A Quantitative Approach for Validating the Building-block Hypothesis

Chatchawit Aporntewan
Department of Mathematics
Faculty of Science, Chulalongkorn University
Bangkok, 10330, Thailand
Chatchawit.A@chula.ac.th

Prabhas Chongstitvatana
Department of Computer Engineering
Faculty of Engineering, Chulalongkorn University
Bangkok, 10330, Thailand
Prabhas.C@chula.ac.th

Abstract- The building blocks are common structures of high-quality solutions. Genetic algorithms often assume the building-block hypothesis. It is hypothesized that the high-quality solutions are composed of building blocks and the solution quality can be improved by composing building blocks. The studies of building blocks are limited to some artificial optimization functions in which it is obvious that the building blocks exist. A large number of successful applications has been reported without a strong evidence that proves the hypothesis. This paper proposes a quantitative approach for validating the building-block hypothesis. We define the quantity of building blocks and the degree of discontinuity by using the chi-square matrix. We test the building-block hypothesis with 15-bit onemax, 5×3-trap, parabola $1-(x^2/10^{10})$, and two-dimensional Euclidian traveling salesman problem (TSP). The building-block hypothesis holds for onemax, 5×3-trap, and parabola. In the case of parabola, Gray coding gives a higher quantity of building blocks than that of binary coding. The hypothesis is accepted for random instances of TSP with a low confidence.

1 Introduction

The building blocks are common structures of high-quality solutions [6, 3]. Genetic algorithms often assume the building-block hypothesis. It is hypothesized that the high-quality solutions are composed of building blocks and the solution quality can be improved by composing building blocks. The studies of building blocks are limited to some artificial optimization functions in which it is obvious that the building blocks exist [7, 14]. A large number of successful applications has been reported without a strong evidence that proves the hypothesis. The genetic algorithms enhanced with building-block identification give promising solutions in real-world applications [11, 4]. However, it is not sufficient to prove the building-block hypothesis. Reaching optimal solution results from a number of parameters that interact with each other, for example, genetic operator, selection method, diversity control, and local search. A fair building-block measurement must decouple itself from the unnecessary parameters that involve finding optimal solutions.

Modern genetic algorithms are capable of identifying and maintaining building blocks [5, 12, 4, 15]. Recent work turns to build a distribution of solutions [13]. The basic concept of the estimated distribution algorithms (EDAs) starts with a uniform distribution of solutions. Next, a number of solutions is drawn according to the distribution. Some good solutions (winners) are selected, and the distribution is adjusted toward the winners (the winner-like solutions will be drawn with a higher probability in the next iteration). These steps are repeated until the optimal solution is found or reaching a termination condition. Building a distribution of solutions involves identifying dependency of solution bits. It can be done by a statistical method, for example, Bayesian network [12]. EDAs focus on the accuracy of building distribution (or identifying the dependency) which is essential to improve solution quality. As mentioned earlier, executing EDAs and obtaining the optimum are not sufficient to prove the building-block hypothesis. In contrast, testing the building-block hypothesis requires measuring the quantity of dependency of solution bits. It will be shown that testing the hypothesis is simple and fast. If the building-block hypothesis holds, one of the EDAs would be applied. Building an accurate distribution needs a great deal of computational time. It might be worth testing the building-block hypothesis before executing EDAs.

This paper proposes a quantitative approach for validating the building-block hypothesis. We define the quantity of building blocks and the degree of discontinuity by using the chi-square matrix [1]. We test the building-block hypothesis with 15-bit onemax, 5×3-trap, parabola $1-(x^2/10^{10})$, and two-dimensional Euclidian traveling salesman problem (TSP). Onemax and trap functions are chosen for evaluation purpose because the building-block information is known beforehand. The quantity of building blocks depends on how the solutions are encoded. In some cases, the performance of genetic algorithms is enhanced by the use of Gray coding [10]. The parabola is tested with binary coding and Gray coding. TSP represents real-world applications. We choose TSP because it is close to practical applications such as vehicle routing, PCB design, X-ray crytallography, etc [8, 2]. The hypothesis is tested for three different codings that are commonly used in the TSP literature.

This paper is organized as follows. Section 2 defines the quantity of building blocks and the chi-square matrix. Section 3 describes a methodology for validating the building-block hypothesis. Section 4 tests the building-block hypothesis with a number of functions. Section 5 concludes the paper.

2 The Quantity of Building Blocks

Building blocks are inferred from a set of solutions. An example of four 15-bit solutions is shown below.

	$v_1v_2v_3$	$v_4v_5v_6$	$v_7v_8v_9$	$v_{10}v_{11}v_{12}$	$v_{13}v_{14}v_{15}$
#1	0 0 0	1 1 1	0 0 0	1 1 1	1 1 1
#2	1 1 1	0 0 0	1 1 1	0 0 0	0 0 0
#3	1 1 1	0 0 0	1 1 1	0 0 0	1 1 1
#4	1 1 1	0 0 0	0 0 0	1 1 1	0 0 0

An inference might be that the above solutions are composed of aligned chunks of 000 and 111. The strong dependency between variables v_i and v_{i+1} and v_{i+2} where $i = 1, 4, 7, 10, 13$ can be detected by a statistical method. We do not give a precise definition of building blocks, but measuring *the quantity of building blocks* is possible.

Given a set of solutions, the quantity of building blocks ranges between min and max. min means no building blocks. This implies that the solutions are random because random bits do not have common structures. max indicates that the solutions are absolutely not random. For instance, all solution bits are zero. The quantity of building blocks inversely relates to randomness. We choose the chi-square matrix for measuring randomness because computing the matrix is simple and fast [1].

Let $M = (m_{ij})$ be an $\ell \times \ell$ symmetric matrix of numbers. Let P be a population or a set of ℓ-bit binary strings. The chi-square matrix is defined as follows.

$$m_{ij} = \begin{cases} ChiSquare(i,j) & ; \text{ if } i \neq j \\ 0 & ; \text{ otherwise.} \end{cases} \quad (1)$$

The $ChiSquare(i,j)$ is defined as:

$$\sum_{xy} \frac{(Count_P^{xy}(i,j) - n/4)^2}{n/4},\ xy \in \{00, 01, 10, 11\} \quad (2)$$

where the observed frequency $Count_P^{xy}(i,j)$ counts the number of solutions in which bit i is identical x and bit j is identical to y. The expected frequencies of observing "00," "01," "10," "11" are $n/4$ where n is the number of solutions. If the solutions are random, the observed frequencies are close to the expected frequencies and therefore the chi-square is low. The quantity of building blocks is defined as the sum of all matrix elements. Since the matrix is symmetric, we sum only a half of the matrix. The quantity of building blocks given a population, P, is defined as follows.

$$Q_P = \sum_{ij} m_{ij},\ i < j \quad (3)$$

The time complexity of computing the quantity of building blocks is $O(\ell^2 n)$.

It is important to note that the chi-square matrix deals only with second-order interactions or marginal dependency. There exist also higher-order interactions that play a crucial role in optimization. However, our goal is not to build an accurate distribution of solutions. A sum of every pairwise couplings is sufficient for measuring the quantity of building blocks, because the sum counts the quantity of higher-order interactions. For examples:

	v_1	v_2	v_3	v_4
#1	0	0	0	0
#2	0	0	1	1
#3	1	1	0	0
#4	1	1	1	1

	v_1	v_2	v_3	v_4
#1	0	0	0	0
#2	0	0	0	0
#3	1	1	1	1
#4	1	1	1	1

The set of four 4-bit solutions on the left represents the second-order interactions of v_1 and v_2, v_3 and v_4. There is no dependency between v_1 and v_3, v_1 and v_4, v_2 and v_3, v_2 and v_4 because the observed frequencies are identical to the expected frequencies. The quantity of building blocks is $ChiSquare(1,2) + ChiSquare(3,4) = 4 + 4 = 8$. The solutions on the right represent the higher-order interactions of v_1, v_2, v_3, v_4. The quantity of building blocks is $ChiSquare(1,2) + ChiSquare(1,3) + ChiSquare(1,4) + ChiSquare(2,3) + ChiSquare(2,4) + ChiSquare(3,4) = 4+4+4+4+4+4 = 24$. The solutions on the right give a higher quantity of building blocks because of the high-order interactions.

3 A Methodology for Validating the Building-block Hypothesis

In the previous section, we have defined the quantity of building blocks that is a fundamental tool for validating the building-block hypothesis. The building-block hypothesis are separated in two hypotheses. The first hypothesis states that the high-quality solutions are composed of building blocks. The second hypothesis states that the solution quality can be improved by composing building blocks. We begin with a methodology for validating the first hypothesis. Then, a different methodology for validating the second hypothesis will be presented.

Let $F : \{0,1\}^\ell \rightarrow \mathbb{R}$ be a fitness function. Let $P_1, \ldots, P_m$ be m sets of n high-quality solutions. Let $T_1, \ldots, T_m$ be m fitness thresholds such that $T_1 < \ldots < T_m$. Each set P_i is made by randomly choosing n solutions of which their fitness is greater than or equal to T_i. The sets of solutions P_1 to P_m simulate a sequence of evolving populations generated by an optimization algorithm. We do not execute an actual optimization algorithm because we want to decouple our methodology from unnecessary parameters such as genetic operator, selection method, diversity control, and local search.

The quantity of building blocks is computed for all P_1 to P_m. The result is shown in Figure 1. min and max are the minimum and the maximum quantity of building blocks. min is the quantity of building blocks in a random population. max is the quantity of building blocks in a population in which all bits are zero. We normalize min and max to 0 and 1, respectively. The first hypothesis holds if and only if the quantity of building blocks does not fall within the rejection area (being closer to min means more randomness). Drawing the plot in Figure 1 requires some parameters. Here is a guideline for parameter settings.

Population size. If the population size is too small, the gap between min and max is narrowed. Subsequently, the resolution is not enough for distinguishing between a random population and a nonrandom population. The min and max should be calculated to make sure that the chosen population size gives a large resolution.

Fitness thresholds. T_1 should not be too small because it makes P_1 random (falling within the rejection area). The first hypothesis involves high-quality solutions. T_1 should

be set at a fitness value that indicates "high quality". A rule of thumb is to set P_1 a little bit greater than the average fitness of a random population. T_m can be at most the optimum. Increasing the number of fitness thresholds, m, likes zooming in the plot. Setting m to be too small makes loss of visual information because of zooming out. It is difficult to make a good-looking plot at the first time without any prior knowledge about the fitness function. However, the sweet spot for setting m is large for trial-and-error method.

Rejection area. The first hypothesis is rejected if the quantity of building blocks is close to min (falling within the rejection area). It is difficult to draw a line that divides building blocks and randomness because the quantity of building blocks is defined over the whole population. However, we can test the randomness separately for every pairwise coupling. There are $\ell(\ell-1)/2$ pairwise couplings (the number of elements in the upper triangle of the chi-square matrix) where ℓ is the number of solution bits. The chi-square test is a test procedure for studying random data [9]. It is summarized as follows. We observe a solution at bit i and bit j. The number of observations is set at a fairly large number such that an expected frequency is five or more. An observation falls into four categories (three degrees of freedom) that are "00," "01," "10," "11." We compute the chi-square value (an element of the chi-square matrix). The selected percentage points of the chi-square distribution is shown below.

Chi-square value	Percentage point
11.345	99%
7.815	95%
6.251	90%

If the chi-square value is greater than 99% entry, the observed data is not sufficiently random. If the chi-square value lies between the 95% and 99% entries, the observed data is suspect. If the chi-square value lies between the 90% and 95% entries, the observed data might be almost suspect. Otherwise, the observed data passes the randomness test. We also reject the building-block hypothesis if more than a half of all pairwise couplings pass the randomness test.

The number of conducting tests. As testing the hypothesis is probabilistic, the hypothesis is accepted by a majority vote or an average quantity of building blocks. The number of conducting tests is up to your desired confidence in accepting the hypothesis.

Generating solutions, P_i, of which their fitness is greater or equal to a constant, T_i, is not obvious. Fortunately, we can choose a small problem size so that we can enumerate all possible solutions (for example, choosing 8-city TSP rather than 100-city TSP). Intuitively, if the hypothesis holds for 8-city TSP, the hypothesis would hold for 100-city TSP. Enumerating all possible solutions may not be required if there is an effective method for generating a set of solutions that is subject to a fitness constraint.

The second hypothesis states that the solution quality can be improved by composing building blocks. In other words, the common structures of P_k are similar to that of P_{k+1} because the population P_{k+1} is produced by composing building blocks from the previous population P_k. The second hypothesis would be accepted if there exist a similarity or

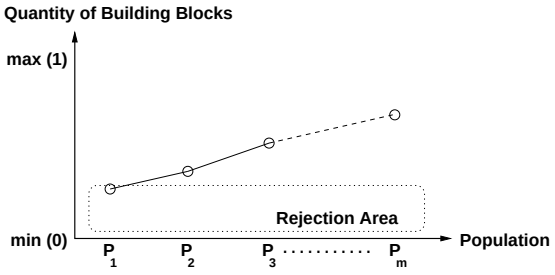


Figure 1: Quantity of building blocks.

continuity between P_k and P_{k+1}. To validate the second hypothesis, we define the degree of discontinuity between two populations. The degree of discontinuity between P_k and P_{k+1} is defined as:

$$D_{P_k,P_{k+1}} = \sqrt{\sum_{ij}(m_{ij}^k - m_{ij}^{k+1})^2},\ i<j \qquad (4)$$

where (m_{ij}^k) is the chi-square matrix of P_k and (m_{ij}^{k+1}) is the chi-square matrix of P_{k+1}. The degree of discontinuity is plotted in Figure 2. min and max are the minimum and the maximum degree of discontinuity. min is always zero (P_k and P_{k+1} are identical). max is the degree of discontinuity between a random population and a population in which all bits are zero. We normalize max to 1. The second hypothesis holds if and only if the degree of discontinuity does not fall within the rejection area (being close to max means a drastic change of common structures between adjacent populations). The other parameter settings are similar to that of the first hypothesis.

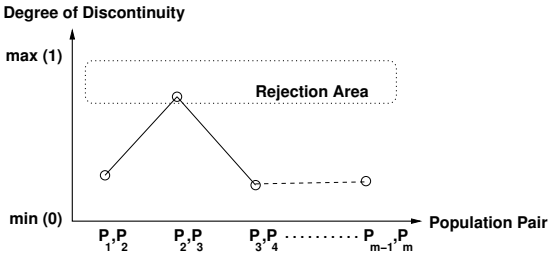


Figure 2: Degree of discontinuity.

4 Testing the Building-block Hypothesis

This section tests the building-block hypothesis with 15-bit onemax, 5×3-trap, parabola, and two-dimensional Euclidian TSP. The 15-bit onemax $F_{\text{onemax}} : \{0,1\}^{15} \rightarrow \{0,\ldots,15\}$ is defined as:

$$F_{\text{onemax}} = \sum_{i=1}^{15} b_i \qquad (5)$$

where b_i is the i^{th} bit of the solution. The 5×3-bit trap $F_{5\times3} : \{0,1\}^{15} \rightarrow \{0,\ldots,15\}$ is defined as:

$$F_{5\times3}(B_1B_2B_3B_4B_5) = \sum_{i=1}^{5} F_3(B_i) \qquad (6)$$

where $B_i \in \{0,1\}^3$. F_3 denotes 3-bit trap $F_3 : \{0,1\}^3 \rightarrow \{0,1,2,3\}$ that is defined as:

$$F_3(b_1b_2b_3) = \begin{cases} 3 & ; \text{ if } u = 3 \\ 2-u & ; \text{ otherwise,} \end{cases} \quad (7)$$

where $u = \sum b_i$ and $b_i \in \{0,1\}$. The quantity of building blocks and the degree of discontinuity are shown in Figure 3 (see also Table 2). The building-block hypothesis holds for 15-bit onemax and 5×3-trap because of high quantity of building blocks and low degree of discontinuity. In addition, the number of pairwise couplings that pass the randomness test drops to zero. It is clear that the populations are composed of building blocks and not random. It may seem counter-intuitive that onemax (first-order interaction) gives a higher quantity of building blocks than that of 5×3-trap (third-order interaction). The high-quality solutions of onemax are composed of "1" more than "0." The 5×3-trap solutions are aligned chunks of "000" and "111." The onemax and 5×3-trap solutions of which the fitness is 12 are shown below.

	15-bit onemax	5×3-trap
#1	000111111111111	000000000111111
#2	001011111111111	000000111000111
#3	001101111111111	000000111111000
#4	001110111111111	000001111111111
#5	001111011111111	000010111111111

By the chi-square definition, the onemax solutions are more dependent (less random). The quantity of building blocks is 1143 and 720 for onemax and 5×3-trap, respectively. The result is counter-intuitive.

The parabola $F_{\text{parabola}} : \{0,1\}^{32} \rightarrow \mathbb{I}$ is defined as:

$$F_{\text{parabola}}(x) = 1 - \frac{x^2}{10^{10}}. \quad (8)$$

There are two alternatives for encoding solutions, the binary coding and Gray coding. The 3-bit coding is shown in Table 1 (actual codings are 32 bits). The quantity of building blocks and the degree of discontinuity are shown in Figure 4 (see also Table 3). It is obvious that Gray coding gives a higher quantity of building blocks. The high discontinuity of Gray coding is typical for the quantity of building blocks that increases sharply. The building-block hypothesis is rejected for binary coding because of the low quantity of building blocks and more than a half of pairwise couplings passing the randomness test. On the other hand, the hypothesis is accepted with a high confidence for Gray coding.

We randomize an instance of TSP (Figure 7). The number of cities is fixed at eight. Each city is randomly placed on a 10×10 grid. The cost of traveling from city i to j is identical to the distance between i and j. The fitness of a solution is 100 subtracted by tour length. There are three codings that are commonly used. The first coding, each city is tagged with a 3-digit binary number ("000" to "111"). The first three bit of a binary solution is the starting city. The next city the salesman visits is the next three bits. The second coding represents a tour by a matrix. If the salesman visits city i before j, the matrix element at row i and column j is one. Otherwise, the matrix element is zero. The third coding is very similar to the second. If the salesman travels from city i to j, the matrix element at row i and column j is one. Otherwise, the matrix element is zero. The first coding results in 24-bit solutions, but the second and the third coding results in 56-bit solutions. We do not count the binary strings that are invalid tours. The expected frequencies of observing a pairwise coupling being "00," "01," "10," "11" are not identical for TSP codings. Equation 2 assumes the identical expected frequencies. In the case of TSP codings, the $ChiSquare(i,j)$ is defined as follows.

Table 1: Binary coding and Gray coding.

Solution	Binary coding	Gray coding
-4	100	110
-3	101	111
-2	110	101
-1	111	100
+0	000	000
+1	001	001
+2	010	011
+3	111	010

$$\frac{(O_{00}-E_{00})^2}{E_{00}} + \frac{(O_{01}-E_{01})^2}{E_{01}} + \frac{(O_{10}-E_{10})^2}{E_{10}} + \frac{(O_{11}-E_{11})^2}{E_{11}} \quad (9)$$

The O_{xy} and E_{xy} are the observed frequencies and the expected frequencies, respectively. The expected frequencies, E_{xy}, are computed by enumerating all possible solutions. The quantity of building blocks and the degree of discontinuity are shown separately in Figure 5 and 6 due to the different lengths in codings (see also Table 4 and 5). The building-block hypothesis is rejected for the first and the second codings because the quantity of building blocks is obviously low. Plus more than a half of pairwise couplings pass the randomness test. The solutions encoded by the first and the second codings are likely random bits rather than composing of some common structures. The third coding gives a higher quantity of building blocks that distinguishes the third coding from the others. However, the quantity of building blocks is a little bit lower than that of 5×3-trap function. The 5×3-trap is invented to have a sufficient amount of building blocks. In terms of quantity of building blocks, the hypothesis would be accepted but more than 40% of pairwise couplings pass the randomness test. For the third coding, we accept the hypothesis with a low confidence. The main point might not be accepting or rejecting the hypothesis. Figure 5 and 6 suggest that the third coding is superior to the others. The third coding would be the best choice if we are going to solve TSP with an optimizer that exploits common structures.

In practical optimization problems, there are many choices for fitness function and coding. Designing fitness function and coding is problematic because it is evaluated by the end result (the quality of a best-so-far solution). The solution quality is affected by many parameters that interact with each other. Therefore it is difficult to find out a proper fitness function and a proper coding. Our method for testing the building-block hypothesis is separated from the

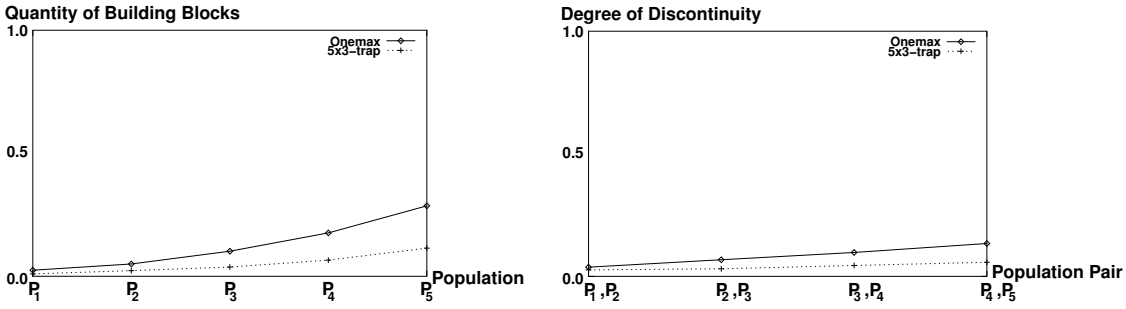


Figure 3: Quantity of building blocks (left) and degree of discontinuity (right). The fitness functions are 15-bit onemax and 5×3-trap. The population size is set at 100. Each point is averaged from 10 independent runs. The fitness thresholds are 8, 9, 10, 11, 12 (see also Table 2).

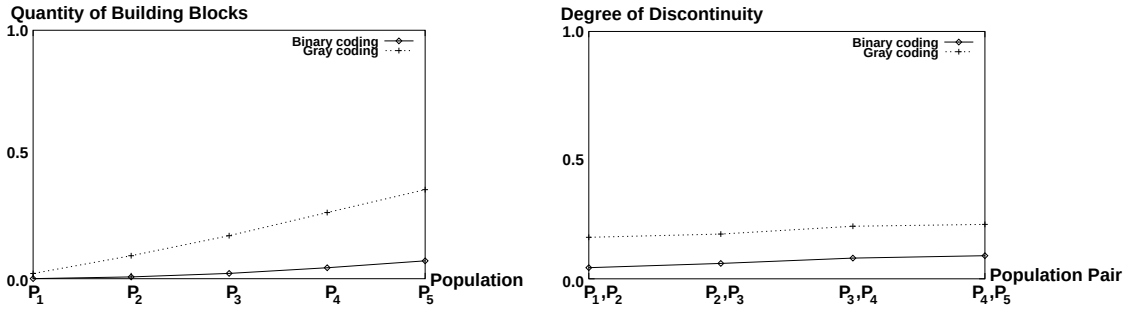


Figure 4: Quantity of building blocks (left) and degree of discontinuity (right). The fitness function is parabola $(1-x^2/10^{10})$ with binary and Gray codings. The population size is set at 500. Each point is averaged from 10 independent runs. The fitness thresholds are 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 0 (see also Table 3).

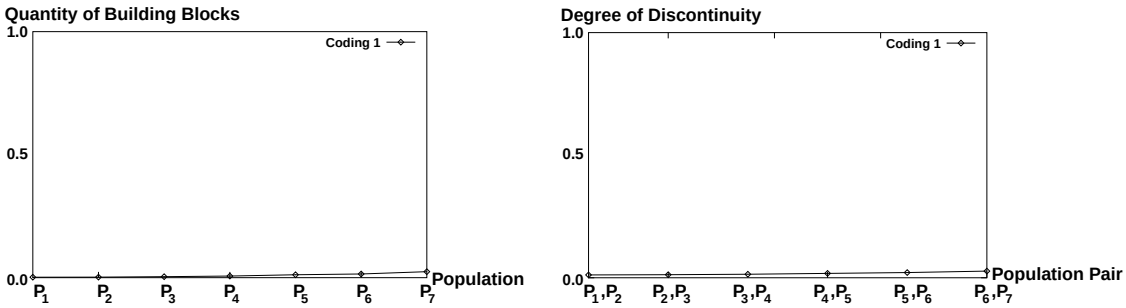


Figure 5: Quantity of building blocks (left) and degree of discontinuity (right). The fitness function is 100 subtracted by a tour length (coding 1). The population size is set at 500. Each point is averaged from 10 independent runs (an instance of TSP is randomized every run). The fitness thresholds are 45, 48, 51, 54, 57, 60, 63 (see also Table 4).

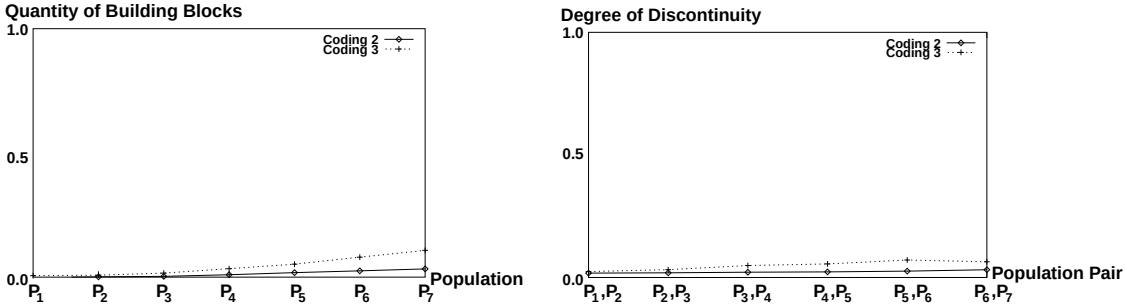
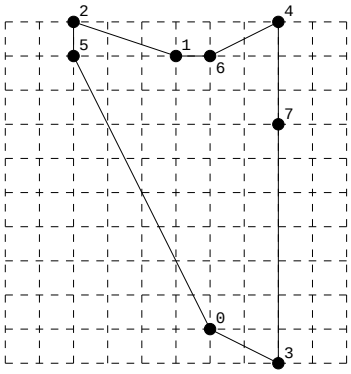


Figure 6: Quantity of building blocks (left) and degree of discontinuity (right). The fitness function is 100 subtracted by a tour length (codings 2 and 3). The population size is set at 500. Each point is averaged from 10 independent runs (an instance of TSP is randomized every run). The fitness thresholds are 45, 48, 51, 54, 57, 60, 63 (see also Table 5).



Coding 1:

000 101 010 001 110 100 111 011

Coding 2:

	0	1	2	3	4	5	6	7
0	X	1	1	1	1	1	1	1
1	0	X	0	1	1	0	1	1
2	0	1	X	1	1	0	1	1
3	0	0	0	X	0	0	0	0
4	0	0	0	1	X	0	0	1
5	0	1	1	1	1	X	1	1
6	0	0	0	1	1	0	X	1
7	0	0	0	1	0	0	0	X

Coding 3:

	0	1	2	3	4	5	6	7
0	X	0	0	0	0	1	0	0
1	0	X	0	0	0	0	1	0
2	0	1	X	0	0	0	0	0
3	1	0	0	X	0	0	0	0
4	0	0	0	0	X	0	0	1
5	0	0	1	0	0	X	0	0
6	0	0	0	0	1	0	X	0
7	0	0	0	1	0	0	0	X

Figure 7: Three codings for TSP. A tour (left) is encoded to a binary string and matrices (right). The tour length is 28.58 and the fitness is 100 - 28.58 = 71.42.

Table 2: Averaged number of pairwise couplings that pass the randomness test ($max = 105$). The fitness functions are 15-bit onemax and 5×3-trap. This table corresponds to the data in Figure 3.

Fitness function	Averaged number of pairwise couplings that pass the randomness test				
	P_1	P_2	P_3	P_4	P_5
Onemax	26%	3%	0%	0%	0%
5×3-trap	72%	45%	21%	3%	0%

Table 3: Averaged number of pairwise couplings that pass the randomness test ($max = 496$). The fitness functions is parabola ($1 - x^2/10$) with binary and Gray codings. This table corresponds to the data in Figure 4.

Coding	Averaged number of pairwise couplings that pass the randomness test				
	P_1	P_2	P_3	P_4	P_5
Binary	88%	86%	76%	77%	67%
Gray	80%	58%	41%	36%	22%

Table 4: Averaged number of pairwise couplings that pass the randomness test ($max = 276$). The fitness functions is 100 subtracted by tour length (coding 1). This table corresponds to the data in Figure 5.

Coding	Averaged number of pairwise couplings that pass the randomness test						
	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1	91%	90%	84%	80%	70%	67%	59%

Table 5: Averaged number of pairwise couplings that pass the randomness test ($max = 1540$). The fitness functions is 100 subtracted by tour length (codings 2 and 3). This table corresponds to the data in Figure 6.

Coding	Averaged number of pairwise couplings that pass the randomness test						
	P_1	P_2	P_3	P_4	P_5	P_6	P_7
2	93%	92%	92%	92%	89%	86%	83%
3	88%	86%	81%	73%	65%	53%	40%

optimization. A large number of fitness functions and codings can be tested in a short time by setting a small problem size.

5 Conclusion

Two important measurements are defined, the quantity of building blocks and the degree of discontinuity. We show that the building-block hypothesis can be tested for a small problem size. The building-block hypothesis holds for 15-bit onemax, 5×3-trap, and parabola (Gray coding). For TSP with the third coding, the hypothesis is accepted with a low confidence. Future work will be to explore the quantity of building blocks in a wide range of real-world applications. The final outcome is a number of problem domains for which the building-block identification and composition are applicable.

Bibliography

[1] C. Aporntewan and P. Chongstitvatana. Chi-square matrix: An approach for building-block identification.

In *Proceedings of 9th Asian Computing Science Conference*, pages 63–77, 2004.

[2] B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. In *Proceedings of 4th Parallel Problem Solving from Nature Conference*, pages 616–621, 1996.

[3] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[4] D. E. Goldberg. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA, 2002.

[5] G. R. Harik and D. E. Goldberg. Learning linkage. In *Foundations of Genetic Algorithms 4*, pages 247–262. Morgan Kaufmann, San Francisco, CA, 1997.

[6] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

[7] T. Jones. A description of Holland's royal road function. *Evolutionary Computation*, 2(4):409–415, 1994.

[8] S. Jung and B. Moon. Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Euclidian TSP. *IEEE Transaction on Evolutionary Computation*, 6(6):557–564, 2002.

[9] D. E. Knuth. *The Art of Computer Programming*, volume 2 / Seminumerical Algorithms. Addison-Wesley, Reading, MA, 2 edition, 1938.

[10] K. E. Mathias and L. D. Whitley. Transforming the search space with Gray coding. In *Proceedings of Evolutionary Computation*, pages 513–518, 1994.

[11] M. Pelikan and D. E. Goldberg. Hierarchical BOA solves Ising Spin Glasses and MAXSAT. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1271–1282, 2003.

[12] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 525–532, 1999.

[13] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[14] R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms. In *Proceedings of Congress on Evolutionary Computation*, pages 1406–1413, 1999.

[15] T. Yu and D. E. Goldberg. Dependency structure matrix analysis: Off-line utility of the dependency structure matrix genetic algorithm. In *Proceedings of Genetic and Evolutionary Computation*, pages 355–366, 2004.