PAPER

# Negative Correlation Learning in the Estimation of Distribution Algorithms for Combinatorial Optimization

**Warin WATTANAPORNPROM**[†a)], *Member and* **Prabhas CHONGSTITVATANA**[†], *Nonmember*

**SUMMARY** This article introduces the Coincidence Algorithm (COIN) to solve several multimodal puzzles. COIN is an algorithm in the category of Estimation of Distribution Algorithms (EDAs) that makes use of probabilistic models to generate solutions. The model of COIN is a joint probability table of adjacent events (coincidence) derived from the population of candidate solutions. A unique characteristic of COIN is the ability to learn from a negative sample. Various experiments show that learning from a negative example helps to prevent premature convergence, promotes diversity and preserves good building blocks.

***key words:*** *combinatorial optimization, estimation of distribution algorithms, negative correlation learning, multimodal*

## 1. Introduction

Combinatorial optimization (CO) plays an important role in real-world applications, including scheduling, balancing, timetabling and routing problems. CO is applicable to problems in which the domains of the feasible solutions are discrete. Often, decision makers must address 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 representations, encodings and operators. There is no standard benchmark for MOCO that contains all of the known solutions and satisfies all of the multi-objective difficulties, as defined by Deb [1]. Moreover, the performance issue has not been sufficiently tested on difficult problems, leaving the unanswered question of how to scale the method to handle large problems.

Recently, Estimation of Distribution Algorithms (EDAs) [2] and Probabilistic Model Building Genetic Algorithms (PMBGAs) [3] have elevated Evolutionary Algorithms (EAs) to a new level. They have replaced traditional random point mutation and crossover operators with new operators that are based on probabilistic models. These probabilistic models are the estimated distribution of positive substructures that are correlated to the selected potential solutions. The new solutions sampled from such models are expected to yield better average results when compared with previous generations. However, applications of EDAs and PMBGAs to permutation-based combinatorial problems have rarely been investigated. Most of the research on EDAs has been in applications to binary or real-value representation domains. The outstanding EDAs for permutation representations found in the literature are the Edge Histogram Based Sampling Algorithm (EHBSA) [4], the EDA with a guided local search (EDA/GLS) [5], the Node Histogram-Based Sampling Algorithm (NHBSA) [6], the Dependency-Tree EDA (dtEDA) [7] and the Coincidence Algorithm (COIN) [8].

Although EDAs and PMBGAs usually outperform traditional EAs, they are ineffective in solving multimodal and multi-objective problems. They either cannot converge or converge to a single optimum. The explanation for this behavior [9] is straightforward. The basins of different global optima can be represented in the population. Because there is no significant selective preference for one of the basins in the population over another, stochastic variations from the selection method make the population drift toward one of them and thus discover at most one global optimum. This phenomenon is known as genetic drift [10]–[12].

Usually, without using an elitist method, a probabilistic model is expected to converge to a single final solution. However, COIN can solve both multimodal [13] and multi-objective [14] problems using a simple probabilistic model. The unique characteristic of COIN is that it exploits knowledge from high-quality solutions as well as low-quality solutions. This additional property is called Negative Correlation Learning (NCL) and promotes diversity in the solutions, which helps to solve multimodal and multi-objective problems.

The purpose of this paper is to describe and analyze the capabilities of COIN for solving problems that contain multimodality, where the problems are varied in size and in the extent to which the substructures are shared. The remaining sections of this paper are organized as follows. The motivation is reviewed in Sect. 2. COIN is introduced in Sect. 3. Empirical studies are revealed in Sect. 4. Section 5 concludes this work.

## 2. Motivation

To guide the search for better solutions in CO, metaheuristics are used. They can be classified as either improvement or constructive methods. In the improvement schemes, most of the optimization methods rely on the Proximate Optimality Principle (POP) [15], which considers the continuity of each candidate solution based on its neighborhoods. The

Manuscript received February 8, 2013.
Manuscript revised June 19, 2013.

[†]The authors are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok, Thailand.
a) E-mail: yongkrub@gmail.com

DOI: 10.1587/transinf.E96.D.2397

improvement methods use move operators to search for better solutions, which are based on strategies defined by the methods. In the constructive schemes, most optimization methods rely on the Building Block Hypothesis (BBH) [16]. The constructive methods use selection operators to choose the solutions containing common substructures of high-quality solutions called building blocks (BBs) and then make use of crossover operators to recombine these BBs to construct better solutions. Both the POP and the BBH are based on the assumption that high-quality solutions share similar characteristics. However, methods based on the POP consider similarities from a local perspective, while methods based on the BBH consider similarities from a global perspective.

Usually, constructive methods have an advantage over improvement methods because they typically produce more diverse solutions. Consequently, constructive methods are the preferred method for solving multimodal and multi-objective problems. The major drawback in using constructive methods for combinatorial problems is that the recombination operators are inefficient. Directly applying classical recombination operators, such as simple one-point or two-point crossovers, to permutation sequences will generate solutions that are invalid and need to be repaired. The crossover operator can cause a BB disruption. A BB disruption caused by a simple crossover is illustrated in Fig. 1.

In Fig. 1, two solutions are recombined using a simple one-point crossover operator. The BBs are indicated in grey. A randomly chosen cutting point divides each solution into two pieces. Then, the pieces of the solutions are exchanged. Unfortunately, a recombination can result in an invalid permutation. Therefore, specialized crossover operators must be developed. Since 1985, a number of permutation-based crossovers have been developed, including the Partially Mapped Crossover (PMX) [17], the Position Based Crossover (PBX) [18], the Order-Based Crossover (OBX) [18], the Cycle Crossover (CX) [19], the Modified Crossover (MX) [20], the Weight-Mapping Crossover (WMX) [21] and the Edge Recombination (ER) [22]. Typical industrial applications are large and complex, and traditional crossover operators are expected to fail.

The problem with traditional permutation-based crossover is twofold: (*i*) the crossover operators do not know which parts of a sequence are BBs; sometimes, the crossover operators disrupt the BBs instead of combining them, and (*ii*) significantly distinct BBs can conflict and are unlikely to co-exist. Additionally, the permutation problem appears

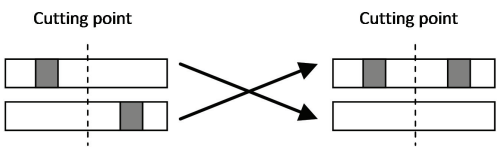



**Fig. 1** Mixing two conflicting BBs.

to encapsulate the reasonable claim that it usually makes little sense to recombine individuals who are genetically very dissimilar. Watson and Pollack [23] noted that *"parents selected from two different fitness peaks are likely to produce an offspring that lands in the valley in between."* Here, using an absolute-order crossover operator is likely to produce offspring that have fitnesses that are similar to their parents.

The problem with permutation-based crossover is well known. Holland [24] suggested combining the BBH with operators that can learn linkage information for recombining the BBs. Subsequently, many methods [25], [26] have been proposed to solve the linkage problem. EDAs and PMBGAs are an effective class of linkage learning models. These algorithms generate solutions by replacing traditional mutation and crossover operators with probabilistic models. Instead of elitism, EDAs and PMBGAs maintain the BBs in the form of probabilities or archives. These BB forms enable a recombination to be based on the entire set of possible BBs, instead of only a few parents. This idea allows EDAs to generate candidate solutions more efficiently. However, there are few studies of EDAs applied to permutation domains. The probabilistic models of EDAs for permutation domains [5]–[8] are usually constructed from a fully connected graph of all of the possible BBs. These types of structures can store and represent multiple search trees. The EDAs for permutation domains generate solution strings in sequences, ensuring that only valid permutations are sampled.

In a permutation representation, objects are distinct. An object can occupy only one position in a sequence. Therefore, BBs in a permutation representation are defined by their order. Two major types of BBs are the absolute order BB [6] and the relative order BB [4], [7], [8]. An absolute order BB is a BB that corresponds to a specific position in a sequence; a relative order BB is a BB that is defined by the distances to the other BBs. Many methods consider relative order BBs to be pairs of adjacent objects, sometimes called edges. A longer chain of pairwise BBs is expected to indirectly link two or more objects that have a longer relative distance. EDAs usually use probabilistic models to represent these order BBs from positive samples. This form is called Positive Correlation Learning (PCL).

Two major components of any metaheuristic method are exploration and exploitation [27]. Exploration generates diverse solutions that explore a search space at a global scale. Exploitation focuses the search in a local region in which a good solution has been found. In EAs, explorations are usually performed through the initialization of the population and through random mutation. Exploitations are usually performed through the selection of the population and recombination. A balance between these two extremes should be carefully maintained, especially during the initialization and selection processes. The initial population must be sufficiently large and diverse to ensure that the distribution of the starting points of the search cover the search space. To ensure that all of the major BBs are preserved, candidate solutions must be chosen to

maintain not only high-quality solutions but also diversity among them. One approach to focusing the search, while maintaining the diversity of the BBs, is to apply Negative Correlation Learning (NCL).

Because the POP and the BBH assume that high-quality solutions share similar structures, it is common for EAs to be driven by PCL. However, low-quality solutions also share similar structures. Combinations of low-quality BBs will result in low-quality solutions. In most EAs, low-quality solutions are ignored by the selection process. These solutions contain useful knowledge that is termed negative knowledge (NK).

In 1994, Minsky [28] introduced the idea of NK in his article "Negative Expertise." He noted that *"competence often requires one to know what one must do, but it also requires one to know what not to do"*. Minsky also stated that *"the creativity of the machine does not only come from the randomization, but it also comes from the reduction of the search space. What distinguishes the performance of a 'smart' or 'creative' artist or problem-solver is not how many trials precede a success, but how few. Therefore, the secret lies not in a disorderly search but in pre-shaping the search space to reduce the number of useless attempts."*

NK has been widely used in classification techniques. NCL is used to promote the diversity of classifiers in neural network ensembles [29]. However, NK has rarely been used in optimization techniques. In the field of EAs, there are some related studies that utilize NK hidden in low-quality solutions. The first technique is the Population-Based Incremental Learning (PBIL) [30] algorithm. It is one of the first EDAs introduced by Baluja in 1994. The PBIL creates a real-valued probability vector that characterizes high fitness solutions that can also be trained with negative examples. In 2000, Michalski [31] proposed an algorithm called the Learnable Evolution Model (LEM). It applies classifiers to develop a population of solutions. The candidates of a population are divided into two categories, the most fit and the least fit. The characteristics of the most fit are strengthened, while the least fit are avoided. Later, in 2003, Llorà and Goldberg [32] proposed an algorithm that combined the Induction of Decision Trees (ID3) and EAs using statistical approaches. In 2004, Miquelez, Bengoetxea, and Larrañaga [33] introduced a new EDA based on Bayesian classifiers called the Evolutionary Bayesian Classifier-Based Optimization Algorithm (EBCOA). Finally, Pelikan, Sastry and Goldberg proposed an incremental version of the Bayesian Optimization Algorithms (BOA) [34] known as the iBOA, which updates the model using the winners and losers of a tournament selection. Unfortunately, none of the studies mentioned here were applied to optimization in the permutation representation.

In 2008, Wattanapornprom and Chongstitvatana proposed an EDA that combined PCL and NCL for combinatorial optimization named the Coincidence Algorithm (COIN) [8]. In addition to the BBH, COIN is based on a complementary concept called the *Negative Building Block Hypothesis* (NBBH), which simply states that *"An algorithm can seek new-optimal performance by avoiding the juxtaposition of short, low-order, low-performance schemata, called negative building blocks."* COIN utilizes the probabilistic models of an adjacency matrix similar to EHBSA. However, instead of using a histogram to estimate the distribution of the BBs, COIN uses reward/punishment learning to adjust the probabilistic model. With the reward/punishment learning method, COIN can learn negative correlations.

The PCL and NCL techniques are both exploitation components. However, they work in different manners. The first aims for quality solutions, while the latter aims for a quantity of solutions. By combining both types of learning methods, COIN gains both quantity and quality. COIN converges faster, and yields more diverse solutions than other methods. COIN has been successfully applied in both single and multi-objective problems [8], [13], [14]. It outperforms the competing algorithms in many performance indicators. The next section provides a brief introduction to COIN.

## 3. Coincidence Algorithm

The Coincidence algorithm (COIN) [8] is an EDA that specializes in solving combinatorial problems. It uses a natural representation as a permutation or combination to encode solution strings. COIN is similar to EHBSA in that they both learn the correlation of mutual edges found in the sequences of promising solutions. Additionally, COIN also includes the NCL of the mutual edges that are found in low-quality solutions.

The selection process of COIN discriminates between better and worse BBs that are found in solutions. This mechanism enhances the algorithm to recognize the better BBs, to compose them. In addition, the worse BBs are discriminated and can be avoided. Figure 2 illustrates this process. $S_1$ has a higher fitness than $S_2$. The BBs that have equal fitness are the common substring of both strings, "13579". The different substring "2468", which is in $S_1$, is a better BB than the substring "4682", which is in $S_2$.

In each iteration, COIN counts the total number of BBs that are found in high-quality solutions and the total number of BBs found in low-quality solutions. The COIN algorithm


**Fig. 2** The differentiation of BBs contained in the better and the worse quality populations.

operates as follows:

Step 1. Initialize the model
Step 2. Sample the population
Step 3. Evaluate the population
Step 4. Select candidates
Step 5. Update the model
Step 6. Repeat steps 2 to 5 until terminated.

COIN uses a transition matrix, $H$, of size $n \times n$ to store the joint probability. $H_{xy}$, where $x$ represents a row and $y$ a column of $H$, has a value between 0 and 1.0. The diagonal elements ($x = y$) are always zero, and the sum over each row equals 1.0. Initially, all off-diagonal elements, $H_{xy}$, are set equal to $\frac{1}{(n-1)}$, where $n$ is the number of variables in the problem.

After each population was evaluated and ranked, two groups of candidates are selected according to their fitness values: better-group and worse-group. The better-group is selected from the top $c\%$ of the rank and is used as a reward, and $H_{xy}$ is increased for every pair of $xy$ found in this group. The punishment is a decrease in $H_{xy}$ for every pair of $xy$ found in the worse-group of the bottom $c\%$ of the population rank. The update equation is

$$H_{xy}(t+1) = H_{xy}(t) + \frac{k}{(n-1)}(r_{xy}(t+1) - p_{xy}(t+1)) + \frac{k}{(n-1)^2}\Big(\sum_{j=1}^{n} p_{xj}(t+1) - \sum_{j=1}^{n} r_{xj}(t+1)\Big),$$

where $k$ denotes the learning step, $n$ is the problem size, $r_{xy}$ is the number of $xy$ found in the better-group, and $p_{xy}$ is the number of $xy$ found in the worse-group. The incremental and decremental step is $\frac{k}{(n-1)}$, and the term $\frac{k}{(n-1)^2}(\sum_{j=1}^{n} p_{xj}(t+1) - \sum_{j=1}^{n} r_{xj}(t+1))$ represents the adjustment of all other $H_{xj}$, where $j \neq x$ and $j \neq y$.

## 4. Empirical Analysis

### 4.1 Experimental Setting

One approach to investigating the behavior of EAs is to use artificial problems in which the solutions are known a priori. For this purpose, researchers usually use deceptive problems, which are difficult multimodal optimization problems. In this article, we benchmark COIN using multimodal combinatorial puzzles whose solutions are known and might mislead the testing algorithm to certain locally optimal points.

#### 4.1.1 Benchmark Problems

COIN and various permutation-based EDAs were compared with two classes of combinatorial problems: permutation and combination. The permutation problems include the 8-queens puzzle, a 3×3 magic square, a 4×4 magic square and the knight's tour problem. The combination problems include the 8-queens, 8-rooks, 14-bishops and 32-knights puzzles. Figure 3, 4, and 5 show a sample solution for each problem.

*A. Magic square*

A magic square of order $n$ is an arrangement of $n \times n$ distinct integers in a square, such that the $n$ numbers in each row, column and diagonal axis sum to the same constant, which is the magic sum, $M$. The magic sum can be calculated by the equation

$$M = \frac{n(n^2+1)}{2}.$$

For a normal magic square of order $n = 3$, 4, and 5, the

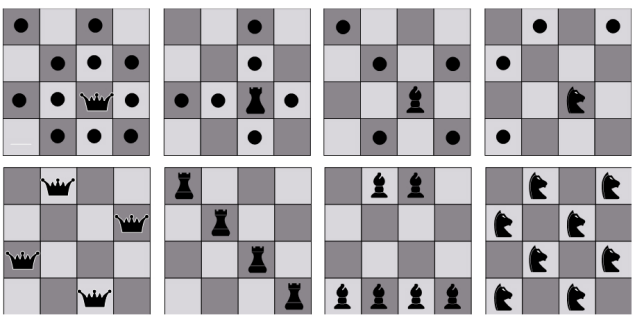
| | | |
|---|---|---|
| 2 | 7 | 6 |
| 9 | 5 | 1 |
| 4 | 3 | 8 |

(a) 3×3 magic square

| | | | |
|---|---|---|---|
| 12 | 6 | 15 | 1 |
| 13 | 3 | 10 | 8 |
| 2 | 16 | 5 | 11 |
| 7 | 9 | 4 | 14 |

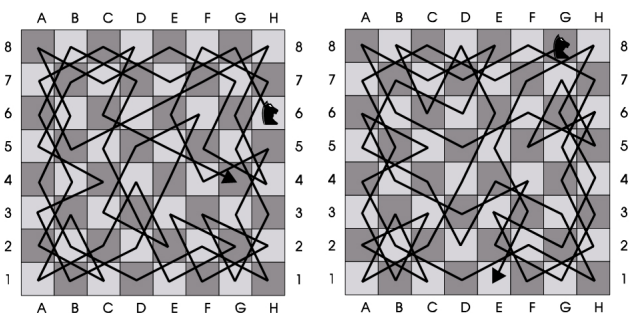
(b) 4×4 magic square

**Fig. 3** Samples of magic square solutions.



(a) $n$-queens (b) $n$-rooks (c) $n$-bishops (d) $n$-knights

**Fig. 4** Available moves and sample solutions to combination problems on a $4 \times 4$ board.



(a) a sample of an open-tour (b) a sample of a closed-tour

**Fig. 5** Two of the solutions generated by COIN.

magic constants are 15, 34, and 65, respectively.

*B. n-pieces chess puzzle*

The $n$-pieces chess puzzle is the problem of placing $n$ of the same class chess pieces on an $8 \times 8$ chessboard so that none of them can capture any other using their moves. A standard $8 \times 8$ chessboard can contain a maximum number of 8-queens, 8-rooks, 14-bishops or 32-knights so that none of them can attack each other. These $n$-pieces chess puzzles are combination problems that are formally defined as: "the optimal $n$-combination of a set, $S$, is a subset of $n$ distinct elements of $S$." These combination problems are more difficult than selection problems because the feasible set of solutions are a fixed size with $k$ distinct elements. Standard mutation and crossover operators in genetic algorithms could produce infeasible solutions. For the 8-queens puzzle, the problem can be recast as both a combination and a permutation problem. It is possible to reduce the number of possibilities by generating the permutations that are solutions of the 8-rooks puzzle. Checking for diagonal attacks further reduces the possibilities from 4,426,165,368 or $\binom{64}{8}$ to 40,320 or 8!.

*C. Knight's tour*

The knight's tour is a well-known classical chess puzzle that has been studied for over a thousand years. The objective of the problem is to find a Hamiltonian path in a graph defined by the legal moves of a knight on a chess board. The knight is required to traverse each square exactly once. Moreover, there are superior solutions, called closed tours, which are solutions with the knight completing the circuit at the starting square. The total number of solutions to the knight's tour problem was researched by Mckay [35] and Mordecki [36] using a very large scale computer. On a standard $8 \times 8$ chess board, McKay calculated the total number of closed tours to be 13,267,364,410,532. Mordecki found the open tours to be approximately $1.305 \times 10^{35}$.

There are studies that have solved the knight's tour problem, including Ant Colonization Optimization (ACO) by Hingston and Kendall [37], which augmented the problem-specific heuristics into their algorithms to increase the chance of finding solutions. The Genetic Algorithm (GA) by Gordon and Slocum [38] and Jarfar Al-Gharaibeh Zakariya Qawagneh and Hiba Al-Zahawi [39] utilized the repair operation and heuristics, respectively. The advantage of using the problem-specific heuristics is that the search space is reduced from 63!, or approximately $1.98 \times 10^{87}$, solutions to $8^{65}$, or approximately $5 \times 10^{58}$, solutions. The performance of COIN compared with these studies can be found in a recent study [40].

#### 4.1.2 Benchmark Algorithms

To contrast the results of permutation-based EDAs with and without negative learning, COIN was compared to EHBSAs [4] and NHBSAs [6] with (EHBSA/wt and NHBSA/wt) and without (EHBSA/wo and NHBSA/wo) templates. In addition, Node-Based COIN (NB-COIN), a variation of COIN, is also implemented and used as a benchmark algorithm to study the behaviors of using NCL in the node-based structure EDAs.

*A. EHBSA and EHBSA/wt*

EHBSA utilize an Edge Histogram Matrix (EHM) to learn the mutual information in the edges contained in the selected solutions and then constructs new solutions by sampling. The idea of EHBSA is to use the edge recombination (ER) in genetic algorithms with the whole selected population instead of a traditional two-parent recombination. We are given matrix $EHM = [e_{ij}]$, where $e_{ij} = P(\sigma_{k+1} = j \mid \sigma_k = i)$ and $i, j \in \{1, 2, \ldots, n\}$ and $k \in \{1, 2, \ldots, n-1\}$. Each $e_{ij}$ is added to an $\varepsilon$ value to control the pressure in sampling and to avoid individuals with probability 0. The variable $\varepsilon$ is given by

$$\varepsilon = \frac{2N}{n-1} B_{ratio},$$

where $N$ is the size of the set of the selected individuals, and $B_{ratio}$ is a constant defined by the authors. To sample the probabilistic model, each candidate is sampled in order starting from the position $\sigma_1$. Once position $\sigma_k$ has been sampled, position $\sigma_{k+1}$ is sampled using the row of EHM corresponding to the index sample at position $\sigma_k$.

EHBSA with template (EHBSA/wt) is an extension of EHBSA, which hybridizes with absolute order crossover. EHBSA/wt generates a new population $P(t+1)$ by filling the template obtained from the previous generation $P(t)$. Then, $k$ random segments are mapped from a randomly selected parent, and $EHM^t$ is used to fill in the remaining space. To maintain the diversity of the population, a newly generated solution is compared with its parent. If the new solution is better, then the parent is replaced with the new solution; otherwise, the newly generated solution is discarded. This template scheme is also used by NHBSA with template (NHBSA/wt).

*B. NHBSA and NHBSA/wt*

In contrast with EHBSA and EHBSA/wt, NHBSA and NHBSA/wt utilize Node Histogram Matrix (NHM) to learn the mutual information of absolute position. Matrix $NHM = [h_{ij}]$, where $h_{ij} = P(\sigma_i = j)$ and $i, j \in \{1, 2, \ldots, n\}$. Hence, $h_{ij}$ represents the probability of the index $j$ to be in the $i$-th position in the selected individuals. Similar to EHBSA, $h_{ij}$ is also added to a $\varepsilon$ value denoted as

$$\varepsilon = \frac{N}{n} B_{ratio}$$

to control the pressure in sampling and to avoid individuals with probability 0.

*C. Node-Based COIN (NB-COIN)*

NB-COIN is a variation of COIN that learns mutual information from absolute positions, similar to NHBSA. Instead of the edge, the matrix $H_{xy}$ represents the probability

of $y$ found in the absolute position $x$. The update equation is changed to

$$H_{xy}(t+1) = H_{xy}(t) + \frac{k}{(n)}\big(r_{xy}(t+1) - p_{xy}(t+1)\big) + \frac{k}{(n)^2}\Big(\sum_{j=1}^{n} p_{xj}(t+1) - \sum_{j=1}^{n} r_{xj}(t+1)\Big),$$

where all of the parameters remain the same as the parameters of the original COIN.

In the experiment, ten runs were performed for each problem. Different configurations of the population size and the number of generations were chosen based on the condition that the total number of evaluations must be less than the solution/space ratio. The bias ratio, $B_{ratio}$, of EHBSAs and NHBSAs was set to 0.005 in all of the experiments. The cutting points of EHBSA and NHBSA with templates were set according to a random parameter $\gamma$ that had values in the range [0.2,0.5). This parameter ($\gamma$) is used to determine the length of each template segment defined in the updated version of EHBSA (Enhanced EHBSA - eEHBSA) [41]. The learning step, $k$, of COIN and NB-COIN, was set to 0.05. The selection pressures of EHBSA and NHBSA without templates were set to 50% of the whole population, while COIN and NB-COIN used 25% of the top ranks for rewards and 25% of the bottom ranks for punishment.

The encoding schemes of each candidate is a straight-forward permutation string, in which each of the items refers to the position on the chess board or table ranging from the top left to the right and then is repeated in the next row until reaching the bottom right. The evaluation function is the total number of legal moves or valid positions that are found in a solution. For all of the $n$-pieces chess problems, the objective functions are to minimize the conflicts of each chess piece. Therefore, the minimum fitness indicates the global optimum. For the knight's tour problem, the objective function is to maximize the legal moves, which are 63 for an open tour and 64 for a closed tour. Therefore, a higher fitness indicates a better performing algorithm. For the magic square problems, the objective functions are to maximize the number of summands that equal the magic number for each row, column and diagonal axis. Therefore, the optimal solutions converge to $2n + 2$, where $n$ is the size of the square.

The performance of each algorithm was evaluated by measuring the average number of evaluations required to find the first global optimum (ANE), the average number of solutions found within the given number of evaluations (#SOL) and the average number of distinct solutions found within the given number of evaluations (#DSOL).

To analyze the behavior of NCL, three additional experiments were performed. The first experiment revealed the role of NCL in maintaining diversity. Additional configurations of COIN were applied to the knight's tour problem, such that the ratio of PCL and NCL were varied from 50:0 to 0:50. The notation of R:P represents the ratio of the better group and that of the worse group, respectively. The snapshots of the EHBSA, EHBSA/wt and variations of COIN were captured every 20 generations to show how the models converged over time. The second experiment revealed another role of NCL that was exploited. The additional configurations of COIN were applied to the $3 \times 3$ magic square such that only PCL or NCL were applied. In this experiment, snapshots of each model were captured every 10 generations. The third experiment verified the properties of NCL in the node-based algorithm. In this experiment, a snapshot of NHBSA, NHBSA/wt and NB-COIN were also applied to the $3 \times 3$ magic square using the same configuration defined in the second experiment.

### 4.2 Analysis of the Results

A summary of benchmarks and their properties are shown in Table 1. The results of the experiments are shown in Table 2. Figure 6 represents box plots of each of the permutation problems, in which magic squares and knight's tour problems are maximization problems and 8-queens-P is a minimization problem. Figure 7 represents the plots of the combination problems that are minimization problems. Each band represents the fitness range of the final population of each benchmark algorithm. In maximization problems, a higher fitness indicates a better convergence of the algorithm. In minimization problems, a lower fitness indicates a better convergence of the algorithm. However, a better convergence does not imply that one algorithm is better than the other when finding multiple solutions. This property only implies that an algorithm can exploit some useful information from the generated population. Some population converge such that most of the population are all similar. Accordingly, the boxes are too narrow to be represented. The results are compared with respect to 5 different aspects, as follows:

#### 4.2.1 Edge-Based vs. Node-Based Algorithms

For this set of problems, edge-based algorithms (EHBSAs, COIN) outperform node-based algorithms (NHBSA, NB-COIN) using the number of distinct solutions as the indicator (#DSOL) (see Table 2). In this set of experiments, each of the benchmarks has more than one optimal solution. The node-based methods are expected to perform poorly, especially on benchmarks that have solutions with less similar BBs. Although the probabilistic models can recognize and maintain all of the possible BBs, a node-based probabilistic model does not provide the linkage information about the search direction. The final models of node-based algorithms should represent a single solution at most. An edge-based probabilistic model generates a candidate solution as a chain such that each position depends on the prior generated sequences. The edge-based model not only represents the information in the solutions but also represents the grouping sequence of different solutions as well. In this study, the edge-based structure is

**Table 1** Properties of each benchmark problem and parameter setting used in the experiments
$c()$ is a conflict test function for each of the pairs, $\sigma_i$,$\sigma_j$ $i \neq j$; it returns 1 if two items are in conflict,
$l()$ is a legal move test function for each of the pairs, $\sigma_i$,$\sigma_j$ $i \neq j$; it returns 1 if the move is a legal move
and $M$ is the $m \times m$ matrix, $row(M_i)$ denotes a function that returns 1 if the sum of row $i$ of the matrix $M$ is $sum(1..m)$, similarly for $col()$ and $diagonal()$

| Problem | Number of Solutions | Search Space | Population Size | Number of Generations | Function Evaluation |
|---|---|---|---|---|---|
| 8-queens-P | 92 | 40,320 | 50 | 50 | $F(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} c(\sigma_i, \sigma_j)$ |
| 8-queens-C | 92 | 4,426,165,368 | 100 | 1,000 | |
| 8-rooks | 40,320 | 4,426,165,368 | 100 | 1,000 | |
| 14-bishops | 8 | 47,855,6999,958,816 | 100 | 3,000 | |
| 32-knights | 2 | 1,832,624,140,942,590,534 | 100 | 3,000 | |
| knight's tour | $1.3 \times 10^{35}$ | $1.268 \times 10^{89}$ | 400 | 1,000 | $F(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sum_{i=1}^{n-1} l(\sigma_i, \sigma_i + 1)$ |
| $3 \times 3$ *magic*$^2$ | 8 | 326,880 | 100 | 100 | $F(M) = \sum_{i=1}^{m} row(M_i) + \sum_{i=1}^{m} column(M_i) + \sum_{i=1}^{m} diagonal(M_i)$ |
| $4 \times 4$ *magic*$^2$ | 7,040 | 20,922,789,888,000 | 200 | 500 | |

**Table 2** Performance of each benchmark algorithm in solving multimodal combinatorial puzzles

| Problem | Algorithm | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Edge-Based Algorithm | | | | | | | | | Node-Based Algorithm | | | | | | | | |
| | EHBSA | | | EHBSA/wt | | | COIN | | | NHBSA | | | NHBSA/wt | | | NB-COIN | | |
| | ANE | #SOL | #DSOL | ANE | #SOL | #DSOL | ANE | #SOL | #DSOL | ANE | #SOL | #DSOL | ANE | #SOL | #DSOL | ANE | #SOL | #DSOL |
| 8-queens-P | 8 | 25 | 4 | 8 | 34 | 4 | 8 | 21 | 13 | 4 | 80 | 4 | 4 | 102 | 4 | 6 | 96 | 10 |
| 8-queens-C | 1821 | 78 | 4 | 323 | 395 | 38 | 3651 | 10 | 9 | N/A | 0 | 0 | N/A | 0 | 0 | N/A | 0 | 0 |
| 8-rooks | 25 | 2457 | 2293 | 43 | 63 | 63 | 454 | 4 | 4 | N/A | 0 | 0 | 65 | 737 | 48 | N/A | 0 | 0 |
| 14-bishops | 419 | 408 | 4 | 865 | 534 | 4 | 1070 | 45 | 8 | 1544 | 456 | 1 | 170 | 2784 | 1 | N/A | 0 | 0 |
| 32-knights | N/A | 0 | 0 | 2851* | 3 | 1 | N/A | 0 | 0 | N/A | 0 | 0 | 724* | 1520 | 1 | N/A | 0 | 0 |
| knight's tour | N/A | 0 | 0 | 805 | 1 | 1 | 154 | 2816 | 2759 | N/A | 0 | 0 | N/A | 0 | 0 | N/A | 0 | 0 |
| $3 \times 3$ *magic*$^2$ | N/A | 0 | 0 | 76 | 2 | 1 | 35 | 40 | 2 | N/A | 0 | 0 | 55 | 8 | 1 | 45 | 2 | 1 |
| $4 \times 4$ *magic*$^2$ | N/A | 0 | 0 | N/A | 0 | 0 | 245* | 1 | 1 | N/A | 0 | 0 | N/A | 0 | 0 | N/A | 0 | 0 |

*the ANEs were obtained only from the successful runs. In the 32-knights problem, EHBSA/wt can find optimal solutions in only 2 out of 10 runs, while NHBSA/wt can find optimal solutions in only 8 out of 10 runs. In the 4x4 magic2 problem, COIN can find optimal solutions in only 5 out of 10 runs.

an important key to solving multimodal and multiobjective problems.

#### 4.2.2 Ad Hoc vs. Incremental Learning Methods

EHBSAs and NHBSAs are considered to be ad hoc learning methods, while COIN and NB-COIN are considered to be incremental learning methods. An advantage of incremental learning methods is that the information from each of the generations is accumulated from generation to generation. The ad hoc learning methods simply throw away information from the prior generations and re-estimate the distribution from the most recent population. If a problem has multiple solutions, then ad hoc methods cannot exploit the information from different peaks of highly fit solutions. The ad hoc methods have an advantage over the incremental learning methods in that if the solutions contain shared BBs, then the ad hoc methods would converge faster compared to the incremental methods. From the overall results, EHBSAs and NHBSAs can find a solution faster than COIN (see the column labeled ANE in Table 2); however, COIN can find more distinct solutions (#DSOL, Table 2).

#### 4.2.3 Template vs. Non-template Methods

In a problem with a single solution, using templates will help the algorithms to converge faster and obtain better solutions [4], [42]. In some situations, templates are used as a local search operator that helps to prevent premature convergence [4], [6]. The advantage of using a template is that it ensures that the BBs are maintained in the population. Additionally, the population-replacing mechanism enhances the template methods to make them capable of maintaining diversity in the population. In this experiment, both EHBSA/wt and NHBSA/wt can find the optimal solution from the 32-knights problem, which is considered to be one of the most difficult problems because it contains only two distinct solutions in which none of the BBs are shared. Without using templates, the chance to maintain the BBs for either one of the two distinct solutions is very low because none of the non-template methods can achieve the optimal solution at all.

However, in the multimodal problems in which each of the solutions rarely share the BBs, the template methods slow down the optimization process because they waste many iterations combining BBs from different peaks of

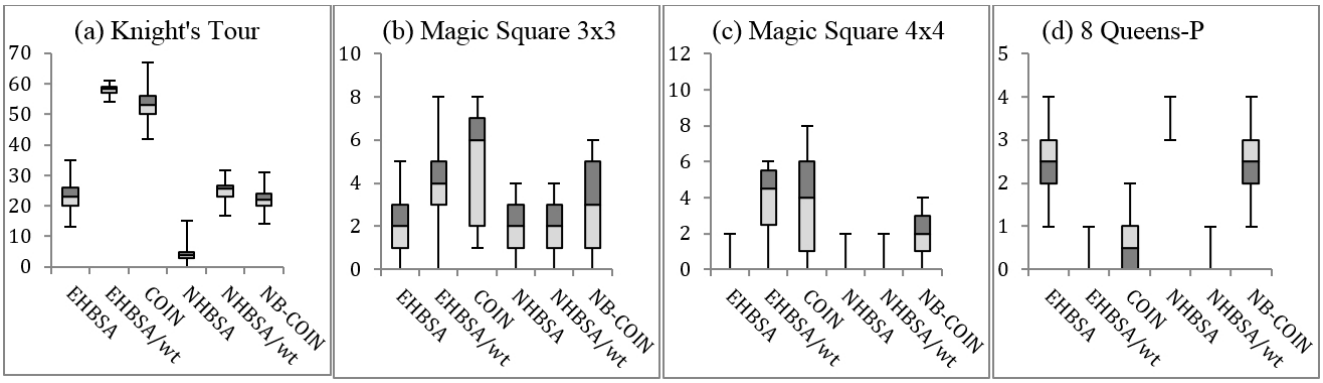



**Fig. 6** Box plots of each of the permutation problems.

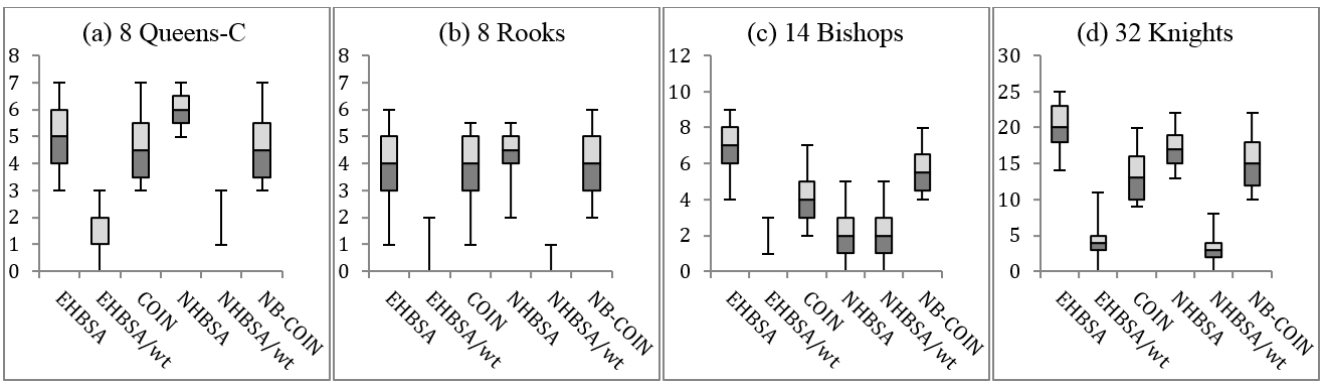



**Fig. 7** Box plots of each of the combination problem.

fitness. Although the BBs are maintained in the population, the negative BBs are not easily eliminated from the population as well.

#### 4.2.4 Permutation vs. Combination Problems

In a permutation problem, a candidate is encoded with all of the possible items, while in a combination problem, a candidate is encoded with a partial set of items. The ad hoc learning methods, such as EHBSAs and NHBSAs, cannot widely explore the solution space because the solutions in the initial generation do not provide all of the essential BBs that are required to compose a new solution. Additionally, node-based methods have difficulty when combining BBs from different peaks, while edge-based methods have the advantage of binding together the BBs that should be grouped together. In general, both types of EHBSAs and NHBSA/wt outperform COIN in combination problems, i.e., EHBSA/wo converges to solutions faster than COIN. In the 8-queens-C and 8-rooks problem, both types of EHBSAs find more solutions than COIN. In the 14-bishops problem, COIN finds all of the distinct solutions, while both types of EHBSAs find only half of them. The reason is that EHBSAs greedily converge to an optimal point because they estimate the probabilistic models only from the latest generation, but COIN attempts to maintain all of the possible BBs starting from the first generation. The 32-knights problem is considered to be a difficult and deceptive problem because there are only two solutions: either all black or all white checkers are filled. EHBSA/wt can find optimal solutions in only 2 out of 10 runs, while NHBSA/wt can find optimal solutions in only 8 out of 10 runs.

#### 4.2.5 PCL vs. NCL

This study focuses on hybridizing NCL into permutation-based EDAs. NCL contributes in filtering good BBs from bad BBs, thus preventing the candidates from being composed by the bad BBs. Additionally, using NCL also contributes to maintaining the BBs found in different potential solutions. However, in a node-based algorithm, the probabilistic model does not provide the linkage information between each of the items; therefore, the algorithm has difficulty in composing the BBs obtained from the different peaks of fitness.

Magic squares are considered to be difficult deceptive problems in which a pattern can be rotated as a new solution. Moreover, each pattern can be mirror-reflected to form another solution. The difficult part of this problem is that there are no overlapping relative building blocks between the solutions. Thus, there is no significant selective preference for one of the basins in the population over another. Currently, there are no promising metaheuristics for these problems.

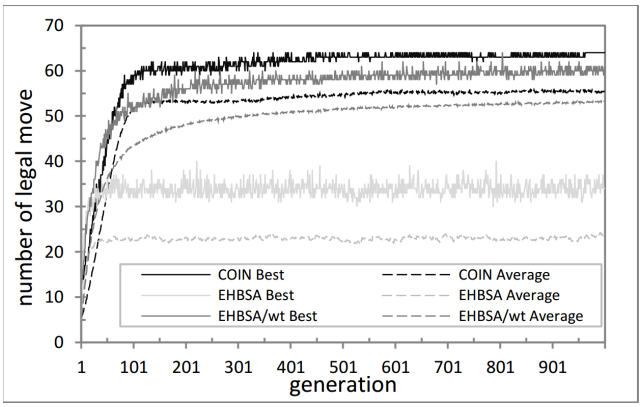



**Fig. 8** The performance of edge based EDAs in the knight's tour problem.

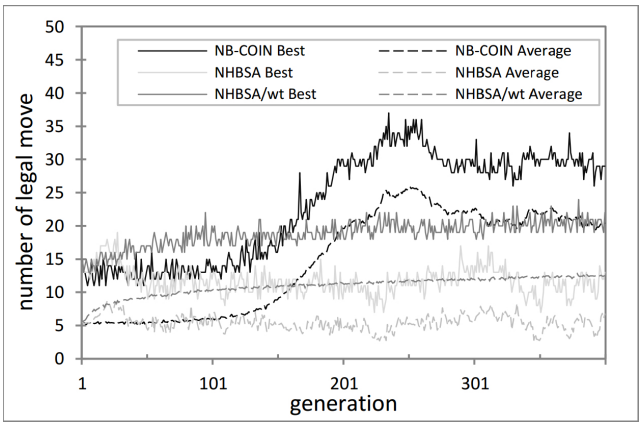



**Fig. 9** The performance of node based EDAs in the knight's tour problem.

COIN and NB-COIN perform better than EHBSAs and NHBSAs in magic square problems. Because COIN and NB-COIN also learn the negative correlations between BBs that are found in the low-quality solutions, COIN prevents the recombination of these BBs and converges closer to the optimal solutions. For example, in a $3 \times 3$ magic square problem, the total number of 3 combination summands is $\binom{9}{3} = 84 \times 3!$. The summands of the magic number $M = 15$ are the $8 \times 3!$ permutations of 1+5+9, 2+4+9, 2+6+7, 3+5+7, 1+6+8, 2+5+8, 3+4+8 and 4+5+6. All other combinations are invalid. Therefore, there are only 24 valid summands out of 504 summands that are considered to be the good BBs. EHBSAs cannot distinguish which summand belongs to which model; they only attempt to construct a solution from the good BBs. In contrast, COIN prevents the construction of 480 negative BBs and finds an optimal solution. NB-COIN discovered that the center of the table should be the number 5 because it maximizes the chance of the alignment of the different summands in 4 out of 8 axes. However, COIN and NB-COIN rarely find the solution of the $4 \times 4$ magic square. The search gets trapped in some local optimums.

It is clear that the properties of the knight's tour problems are similar to the TSP. The BBs in absolute order are heavily conflicting. The node-based algorithms are not suitable for solving these types of problems. [6], [41]

COIN has an ability to learn the negative knowledge that is contained in the low-quality solutions. In a complete graph, a knight can have no more than 8 legal moves; the remaining 56 moves are prohibited. Because there are a greater number of prohibited moves, learning the prohibited moves becomes easy. Negative correlation learning of COIN plays two major roles in this benchmark. First, it helps to recognize and eliminate the illegal knight's paths from the complete graph, which leads to diversity among the legal paths. Second, once the probability matrix converges, it unlearns some of the occurrences in the previous generation.

Figure 8 compares the performance of COIN, EHBSA/wo and EHBSA/wt in the knight's tour problem. The two darkest top lines are the best and the average tour generated by COIN. The algorithm converges to the first open tour at generation 150. More complete tours are rapidly generated until the first closed tour is found at generation 301. The performances of EHBSA/wt are illustrated using the dark gray lines. EHBSA/wt can find the first and only tour in the generation 819. In the knight's tour problems, EHBSAs cannot converge to an optimal solution at all.

Figure 9 compares the performance of the node based EDAs. The two darkest top lines are belong to NB-COIN. The dark gray lines and the light gray lines are belong to NHBSA/wo and NHBSA/wt respectively. The NCL contributes by exploiting the common negative BBs shared by the low-quality solutions. However, NB-COIN cannot find the optimal solution because the node based representation is not appropriate for this problem.

Figure 10 shows the snapshots of the probabilistic models of EHBSA, EHBSA/wt and COIN for the knight's tour problem. The snapshots of COIN show many configurations, from the positive correlation only (50:0) to the total negative correlation (0:50). Because of the enormous size of the model, the snapshots were sampled only from the $26^{th}$ row. Such a row can be transformed to a set of possible paths of a knight from the coordinate 5B, which is indicated by the open black square. (The reference coordinate can be seen in Fig. 5.) The brighter blocks indicate the lower probabilities, while the darker blocks indicate the higher probabilities. The probability ranges from the whitest to the blackest (0 to 1/8). To observe the behavior of the algorithm more clearly, a problem-specific heuristic is not used in this experiment. EHBSA does not converge to a stable state at all because it estimates the distribution only from the latest generation. EHBSA/wt can find all of the essential BBs; however, EHBSA/wt cannot eliminate the probability of some false BBs that result in inefficiently composing the BBs. COIN incrementally learns from all of the previous generations. COIN slowly adjusts the model toward a stable state. In the stable state,

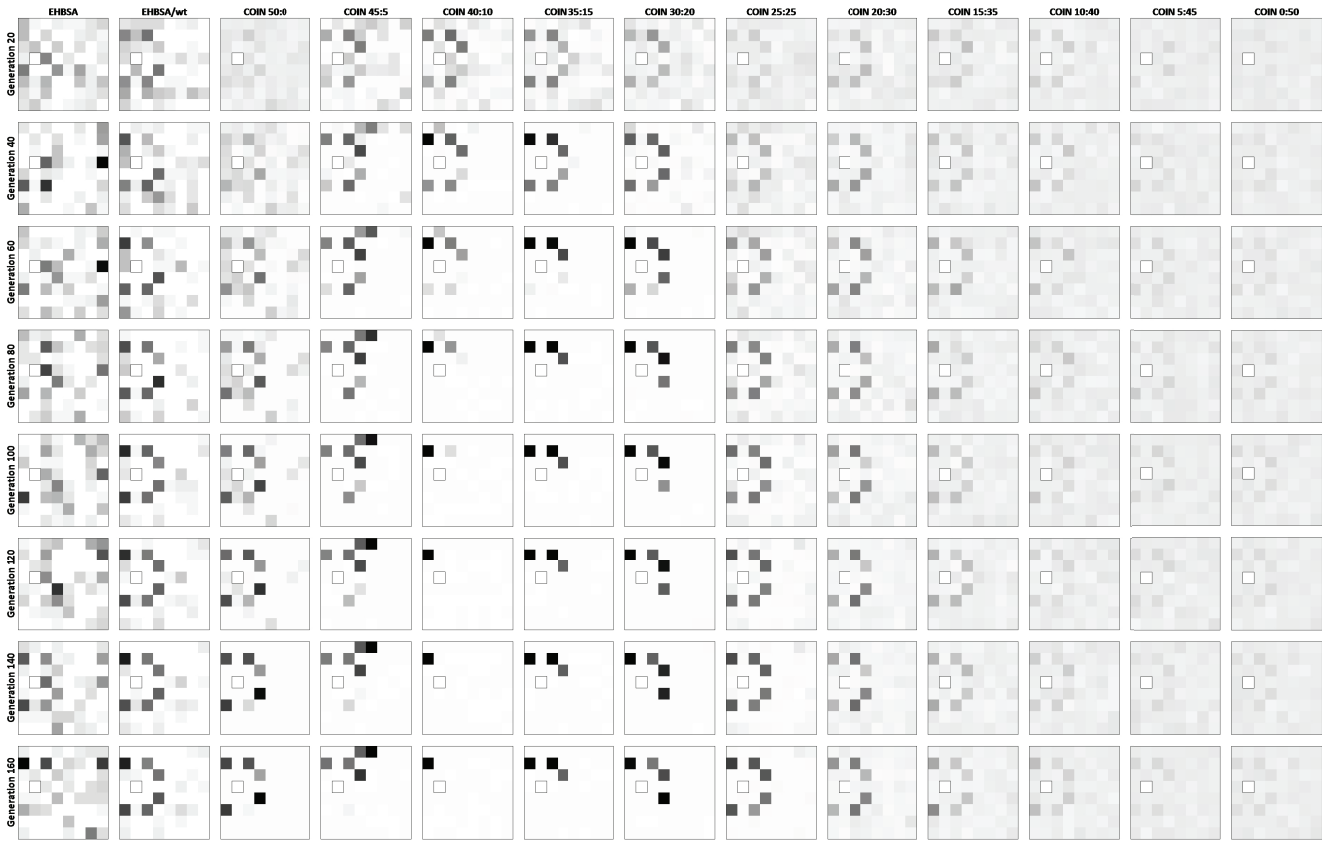



**Fig. 10** Probabilistic model snapshots of EHBSA, ESBSA/wt and variations of the positive and negative correlation learning ratio of the COIN for the knight's tour problem.The brighter blocks indicate the lower probabilities while the darker blocks indicate the more potential probabilities.

the model constrains the possible moves to legal moves. In COIN, between PCL and NCL, PCL has a higher weight in the model than NCL. Although NCL identifies all of the BBs that satisfy legal moves, the probabilities are not strong enough to be sampled. When COIN learns the positive correlation, it is likely to converge to a local optima because the probabilities are dominated by a few strong legal moves. In addition, the imbalance of positive and negative BBs is likely to mislead the search direction, as observed in COIN (45:5); some probabilities are given to illegal moves. Determining the proper weight for the ratio between PCL and NCL is still based on empirical study.

Figure 11 shows the probabilistic model snapshots of edge based EDAs for the $3\times3$ magic square problem. In this experiment, both types of EHBSAs were compared with the three versions of COIN, including PCL COIN (25:0), NCL COIN (0:25) and COIN (25:25) (which combines PCL and NCL). The brighter blocks indicate the lower probabilities, and the darker blocks indicate the higher probabilities. The probability ranges from the whitest to the blackest (0 to 1). Because the $3 \times 3$ magic square is a multimodal problem in which the building blocks are rarely shared; EHBSA and the PCL COIN (25:0) cannot converge to a single solution. In this case, the PCL COIN (25:0) cannot exploit the knowledge found in the population at all. The NCL

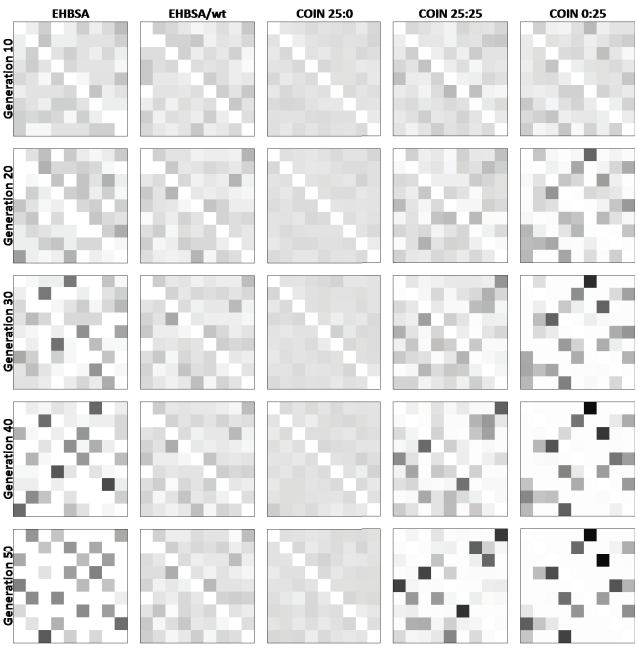



**Fig. 11** Probabilistic Model snapshots of EHBSA and variations of the Positive and Negative Correlation Learning ratio of the COIN for the $3 \times 3$ magic square problem.

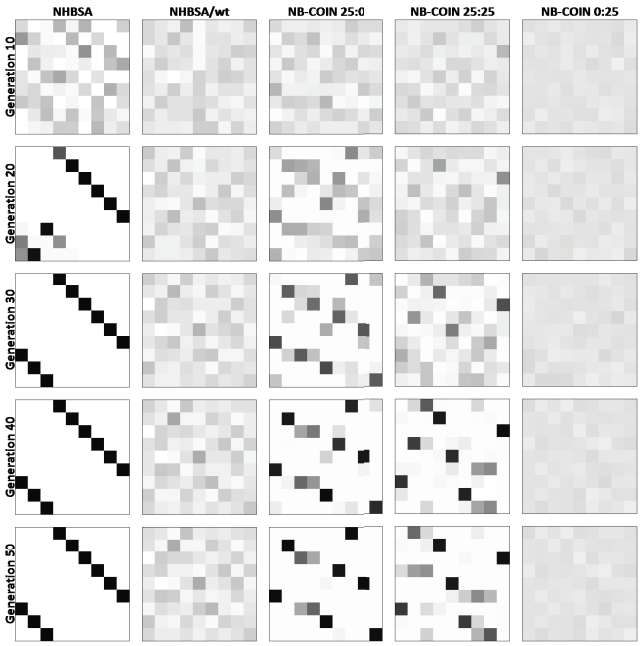



**Fig. 12** Probabilistic Model snapshots of NHBSA and variations of the Positive and Negative Correlation Learning ratio of the NB-COIN for the $3 \times 3$ magic square problem.

COIN (0:25) appears to be smarter in exploiting the search space. The NCL contributes by exploiting the common negative BBs shared by the low-quality solutions. The combination of positive and negative correlation learning enhances COIN (25:25), and it converges faster than the others. Figure 12 shows the probabilistic model snapshots of node based EDAs for the $3 \times 3$ magic square problem. In this experiment, both types of NHBSAs were compared with the three versions of NB-COIN, including PCL NB-COIN (25:0), NCL NB-COIN (0:25) and NB-COIN (25:25) (which combines PCL and NCL). The probabilistic models in this experiment represent the probabilities of a number to be sampled in each node. NHBSA and PCL NB-COIN (25:0) rapidly get stuck in some local optimums where BBs are conflicting with each other. NHBSA/wt can slowly exploit the information and can find a solution, while NCL NB-COIN (0:25) cannot exploit any useful information from the population at all. By combining the power of PCL and NCL, NB-COIN (25:25) discovers that every optimal solution shares a BB in which the number 5 is shared at the center position.

## 5. Conclusions

This article introduces an estimation of distribution algorithm called the Coincidence Algorithm (COIN). This algorithm was applied to several multimodal combinatorial puzzles, including both permutation and combination problems. COIN has shown the role of NCL in solving globally multimodal problems. There are three main advantages in using COIN. First, it increases the probability of selecting and composing good BBs by filtering out the bad BBs contained in the low-quality solutions. Second, the negative learning in COIN also contributes to solving multi-objective problems [8], [13], [14]. Finally, it prevents solutions from being composed of the BBs that are not likely to satisfy any objectives being optimized. COIN can be adapted easily to solve other problems that can be encoded in a permutation representation.

### References

[1] K. Deb, "Multi-objective genetic algorithms: Problem difficulties and construction of test problems," Evolutionary Computation, vol.7, no.3, pp.205–30, 1999.
[2] P. Larrañaga and J.A. Lozano, Estimation of Distribution Algorithms, Kluwer Academic Publishers Boston, 2002.
[3] M. Pelikan, D.E. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic models," Computational Optimization and Applications, vol.21, no.1, pp.5–20. IlliGAL Report no.99018, 2002.
[4] S. Tsutsui, "Probabilistic model-building genetic algorithms in permutation representation domain using edge histogram," Proc. 7th Int. Conf. on Parallel Problem Solving from Nature (PPSN VII), pp.224–233, 2002.
[5] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem," Workshop Proceddings at the Genetic and Evolutionary Computation Conference (GECCO-2003), pp.42–48, 2003.
[6] S. Tsutsui, "Node histogram vs. edge histogram: A comparison of probabilitstic model-building genetic algorithms in permutation domains," Proc. IEEE Congress on Evolutionary Computation, pp.1939–46, Vancouver, BC, Canada, 2006.
[7] M. Pelikan, S. Tsutsui, and R. Kalapala "Dependency trees, permutations, and quadratic assignment problem," MEDAL Report no.2007003, 2007.
[8] W. Wattanapornprom, P. Olanviwitchai, P. Chutima, and P. Chongstitvatana, "Multiobjective combinatorial optimisation with coincidence algorithm," Proc. IEEE Congress on Evolutionary Computation, pp.1675–1682, Norway, 2009.
[9] J.M. Peña, J.A. Lozano, and P. Larrañaga, "Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks," J. Evolutionary Computation, vol.13, Issue 1, pp.43–66, Jan 2005.
[10] K.A. De Jong, "An Analysis of the Behavior of a Class of Genetic Adaptive Systems," PhD Thesis, University of Michigan, 1975.
[11] D.E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wisley, 1989.
[12] D.E. Goldberg and P. Segret, "Genetic algorithms with sharing for multimodal function optimization," Proc. 2nd Internation Conference on Genetic Algorithms, pp.41–49, 1987.
[13] W. Wattanapornprom and P. Chongstitvatana, "Solving multimodal combinatorial puzzles with edge-based estimation of distribution algorithm," Proc. Genetic and Evolutionary Computation Conference, GECCO, pp.67–68, Dublin, Ireland, 2011.
[14] R. Sirovetnukul and P. Chutima "The impact of walking time on U-shaped assembly line worker allocation problems" Chulalongkorn University's Engineering Journal, vol.14, issue 2, April 2010.
[15] F. Glover and M. Laguna, Tabu Search, Boston, Kluwer Academic Publishers, MA, 1998.
[16] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wisley, Reading, MA, 1989.
[17] D.E. Goldberg and R. Lingle, "Alleles, Loci and the Traveling Salesman Problem," Proc. 1st International Conference on Genetic Algorithm (ICGA), pp.154–159, 1985.
[18] G. Syswerda, "Schedule Optimization Using Genetic Algorithms,"

in Handbook of Genetic Algorithms, 1991.
[19] I.M. Oliver, D.J. Smith, and J.R.C. Holland, "A study of permutation crossover operators on the travelling salesman problem," Proc. 2nd International Conference on Genetic Algorithm (ICGA), pp.224–230, 1986.
[20] L. Davis, "Applying Adaptive Algorithms to Epistactic Domains," Proc. International Conference on Artificial Intelligence (IJCAI), pp.162–164, 1985.
[21] J. Lee, M. Gen, and K. Rhee, "Designing a multistage reverse logistics network problem by hybrid genetic algorithm," Proc. 10th Annual Conference on Genetic and Evolutionary Computation (GECCO), 2008.
[22] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and traveling salesman problem: The genetic edge recombination operator," Proc. 3rd International Conference on Genetic Algorithms, 1989.
[23] R.A. Watson and J.B. Pollack, "Recombination without respect: Schema combination and disruption in genetic algorithm crossover," Proc. 2000 Genetic and Evolutionary Computation (GECCO), pp.112–119, 2000.
[24] J.H. Holland, Adaptation in Natural and Artificial Systems. Ann Arbor, University of Michigan Press, MI, 1975.
[25] G.R. Harik and D.E. Goldberg, "Learning linkage," Foundations of Genetic Algorithms, vol.4, pp.7–12, 1996.
[26] M. Munetomo and D.E. Goldberg, "Identifying linkage groups by nonlinearity/non-monotonicity detection," Proc. 2000 Genetic and Evolutionary Computation (GECCO), vol.1, pp.433–440, 1999.
[27] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," ACM Computing Surveys, vol.35, no.3, pp.268–308, 2003.
[28] M. Minsky, "Negative expertise," International J. Expert Systems 7, 1994.
[29] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," IEEE Trans. Evol. Comput., vol.4, pp.380–387, Sept. 2000.
[30] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Tech. Rep. No.CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA.
[31] R.S. Michalski, "Learnable execution model: Evolutionary processes guided by machine learning," Mach. Learn., vol.38, pp.9–40, 2000.
[32] X. Llorà and D.E. Goldberg, "Wise breeding GA via machine learning techniques for function optimization," Proc. Genetic and Evolutionary Computation Conference, GECCO, pp.1172–1183, 2003.
[33] T. Miquélez, E. Bengoetxea, and P. Larrañaga, "Evolutionary computation based on Bayesian classifiers," International. Journal of Applied Mathematics and Computer Science, vol.14, no.3, pp.101–115, 2004.
[34] M. Pelikan, K. Sastry, and D.E. Goldberg, "iBOA: The incremental bayesian optimization algorithm," Proc. Genetic and Evolutionary Computation Conference, GECCO, pp.455–62, Atlanta, Georgia, USA, 2008.
[35] B.D. McKay, Knight' tours of an 8×8 chessboard, Ph.D. dissertation, Department of Computer Science, Australian National University, 1997.
[36] E. Mordecki, "On the number of knight's tours," Pre-publicaciones de Matematica de la Universidad de la Republica, Uruguay 2001/57, 2001.
[37] P. Hingston and G. Kendall, "Enumerating knight's tours using an ant colony algorithm," Proc. IEEE Congress on Evolutionary Computation, pp.1003–1010, Edinburgh, Scotland, 2005.
[38] V.S. Gordon and T.J. Slocum, "The knight's tour - evolutionary vs. depth-first search," Proc. IEEE Congress on Evolutionary Computations, pp.1435–40, 2004.
[39] J. Al-Gharaibeh, Z. Qawagneh, and H. Al-zahawi, "Genetic algorithms with heuristic - Knight's tour problem," Proc. International Conference on Genetic and Evolutionary Methods, pp.177–81, 2007.
[40] R. Sirovetnukul, P. Chutima, W. Wattanapornprom, and P. Chongstitvatana, "The effectiveness of hybrid negative correlation learning in evolutionary algorithm for combinatorial optimization problems," IEEE Int. Conf. on Industrial Engineering and Engineering Management, 2011.
[41] S. Tsuisui, "Parallelization of an evolutionary algorithm on a platform with multi-core processors," Artificial Evolution, Springer, Lecture Notes in Computer Science, vol.5975, pp.61–73, 2009.
[42] J. Ceberio, E. Irurozki, A. Mendiburu, and J.A. Lozano, "A review on estimation of distribution algorithms in permutation based combinatorial optimization problems," Progress in Artificial Intellgence, vol.1, no.1, pp.103–117, 2012.

**Warin Wattanapornprom** received a B.S. in Information Technology from Sirindhorn International Institute of Technology, Thammasat University in 1999 and an M.S. in Computer Science (2003) and a Ph.D. in Computer Engineering (2012) from Chulalongkorn University.

**Prabhas Chongstitvatana** has been a professor in the Department of Computer Engineering, Chulalongkorn University, since 2007. He earned a B.Eng. in Electrical Engineering from Kasetsart University, Thailand, in 1980, and a Ph.D. from the Department of Artificial Intelligence, Edinburgh University, U.K., in 1992. His research involves robotics, evolutionary computation, computer architecture, grid computing and bioinformatics.