The Effectiveness of Hybrid Negative Correlation Learning in Evolutionary Algorithm for Combinatorial Optimization Problems

R. Sirovetnukul[1], P. Chutima[2], W. Wattanapornprom[3], P. Chongstitvatana[4]
[1]Department of Industrial Engineering, Mahidol University, Nakhonpathom, Thailand
[2]Department of Industrial Engineering, Chulalongkorn University, Bangkok, Thailand
[3,4]Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand
([1]egrsr@mahidol.ac.th, [2]cparames@chula.ac.th, [3]warin.w@chula.ac.th and [4]prabhas@chula.ac.th)

Abstract – Most evolutionary algorithms optimize the information from good solutions found in the population. A selection method discards the below-average solutions assuming that they do not contribute any information to update the probabilistic models. This work develops an algorithm called Coincidence algorithm (COIN) which merges negative correlation learning into the optimization process. A knight's tour problem, one of NP-hard multimodal Hamiltonian path problems, is tested with COIN. The results show that COIN is a competitive algorithm in converging to better solutions and maintaining diverse solutions to solve combinatorial optimization problems.

Keywords – **Negative knowledge, Coincidence algorithm, Combinatorial optimization problems**

I. INTRODUCTION

Combinatorial optimization plays an important role for application to real world problems including scheduling, balancing, timetabling and routing problems, where the domains of feasible solutions are discrete. Most combinatorial problems find a natural mapping in permutation spaces where mathematical programming is very difficult to solve and most metaheuristics methods tend to produce infeasible solutions. In addition, it is also well known that decision makers have to deal with more than one objective. Unfortunately, multi-criteria or multi-objective combinatorial optimization (MOCO) has not been widely studied. Many methods use either problem-specific or ad-hoc representation, encodings and operators. There is no standard benchmark for MOCO. Moreover, the performance has not been sufficiently tested on hard problems, leaving the question of scaling up the method to handle large problems.

Recently, there has been a growing interest in developing evolutionary algorithms (EAs) based on probabilistic models called Estimation of Distribution Algorithms (EDAs) [1]. In this scheme, the candidate solutions are generated according to the estimated probabilistic model of the candidate solutions instead of using traditional solution modification operators. The outstanding algorithms used to solve the problem in permutation representation domains are Edge Histogram Based Sampling Algorithms (EHBSAs) and Node Histogram Based Sampling Algorithm (NHBSAs) [2].

Previously, Wattanapornprom et al. proposed a new EDA called Coincidence Algorithm (COIN) [3] which has been successfully applied to many single and multiple objective combinatorial problems such as the traveling salesman problem (TSP) [3], line balancing [3], sequencing [4] and worker allocation [5] respectively. From the experiments, without maintaining the elitists, COIN defeats NSGA-II for all measurement aspects including convergence and spreads to the Pareto-optimum set and ratio of non-dominated solutions. Such experiments leave the research question of why COIN can perform such results.

COIN can be considered like EHBSA and NHBSA. However, the distinctiveness of COIN is that it is implemented as an incremental learning model instead of traditional methods that re-estimate the probabilistic model from an ad-hoc selected population. Such an incremental method allows COIN to learn the information from not-good solutions that are usually truncated and simply ignored. Later, Wattanapornprom and Chongstitvatana compared COIN and EHBSA in multimodal combinatorial problems [6] and discovered that such information, called the negative knowledge, contributes not only in converging to better solutions but also in producing more diverse solutions. For a better understanding, this paper aims to reveal the mechanism behind this phenomenon: why COIN can converge and maintain diverse solutions at the same time.

The difficulties in solving multi-objective problems [7] including multimodality, deception, discontinuity of promising solutions, shape of the Pareto front and constraints. Although Deb [7] has proposed benchmarks for multi-objective problems according to the above difficulties, none of them are an order-based combinatorial problem. Due to the complex geometry of the permutation space, it is hard to design a good benchmark for MOCO which contains all the major difficulties. The main reason is that the priori known solutions and the shape of the optimal Pareto front are not easy to determine. In this paper the knight's tour, a well-known Hamiltonian path problem, is tested. The knight's tour is a special case of the TSP which contains multimodality, deception and constraints.

The remaining sections of this paper are organized as follows. The negative correlation learning is reviewed in Section II. The Edge Histogram Based Sampling Algorithm is reviewed in Section III. The mechanism of combining the negative correlation learning in EA for combinatorial optimization problems is described in Section IV. The empirical studies are revealed in Section V. Finally, Section VI concludes the work.

978-1-4577-0739-1/11/$26.00 ©2011 IEEE

II. NEGATIVE CORRELATION LEARNING

It is common for EAs to be driven by good solutions. Most selection process sampling solutions are based on their fitness. Some selection processes such as truncated selection ranks the solutions and admits only the better ones. The proximate optimality principle (POP) [8] and Building Block Hypothesis (BBH) [9] assume that good solutions share similar structures. The combination of the good building blocks leads to better solutions. On the other hand, not-good solutions also contain the common substructures which lead to undesirable solutions. In most EAs, undesirable solutions are usually ignored by the selection process. COIN uses the negative knowledge in the optimization process to avoid reproducing the not-good solutions.

In 1994, Minsky [10] pointed out that competence often requires one to know what one must do, but it also requires one to know what not to do. The negative knowledge also claimed by Oser and Spychiger [11] in 2005.

In 2006, Parviainen and Eriksson [12] extended Minsky's idea by identifying four features of negative knowing as (i) to know what one does not know (ii) to know what not to do (iii) to unlearn and to bracket the knowledge and (iv) to regard the value of failures and disappointments as emotions, as well as to recognize the creativity that emerges from making mistakes.

In 2008, Gartmeier et al. [13] described the three functions of the negative knowledge as (i) the negative knowledge increases certainty through awareness of possible positive as well as negative outcomes of their actions; (ii) the negative knowledge increases efficiency and (iii) the negative knowledge promotes reflection.

In the field of Evolutionary Algorithms (EA), there are some related works taking the benefit of the negative knowledge. The population-based Incremental Learning (PBIL) [14] algorithm is one of the first EDAs introduced in 1994. PBIL creates a real-value probability vector characterizing high fitness solutions which can also be trained with negative examples. In 2000, Michalski [15] proposed an algorithm called the Learnable Evolution Model (LEM) that applies classifiers to develop a population of solutions. The candidates of a population are divided as the fittest and the less fit ones. Then the characteristics of the good ones are strengthened, but bad ones are avoided. Later in 2003, Llorà and Goldberg [16] proposed an algorithm that combined the Induction of Decision Tree (ID3) and evolutionary algorithm using statistical approaches. In 2004, Miquelez, Bengoetxea, and Larrañaga [17] introduced a new estimation of distribution algorithm based on Bayesian classifiers called Evolutionary Bayesian Classifier-Based Optimization Algorithm (EBCOAs). Finally, Pelikan, Sastry and Goldberg in 2008 [18] propose an incremental version of Bayesian Optimization Algorithms (iBOAs) which update the model using the winners and losers of the tournament selection. However, none of the mentioned works were applied to the optimization in permutation representation.

III. EDGE HISTOGRAM BASED SAMPLING ALGORITHM

Edge Histogram Based Sampling Algorithms (EHBSAs) proposed by Tsutsui in 2002 [2] were designed to solve combinatorial problems and have shown competitive performance in solving many real world applications including traveling salesman problems (TSP), flow shop scheduling problems (FSSP) and capacitated vehicle routing problems (CVRP) [2].

In permutation schemes, the models of solutions can be represented as a graph of nodes connected by edges. EHBSAs utilize Edge Histogram Matrix (EHM) to learn the mutual information of edges contained in the selected solutions and then construct new solutions by sampling from it. The idea of EHBSA is to use the edge recombination (ER) [19] in genetic algorithms with the whole selected population instead of traditional two-parent recombination.

A. Constructing Edge Histogram Matrix

An edge histogram matrix is a matrix that simply stores the summation of edges counted from the selected population plus a bias. Let string of kth individual in population $P(t)$ at generation t represent as $s_k^t = (\pi_k^t(0), \pi_k^t(1), \dots, \pi_k^t(L-1)$. $\pi_k^t(0), \pi_k^t(1), \dots,$ and $\pi_k^t(L-1)$ are the permutation of $(0,1, \dots, L-1)$ where L is the length of the permutation. Edge histogram matrix $EHM^t\ (e_{i,j}^t)(i,j = 0,1,\dots,L-1)$ of population $P(t)$ is symmetrical and consists of L^2 elements as follows [2]:

$$e^t{}_{i,j} = \begin{cases} \sum_{k=1}^{N} (\delta_{i,j}(s_k^t) + \varepsilon \text{ if } i \neq j \\ 0 \text{ if } i = j \end{cases} \quad (1)$$

where N is the population size, $\delta_{i,j}(s_k^t)$ is a delta function defined as

$$\delta_{i,j}(s_k^t) = \begin{cases} 1 \text{ if } \exists h[h \in \{0,1,\dots L-1\} \wedge \\ \pi_k^t(h) = i \wedge \pi_k^t\big((h+1) \bmod L\big) = j] \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

and ε $(\varepsilon > 0)$ is a bias to control pressure in sampling nodes just like those used for adjusting the selection pressure in the proportional selection in GAs. The average number of edges of element $(e_{i,j}^t)$ in EHM^t is $2LN/(L^2 - L) = 2N/(L-1)$. So, ε is determined by a bias ratio B_{ratio} $(B_{ratio} > 0)$ of this average number of edges as

$$\varepsilon = \frac{2N}{L-1} B_{ratio} \quad (3)$$

A smaller of value of B_{ratio} reflects the real distribution of edges in sampling of nodes and a bigger value of B_{ratio} will give a kind of perturbation in the sampling. An example of EHM^t is shown in Fig. 1.

$$\begin{matrix} s_1^t = (0,1,2,3,4) \\ s_2^t = (1,3,4,2,0) \\ s_3^t = (3,4,2,1,0) \\ s_4^t = (4,0,3,1,2) \\ s_5^t = (2,1,3,4,0) \end{matrix} \qquad \begin{bmatrix} 0 & 1.1 & 0.1 & 1.1 & 0.1 \\ 1.1 & 0 & 2.1 & 2.1 & 0.1 \\ 1.1 & 2.1 & 0 & 1.1 & 0.1 \\ 0.1 & 1.1 & 0.1 & 0 & 4.1 \\ 2.1 & 0.1 & 2.1 & 0.1 & 0 \end{bmatrix}$$

$P(t)$ $\qquad$ EHM^t

Fig. 1. An example of asymmetric EHM $(N = 5, L = 5, B_{ratio} = 5)$.

B. Sampling from Edge Histogram Matrix

The sampling algorithm of EHBSA is similar to Ant Colony Optimization. A new individual permutation $c\,[x]$ is generated straightforwardly as follows:

Step 1. Set the position counter $p \leftarrow 0$
Step 2. Obtain first node $c[0]$ randomly from $[0, L-1]$
Step 3. Construct a roulette wheel vector $rw[]$ from matrix as $rw[j] \leftarrow e^{t}{}_{c[p],j} (j = 0,1, \dots, L-1)$
Step 4. Set to 0 previously sampled nodes in $rw[]$ $(rw[c[i]] \leftarrow 0$ for $i = 0, \dots, p)$
Step 5. Sample next node $c[p+1]$ with probability $rw[x] / \sum_{j=0}^{L-1} rw[j]$ using roulette wheel $rw[]$
Step 6. Update the position counter $p \leftarrow p+1$
Step 7. If $p < L-1$, go to **Step 3**
Step 8. Obtain a new individual string $c[]$.

IV. COINCIDENCE ALGORITHM

Coincidence algorithm (COIN) [3] can be considered as an incremental version of EHBSA. However, the idea of COIN is to allow learning from the below average solutions as well as traditional learning from the good solutions. The coincidences (referred to as edge in EHBSA) found in a situation should be able to statistically describe the chance of the situation occurring whether the situation is good or not. Thus the learning of the coincidence found in not-good solutions should be used to avoid producing a not-good solution.

A. Selection of Candidates

COINs utilize the negative knowledge to describe why a solution does not survive in each generation. This helps avoiding candidates containing bad substructures. After the population is evaluated, the whole population is ranked by its fitness. The unique characteristic of COIN is that it selects two groups: better-group candidates and worse-group candidates. This notion of "better or worse" is relative to the average fitness of the population. The exact selection method can be varied according to problems, for example, best 25% and worst 25%.

B. Updating the Joint Probabilities Matrix

Instead of an edge histogram matrix, COIN adapts the first order matrix of Markov Chain Monte Carlo (MCMC) as a data structure. The joint probabilities matrix, *H*, is a square matrix of size $L \times L$, where *L* is the size of the permutation string. The sum of each row $H_{i,j}$ where *i* ranges from 0 to $L-1$ equals 1. It denotes the probability of the occurrence of coincidence *i,j* found in the candidate solution. Each entry of $H_{i,j}$ has a value of 0 to 1. Each entry of diagonal $H_{i,j}$ is 0. In this paper the notation is applied the same way as EHBSA.

COIN starts with an initialization step. The matrix *H* is initially filled with a value 1/(*L*-1) except the diagonal $H_{i,j}$where $i = j$ is set to 0. The update of *H* is separated into two components: reward and punishment. The reward is the increase of $H_{i,j}$ by the occurrence of the pair *i,j* found in the better-group candidates. The incremental step is $k/(L-1)$ where k denotes the learning coefficient, *L* the length of the permutation. The punishment is the decrease of $H_{i,j}$ by the occurrence of the pair *i,j* found in the worse-group candidates with similarly calculation to the reward. The equation is as follows:

$$H_{i,j}^{t+1} = H_{i,j}^{t} + \frac{k}{(L-1)}(r_{i,j} - p_{i,j}) + \frac{k}{(L-1)^2}(\sum_z p_{i,z} - \sum_z p_{i,z}) \qquad (4)$$

where $r_{i,j}$ and $p_{i,j}$are defined similar to $\delta_{i,j}(s_k^t)$. $r_{i,j}$ is obtained from the desired good selected solutions while $p_{i,j}$ is obtained from the undesired bad selected solutions. The last term of $\frac{k}{(L-1)^2}\left(\sum_z p_{i,z} - \sum_z p_{i,z}\right)$ represents the adjustment step for all others, $H_{i,j}^{t}$ $(z \neq i, z \neq j)$ in the opposite direction and keeping the sum of all probabilities in a row constant to 1. An example of updating the joint probability matrix is shown in Fig. 2.

Candidates	Fitness	Action
$s_1^t = (0,1,2,3,4)$	High	Reward
$s_2^t = (1,3,4,2,0)$	Average	Discard
$s_3^t = (3,4,2,1,0)$	Average	Discard
$s_4^t = (4,0,3,1,2)$	Average	Discard
$s_5^t = (2,1,3,4,0)$	Low	Punishment

$P(t_0)$

$$\underset{H^{t_0}}{\begin{bmatrix} 0 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 & 0 \end{bmatrix}} \Rightarrow \underset{H^{t_1}}{\begin{bmatrix} 0 & 0.4 & 0.2 & 0.2 & 0.2 \\ 0.25 & 0 & 0.35 & 0.05 & 0.25 \\ 0.25 & 0.05 & 0 & 0.35 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0 & 0.25 \\ 0.1 & 0.3 & 0.3 & 0.3 & 0 \end{bmatrix}}$$

Fig. 2. An example of joint probability matrix of COIN (*L*=5, *k*=0.2).

Since the joint probability is updated by increasing or decreasing at a constant rate, a joint probability must not become negative. Therefore, it is necessary to maintain the probability value by disallowing the punishment if $H_{i,j}$ will decrease the probability down below 0.

C. Sampling from Joint Probabilities Matrix

The sampling method of COINs is similar to the EHBSA which is already described in Section III.

V. EMPIRICAL STUDY

A. Experimental Methodology

The steps of both EHBSA and COIN are similar to standard EDAs except for **Step 4** of COIN which needs to select two groups of candidates to update the joint probability matrix. The steps are as follows:

Step 1. Initialize the probability model (EHM and joint probability matrix)
Step 2. Sample a population from the probability model
Step 3. Evaluate the population
Step 4. Select the population

Step 5. Update the probability model using the equation described in Section III and IV
Step 6. Repeat **Step 2** until a condition is terminated.

B. Knight's Tour Problem

Knight's Tour is a well-known classical chess puzzle which has been studied over the last thousand years. The objective of the problem is to find a Hamiltonian path in a graph defined by the legal move of a knight on a chess board in which the chess knight has to traverse each square exactly once in Fig. 3. Moreover, there are superior solutions called closed tours which are the solutions that allow the knight to have an extra move to complete the circuit at the starting square. The total number of solutions to the knight's tours problems was researched by McKay [20] and Mordecki [21] using very large scale clustering computers. On a standard 8x8 chess board, McKay calculated the total number of closed tours to be 13, 267, 364, 410, 532 while Mordecki found the open tours to be approximately 1.305×10^{35}.

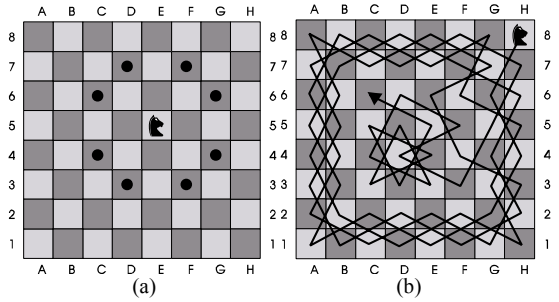

Fig. 3. (a) The legal knight's move
(b) The first recorded solution by Ali C. Mani.

The encoding scheme of both COIN and EHBSA is a straight forward permutation string, where each of the items refers to the position of the chess board ranging from the top left toward right as 1 to 8 and then repeats to the next row until 64 at bottom right. The evaluation function of our approach is the number of legal moves found in a tour.

C. Performance Measurement

For analysis purposes, we implement the algorithms using Codegear Delphi 2007 and test them using an Intel Core 2 duo 2.16 GHz with 2 GB of RAM. Ten runs were performed for each problem. The different configurations of population size and number of generation are applied in which the total number of evaluations is smaller than the solution/space ratio. The bias ratio B_{ratio} of EHBSA is 0.005 was used in all experiments while the learning rate k of COIN is set to be 0.05. The selection pressure of EHBSA is 50% of the whole population, while COIN uses 25% for both reward and punishment. The population size and number of generations are set to be 400 and 800 respectively.

The performance of COIN is compared with MACE [22], GA with repair [23] and GA with heuristic [24] according to the result reported in the original literatures.

D. Computational Results

Fig. 4 compares the performance of COIN and EHBSA. The two red top lines (Best COIN and Average COIN) are the results of maximum and average tours generated by COIN, which can converge to the first open tour at generation 150. Then more of the complete tours are rapidly generated until the first closed tour is found at generation 301. The experimental results of COIN are shown in Fig. 5. EHBSA cannot converge to a single optimal tour. The explanation is that in this benchmark, there are substructures in the population in which there is no significant selective preference to differentiate which one is better. EHBSA cannot even make the population drift towards to a single peak.

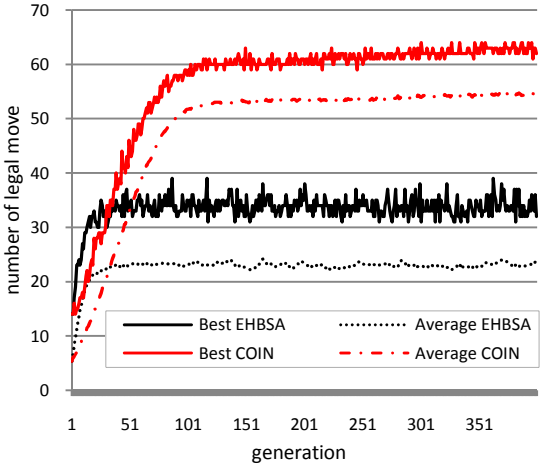

Fig. 4. Comparison of the performance of COIN vs. EHBSA.

While EHBSA became stuck in local optima, COIN has an ability to learn the negative knowledge contained in the not-good solutions. In a complete graph, a knight can have at most 8 legal moves while the remaining 56 moves are prohibited. Learning the prohibited moves is easier as there are a greater number of prohibited moves.

Negative correlation learning of COIN plays two major roles in this benchmark. First, it helps recognizing and eliminating the illegal knight's path from the complete graph, which leads to diversity amongst the legal paths. Second, once the probability matrix converges, it unlearns some of the occurrences in the previous generation in order to find more solution models.

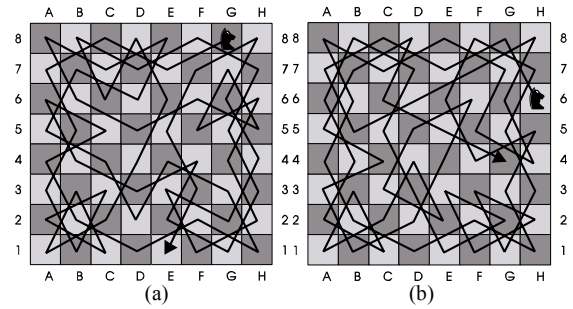

Fig. 5 Two of the solutions generated by the coincidence algorithm.
(a) the first open-tour found in the generation 150.
(b) the first closed-tour found in the generation 301.

Fig. 6 shows the probabilistic model snapshots of EHBSA, COIN that learns the positive correlation only, COIN that learns the negative correlation only and COIN for the knight's tour problem. Due to the enormous size of the probabilistic model, the snapshots were taken only from the 27th row in each generation. Such a row can be transformed to the possible path of a knight from the coordinate 5B to all the rest of the coordinates ,e.g., the row size $= 1 \times 64$ can be transformed into the matrix of 8×8. In these experiments, the problem-specific heuristic is not embedded in order to observe the behavior of the four algorithms. The probabilistic model of EHBSA cannot converge to a stable stage at all as it estimates the distribution from the latest generation while COIN (Pos), COIN (Neg) and COIN incrementally learn from all the previous generations. The probabilistic models of COIN (Pos), COIN (Neg) and COIN slowly adjust themselves toward a stable stage. Such stable stages show that the probability models try to constrain the probabilities of the possible moves to satisfy the legal knight's moves.

According to [23] and [24], a pure genetic algorithm was proven to fail to find solutions to the knight's tour problem. One of the reasons is that there are too many global optima which contain diverse substructures. These substructures are not only hard to recognize but also are in conflict with each other. Even if all of the substructures are identified, the problem of how to compose these substructures remains. The probability matrix of COIN not only learns to compose substructures in order to form a good solution, but it also learns not to compose the substructures likely to form an undesirable solution.

For the comparison, a number of generations are increased to 2,000 generations in order to compare the results of GA with repair operation and GA with heuristic proposed by Gordon and Slocum [23] and Al-Gharaibeh, Qqwagneh and Al-zahawi [24] respectively. This work is also compared with an improved version of ACO called the Multiple-restart Ant Colony Enumeration (MACE) algorithm proposed by Hingston [22]. However, the number evaluation used by Hingston is incredibly enormous. Therefore it is reasonable to mainly compare the result using the hit ratio obtained from the number of tours found from the invested function evaluation.

From Table I, the performance of MACE is superior. Within 172,800,000 evaluations, MACE has found over 13 millions tours on average as the ants quickly bias by not laying pheromones on the prohibited moves. Once a complete tour is found, a population of ants repeatedly performs local searching over the shared information. While MACE evolves a tour from the sub-tours containing only the legal moves, COIN learns the whole permutation strings where both legal and illegal moves are mixed and tries to differentiate the legal and illegal moves among the whole populations. This is considered as a totally blind search. Without problem specific heuristics or bias, COIN can find an average of 10,531 tours within a million function evaluations with 1.05% hit rate compared to 7.5% obtained from MACE. GA with heuristics also performs a great job as it improves the odds of finding a solution of pure heuristics from 0.25 up to 1.51%. GA with repair can improve the odds to find a solution from an iterated repair operation up to only 0.57%. However, the numbers of closed tours are not mentioned in most literatures. On average, COIN can find up to 921.4 closed tours out of 10,531 open tours.

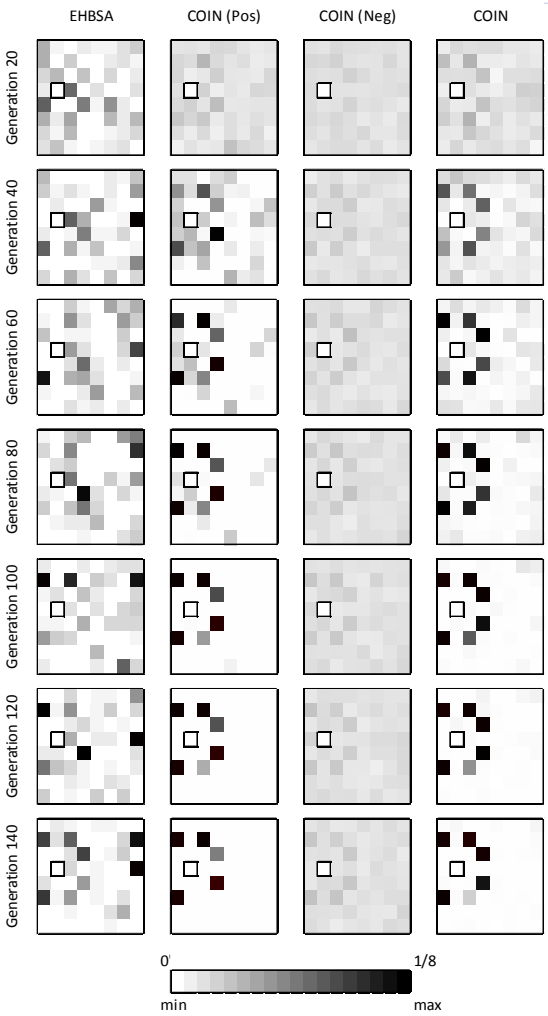

Fig. 6. The 27th row generator snapshots of EHBSA, COIN positive, COIN negative and COIN for the knight's tour problem.

TABLE I
RESULTS OF APPLYING DIFFERENT APPROACHES TO SOLVE THE KNIGHT'S TOUR PROBLEM.

Algorithms	Evaluations	Tours Found	Hit Ratio
MACE	172,800,000	13,124,464	7.5%
COIN	1,000,000	10,531	1.05%
GA+Repair	1,000,000	5,696	0.57%
Repair only	1,000,000	192	0.02%
GA+Heuristic	800,000	12,084	1.51%
Heuristics only	800,000	1,979	0.25%

VI. CONCLUSION

In this paper, the authors introduce an estimation of distribution algorithm based on an edge representation named Coincidence Algorithm (COIN) and apply it to solve a knight's tour problem which is considered to be a globally multimodal problem. COIN has shown the contribution of negative learning in solving a global multimodal as it increases the probability of select and composing good substructures by filtering out the undesired substructures contain in the below average solutions. The negative correlation learning property of COIN contributes in maintaining the diversity for solving the globally multimodal problems which is one of the main difficulties in multi-objective problems as it prevents the solutions from being composed by the unsatisfied substructures. Without using additional heuristics or bias, the results show that COIN is a competitive algorithm in solving the knight's tour problem. Thus, COIN should be adapted to solve combinatorial optimization problems which can be encoded in permutation.

ACKNOWLEDGMENT

The authors would like to thank Mr. Graham Keith Rogers for his editing suggestions. We would also express our gratitude to anonymous reviewers and especially to the program chairs for their comments and proofreading.

REFERENCES

[1] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms*, Boston: Kluwer Academic Publishers, 2002.

[2] S. Tsutsui, "Node histogram vs. edge histogram: A comparison of probabilitstic model-building genetic algorithms in Permutation Domains," in *Proc. of the IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, 2006, pp. 1939-46.

[3] W. Wattanapornprom, P. Olanviwitchai, P. Chutima, and P. Chongstitvatana, "Multiobjective combinatorial optimisation with coincidence algorithm," in *Proc. of IEEE Congress on Evolutionary Computation*, Norway, 2009, pp. 1675-82.

[4] P. Chutima and N. Kampirom, "A multi-objective coincidence memetic algorithm for a mixed-model U-line sequencing problem," in *International Journal of Advanced Operations Management*, vol. 2, no. 3/4, pp. 201-48, 2010.

[5] R. Sirovetnukul and P. Chutima, "The impact of walking time on U-shaped assembly line worker allocation problems," in *Engineering Journal*, vol. 14, no. 2, pp. 53-78, 2010.

[6] W. Wattanapornprom and P. Chongstitvatana, "Solving multimodal combinatorial puzzles with edge-based estimation of distribution algorithm," in *Proc. of Genetic and Evolutionary Computation Conference, GECCO*, Dublin, Ireland, 2011, pp. 67-68.

[7] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," in *Evolutionary Computation*, vol. 7, no. 3, pp. 205-30, 1999.

[8] F. Glover and M. Laguna, *Tabu Search*, Boston, MA: Kluwer Academic Publishers, 1998.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wisley, 1989.

[10] M. Minsky, "Negative expertise," in *International Journal of Expert Systems*, vol. 7, no. 1, pp. 13-19, 1994.

[11] F. Oser and M. Spychiger, *Learning is Painful. On the Theory of Negative Knowledge and the Practice of Error Culture*, Weinheim: Beltz, 2005.

[12] J. Parviainen and M. Eriksson, "Negative knowledge, expertise and organizations," in *International Journal of Management Concepts and Philosophy*, vol. 2, no. 2, pp. 140-53, 2006.

[13] M. Gartmeier, J. Bauer, H. Gruber, and H. Heid, "Negative knowledge: Understanding professional learning and expertise," in *Vocations and Learning*, vol. 1, no. 2, pp. 87-103, 2008.

[14] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," in Tech. Rep. No. CMU-CS-94-163, Pittsburgh, PA: Carnegie Mellon University.

[15] R. S. Michalski, "Learnable execution model: Evolutionary processes guided by machine learning," in *Machine Learning*, vol. 38, pp. 9-40, 2000.

[16] X. Llorà and D. E. Goldberg, "Wise breeding GA via machine learning techniques for function optimization," in *Proc. of Genetic and Evolutionary Computation Conference*, *GECCO*, 2003, pp. 1172-83.

[17] T. Miquélez, E. Bengoetxea, and P. Larrañaga, "Evolutionary computation based on Bayesian classifiers," in *International. Journal of Applied Mathematics and Computer Science*, vol. 14, no. 3, pp. 101-115, 2004.

[18] M. Pelikan, K. Sastry, and D. E. Goldberg, "iBOA: The incremental bayesian optimization algorithm," in *Proc. of Genetic and Evolutionary Computation Conference*, *GECCO*, Atlanta, Georgia, USA, 2008, pp. 455-62.

[19] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and traveling salesman problem: The genetic edge recombination operator," in *Proc. of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[20] B. D. McKay, "Knight's tours of an 8×8 chessboard," Ph.D. dissertation, Department of Computer Science, Australian National University, 1997.

[21] E. Mordecki, "On the number of knight's tours," in Pre-publicaciones de Matematica de la Universidad de la Republica, Uruguay 2001/57, 2001.

[22] P. Hingston and G. Kendall, "Enumerating knight's tours using an ant colony algorithm," in *Proc. of the IEEE Congress on Evolutionary Computation*, Edinburgh, Scotland, 2005, pp. 1003-10.

[23] V. S. Gordon and T. J. Slocum, "The knight's tour – evolutionary vs. depth-first search," in *Proc. of the IEEE Congress on Evolutionary Computations*, 2004, pp. 1435-40.

[24] J. Al-Gharaibeh, Z. Qawagneh, and H. Al-zahawi, "Genetic algorithms with heuristic – Knight's tour problem," in *Proc. of International Conference on Genetic and Evolutionary Methods*, 2007, pp. 177-81.