Solving Multimodal Problems by Coincidence Algorithm

Kiatsopon Waiyapara
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University,
Bangkok, Thailand
kiatsopon.w@gmail.com

Prabhas Chongstitvatana
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University,
Bangkok, Thailand
Prabhas.C@chula.ac.th

Abstract—**In general, Multimodal optimization is hard problems even for Evolutionary Algorithm. Using a Genetic Algorithm (GA) to solve these problems, the algorithm cannot converge to solutions easily. This work presents a study of Coincidence Algorithm (COIN) to solve these problems. COIN has an ability to retain multiple solutions in its model; hence it is suitable for Multimodal optimization problems. The experiment is carried out to illustrate this capability. The benchmarks are designed for comparing the problem solving behavior of COIN against a Genetic Algorithm.**

Keywords-component; Coincidence Algorithm; Multimodal Problem; Genetic Algorithm

I. INTRODUCTION

Coincidence Algorithm (COIN) is an evolutionary algorithm that was designed recently [1]. The algorithm belongs to a class of so called *Competent Genetic Algorithm* or *Estimation of Distribution Algorithm* (EDA) [2] which is different from Genetic Algorithm (GA) in a way that a model of solution is used. COIN is suitable for solving combinatorial problems. It composes of a probabilistic model for generating the solutions and a learning method to learn patterns from solutions that are used to update the model. A distinct characteristic of COIN is that its selection method that provides both good and bad solutions for updating the probabilistic model.

COIN is powerful for a wide range of applications. Many multi-objective problems [1] can be solved by COIN. One of the key difficulties of multi-objective problems is that they contain multimodality [3]. This work is set out to illustrate how COIN solves multimodal problems [4]. The problems used in the experiment are Traveling Salesman Problems with multiple satisfactory solutions called Multimodal Traveling Salesman Problem. It is one of the interesting problems that represent the class of multimodal problems.

To understand the behavior of COIN, a Genetic Algorithm with a greedy crossover [5] is used for comparison. This class of problems is difficult for permutation-based GA to solve because there may be conflicting building blocks in which GA cannot possibly do a good job of composing them. However, it is suitable for COIN which can provide multiple solutions.

Generally, multimodal combinatorial problems are difficult to design. Traveling Salesman Problems are used as a basis to design the benchmark problems. All of the benchmark problems have many solutions sharing either a few or a lot of building blocks. Even though GA can identify good solutions, it does not know which building blocks in the solutions are suitable for generating the new solutions. In addition, the permutation representation constraints the recombination such that building blocks of the two distinct solutions are unlikely to co-exist, therefore the probability that GA can recombine building blocks to form good solutions is low [6].

In many multimodal problems, building blocks are in conflict to each others. The mechanism for composing a solution in COIN exploits the knowledge of building blocks, therefore it has high probability to compose multiple and good solutions. In addition, besides using the good solutions to learn the model, COIN also uses negative correlation learning [7] where the *bad* solutions are also take part in guiding the search. This provides additional exploratory power for solving multimodal problems.

II. COINCIDENCE ALGORITHM

In many ways COIN is similar to EDAs. The procedures are: using a probabilistic model, sampling a population, evaluating the population, and selecting the good solutions and updating the model. A specific characteristic of COIN is that COIN uses *bad* solutions to avoid unnecessary search into unpromising solution space. Both good solutions and bad solutions are provided to update the model. Steps of the algorithm are presented in the flowchart shown in Fig. 1. Each candidate generated by COIN represents adjacent pairs of nodes. For Traveling Salesman Problems, the candidate refers to a string of the number. Each element of the string refers to a city in the tour. For example, assuming that the string is *01234*, and a city i is the city such that i denotes the city number. This string can be read as $City_0$ connects to $City_1$, $City_1$ connects to $City_2$, and $City_2$connect to $City_3$, and so on.

978-1-4673-1921-8/12/$31.00 ©2012 IEEE

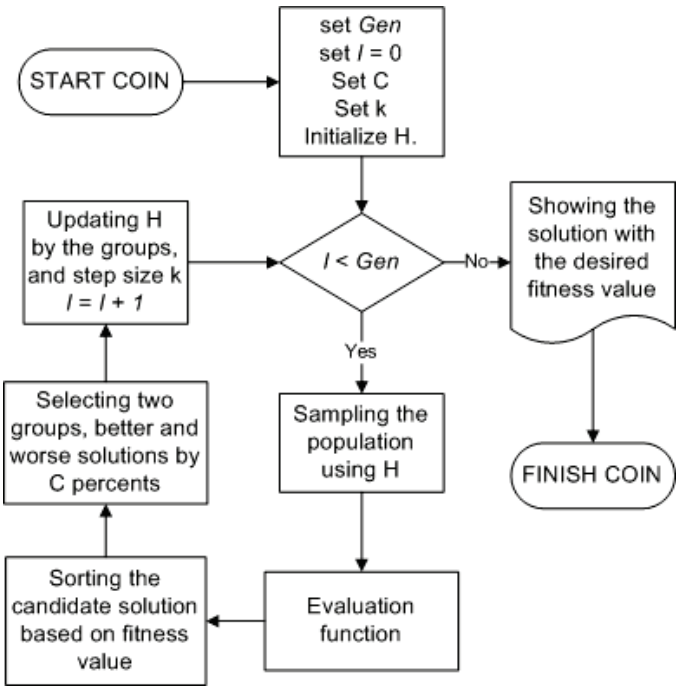

Figure 1. A flowchart of Coincidence Algorithm

TABLE I. AN EXAMPLE OF THE JOINT PROBABILITY MATRIX

	0	1	2	3	4
0	**0.00**	**0.25**	**0.25**	**0.25**	**0.25**
1	**0.25**	**0.00**	**0.25**	**0.25**	**0.25**
2	**0.25**	**0.25**	**0.00**	**0.25**	**0.25**
3	**0.25**	**0.25**	**0.25**	**0.00**	**0.25**
4	**0.25**	**0.25**	**0.25**	**0.25**	**0.00**

The problem size = 5.

The probabilistic model in COIN is known as a joint probability matrix, denoted by H. H is used to generate the candidate solutions. The population is evaluated for the fitness of each candidate. In the update step, the matrix is adjusted according the evidence extracted from selected candidates. For example, the matrix in Table I illustrates an initial state of the joint probability matrix which size of the problem is 5.

Each element of the matrix is denoted by H_{xy} . x is a row index number, and y is a column index number; that is, x and y refer to a position in the matrix. In an initial state, all elements of the matrix except H_{xx} equals to 1/(n-1) where n is the problem size. H_{xx} (The diagonal) equals 0. This represents a state where solutions are all equally likely (uniform distribution). The summation of each row equals to 1.0. For instance, if the problem size is 5, then in the initial state, all the elements of H are 0.25 except for the H_{xx} which are 0.0.

The population is sampling from this matrix. After all candidates are evaluated for their fitness values, they are selected into two groups: better and worse. The size of the group is controlled by a parameter C as a percentage of the population size.

These two groups are used to update the matrix. The *coincidence* found in these candidates is used to adjust the weight of the matrix. For the better group, this weight is used to adjust the value in the matrix upward (more probability) according to (1).

$$H_{xy}(t+1) = H_{xy}(t) + \frac{k}{(n-1)}\left(r_{xy}\right) - \frac{k}{(n-1)^2}\left(\sum_{i=1}^{n} r_{xi}\right) \quad (1)$$

Where r_{xy} is the number of time the concidence xy is found in the candidate in the better group. k denotes the step size and n denotes the problem size. t denotes the current time step. $t+1$ denotes the next time step. Because the summation of the probability in each row is kept to be 1.0, the weight of the others in the same row must be decreased by $\frac{k}{(n-1)^2}\left(\sum_{i=1}^{n} r_{xi}\right)$. The worse group is used similarly to adjust the weight in the matrix downward according to (2).

$$H_{xy}(t+1) = H_{xy}(t) - \frac{k}{(n-1)}\left(p_{xy}\right) + \frac{k}{(n-1)^2}\left(\sum_{i=1}^{n} p_{xi}\right) \quad (2)$$

Two equations can be recombined into (3).

$$H_{xy}(t+1) = H_{xy}(t) + \frac{k}{(n-1)}\left(r_{xy} - p_{xy}\right) - \frac{k}{(n-1)^2}\left(\sum_{i=1}^{n} r_{xi} - \sum_{i=1}^{n} p_{xi}\right) \quad (3)$$

To prevent a zero probability to occur in H_{xy} where $x \neq y$, (this is equivalent to an absolute decision not to use this path), the value is limited to be higher than $\frac{1}{10}$ of the initial value.

III. EXPERIMENT

Coincidence algorithm is implemented in C++, and a tool to visualize the algorithm is implemented based on OpenGL. The Genetic Algorithm with greedy crossover is implemented in C# and is visualized based on GDI. The parameters setting for the experiments are as follows. For GA, the population size is 400; the maximum generation is 10000000; the mutation probability is 10%. For COIN, the population size is 400; the maximum generation is 10000000; the selection size is 5%; k is 0.1. Each experiment is run 10 times and the reported data is averaged from all runs.

TABLE II. COMPARISION OF THE NUMBER OF UNIQUE SOLUTIONS FOUND AND THE AVERAGE NUMBER OF GENERATION BETWEEN GA AND COIN.

PROBLEM	PSIZE	OPSOL	AVG GEN		AVG SOL	
			GA	*COIN*	*GA*	*COIN*
I	9	8	55	2	8	8
II	13	24	440	9	5	24
III	13	96	430	5	76	96
IV	17	8	496642	39	4	8
V	24	10	968397	46	1	8

Population Size = 400, Maximum Generation Number = 10000000

GA: Mutation Probability = 10 %, COIN: k = 0.1, C = 5%

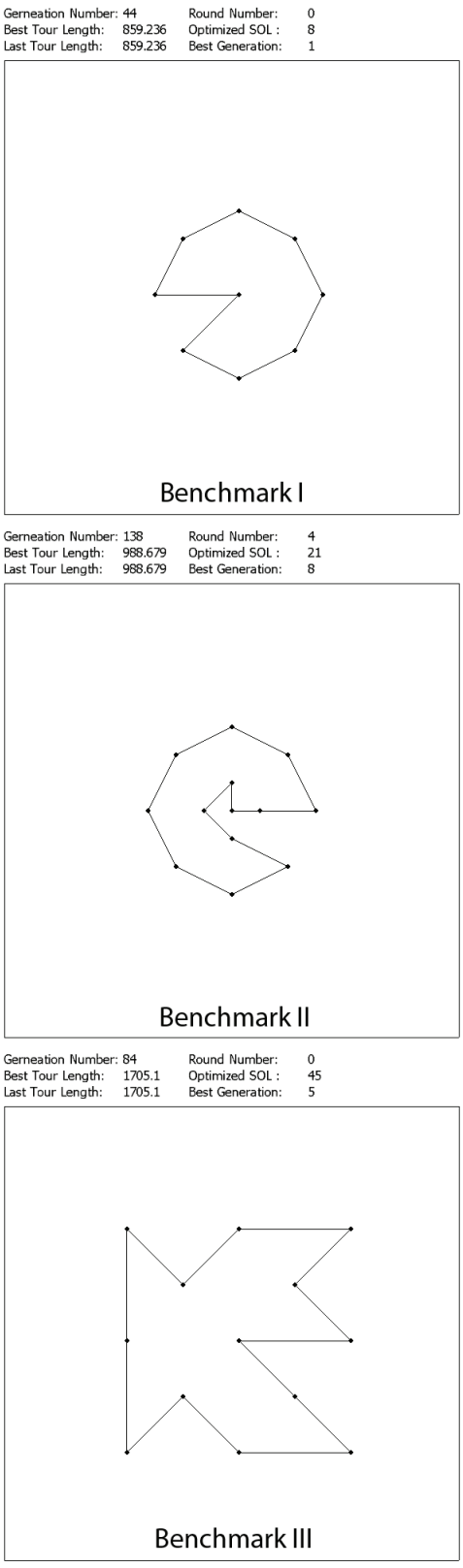

Figure 2. The benhmarks representing the example of Multimodal Traveling Salesman Problem

To compare COIN with GA, the performance measurements are the number of unique solutions found and the average generation. The experiment is run 10 times and the average of data is reported. Table II shows the results from the experiment. PROBLEM denotes the problem number of Multimodal Traveling Salesman Problem; PSIZE denotes the size of the problem, OPSOL denotes the number of possible solution for each of the problems, AVG GEN denotes the last generation number the solution is found, and AVG SOL denotes the number of unique solutions found. Five benchmarks are used for the experiment. Each benchmark is designed to be different from the others. In benchmark I, the problem size is 9, the number of optimized solutions is 8, and a lot of sharing building blocks. In benchmark II, the problem size is 13, and the number of optimized solutions is 24. This problem has a moderate number of sharing building blocks. In benchmark III, the problem size is 13, the number of optimized solutions is 96, and there are a lot of sharing building blocks. In benchmark IV, the problem size is 13, and the number of optimized solutions is 8. In benchmark V, the problem size is larger than the other problems, 24, the number of optimized solutions is 10, and it has different patterns of building blocks.

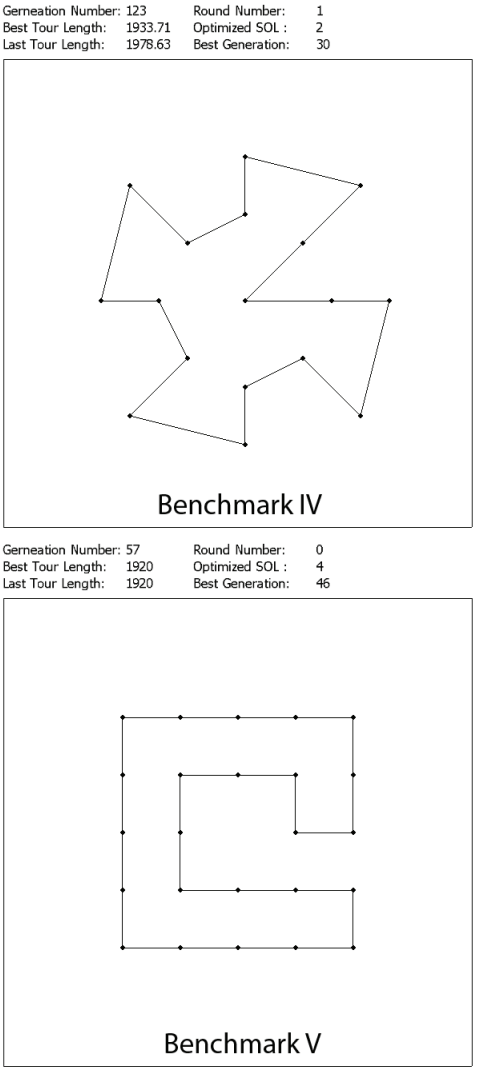

Figure 3. The benchmarks represneting the example of Multimodal Traveling Salesman Problem (Cont.)

For the discussion of the results, the problems are divided into two groups: easy and difficult problems. Benchmark I, II and III are easy because the solutions share a number of building blocks. Benchmark IV and V are hard (by design). According to Table II, GA can solve the problem in benchmark I, II, and III. Comparing to COIN, COIN found more unique solutions and was faster in finding the solutions. Fig. 4 shows examples of the solutions which have a lot of sharing building blocks. In Benchmark IV, the building blocks are designed to be conflicting with each others. GA performs badly on this problem. The example of two solutions which have a different pattern of building blocks is shown in Fig. 5. In the Difficult problems, COIN still performs quite well and finds almost all of the solutions. In the Benchmark V, some of unique solutions cannot be found by COIN, since the solutions have too many conflicting and too few sharing building blocks (Fig. 6).

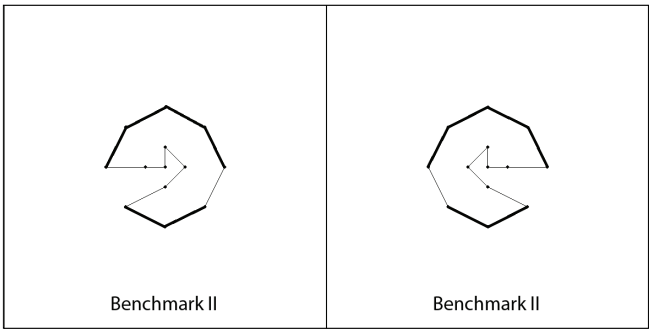

Figure 4. The examples of the best two solutions with a lot of sharing building blocks.

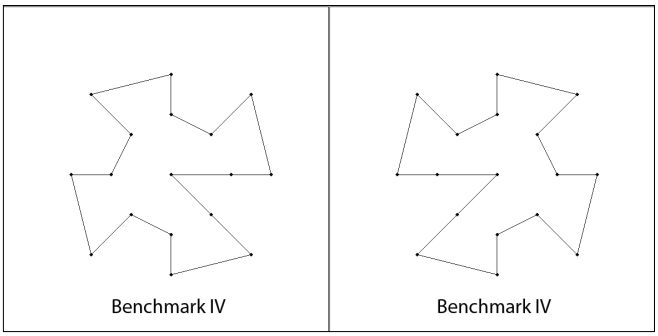

Figure 5. The examples of the best two solutions without sharing building blocks.

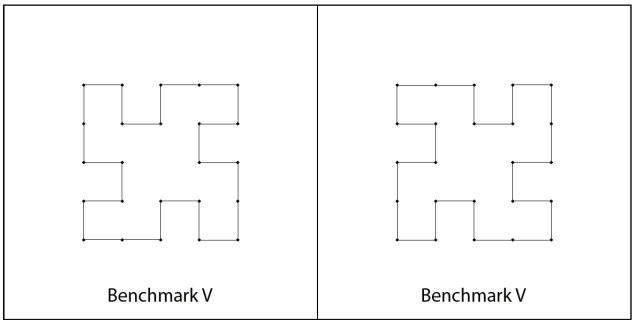

Figure 6. The examples of the two solutions that have very conflicting and few sharing building block

IV. CONCLUSION

Coincidence algorithm has capability to retain multiple models of solution. The central idea of its model, the matrix, provides not only the efficient search that avoids unpromising solutions, but also the ability to find multiple (even conflicting) solutions. The experiments in this work are designed to bring out this behavior. In comparing with Genetic Algorithm, GA struggles with multimodal problems while COIN finds almost all the solutions.

ACKNOWLEDGMENT

We would like to express our gratitude to Warin Wattanapornprom who initiates us to the special characteristic of COIN.

REFERENCES

[1] Wattanapornprom, W., and Chongstitvatana, P., "Multi-objective Combinatorial Optimisation with Coincidence Algorithm," IEEE Congress on Evolutionary Computation, Norway, May 18-21, 2009.

[2] Larrañaga, P., and Lozano, J. A., (Eds.), "Estimation of distribution algorithms: A new tool for evolutionary computation," Kluwer Academic Publishers, Boston, 2002.

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

[4] Wattanapornprom, W., and Chongstitvatana, P., "Solving Multimodal Combinatorial Puzzles with Edge-Based Estimation of Distribution Algorithm," Genetic and Evolutionary Computation Conference (GECCO), July 12-16, 2011, Ireland.

[5] Lalena, M., TSP solver. Online link : http://www.lalena.com/ai/tsp

[6] Watson, R.A., and Pollack, J.B., "Recombination Without Respect: Schema Combination and Disruption in Genetic Algorithm Crossover," in Proc. Genetic Evol. Comput. Conf., 2000 , pp. 112-119.

[7] Sirovetnukul, R., Chutima, P., Wattanapornprom, W., and Chongstitvatana, P., "The Effectiveness of Hybrid Negative Correlation Learning in Evolutionary Algorithm for Combinatorial Optimization Problems," IEEE Int. Conf. on Industrial Engineering and Engineering Management, Singapore, 6-9 Dec 2011.