the following description. The initial solutions are generated randomly. Then the fitness of each of the solutions is computed. We will select the solution with the higher competition to update the probability matrix according their performance. During the evolutional process, the occasion of injecting AC is decided according to the convergence angle and iteration number. Whenever the AC is decided to inject, the injected AC will be selected and combine the blocks according the probability matrix. The combined blocks will continue to be adopted to assemble the new AC and the blocks in each AC will be exchanged. The EHBSA here is adopted to divide and be optimal. Finally, the new population will be generated for the next evolution until the stopping criteria are met. The flowchart is represented as Figure 3.

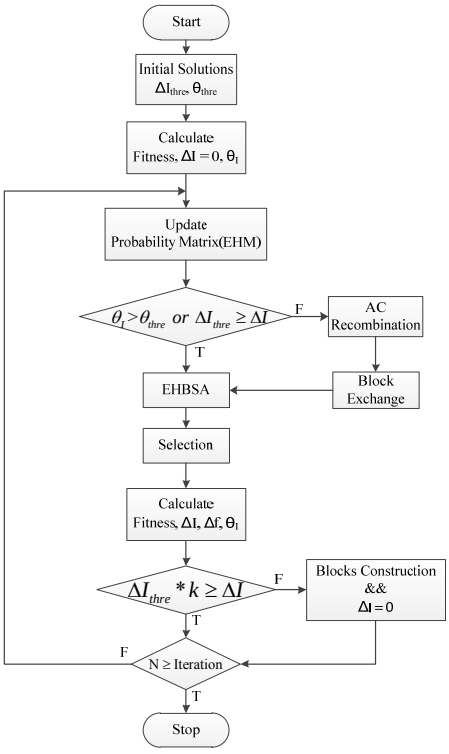

Fig. 3 BBEA flowchart

For further explanation of the flow, the pseudo-code is given as follows. The number of the initial solutions is assigned as 100. The next step is to select the top 20 performance to update the probability matrix. Since the matrix is built, the criterion is given to decide the occasion of injecting AC. If the criterion judge not to inject, every solution will be reassembled, then select the solutions with the top 100 performance via the binary selection for the next evolution. If it decide to inject, then the solutions will be stochastically selected, then if the injections do not update the best known solution, it will use the Nearest Neighbor (NN) to construct the AC and exchange the blocks until all the solutions are been proceed.

$\lambda[\ ]$: AC matrix

$\mu[\ ]$: Generations matrix

1. $\mu[\mu_1, \mu_2, ..., \ _m]\mu \leftarrow$ Random Initialization
2. Calculate fitness
3. **while** The stopping criterion is not met **do**
4. Update probability matrix
5. **while** The optima has not been update for *i* iteration **do**
6. Blocks Re-construction by probability matrix
7. Blocks competition by Conflict Block
8. End **while**
9. **while** The convergence angle $\theta \le \theta_{thre}$ or optima has not been update for *j* iteration **do**
10. $\lambda[\lambda_1, \lambda_2, ..., \ _n]\lambda \leftarrow$ AC re-production based on Block
11. $\lambda[\lambda_1', \lambda_2', ..., \ _n']\lambda \leftarrow$ Block interexchange in $\lambda[\lambda_1, \lambda_2, ..., \ _n]\lambda$
12. End **while**
13. $\lambda[\lambda_1'', \lambda_2'', ..., \ _n'']\lambda, \mu[\mu_1', \mu_1', ..., \ _m']\mu \leftarrow \lambda[\ ], \mu[\ ]$ partial solns. recombination by EHBSA
14. current $\mu[\mu_1, \mu_2, ..., \ _m]\mu \leftarrow$ Selection by $\lambda[\ ], \mu[\ ]$
15. Fitness Evaluation
16. End **while**
17. **End**

1. A Gene Linkage Probability Matrix

A Gene Linkage Probability Matrix is applied to identify the strength between city to city by the probability updating strategy. In the beginning, a Gene Linkage Probability matrix is initialized by selecting chromosomes μ_i with best fitness from the population in the first generation as shown in Figure 4. The initial probability matrix for the index to evaluate the probability from $city_i$ to $city_j$ is represented as P_{ij}. The P_{ij} is calculated via the specific probability P_{ij} among the whole routes.

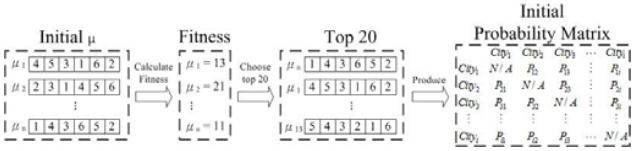

Fig. 4 Initial Gene Linkage Probability Matrix

Later on, the probability matrix is updated by population stored in the μ archive. The population is composited by the chromosomes with better performance which are evaluated by the EDA. For example, in this research, the population is assigned as 100. From the end of evolution at the last iteration, the population contains two sources, which are the solutions of the current and last iterations. The number of the population is larger than what we really need.

Therefore, this research takes advantage of the strategy of EDA to filter the population into 100 to decide the population of the next iteration. Next, we select the top 20 best fit chromosomes in μ from the archive and compute P_{ij} of each pair of cities. Finally, the probability matrix will be updated by these 20 chromosomes as mentioned in the initial probability matrix. Due to the composition of the matrix is updated by the accumulation of the probability to maintain the competitive approach. The probability matrix is so-called Accumulative Probability Matrix which is represented as Figure 5.

Fig. 5 New Gene Linkage Probability Matrix

2. A Block Mining Procedure

A block mining procedure is applying the Gene Linkage Probability Matrix to extract the blocks from the set of high fit chromosomes. It is a process of linkage learning which is applied to discover the hidden knowledge within the dependent variables. The block consists of a series of genes linked to each other continuously. To mine the blocks from the set of high fit chromosomes, two methods can be applied: static block size, on which equally sized blocks are created or dynamic block size where are created blocks with random sizes. In this research, we will focus on static block size.

A static block with size K can be generated according to the following procedures:

1. According to the gene linkage probability matrix, a city is picked randomly and then the *K* connected cities with highest probabilities will be branched.
2. The probability for each block will be calculated from city to city among these *KK*-1 combinations.
3. The one with the shortest distance will be saved into the block archive.
4. The procedures will be repeated again until a pre-defined number of blocks is met, i.e., M, are identified. If any block with a city overlaps with the blocks previous generated, it will be abandoned.

For example, city 38 is selected randomly and the next K, i.e., 5 here, cities with the probability in non-descending orders will be 9, 33, 47, 5, and 8. Again, for each city branched it can be further expanded in the next level. For example, city 9 will be selected and further branched for next 5 cities, until all cities in the same level are branched. The same procedure repeats again and again until it reaches the fourth level. In the final level, there will have 625 cities, i.e., 5x5x5x5 cities, exploded. Finally, block {38, 9, 40, 18} will be selected since it has the shortest distance, i.e., 15. A branching strategy is applied in generating the possible blocks.

The branching rule prescribes how the current problem should be partitioned into sub problems. In this research, we adopt a forward branching strategy that prescribes which sub problems should be expanded next and Best First Search (BFS) which solves the most promising sub problem first, usually the sub problem with the largest value of the probability.

A legal block is a set of cities connecting cities from level one up to level 4. In this research, we branch three times to form a legal block with the length of 4. The reason is to decrease the computational times especially when the problem size is large, i.e., thousand or up to ten thousands of cities. In addition, the block is just like a micro-structure of the chromosome. They can be easily recombined by a good heuristic function R in the recombination process to form a longer block.

The final set of blocks, i.e., puzzles, mined from the probability matrix are stored in the archive as shown in Figure 6.

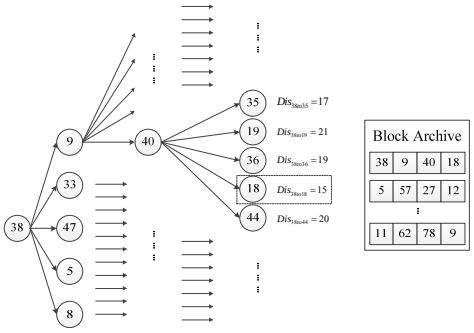

Fig. 6 A set of Blocks mined from the Probability Matrix

3. Blocks Recombination

The common sequences are regarded as having high potential for good substructure or the Blocks because they appear identically in different selected chromosomes which are assumed to be good or highly fit. Therefore they will be retained in the original structures. The next problem is how to compose these blocks with the rest of the cities to form a legal chromosome.

Once the set of blocks are identified and stored in the archive, the rest of cities not in the blocks are also saved together. We name this archive as a Puzzle archive as shown in Figure 7.

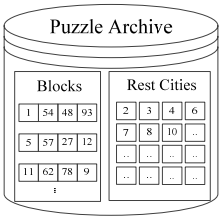

Fig. 7 The composition of Puzzle Archive

The next mission is to reconstruct these blocks and cities to forma legal chromosome. There are numerous methods to recombine these blocks and cities for solving the puzzle to construct a feasible chromosome.

The mission next is how to reconstruct these blocks and cities to form legal chromosomes. There are numerous methods to recombine these blocks and cities for solving the puzzle to construct a feasible chromosome. The Nearest Neighbor (NN) approach has been applied here in this research. The NN method was initially introduced by Skellam [18] where the ratio of expected and observed mean value of the nearest neighbor distances is used to determine if a data set is clustered. A diagram by using the NN to form a legal tour is illustrated in Figure 7.

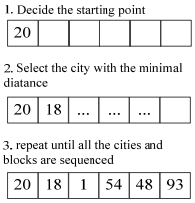

Fig. 7 The composition of Puzzle Archive

The recombined chromosome here is called AC, which is explained in previous section. AC is the key to maintain the population in GA. Since AC is produced via the block recombination and stored in the AC archive, shown as Figure 8, these β chromosomes will be injected to the population mating pool. These ACs with very good infrastructures can play a paramount role in speeding up the convergence process. In addition, these small blocks within the ACs also provide a good chance to come out with a better fit chromosome when crossover or mutated with other chromosomes.

20	19	...	...	...	54
54	33	...	...	...	98
5	9	...	...	...	76
86	44	...	...	...	13

Fig. 8 A new population of Chromosomes

B. Artificial Chromosome

The mechanism of the AC in this research represented as follows. The first step is to compute the average length of all effective routes. From those routes, we stochastically select 10 cities and pick those cities whose routes are shorter than the previous average length. If the selections do not update the optima for three times, NN will be adopted for searching the routes for the next connected cities. These processes will be repeated until all of the remaining cities are assigned. Figure 9 shows the idea of the design of BBEA.

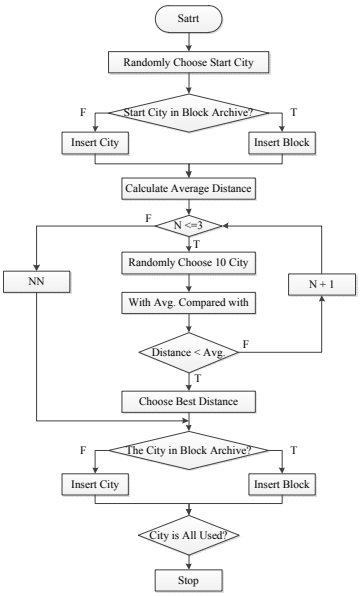

Fig. 9 The mechanism of AC generation

C. Population Recombination

EHM_t described in a marginal edge histogram. It has no explicit graphical structure. EHBSA/WT is intended to make up for this disadvantage by using a template in sampling a new string. In generating each new individual, a template individual is chosen from *P(t)* (normally, randomly). The *n* (*n*>1) cut points are applied to the template randomly. When n cut points are obtained for the template, the template should be divided into n segments. Then, we choose one segment randomly and sample nodes for the segment. Nodes in other *n*-1 segments remain unchanged. We denote this sampling method by EHBSA/WT/*n*. Since average length of one segment is *L/n*, EHBSA/WT/*n* generates new strings which are different at most *L/n* nodes on average from their templates.

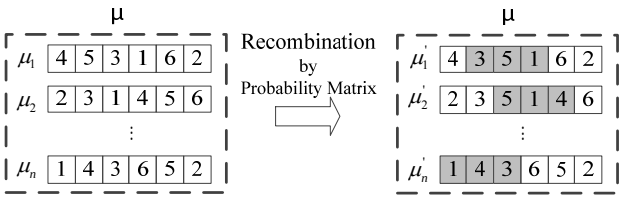

Fig. 10 Population Recombination

IV. Some Experimental Result

A. Environment description

In this section we present the experimental results of the BBEA and compare the performance of BBEA with other algorithms. Each algorithm is executed for 30 times on each instance and the computing hardware consists of Intel Core2 (1.86GHz) and with DDR2 800 (2GB Memory). The programming language is Microsoft Visual C++ 2008 Express. All test cases were chosen from website TSPLIB and with the best known solutions.

All algorithms implemented have a number of parameters to be tuned before they can be applied to a given TSP instance.

B. Comparisons with Meta-heuristics

Due to BBEAs adopts the concept of pheromone update for block mining and the evolving operators of GA, two well-known algorithms, GA and ACO are applied to compare. GA code is from http://www.codeproject.com and ACO is from http://www.aco-metaheuristic.org.The comparisons of BBEAs with GA and ACO are listed in Table I.

TABLE I
COMPARISON WITH OTHER META HEURISTIC

TSP Instances	BBEAs				ACO				GA			
	Mean	Std.	Best	Error Rate	Mean	Std.	Best	Error Rate	Mean	Std.	Best	Error Rate
kroA100	21285	0.0	21285	0.01	26577.2	520.3	26019	24.9	27230.2	24960.7	1866.39	27.95
kroA150	26881.5	161.3	26598	1.4	33860.0	924.6	32470	27.7	34770.2	31504.1	3090.91	31.09
kroA200	30038.0	236.2	29693	2.3	39927.0	601.8	39058	36.0	44577.2	39609.5	4581.08	51.79
pr299	49709.5	329.0	48894	3.2	66697.5	1392.0	63795	38.4	115648.8	99315	10242.23	139.98
pcb442	53616.3	334.0	52742	5.6	68547.5	1734.5	65741	35.0	136426.3	125779	7768.42	168.67
pr1002	279270.5	1009.1	278040	7.8	360559.1	4764.7	353632	39.2	1229439	1193770	25263.57	374.60
pcb1173	62066.1	219.7	61638	9.1	78336.1	1185.0	76086	37.7	282746.4	270264	8952.91	396.99
pr2392	414710.0	1753.3	411449	9.7	539686.2	7023.9	530915	42.8	3418356	3355720	52932.60	804.25
pcb3038	151787.2	327.3	151305	10.2	245724.2	2698.8	242969	78.2	1282795.0	1257040	13282.15	831.63
Avg.				5.5				40.0				314.1

C. Comparisons with the-state-of-the-art

The results to be presented here are the best, mean and standard deviation of the cost (tour length) taken over 30 runs. These two approaches, i.e., RABNET-TSP [19] and SME [20], are selected for comparison with our proposed approach. These two approaches are based on Self-Organized Map (SOM) network with very effective and efficient performances. The comparisons of the experimental results for EDAs, p-ACGA, RABNET, and SME are presented in Table II. The results show that EDAs algorithm outperformed than these two researches. The effectiveness of the proposed approach can be observed in the reduced average error rate for all instances. The results of EDAs show that the algorithm is capable of finding the best solution in most cases even for those instances with larger numbers of cities.

From the results in Table II and III, the efficiencies of BBEA and other compared algorithms all have satisfactory performance. Especially the comparison of computational time, BBEA has significant performance than the other algorithms. However, not for every instance, BBEA's searching abilities are worse than the compared algorithms on four specific instances of all the instances.

TABLE II
COMPARISON WITH THE STATE OF THE ART

TSP Instances	BBEA			p-ACGA			RABNET			SME		
	Mean	Std.	T-test	Mean	Std.	T-test	Mean	Std.	T-test	Mean	Std.	T-test
eil51	428.0	0.0		430.7	4.3	0.3783	437.5	4.2	0.1263	440.6	3.4	0.0329
eil76	546.7	3.1		551.6	6.6	0.4495	556.3	5.3	0.2132	562.3	5.2	0.0970
eil101	649.9	4.1		645.4	9.1	0.2967	648.6	3.9	0.4531	655.6	6.0	0.3443
berlin52	7544.0	0.0		7608.9	115.2	0.3862	7932.5	277.3	0.2375	8025.1	248.8	0.1632
bier127	127465.1	405.2		120904.4	1450.7	0.0182	120886.3	1158.8	0.0050	121733.3	1240.0	0.0156
ch130	6222.7	22.2		6292.2	99.8	0.3295	6282.4	60.2	0.3175	6307.2	63.0	0.2595
ch150	6628.1	45.6		6649.1	83.5	0.4566	6738.4	76.1	0.2630	6751.1	62.2	0.2085
rd100	8031.4	30.0		8061.6	110.3	0.4720	8199.8	80.8	0.1608	8239.4	103.9	0.1646
lin105	14382.0	0.0		14519.7	205.7	0.2819	14400.2	44.0	0.4160	14475.6	118.2	0.3428
lin318	43667.3	189.2		43735.2	333.8	0.4460	43696.8	410.1	0.4866	43922.9	383.3	0.3798
kroA100	21285.0	0.0		21603.3	316.5	0.2890	21522.7	93.3	0.0994	21616.8	164.2	0.1529
kroA150	26881.5	161.3		27485.4	458.5	0.2803	27356.0	327.9	0.2539	27401.3	252.0	0.1887
kroA200	30038.0	236.2		30303.6	444.4	0.4645	30190.3	273.4	0.4145	30415.7	132.9	0.2387
kroB100	22241.7	77.8		22404	157.6	0.2890	22661.5	193.5	0.1538	22622.5	75.3	0.0400
kroB150	26351.1	135.6		26655.8	464.3	0.2773	26631.9	232.9	0.2974	26806.3	250.1	0.2079
kroB200	30348.9	167.2		30469.2	326.0	0.4913	30135.00	276.8	0.3675	30286.5	301.2	0.4629
kroC100	20751.9	6.0		21099.5	295.5	0.2947	20971.2	108.2	0.1525	21149.9	188.0	0.1419
kroD100	21345.6	95.8		21831.8	358.8	0.2318	21697.4	157.0	0.1658	21845.7	154.3	0.0830
kroE100	22168.4	65.7		22420.8	233.9	0.3136	22714.6	260.2	0.1511	22682.5	214.1	0.1230
rat575	7507.7	39.7		7132.2	39.9	0.0118	7115.7	37.5	0.0005	7173.6	39.5	0.0023
rat783	9506.5	41.3		9347.4	44.3	0.2277	9343.8	47.0	0.0950	9387.6	39.4	0.1456
rl1323	291142.5	2369.2		296926	2070.4	0.2068	305314.3	2315.8	0.0178	300899.0	2717.1	0.0865
fl1400	20933.8	112.5		21266	210.8	0.4704	21110.0	163.3	0.3249	20742.6	115.8	0.2727
d1655	68574.0	300.2		65924.6	299.2	0.0066	72113.2	698.6	0.0115	68046.4	379.3	0.2890

TABLE III
COMPARISON WITH THE STATE OF THE ART OF TIME

TSP Instances	BBEA			*p*-ACGA			RABNET-TSP			SME		
	Time(s)	Best	Error Rate	Time(s)	Best	Error Rate	Best	Error Rate		Best	Error Rate	
eil51	1.0	428	0.5	48.1	427	1.1	427	2.7		433	3.4	
eil76	4.0	544	1.6	98.9	545	2.5	541	3.4		552	4.5	
eil101	8.0	642	3.3	166.9	630	2.8	638	3.1		640	4.2	
berlin52	2.0	7544	0.03	40.7	7542	0.9	7542	5.2		7715	6.4	
bier127	13.0	126946	7.8	246.0	119106	2.2	118970	2.2		119840	2.9	
ch130	14.0	6178	1.8	266.0	6221	2.9	6145	2.8		6203	3.2	
ch150	19.0	6549	1.5	348.3	6549	1.9	6602	3.2		6631	3.4	
rd100	8.0	8003	1.5	162.9	7910	1.9	7982	3.7		8028	4.2	
lin105	8.0	14382	0.02	49.0	14379	1.0	14379	0.2		14379	0.7	
lin318	113.7	43269	3.9	420.3	42820	4.1	42834	4.0		43154	4.5	
kroA100	7.7	21285	0.01	45.0	21305	1.5	21333	1.1		21410	1.6	
kroA150	19.0	26598	1.4	96.2	26875	3.6	26678	3.1		26930	3.3	
kroA200	38.0	29693	2.3	167.7	29668	3.2	29600	2.8		30144	3.6	
kroB100	7.0	22141	0.5	45.0	22199	1.2	22343	2.4		22548	2.2	
kroB150	19.2	26192	0.9	96.3	26130	2.0	26264	1.9		26342	2.6	
kroB200	38.9	30047	3.1	167.9	30108	3.5	29637	2.4		29703	2.9	
kroC100	7.8	20750	0.01	45.0	20769	1.7	20915	1.1		20921	1.9	
kroD100	7.8	21294	0.2	45.0	21389	2.5	21374	1.9		21500	2.6	
kroE100	8.0	22106	0.5	45.0	22111	1.6	22395	2.9		22379	2.8	
rat575	2174.3	7428	10.9	1382.4	7065	5.3	7047	5.1		7090	5.9	
rat783	1213.2	9422	8.0	2558.7	9317	6.1	9246	6.1		9316	6.6	
rl1323	2853.4	287234	7.8	7101.2	295671	9.9	300770	13.0		295780	11.4	
fl1400	3185.4	20743	4.0	6531.4	20946	5.7	20851	4.9		20558	3.1	
d1655	4875.7	67917	10.4	8769.4	65059	6.1	70918	16.1		67459	9.5	
Avg.			3.00			3.13		3.97			4.06	

Therefore, BBEA is with fast speed and an effective algorithm in solving most instances. Due to the lack of the execution of RABNET-TSP and SME, the result is not presented here.

V. CONCLUSION

In recent academic researches, most focus on developing to enhance the conventional evolutional algorithms and facilitate the local heuristics, such as VNS, 2-opt and 3-opt. Despite the performances of the introduction of the local strategies are significant, however, these improvement cannot improve the performance for solving the different problems. Therefore, this research proposes a meta-heuristic evolutional algorithm which can be applied to solve several types of problems. The performance validates BBEA has the ability to solve the problems even without the design of local strategies.

In this paper, we proposed a block-based AC generation approach which is for better combination and competitive AC. However, due to the computation of the probability matrix, we adopt the Estimation of Distribution Algorithms (EDAs) for the conditional probabilities. That means, all of the connections of pairs of cities are considered with better probabilities to link. Therefore, the composed probability matrix represents the dominance matrix of the whole cities. This idea of the AC injection is used to improve the slow convergence and escape trapped in local optima. The results from the experiments validate the idea of application of the AC injection and block mining can help the evolutional algorithm to enhance the searching ability.

REFERENCES

[1] P. C. Chang, S. H. Chen, C. Y. Fan, "Mining gene structures to inject artificial chromosomes for genetic algorithm in single machine scheduling problems," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 767-777, 2008.

[2] P. C. Chang, S. H. Chen, C. Y. Fan, C. L. Chan, "Genetic Algorithm with Artificial Chromosomes for Multi-Objective Flow shop Scheduling Problems," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 550-561, 2008.

[3] P. C. Chang, S. H. Chen, C. Y. Fan, "Generating Artificial Chromosomes with Probability Control in Genetic Algorithm for Machine Scheduling Problems," *Annals of Operations Research*, vol. 180, no. 1, pp. 197-211, 2010.

[4] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 345-358, 1992.

[5] G. C. Onwubolu, M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization," *International Journal of Production Research*, vol. 44, no. 3, pp. 473-491, 2004.

[6] M. Affenzeller, S. Wanger, "A Self-Adaptive Model for Selective Pressure Handling within the Theory of Genetic Algorithms," *Lecture Notes in Computer Science*, vol. 2809, no. 1, pp. 384-393, 2003.

[7] M. Budinich, "A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing" *Neural Computing*, vol. 8, 416-424, 1996.

[8] G. Liu, Y. He, Y. Fang, Y. Oiu, "A novel adaptive search strategy of intensification and diversification in tabu search," *Proceedings of Neural Networks and Signal Processing*, Nanjing, China, 2003.

[9] L. Bianchi, J. Knowles, J. Bowler, "Local search for the probabilistic traveling salesman problem: Correction to the 2-p-opt and 1-shift algorithms." *European Journal of Operational Research*, vol. 162, no. 1, pp. 206-219, 2005.

[10] S. C. Chu, J. F. Roddick, J. S. Pan, "Ant colony system with communication strategies," *Information Sciences*, vol. 167, no. 1-4, pp. 63-76, 2004.

[11] K. S. Leung, H. D. Jin, Z. B. Xu, "An expanding self-organizing neural network for the traveling salesman problem," *Neural computing*, vol. 62, pp. 267-292, 2004.

[12] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.
[13] J. Grefenstette, R. Gopal, B. Rosimaita, D. van. Gucht, "Genetic algorithms for the traveling salesman problem," in *Proc. Int. Conf. Genetics Algorithms and Their Applications*, pp.160-168, 1985.
[14] H. C. Braun, "On solving traveling salesman problems by genetic algorithm," *Lecture Notes in Computer Science*, vol. 496, pp. 129-133, 1991.
[15] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed," *Berlin*. Germany: Springer-Verlag, 1996.
[16] J. Fan, D. Li, "An overview of data mining and knowledge discovery," *Journal of Computer Science and Technology*, vol. 13, no. 4, pp. 348-368, 1998.
[17] P. Moscato, M. G. Norman, "A memetic approach for the traveling salesman problem—implementation of a computational ecology for combinatorial optimization on message-passing systems," *International conference on parallel computing and transputer application, IOS Press*, Amsterdam, Holland, pp. 177–186, 1992.
[18] J. G. Skellam, "Studies in statistical ecology," *I. Spatial pattern Biometrica*, vol. 39, pp. 346-362, 1952.
[19] R. Pasti, L. N. de. Castro, "A Neuro-immune network for solving the traveling salesman problem," *Proceedings of International Joint Conference on Neural Networks*, vol. 6, pp. 3760-3766, 2006.
[20] S. Somhom, A. Modares, T. Enkawa, "A self-organizing model for the travelling salesman problem," *Journal of the Operational Research Society*, vol. 48, pp. 919-928, 1997.