A Performance Analysis of Compressed Compact Genetic Algorithm

Orawan Watchanupaporn,
Nuanwan Soonthornphisaj, Members, and **Worasait Suwannik**, Non-member

ABSTRACT

Compressed compact genetic algorithm (c^{2}GA) is an algorithm that utilizes the compressed chromosome encoding and compact genetic algorithm (cGA). The advantage of c^{2}GA is to reduce the memory usage by representing population as a probability vector. In this paper, we analyze the performance in term of robustness of c^{2}GA. Since the compression and decompression strategy employ two parameters, which are the length of repeating value and the repeat count, we vary these two parameters to see the performance affected in term of convergence speed. The experimental results show that c^{2}GA outperforms cGA and is a robust algorithm.

Keywords: Compresssed Compact Genetic Algorithm, Compact Genetic Algorithm, Compression Encoding, OneMax, Royal Road, De Jong, Robot arm control programs.

1. INTRODUCTION

Genetic Algorithm (GA) is an algorithm inspired by natural evolution [1]. In order to solve problems, GA encodes a candidate solution in a chromosome. A typical chromosome is in binary encoding. The evolution process of GA aims to explore the search space to find the optimal solution. The fitness of each chromosome is evaluated. Good chromosomes are selected to create the next generation. The cycle of fitness evaluation, selection, and creating the next generation continue until the optimal solution is found or maximum generation is reached. GA has an advantage that it can be used to solve a variety of problems with large search space [2,3].

Traditional GA requires large amount of memory to store population of chromosome and spends quite long time during the evolutional process. We proposed a new algorithm called *compressed compact genetic algorithm* (c^{2}GA) [4]. We found that c^{2}GA consumes less memory and takes less computation time in fitness calculation. In this paper, we investigate the performance of c^{2}GA on OneMax and Royal Road problem in order to analyze its behavior. Furthermore, different compressed encoding format which is the run length encoding method is investigated in our experiments.

The remainder of this paper is organized as follows. Section 2 presents some related works done on compact genetic algorithm. Section 3 gives details about c^{2}GA. Section 4 gives experimental results. Section 5 gives discussions about experimental results. Finally, we conclude our work in Section 6.

Manuscript received on December 7, 2005; revised on March 20, 2006.

The authors are with the Department of Computer Science , Kasetsart University,Bangkok,10900 Thailand; E-mail: orawan@src.ku.ac.th, fscinws@ku.ac.th, worasait.s@ku.ac.th

2. RELATED WORKS

2.1 Compact Genetic Algorithm

To improve the performance of GA, different chromosome encoding techniques are proposed. For example, Harik et al. [5] introduced a compact genetic algorithm (cGA) with uniform crossover (see Figure 1). The algorithm uses a single probability vector to represent the whole population that makes cGA consumes less memory than traditional Genetic Algorithm. Mutation is not used because cGA was designed to model uniform crossover behavior in simple GA. GA needs more memory than cGA since GA uses $n \times l$ bits, whereas cGA requires only $log_2 n \times l$ bits (cGA requires only $L(log_2 N)$ bits of on-chip RAM to represent a population of size N, in which L is the length of each bit string. On the other hand, a standard GA, which explicitly stores the population as a set of bit strings requires $L \times N$ bits).

Aporntewan and Chongstitvatana [6] implemented the cGA on FPGA using Verilog HDL. Hardware implementation of cGA runs 1,000 times faster than running on software executing on a workstation.

Figure 1 shows the cGA algorithm. The first step is to initialize all bits in chromosome to be 0.5. The value 0.5 means that each bit in the chromosome has equal chance to be 1 or 0. Then we randomly generate two individuals from the probability vector. Next, both individuals are evaluated for the fitness value. Note that the individual with higher fitness score is called the winner, whereas the one with the lower score is called the loser. Each number in the probability vector is updated as follows.

- Increase the probability value by $\frac{1}{n}$, if the winners bit = 1 and the losers bit = 0
- Decrease the probability value by $\frac{1}{n}$, if the winners bit = 0 and the losers bit = 1 (n is the population size)

The last step is to check whether the probability

vector has been converged. If not, the evolutional process is continued started from step 2 through step 5.

Some approaches use heuristic function in an evolution process to reduce search space of GA, which make the algorithm quickly converged. In [7, 8], the specific type of crossover that preserves some constraints can beneficially reduce the search space. The result shows that the proposed crossover can find better solution in a flow shop scheduling problem.

Parameters
n : population size
l : chromosome length

1) Initialize probability vector.
for $i := 1$ to l do
$p[i] := 0.5$;

2) Generate two individuals from the vector
$a := generate(p)$;
$b := generate(p)$;

3) Let them compete.
$winner, loser := evaluate(a, b)$;

4) Update the probability vector towards the better individual.
for $i := 1$ to l do
if $winner[i] \neq loser[i]$ then
if $winner[i] = 1$ then $p[i] := p[i] + \frac{1}{n}$
else $p[i] := p[i] - \frac{1}{n}$;

5) Check if the vector has converged.
for $i := 1$ to l do
if $p[i] > 0$ and $p[i] < 1$ then
return to step 2;

6) p represents the final solution.

Fig.1: *Compact Genetic Algorithm.*

2.2 Compressed Encoding

2.21 Encoding used in CompressedGA

Another approach aims to reduce the search space is *compressedGA* [9]. It encodes chromosomes in compressed format. The algorithm can effectively reduces the search space. The result shows that compressedGA solved OneMax and a robotic problem using less fitness evaluation than GA.

Watchanupaporn et al. [4] introduced a new approach in order to improve the performance of cGA by encoding a chromosome in a compressed format in order to reduce the size of the chromosome. This method is called Compressed Compact Genetic Algorithm (c^{2}GA). The advantage of c^{2}GA is that it consumes less memory by representing population as a vector of probabilities. The experimental results show that c^{2}GA requires less fitness evaluations than cGA on OneMax and the Royal road problems by 4 and 9 times, respectively.

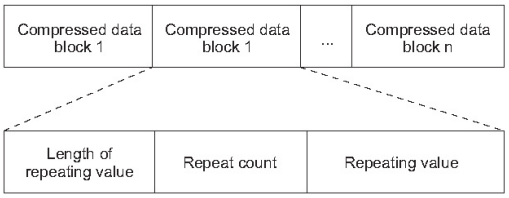

Fig.2: *Compressed format of CompressedGA.*

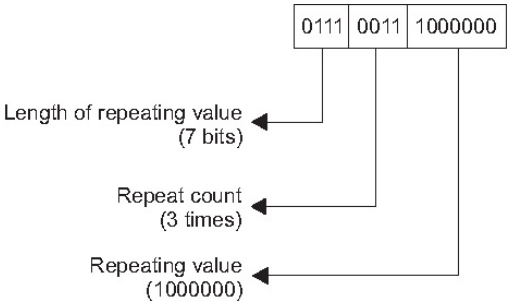

Fig.3: *The compressed chromosome which will be decompressed as 10000001000000001000000.*

Furthermore, c^{2}GAs performance has been investigated using another benchmark called De Jong problem [10]. The experimental results show that c^{2}GA outperforms cGA on function F1 to F4 and quickly converges to the solution since it uses less number of fitness calculation time than cGA.

2.22 Chromosome Encoding

As shown in Figure 2, the compressed chromosome consists of compressed data blocks. Each block has 3 fields: length of repeating value, repeat count and repeating value.

The chromosome in c^{2}GA needs to be decompressed before the fitness evaluation starts. The length of the decompressed chromosome is varied. If the length of decompressed chromosome is longer than the size of the solution encoding, the excess string is discarded. If the length is less than the size of the solution encoding, the zero bits are added (see Figure 3).

3. COMPRESED COMPACT GENETIC ALGORITHM

This section describes our approach in details. Our algorithm (c^{2}GA) applies compressed encoding to cGA.

As shown in Figure 4, the algorithm starts with the probability vector initialization, then two compressed individuals are randomly generated based on

Parameters
n : population size
l : chromosome length

1) Initialize probability vector.
for $i := 1$ to l do
$p[i] := 0.5$;

2) Generate two individuals from the vector
$a := generate(p)$;
$b := generate(p)$;

3) **Decompress chromosome.**

4) Let them compete.
$winner, loser := evaluate(a, b)$;

5) Update the probability vector towards the better individual.
for $i := 1$ to l do
if $winner[i] \neq loser[i]$ then
if $winner[i] = 1$ then $p[i] := p[i] + \frac{1}{n}$
else $p[i] := p[i] - \frac{1}{n}$;

6) **If a and b is not final solution**
return to step 2;

Fig.4: *Compressed Compact Genetic Algorithm.*

the probability values. Our algorithm differs from cGA as follows:

1)The chromosome is decompressed before the fitness evaluation (step 3).

2)The terminating criteria are different. c^{2}GA will terminate when it find the solution. In our experiment, we changed the terminating criteria of the original cGA to be the same as c^{2}GA in order to make a fair comparison.

4. EXPERIMENTAL RESULTS

4.1 Benchmark Problems

The objective of our experiments is to investigate the robustness of c^{2}GA with respect to the compressed encoding parameters. Since the compression format has two important parameters which are the length of repeating value and the repeat count. The ranges of these parameters depend on the number of bits allocated for them. Changing the number of bits might affect the performance of c^{2}GA. We measure the robustness of c^{2}GA using the number of fitness evaluations on OneMax, Royal Road and robot arm control programs. In order to empirically investigate the performance of c^{2}GA, we conducted several experiments using different problems. Four benchmark problems are introduced in our study as follows.

4.11 OneMax problem

The OneMax Problem [11] (or BitCounting) is a simple problem that aims to maximize the number of 1s in a bit string. Formally, this problem can be described as finding a string $\overrightarrow{x} = \{x_1, x_2, \ldots, x_n\}$, with $x_i \in \{0, 1\}$, that maximizes the following function:

$$F(\overrightarrow{x}) = \sum_{i=1}^{N} x_i \tag{1}$$

For this problem, we experimented on different string lengths, which are 128, 256 and 512 bits, respectively.

4.12 Royal Road problem

A simple Royal Road functions [12,13] denoted by R are defined as:

$$R(x) = \sum_{i=1} c_i \delta_i(x), \text{where } \delta_i(x) = \begin{cases} 1 & if x \in s_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

For a problem with block size k, s_i is a schema that have 1 defined in the range $i \times k$ to $((i+1) \times k) - 1$. All other positions contain a wildcard '*'. Each schema s_i is given with a coefficient c_i.

Each fixed-length chromosome consists of n blocks; each block is k bits long. In each block, a candidate bit string makes no contribution to the fitness unless it is a perfect match to the corresponding block on the target. Conventionally, the fitness of a candidate is taken to be the number of such perfectly matched blocks. The objective is to evolve some bit strings to perfectly match the target. We vary the problem size from 60 to 240 bits and use different block sizes which are 2, 4 and 6, respectively.

4.13 De Jong problems

De Jong problems [14] consists of five test functions (F1-F5). These functions represent different levels of difficulty dealing with the optimization.

The test functions F1 to F5 are given by:

$$F_1(\overrightarrow{x}) = \sum_{i=1}^{3} x_i^2 \tag{3}$$

$$F_2(\overrightarrow{x}) = 100 \times (x_1^2 - x_2)^2 + (1 - x_1)^2 \tag{4}$$

$$F_3(\overrightarrow{x}) = \sum_{i=1}^{5} \lfloor x_i \rfloor \tag{5}$$

$$F_4(\overrightarrow{x}) = \sum_{i=1}^{30} i x_i^4 + GAUSS(0, 1) \tag{6}$$

$$\frac{1}{F_5(\overrightarrow{x})} = \frac{1}{K} + \sum_{j=1}^{25} \frac{1}{f_j(\overrightarrow{x})} \tag{7}$$

where $f_j(\overrightarrow{x}) = c_j + \sum_{i=1}^{2}(x_i + a_{ij})^6$

All of the parameter settings can be found in [14]

F1 is the sphere function that smooth, unimodal, and symmetric. It is a simple optimization problem and serves as a basic benchmark for the performance measurement. F2, the Rosenbrock function, has a very narrow ridge that runs around a parabola. F3, the step function, is the representative of optimization problems with flat surfaces that provide no information on which direction is favorable. F4, the random quadratic function is a simple, smooth, and unimodal error function polluted with random noise. F5, the foxholes function, provides an error landscape in which there are many local optima [15].

4.14 Robot arm control programs [9,16]

This problem simulates a robot arm in a working environment. Figure 5 shows the initial robot arm configuration. The three lines are the obstacles for the robots arm. The arrow points to a target. The objective is to moves the tip of robot arm to the target (see Figure 6). The rotation angle for each joint is 15 degrees clockwise and 15 degrees counter clockwise. The arm can sense if it hits an obstacle and knows the distance between the tip and the target. The arm cannot move any of its parts out of the boundary.

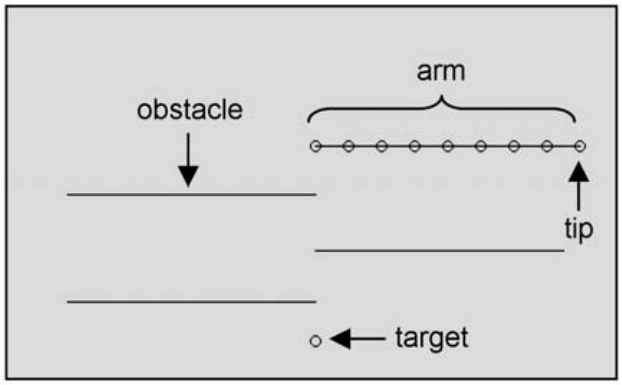

Fig.5: *Initial robot arm configuration.*

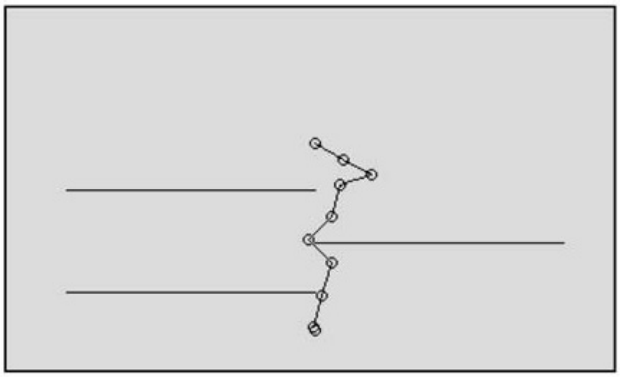

Fig.6: *The robot arm reaches the target.*

4.2 Performance of c^{2}GA compared to cGA

We conducted experiments to compare the performance of c^{2}GA and cGA on OneMax, Royal Road, De Jong and Robot arm control programs. We set the length of repeating value and repeat count to 4. The size of compressed chromosome is one fourth of that of decompressed chromosome for OneMax, Royal Road, and Robot arm control programs. The experimental results show that c^{2}GA outperforms cGA on OneMax and Royal Road problems (see Figures 7 and 8). We found that c^{2}GA used less number of fitness evaluation than cGA.

The experimental results obtained from OneMax and Royal Road problem experiment are averaged over 70 runs. The experimental results obtained from the Robot arm experiment is averaged over 30 runs.

The experimental results show that c^{2}GA requires less fitness evaluations less than cGA. c^{2}GA takes only one fourth and one ninth fitness evaluations of that of cGA to solve OneMax and Royal Royal problems respectively.

The length of the chromosome for each of De Jong problem varies. The length is equal to the number of variables multiply by the number of bits that store data. The length of compressed chromosome is half of that of uncompressed chromosome. The results are averaged over 50 runs. The experimental results show that c^{2}GA outperforms cGA on function F1 to F4. We found that c^{2}GA can quickly converge to the solution and use less number of fitness calculation time than cGA. However, cGA performs better than c^{2}GA in F5 problem. This might be because F5 problem has too many local optima and considered as the most difficult problem (see Table 1).

Table 2 shows the experimental result of cGA and c^{2}GA on Robot arm control programs problem. On average, c^{2}GA can solve the six hundred bits. c^{2}GA can find solution using 2 times less number of fitness calculation than cGA.

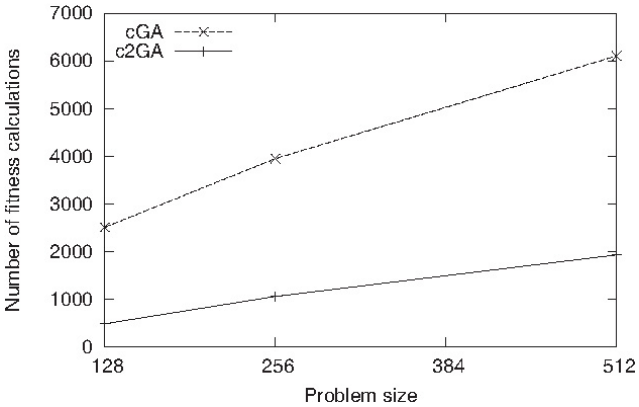

Fig.7: *The plots illustrate the number of fitness calculation times of cGA and c2GA for the various problem sizes on OneMax problem. The solid lines are for c^{2}GA, and the dashed lines are cGA.*

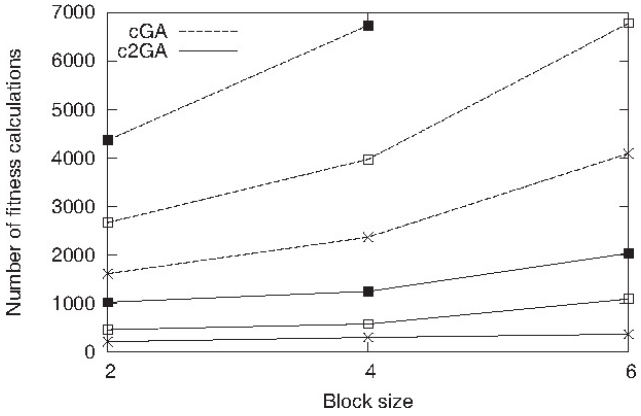

Fig.8: *These plots compare cGA and c2GA on Royal Road problem. The graphs plot the number of fitness calculation times. The solid lines are for c2GA, and the dashed lines are cGA. Problem sizes are 60, 120, and 240 bits, which are marked with crosses, white squares, and black squares respectively.*

Table 1: *The chance in finding solution of cGA and c^2GA on De Jong problem.*

Function	cGA(%)	c^2GA(%)
F1	20	86
F2	66	100
F3	100	100
F4	18	100
F5	96	82

4.3 Analysis of two parameters in c^{2}GA

We study the affect of repeat count (R) and the length of repeating value (L). These parameters were varied on many experiments, which are OneMax, Royal Road, and Robot arm control program problem.

4.3.1 Experimental results on OneMax problem

c^{2}GA was tested against 128, 256 and 512-bit OneMax problems. For each problem setting, we varied the parameters, L and R, from 1 to 10. The result of 128-bit OneMax experiment is shown in Table 3. In Table 4, the column titled "128 bits" summarizes the result in Table 3 by selecting only the best result for each row (or for each length of repeating value (L)). Similarly, the columns titled "256 bits" and "512 bits" in Table 4 summarized the best result in 256 and 512-bit respectively.

Table 2: *Average fitness evaluations of cGA and c^2GA on robot arm control programs.*

Algorithm	Fitness calculations
cGa	6510
c^{2}GA	3184

Table 4: *Performance of c^2GA on OneMax problem with different problem sizes.*

128 bits		256 bits		512 bits	
L, R	Fitness calc	L, R	Fitness calc	L, R	Fitness calc
1,10	3.43	1,10	3.37	1,10	3.40
2,10	4.09	2,10	4.71	2,10	5.57
3,10	7.09	3,10	9.74	3,10	11.80
4,10	14.14	4,10	19.83	4,10	20.66
5,10	30.77	5,10	35.91	5,10	50.43
6,10	59.14	6,10	61.54	6,10	73.57
7,10	94.80	7,10	95.20	7,10	127.60
8,10	164.83	8,10	158.89	8,10	207.97
9,10	238.34	9,10	270.06	9,10	262.20
10,10	379.31	10,10	373.86	10,10	399.49

From the experimental result, we found that the repeat count (R) affects the performance of c^{2}GA: greater repeat count causes less fitness calculation. From Table 4, we found that the smaller number of the length of repeating value (L) can enhance the performance of c^{2}GA in solving OneMax problem.

4.3.2 Experimental results on Royal Road problem

c^{2}GA was tested against 60, 120 and 240-bit Royal Road problems. For each size of problem, we experiment on the problem with block size 2, 4, and 6. For each problem, we varied the parameters, L and R, from 1 to 10 as we did in OneMax problem. The result of 60-bit Royal Road with block size 2 is shown in Table 5. In Table 6, the column titled "block size=2" summarizes the result in Table 5 by selecting only the best result for each row (or for each length of repeating value (L)). Similarly, the columns titled "block size=4" and "block size=6" in Table 6 summarized the best result in block size 4 and 6 on 60-bit Royal Road experiments. Table 7 and 8 show the result of 120 and 240-bit Royal Road experiments respectively.

For Royal Road problem, we found that the larger block size needs more fitness evaluation. The greater repeat count (R) and smaller length of repeating value (L) are preferred. As shown in Tables 6, 7 and 8, when the block size is large, the algorithm needs more fitness calculation times. Considering the problem size, we found that the bigger problem size needs more fitness calculation time.

4.3.3 Experimental results on Robot arm control programs

We conducted another set of c^{2}GA experiment on a simulated robot arm reaching problem. Table 9 shows the number of fitness evaluation for different values of the repeat count (R) and the length of repeating value (L) in the robot arm experiment. The best

Table 3: *Performance of c^2GA on 128-bit OneMax problem. Parameters L and R are varied from 1 to 10. The dash means that the algorithm cannot find a solution after 20,000 fitness evaluations. The bold number is the minimum number.*

	R = 1	R = 2	R = 3	R = 4	R = 5	R = 6	R = 7	R = 8	R = 9	R = 10
L = 1	-	-	-	-	-	84.60	16.71	3.97	3.51	**3.43**
L = 2	-	-	-	829.91	176.66	27.74	9.17	5.80	5.37	4.09
L = 3	-	-	-	516.43	122.71	40.29	16.60	9.54	9.31	7.09
L = 4	-	-	-	473.63	196.37	83.91	41.97	20.60	17.83	14.14
L = 5	-	-	1056.24	933.74	311.31	139.89	81.34	41.77	30.97	30.77
L = 6	-	-	1142.09	934.77	418.37	196.40	121.46	71.83	63.03	59.14
L = 7	-	-	1257.06	781.31	497.43	287.43	161.43	113.91	104.40	94.80
L = 8	-	-	1120.26	757.14	527.51	374.57	268.20	182.09	165.57	164.83
L = 9	-	-	1419.75	818.99	668.60	427.94	402.49	279.49	266.63	238.34
L = 10	-	-	1621.90	984.58	781.59	618.94	530.35	501.11	399.23	379.31

Table 5: *Performance of c^2GA on 60-bit Royal Road problem. Parameters L and R are varied from 1 to 10. The dash means that the algorithm cannot find a solution after 20000 fitness evaluations. The bold number is the minimum number.*

	R = 1	R = 2	R = 3	R = 4	R = 5	R = 6	R = 7	R = 8	R = 9	R = 10
L = 1	-	-	-	-	321.23	43.71	8.43	6.23	4.94	**4.54**
L = 2	-	-	-	-	47.29	13.26	6.89	6.69	5.53	5.32
L = 3	-	-	-	137.37	53.43	27.80	13.11	11.40	10.22	12.28
L = 4	-	-	-	219.54	79.66	51.49	25.79	22.45	23.51	29.17
L = 5	-	-	-	475.00	201.49	100.89	60.00	57.74	69.57	*
L = 6	-	-	-	922.75	483.31	212.09	159.39	197.00	*	*
L = 7	-	-	-	2365.52	1280.45	1114.36	504.71	*	*	*
L = 8	-	-	-	-	2345.50	1935.69	*	*	*	*
L = 9	-	-	-	-	-	*	*	*	*	*
L = 10	-	-	-	-	*	*	*	*	*	*

performance is obtained when $R = 6$ and $L = 5$.

Note that the number of active paths required to solve the problem depends on an instance of the task. Large number of paths does not guarantee high level of robustness. But for the same instance of the problem, a larger number of paths are usually more robust.

4.4 Comparison of Compressed Encoding Formats

Originally, c^{2}GA used a compressed encoding proposed in CompressedGA. However, the algorithm does not require that we have to use such compressed encoding. In this subsection, we conducted the experiment to compare the performance of compressed encoding in solving OneMax and a Robot arm problems. Those compressed encodings are the original compressed encoding and run length encoding (RLE).

RLE is a straightforward way to encode data so that it takes up less space. It is a simple compression algorithm that counts the number of repeating bits in the chromosome. This encoding method is chosen because it fits to our approach which operates on a compressed chromosome and needs the decompression function in order to measure the fitness value. Moreover, this encoding is well known, simple, and widely used in a variety of applications that need the compression mechanism such as computer network and image comparison [17].

In RLE, if the original string is "aaaaabcddccc", for example, the encoder should output a,a,4,b,c,d,d,0,c,c,1. The decompression is the reversed of compression. Here, we used RLE to decompress a binary string. For example, the input and the output of RLE uncompression is shown in Figure 9.

The experimental results show that c^{2}GA using compressedGA format outperforms c^{2}GA using run length encoding format on OneMax and a robot arm problems (see Tables 10 and 11).

5. DISCUSSIONS

By experimenting with various compression parameters in OneMax and Royal Road problems, we found that c^{2}GA performs better if we assign a small number to the repeat length and large number to the number of repeat bits. For example, (see Figure 10) it shows that when we uncompress the first data set,

Table 9: *Performance of c^2GA on robot arm control programs.*

	R = 1	R = 2	R = 3	R = 4	R = 5	R = 6	R = 7	R = 8	R = 9	R = 10
L = 1	-	-	-	-	-	-	-	-	-	-
L = 2	-	-	-	-	5778	6638	6383	-	-	-
L = 3	-	-	10798	11793	1939	1841	2921	2957	2250	1909
L = 4	-	-	12959	3184	1492	1712	1910	1753	2064	1676
L = 5	-	16230	9419	2072	1465	**1261**	1368	1825	1529	1794
L = 6	-	-	10555	2963	2304	1934	2300	2021	3089	2349
L = 7	-	17802	8205	3110	3210	1738	2905	2259	3381	3072
L = 8	-	11606	7447	5547	3780	3440	3253	2187	3334	3514
L = 9	-	11730	6092	5937	4843	3221	5389	3723	3227	2411
L = 10	-	11671	9208	5278	5795	3968	4423	4117	3168	3800

Table 6: *Performance of c^2GA on Royal Road problem (problem size = 60 bits).*

Block size = 2		Block size = 2		Block size = 2	
L, R	Fitness calc	L, R	Fitness calc	L, R	Fitness calc
1,10	4.54	1,10	4.49	1,10	4.83
2,10	5.32	2,10	5.49	2,10	5.77
3,9	10.22	3,9	10.94	3,9	10.45
4,8	22.45	4,8	21.14	4,8	24.97
5,8	57.74	5,8	52.14	5,8	56.17
6,7	159.39	6,7	132.83	6,7	142.34
7,7	642.51	7,7	408.67	7,7	493.47
8,6	1935.70	8,6	1963.50	8,6	1986.74
9	-	9	-	9	-
10	-	10	-	10	-

Table 7: *Performance of c^2GA on Royal Road problem (problem size = 120 bits).*

Block size = 2		Block size = 2		Block size = 2	
L, R	Fitness calc	L, R	Fitness calc	L, R	Fitness calc
1,10	3.11	1,10	3.06	1,9	3.26
2,9	4.54	2,9	3.71	2,9	5.00
3,10	8.00	3,10	8.23	3,10	6.89
4,10	15.77	4,9	17.00	4,10	13.71
5,8	34.57	5,10	35.37	5,10	31.49
6,10	67.46	6,10	71.60	6,10	75.94
7,9	116.97	7,8	118.63	7,10	146.60
8,10	192.31	8,10	255.74	8,10	346.72
9,9	475.45	9,10	569.97	9,9	486.03
10,10	917.23	10,10	947.35	10,10	923.90

Table 8: *Performance of c^2GA on Royal Road problem (problem size = 240 bits).*

Block size = 2		Block size = 2		Block size = 2	
L, R	Fitness calc	L, R	Fitness calc	L, R	Fitness calc
1,10	3.23	1,10	2.74	1,10	3.17
2,10	4.83	2,10	4.74	2,10	3.94
3,9	9.97	3,10	8.49	3,10	9.77
4,10	17.11	4,10	20.77	4,9	18.63
5,10	31.91	5,10	40.34	5,10	40.66
6,10	58.80	6,10	70.11	6,9	69.74
7,10	114.29	7,9	122.51	7,10	151.14
8,10	156.74	8,10	225.57	8,10	272.34
9,10	309.51	9,10	446.66	9,10	595.11
10,10	552.49	10,9	889.43	10,10	1094.75

Table 10: *The average number of fitness calculation times measured by c^2GA on OneMax problem using CompressedGA and Run length encoding.*

Problem size (bits)	Compression Algorithm	
	CompressedGA	Run length encoding
128	2.54	4.20
256	2.83	5.57
512	3.23	5.86

Table 11: *The average number of fitness calculation times measure by c^2GA on robot arm control programs using different encoding formats (compressedGA vs run length).*

Compression algorithm	Fitness calculations
CompressedGA	1,032
Run length encoding	1,246

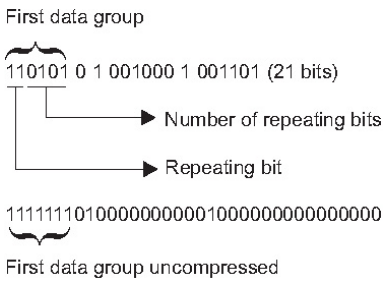

Fig.9: *Uncompression of run length encoding.*

we can get a chromosome with 672 ones which is the solution to 512-bit OneMax or 240-bit Royal Road problems. However, the best number of bits for L and R in the robot problem is different from OneMax and Royal Road. c^{2}GA performs best in the robot problem when L=5 and R=6.

The search space of c^{2}GA is much smaller than that of cGA. However, if the search space is too small, a solution might not exist in this space. For example to solve 128-bit OneMax problem, if the length of the chromosome is 5 bits, the chromosome cannot be decompressed to a solution. In general, if we set the c^{2}GA chromosome too short, it will not be able to find a solution. On the contrary, if we set the length too long, the search space will become larger and it might take longer time to solve a problem.

In the compressed encoding experiments, even though the original compressed encoding outperforms RLE. RLE might be more attractive because it has one less parameter to be adjusted. The number of parameter settings in the original compressed encoding and RLE are different. The original compressed encoding uses three parameters, which are the length of compressed chromosome, the length of L, and R. For RLE, there are two parameters, which are the length of compressed chromosome and the length of number of repeating bits.

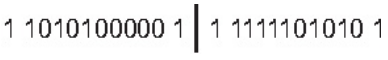

Fig.10: *The example data when the repeat length is 1 and the number of repeat bits is 10.*

6. CONCLUSIONS

c^{2}GA employs the compressed encoding. The main feature of c^{2}GA is an ability to reduce the search space so that it can effectively find a solution. We investigate two parameters used in the compressed chromosome of c^{2}GA and found that the length of L and R affects the performance of c^{2}GA. From the problems that we experimented, the best length of L and R depends on the problems. We also compared CompressedGA encoding and RLE and found that the original encoding outperformed RLE.

ACKNOWLEDGEMENT

We would like to thanks the anonymous reviewer for helpful comments.

References

[1] M. Mitchell, *Genetic Algorithms: An Overview. An Introduction to Genetic Algorithm*, A Bradford book, United Stated of America, 1996.

[2] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, January 1, 1989.

[3] D.E. Goldberg, *The Design of Innovation*, Kluwer Academic Publishers, USA, 2002.

[4] O. Watchanupaporn, W. Suwannik and P. Chongstitvatana, "Compressed Compact Genetic Algorithm," Proceedings of National Computer Science and Engineering Conference (NCSEC), October 27-28, 2005, pp. 801-811. (abstract in English)

[5] G.R. Harik, F.G. Lobo and D.E. Goldberg, "The Compact Genetic Algorithm," IEEE World Congress on Intelligent Control and Automation, pp. 523-528, 1998.

[6] C. Aporntewan and P. Chongstitvatana, "A Hardware Implementation of the Compact Genetic Algorithm," IEEE Congress on Evolutionary Computation Seoul, Korea, May 27-30, pp. 624-629, 2001.

[7] S. Chen and S. Smith, "Improving Genetic Algorithms by Search Space Reduction (with Applications to Flow Shop Scheduling)," GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, 1999.

[8] Z. Yong and N. Sannomiya, "An Improvement of Genetic Algorithms by Search Space Reductions in Solving Large-scale Flowshop Problems," T.IEE Japan. 121-c(6), 2001, pp. 1010-1015.

[9] W. Suwannik, N. Kunasol and P. Chongstitvatana, "Compressed Genetic Algorithm," Proceedings of Northeastern Computer Science and Engineering Conference, March 31-April 1, 2005, pp. 203-211. (abstract in English)

[10] O. Watchanupaporn, W. Suwannik, N. Soonthornphisaj, "Compressed Compact Genetic Algorithm : A Case Study on De Jong Problems," Proceedings of Joint Conference on Computer Science and Software Engineering (JCSSE), November 17-18, 2005, pp. 91-97. (abstract in English)

[11] J.D. Schaffer and L.J. Eshelman, "On crossover as an evolutionary viable strategy," In R.K. Belew and L.B. Booker, editors, Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufmann, 1991, pp. 61-68.

[12] M. Mitchell, J. Holland, S. Forrest, "When Will a Genetic Algorithm Outperform Hill Climbing?," Advances in Neural Information Processing Systems, vol. 6, pages 51-58, 1994.

[13] E. van Nimwegen, J.P. Crutchfield and M. Mitchell, "Statistical dynamics of the Royal Road genetic algorithm," Theoretical Computer Science, Special Issue on Evolutionary Computation, A. E. Eiben and G. Rudolph (eds.), 1999, pp. 41-102.

[14] A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systemsh*, Ph.D. thesis, Michigan University, 1975.

[15] J.C. Gallagher, S. Vigraham, and G. Kramer, "A Family of Compact Genetic Algorithms for Intrinsic Evolvable Hardware," IEEE Transactions on Evolutionary Computation vol. 8 (2004), pp. 111-126, 2004.

[16] C. Jassadapakorn and P. Chongstitvatana, "Reactive Planning with Evolutionary Computation," National Computer Science and Engineering Conference, Pattaya, Thailand, 2002, pp. 357-361.

[17] D. Salomon, *Data Compression*, Springer-Verlag, Inc., New York, 2004.

Orawan Watchanupaporn received a bachelor's degree in Computer Science from Bangkok University in 2004 and master's degree from Kasetsart University in 2006. She is a lecturer at the Department of Computer Science, Kasetsart University, Sriracha Campus, Thailand. Her research interests include the compressed compact genetic algorithms

Nuanwan Soonthornphisaj is an Assistant Professor at the Department of Computer Science at Kasetsart University, Bangkok, Thailand. She received Ph.D. (Computer Engineering) from Chulalongkorn Univeristy in 2002. Her research interests include data mining, machine learning and medical informatics.

Worasait Suwannik received a Ph.D. in computer engineering from Chulalongkorn University, Thailand, in 2006. At present, he is a lecturer at the Department of Computer Science, Kasetsart University, Bangkhen Campus, Thailand. His research interests include evolutionary robotics and compressed genetic algorithms.