Practical Search Index as a Hardness Measure for Genetic Algorithms

Zhenhua Li
School of Computer Science
China Univ. of Geosciences
Wuhan 430074, China
zhli@cug.edu.cn

Erik D. Goodman
Dept. of Electrical and Computer Engineering
2120 Engineering Building
Michigan State Univ., E. Lansing, MI 48824
goodman@egr.msu.edu

ABSTRACT

Use of the Building Block Hypothesis to illuminate GA search behavior, as pursued by J. H. Holland and D. E. Goldberg, invites additional investigation.This paper investigates the space actually searched by a GA, in light of the Building Block Hypothesis, GA sampling and population size, in an effort to develop more quantitative measures of GA hardness for problems where building block sizes can be estimated. A Practical Search Index (PSI) is defined, related to the size of the space actively searched by the GA, in terms of sizes and numbers of building blocks. The results of the analysis suggest that hardness depends strongly on the sizes of the largest building blocks, premature convergence prevails when population size is not big enough to allow sampling and assembly of building blocks, and appropriate sizing depends on balancing the BB sampling and mixing costs. A set of simple GA experiments on classical test functions at various population sizes, illustrates the relationship between the PSI, population size, and efficiency of search.

General Terms

Algorithms, Performance

Keywords

genetic algorithm, building blocks, search space, practical search index,building block sampling, population size, GA hardness

[0]This paper is an extension of the short paper [14] presented at the the 10th annual conference on Genetic and evolutionary computation

1. INTRODUCTION

When solving some problems by search algorithms, we are usually looking for some solutions, which will be the best among others.Here, the set of all possible solutions is called search space. However, nearly all search algorithms won't look through the whole space,except the brute-force search or method of exhaustion. Any Genetic Algorithms(GA,or Evolutionary Algorithms,EA) will not normally search the whole space either. For Example, the GA search on a One-Max problem will never sample the all-"0" string unless it happens to be created in the initial stage.

Currently,numerous GA research aimed to shrink its search or reduce the search space, in order to make the GA fast, but so far, there is seldom an approach to tell how much the space actually searched by a GA is, and brings the answer for the following questions depends on experiences rather than theoretical analysis.

1. Why some problems is GA-hard, while the others are not. What does contribute the hardness?
2. How to measure to GA-hardness if the landscape is unknown?
3. How much the population size is big enough to find the best out?

Before these questions are discussed, we should firstly investigate the Building Block Hypothesis(BBH), so-called footing stone of GA. Building Blocks(BBs)is declared by John H. Holland as a ubiquitous feature at all levels of human understanding, from perception through science and innovation; and genetic algorithms are designed to exploit this prevalence[9, 5].And the concept has been shown to be useful in numerous experiments, with many GA variants created based on it. For example, messy GA (mGA) [7], and its later improvements, fast mGA [6] and Gene Expression mGA [12], generate all substrings up to a certain length during the initialization procedure, then filter out Building Blocks; The cohort genetic algorithm[9] is designed both to explore search spaces for building blocks and to exploit building blocks already found.

The above works have concentrated mainly on how to find and maintain the BBs, in order to speed GA search; however, apart from Holland's schema theorem and Building Block Hypothesis, few papers discuss BB theory or explain why

doi:10.4304/jcp.8.8.2034-2041

we should track them — in other words, how to explain the GA search mechanism from a BB viewpoint. Though some works [20, 15] considers GA hardness, they didn't involve the the BBH. In this paper, in light of the BBs Hypothesis, we shall define a Practical Search Index (PSI), in order to seek insight into the fraction of the sample space actually visited in a typical GA search, as a first step. Then we reexamine BB theory, explain the GA adaptive mechanism,review GA sampling, and discuss sizing of populations.

The rest of the paper is organized as follows: Firstly, the Building Block Hypothesis is reviewed and questioned as mostly descriptive rather than quantitative in Section 2; Secondly the Practical Search Index(PSI) is defined on the analysis of Sample Space and Practical Search Space in the following section; Then GA adaptive mechanism is discussed both in GA sampling and population size; And a couple of experiments are conducted in Section 5, in order to test the theory of PSI and figure out the relation between the population size and BB construction; Finally, conclusions are drawn in Section 6.

2. THE BUILDING BLOCK HYPOTHESIS: REVIEW

The Building Block Hypothesis could be seen as rooted in Fisherąŕs theories of sexual evolution: small groups of closely located co-adaptive alleles perhaps propagate within an evolving population of genomes in much the same way that single adaptive alleles do [3]. To extend this idea into genetic algorithms, Goldberg[5] formulated the Building Block Hypothesis, in his 1989 book, as follows.

"Short, low order, and highly fit schemata are sampled, recombined, and resampled to form strings of potentially higher fitness. In a way, by working with these particular schemata [the building blocks], we have reduced the complexity of our problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings."

This theory set up a roadmap for an efficient GA search for many types of problems: namely, creating, growing and mixing BBs until finding a solution, in both difficult real-world problems and in "toy problems." As Goldberg stated[5]:

"...the building block hypothesis has held up in many different problem domains. Smooth, unimodal problems, noisy multimodal problems, and combinatorial optimization problems have all been attacked successfully using virtually the same reproduction-crossover-mutation GA."

Obviously, Building Block theory [8, 5, 16, 9] provides an often successful guide for GA design and clearly explains how and why a GA works quickly on many problems. However, this theory, to date, is mostly descriptive rather than quantitative. We try to quantitatively or at least semi-quantitatively analyze the theory, with respect to the following topics:

1. The biasing of search based on BB theory and calculation of a practical index of search difficulty.
2. Analysis of a GA sampling in a small population: is it sampling individuals or BBs?
3. Population size effects. What is the appropriate population size for a GA, as a function of GA-hardness of the problem.

3. SAMPLE SPACE, PRACTICAL SEARCH SPACE, AND PRACTICAL SEARCH INDEX

We argue that why GA is efficient is that it only searches part of the whole problem space, but up to date, no one define how much the search space is. We firstly discuss the sample space and the practical search space, then give the definition of PSI.

3.1 GA Search Space

When we refer to the search space or sample space of a GA, we refer to the space of all possible samples. For example, the search space for a binary string of length l is 2^l. But we argue that any GA will not normally search the whole space. The GA search on a One-Max problem will almost never sample the all-"0" string unless it happens to be created in the initial stage (and that probability is also small). And indeed, for any biased search algorithm, only a small fraction of the sample space is normally searched (were that not true, enumeration would be superior to GA search for such problems). We will call that subset the Practical Search Space (PSS).

Here we define the *s*ample space and the *p*ractical search space as follows.

Definition 1. The *s*ample space S_s is the set of all possible instances, S_s, of cardinality n^l, for a string of length l with n choices at each position.

Definition 2. The *p*ractical search space S_p is the space which an algorithm searches with high probability.

Of course, $S_p \leq S_s$, and in most practical searches, S_p is much smaller. For example,$S_p = S_s$ for random search and $S_p \ll S_s$ for most GA search.

3.2 Practical Search Space Estimation

We cannot easily quantify the relationship of the PSS to the whole search space, given the GA parameters and problem characteristics. As a first step, we shall define a Practical Search Index (PSI), in order to seek insight into the fraction of the sample space actually visited in a typical GA search.

For those problems exhibiting a strong building block structure, or so-called decomposable problems, a Practical Search Index can be defined at the initial stage of GA search:

Definition 3. The *P*ractical Search Index (PSI) of a problem P to be searched by a GA, PSI_P is

$$\sum_{i=1}^{k} n^{l_i}$$

where k is the number of BBs, l_i is the length of the i-th BB, and n is the number of alleles at each position (2, for example, for binary chromosomes).

Based on this definition, the PSIs of some famous GA problems could be computed as follows:

1. For a One-Max problem of length l, $PSI_{1-max} = 2l$, since the size of each of its BBs is 1.

   For instance, for a 64-bit 1-max problem,

   $$PSI_{1-max} = \sum_{i=1}^{64} 2^1 = 2^7$$

   and if we underline each building block individually, would be written as

   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
   1 1 1 1 1 1

2. For Royal Road Function 1 (RR1) with m n-bit blocks, $PSI_{RR1} = m2^n$. For a 64-bit problem with 8 8-bit BBs,

   $$PSI_{RR1} = \sum_{i=1}^{8} 2^8 = 2^{11}$$

   and it would be displayed as

   11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111

3. For a Needle-in-a-Haystack problem with length l and n choices for each building block locus, $PSI_{NinH} = n^l$. For a 64-bit binary problem,

   $$S_{NinH} = \sum_{i=1}^{1} 2^{64} = 2^{64}$$

   and it would be displayed as

   111111111111111111111111111111111111111111111111
   1111111111111111

The three examples above show that the PSIs for different problems with the same string length vary dramatically, due to the differences in their BB structures. The result, $PSI_{1-max} < PSI_{RR1} < PSI_{NinH}$, agrees with our experience that their difficulty increases in that order. The index is intended to indicate that problems with smaller BB sizes have exponentially smaller practical search spaces. It seems obvious that the size (length) of the largest BB plays an overwhelming role in accounting for the size of the PSS. For example, a problem with BB structure

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 11111111

has search space

$$S = \sum_{i=1}^{57} 2^{l_i} = 56 * 2^1 + 2^8 = 368$$

so the last 8-bit BB contribute nearly 70% to the PSI of this 64-bit problem.

3.3 Hierarchical Building Block Structure and Practical Search Index

Assessing the PSI for a Hierarchical Building Block Structure (HBBS) will be more complex than for the flat structure problems discussed in the last subsection.

Li & Goodman illustrated an evolutionary process on a classical HBBS problem in Figure 1 [13].

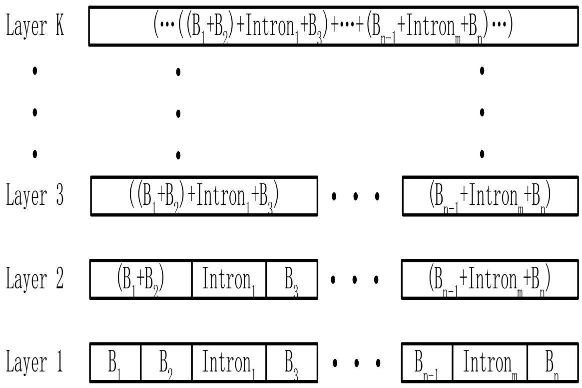

Figure 1: Hierarchical Organization of Building Blocks

Figure 1 shows a pyramid (or tree) structure, where BB exist at different levels, and the higher-level BBs are composed of lower-level ones. In such cases, the BB-related structure of the search is dynamic, changing during the search process. At each point in the evolutionary process, some subset of the building blocks has been found, and other BBs are still being sought. There is a relationship between spatial structure and temporal stage: the lower-level BBs tend to be assembled at an earlier time in the GA's evolutionary process, while higher-level BBs appear at later stages.

For HBBS problems, Figure 1 illustrates that the BB size will continue to grow during search, sometimes finally expanding to the whole length when the highest-level BBs are assembled. But, as stated in the previous section, a bigger BB size means a bigger PSI. So, does Figure 1 suggest that the PSI enlarges more and more in the later stages? Will the PSI in the final stage be equal to that of the Need-in-a-Haystack problem, since the BB size grows to be the whole string, as the highest level shows in Figure 1?

The answer is NO. On the contrary, the PSI will shrink quickly after the lower BBs are formed, so long as they are not lost, and each BB can be regarded as one locus with n states, where n is the number of alleles that contribute sufficiently to high fitness to be regarded as part of the schema for that particular building block. In this way, the set of BBs at later stages has fewer states than the set at earlier stages, and the PSI decreases.

The idea of regarding a formed BB as one "bit" or unit also appeared in Yu & Goldberg [23] and Howard & Sheppard's[10] work. The former paper proposed an explicit

chunking scheme, called *substructural chromosome compression*, which expresses a building block by one single variable when the alleles in the building block nearly converge to only a few schemata, to conquer hierarchical difficulty by reducing the complexity (number of states) of higher-level BBs. The latter paper grouped all bit strings with the same arrangement of schemata, and this representation space from the set of 2^{64} 64-bit binary strings to the set of 2^8 8-bit binary strings, when they built a model for an eight-8-bit-BB Royal Road Problem.

Taking a BB as an element, we can calculate the PSI of some HBBS problems.

Definition 4. After the lower BBs are formed,the GA search space PSI_{BB} is

$$\sum_{i=1}^{k} A_i$$

where k is the number of BBs, and A_i is the number of alleles of the i-th BB.

Below are some HBBS examples.

1. Royal Road function 2 (RR2) is composed of eight 8-bit BBs, where each pair of odd-even-numbered adjacent BBs constitutes a higher-level BB. It can be expressed as:

   $$\underline{\underline{11111111\ 11111111}}\ \underline{\underline{11111111\ 11111111}}$$
   $$\underline{\underline{11111111\ 11111111}}\ \underline{\underline{11111111\ 11111111}}$$

   The PSI changes as BBs are found. At level 1, the initial stage, the PSI is 2^{11}, as in RR1. At level 2, with the 8-bit BBs formed, and the PSI shrinks to 8. It then decreases to 4, 2, and 1 at the later levels. It makes it appear that the GA search will proceed more quickly after the basic BBs have been discovered. That is not to say that assembly of the higher-level BBs is trivial, however, as Holland has pointed out for this problem.

   The computation above suggests that the PSI of RR1 is slightly different from that of RR2, and that it is consistent with the original result in [17, 4] and later experiments in[10].

   Of course, the notion of the BB levels advancing one level at a time is an idealization rarely realized in practice. Different levels typically overlap during a GA run.

2. Holland's Royal Road Challenge, RR(JH), presented by John Holland at ICGA '93, is a HBBS problem with deception. The total score sums the bonus from good BB combinations and the first-level BBs themselves with correct schemata.

   The PSI for a 238-bit string, with every 8-bit BB separated from its neighbor by a 7-bit intron, is 2^{12} at leve1 1, where only 16 BBs contribute to the PSI and the PSI for each BB is 2^8; the PSIs are 8, 4, and 2 at the subsequent levels. Thus, the PSI indicates that the problem requirements, in terms of population sampling, are most stringent at first level, as they were for RR2.

3. Hierarchical-if-and-only-if (H-IFF)[21, 22], is a recursive HBBS problem, which defines a function whose score gets a bonus if its lower parts are consistent (match). For instance, although "00000" and "1111" are good building blocks respectively, their combination "000001111" gets no bonus; only if they are the same do they garner the reward.

   Unlike the preceding examples, HIFF's PSI decreases slowly over time. For example, a 64-bit HIFF's PSI is just 2^7, for 32 2-bit BBs; but it will later be 2^6 for 16 4-bit BBs, and then 2^5,2^4,2^3,2^2,2^1 at the later levels. This suggests that it advances slowly, and moreover, the diversity must be continually safeguardes, to preserve enough consistent BBs for recombinations in later stages.

In the examples above, we have not discussed deception, although it influences the difficulty of BB discovery and problem solution.

Deception misguides GA search direction and therefore makes the problem harder, but any deceptive BB is still a BB, and a PSI contribution for that BB can be calculated as above, and at later stages, acts as other BBs. The deception per se does not increase the search space. But that points up that a small PSI does not necessarily imply that the search is easy. However, a large PSI does indicate some level of search difficulty.

4. GA ADAPTIVE MECHANISM

Base the previous work, we re-examine the GA sampling and population size on the viewpoint of PSI.

4.1 GA Sampling:Individual vs. BB

It is generally accepted that effective GA search requires larger population sizes as the problem difficulty increases. However, despite work that bounds population sizes as a function of certain problem characteristics, there is still uncertainty in this decision process, so it is worth attention.

As we discussed in Section 3, to ensure finding a global optimal solution, the population size should be as big as the search space, and an exhaustive search might as well be used. However, we usually use much smaller population sizes, and most of time, can still find the best solution for many applications.

Here, we argue (after Goldberg) that although the population size is much smaller than the whole search space, we will proceed toward optimality not by repeatedly sampling for the best individual, but by ferreting out BBs, for combining into the solution in later stages.

For example, for an 8×8 Royal Road problem, if we set the population size to 256, obviously, the size is just $1/2^{56}$ of the whole sample space, but this size is enough for sampling all 8-bit building blocks, from which the global optimum comes. But, of course, a population of 256 will not typically sample all of the 8-bit BBs, so will not be adequate for assembly-only operations, even if none of those discovered were ever lost.

But to even *a*llow for sampling all BBs during the initial stage, the population size must not be smaller than the sample space of the largest single BB. However, setting the population size to the size of the sample space for the BBs does not ensure that the global solution will be assembled, even if every BB appears in the initial stages. The problem is compounded if BBs are deceptive, as in RR(JH), so that random creation is more likely to create the BBs than are crossover or mutation operations. The question still exists, how big should the population size be?

4.2 Population Size: BB Sampling vs. BB Mixing

For many crossover-dominated GAs, their search process could divided into two stages: the initial stage and the evolutionary stage. In the former phase, the BBs are sampled; in the latter, the BBs are mixed. BB sampling is easier when more individuals are available, so the BB can appear and have more instances present (which means more chances to persist to the later stages); while the BB mixing often works well with fewer individuals, since too many individuals, especially in later stages when the population is almost homogenous, result in many wasted mixing operations among similar individuals.

So, BB sampling and mixing are somewhat competitive in population sizing. When the population size is small (but not smaller than the BB sample space size), although the cost of mixing is less, the availability of BB instances is also less. The GA still needs a long time to collect them. When the population is larger, the number of BB instances goes up, however the evaluations in each generation also jumps, therefore making the simple GA more costly to run. An appropriate population size should balance the BB sampling and mixing costs.

This also suggests the well-known fact that a small population size and long evolution time is not equivalent to a large population size and a short evolution time. This tradeoff can be made only on a small scale near the balance point, where neither the sampling nor mixing are overwhelming each other and these two operations can compensate for each other to some extent.

5. EXPERIMENTS AND RESULTS

To test our formulation of the PSI, which is calculated from the BB structure, and the the relationship between the population size and BB construction, we designed a series of Simple GA (SGA) experiments on different BB structures and population sizes.

5.1 Test Examples

We choose some frequently used functions.

1. One-Max. A GA-easy function, that counts the “1”s on the string.
2. Royal Road Function 1 (RR1). A 64-bit string composed of eight blocks, each of which earns a reward only if it is all “1”s.
3. Royal Road Function 2 (RR2). It is a typical hierarchical example. In addition to each block’s score, as in RR1, two adjacent odd-even BBs double their scores if both are present.
4. Holland’s Royal Road Challenge. It has eight 8-bit BBs (RRJH64). A hierarchical function with deception. Its length was set to 64 bits, rather than the usual 128 bits of BBs, to allow comparison with the other functions used here.
5. Hierarchical-if-and-only-if (HIFF). A challenging example with a number of traps. It is increasingly hard to assemble BBs at the later stages.

5.2 Algorithm

For testing the GA behavior over BBs , we chose a (traditional) simple GA, rather than using a GA variant, such as a messy GA, whose objective is to speed the search.

The algorithm is:

```
//start with an initial time
  t := 0;
  //initialize population randomly
  Initpopulation P (t);
  //evaluate all initial individuals
  Evaluate P (t);
  //test for fitness or time limit reached
  WHILE not done
     //increase the time counter
     t := t + 1;
     //select two individuals
     P' := Selectparents P (t);
     //recombine the "genes"
     Crossover P' (t);
     //perturb them
     Mutate P' (t);
     //evaluate its new fitness
     Evaluate P' (t);
     //offspring replace parents
     P := P' (t);
  END WHILE
```

5.3 GA Setting

To simplify the experiments, the only operators are one-point crossover and bit-wise mutation. The population size ranges from 2 to 8192, to test how the size impacts the BB sampling. The settings are shown in Table 1

Table 1: GA Parameters

	One-Max	R1	R2	RRJH64	HIFF
Crossover	one-point crossover				
Mutation	bit-wise mutation at 0.005				
Selection	tournament, size 4				
Pop. size	2-8192	128-8192		64-8192	
No. of Gens.	500				
No. of Runs	50				

5.4 Result

5.4.1 Search Space

R1 with Different BB Sizes. We tested the average evaluation counts for a 64-bit RR1 function at BB sizes of 2,

4, 8, and 16 using a population size of 1024, running each experiment 50 times, independently.

Table 2: Evaluations and search space of RR1 with different BB sizes

BB Size	2	4	8	16
PSI	128	256	2048	262144
Mean	4503	6902	17265	1135084
Std. Deviation	2324	3899	19005	696020
Success Rate	100%			

The correlation (1.00) between PSI and BB size is not surprising... PSI is calculated from the BB size. Given that, the bigger the BB size, the larger the PSI and the more evaluations were needed, as expected.

HBBS and Evaluations. We believed that the hierarchical BB structure contributes less to the PSI in section 3.3, and the result of R1, R2, and RRJH64, whose BB structures are the same at the bottom level but differ on upper levels (RR1 is flat, RR2 is fully hierarchical, and RRJH64 has a stronger hierarchical structure and weaker deception within a single BB), corresponds with the expectations, as shown in Table 3.

Table 3: Evaluations of RR1, RR2, and RRJH64 in different population sizes, based on 50 runs

Pop. Size	256	512	1024	2048	4096	8192
RR1 Mean	36933	24381	17265	24863	45957	88801
Std. Dev.	27373	22141	19004	2069	2901	6271
RR2 Mean	39972	24607	14643	23798	44483	84378
Std. Dev.	27882	29970	8190	1750	2618	5034
RRJH64 Mean	Success Rate			24658	46449	88310
Std. Dev.	less than 95%			2067	3269	5802
RR1 RR2 T-Test	0.587	0.966	0.374	0.007	0.009	0.000
One-Way ANOVA	N/A			0.018	0.003	0.000

Table 3 suggests that the HBBS have weak effects on the GA process when the population size is small, and gradually lose their impact as the population size increases. The lack of a striking difference in performance on RR1 and RR2 indicates that, once again, the Royal Road is "not taken", i.e., the additional reward for higher-level building blocks in RR2 does not speed its search at any of the population sizes tested.

5.4.2 Population Size

Five functions with the same string length are tested under different population sizes (see Fig.2). All results for which success rates were bigger than 95% are presented here. All runs were stopped at the 500th generation, except the population-size-2 One-Max example, which was run to 50000 generations.

Fig.2 suggests:

1. Smaller-BB-size problems demand smaller population sizes. A 64-bit One-Max function, whose BB size could be regarded as 1, could reach its global optimum by using just 2 individuals, the smallest population size to implement crossover, the typical operation of GA.

2. Population size should be at least big enough to sample the BBs. In our experiments, One-Max, RR1 and RR2 could find the best solution when their population sizes were set to their BB Sample Sizes; but HIFF and RRJH64 needed more individuals than that to sample the BBs. Searches failed to succeed at above the 95% rate if their population sizes were less than the minimum size to sample the BBs.

   The phenomenon that a GA is stuck at a local, non-global optimum, often attributable to insufficient population size, is called premature convergence. Another effect of small population sizes is an increase in genetic drift, or loss of alleles that do not convey strong selective advantage in their current context, due to chance breeding effects. Here, we give further explanation based on the BB sampling analysis: the population size is not enough to sample all BBs. And we regard that in order to avoid the premature convergence, the population size should not be smaller than the size for sampling BBs, for crossover-dominated GAs. Introduction of deception in the BBs requires still larger sampling in the initial population for efficient solution.

3. It is interesting that the curves are divided into two parts at population size 1024. For populations smaller than that, the problems exhibit very different performance. But at population sizes of 1024 or larger, the GA shows similar performance on all of the functions, both easy and hard.

   We hypothesize that 1024 is the balance point of BB sampling and BB mixing for the RR1 and RR2 problems at the sizes run, and is the bottom point for other harder functions with eight 8-bit BBs. As the population size increases from 1024, the cost of BB mixing becomes dominant, and that the SGA used might be able to complete mixing of similar numbers of any types of BB, deceptive or non-deceptive, in similar numbers of operations.

The experiments indicate that the population size should be at least somewhat larger than the minimal size needed to contain all values for potential BBs, but not an order of magnitude larger, and not of a size that would probabilistically guarantee that all bit combinations within a BB would be present in an initial randomly generated population. The value of an initialization method that systematically generates all bit combinations below a certain size may allow a tightening of that bound, but was not tested in these experiments.

5.5 Discussion

As a naturel extension of PSI theory, we must discuss how it can work threose weak BB problems and more, can it be an index of GA hardness?

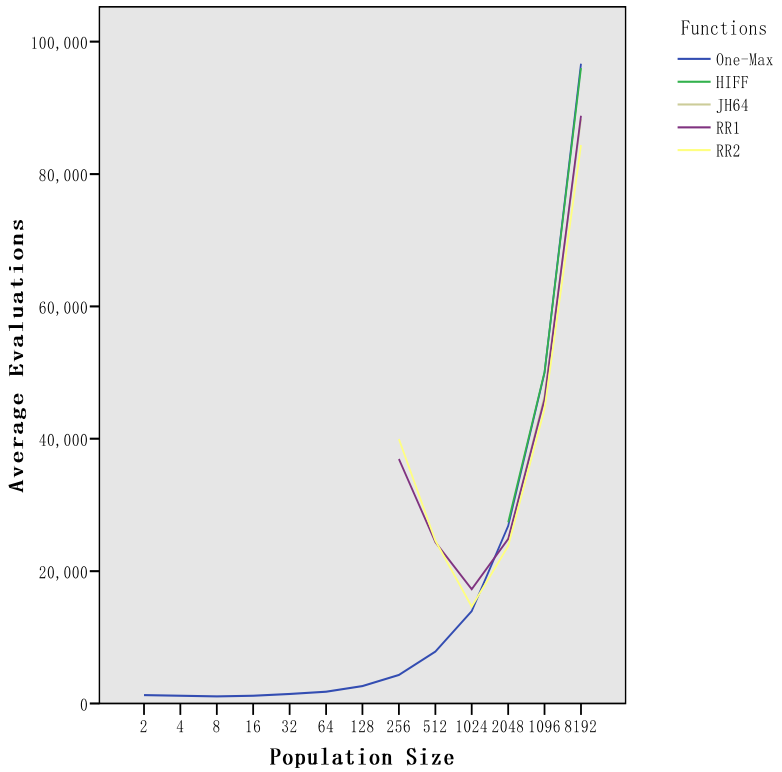

Figure 2: Average Evaluations of 64-bit functions under different population sizes over 50 runs

5.5.1 *Tight BBs and epistasis*

The examples we discussed above are all defined as strong BB-structured functions. We suppose that the conclusion may also be applicable for many other functions with strong linkage effects, but clearly could not press a claim that they are true for GA search in general, as one would be hard-pressed even to define the PSI or relevant BB sizes in general.

The efficiency of GA operators depends strongly on the properties of the search domain, and 1- and 2-point crossover, for example, typically work well in problems with tightly structured BBs. Other operators might be better suited to domains with strong epistasis. Uniform crossover, on the other hand, shows little respect for BB structures. But perhaps analogous behavior to that found here may be observed under other operators in other environments; that has also not yet been tested.

5.5.2 *GA hardness*

Fitness Distance Correlation [11, 1, 19] and epistasis variance [2, 18] give measures of GA hardness based on the fitness landscape; we would like to pursue another measure based on the discussion of the Practical Search Index and population sizing above, but it cannot be done absent consideration of the types of operators chosen.

For example, we might want to define GA easiness (or GA success in a given timeframe) for a problem as

$$E = Avg(P(p_1, p_2))$$

where E denotes Easiness, p_1 means population size and p_2 means perturbation (or the genetic operators used). P is the success probability based on p_1 and p_2. To date, we have explored only the effects of population size, but further exploration is planned.

1. GA-Easy: $E = 1$, even for small population sizes. For example: the One-Max problem.
2. GA-Medium: $0 < E < 1$, if p_1 increases, E also increases. For example: the HIFF function.
3. GA-Hard: $E \approx 0$, even for large population sizes. For example: a Needle-in-a-Haystack problem.

6. CONCLUSIONS

A Practical Search Index measure of GA hardness for problems with strong building block structure has been defined. According to this measure, large BB sizes, rather than long strings per se, account for the exponential increases in the space that must actually be sampled by a GA in solving a problem, and therefore make the problem hard.

Hierarchical Building Block structures contribute less to the PSI since once formed, lower-level Building Blocks can, under appropriate operators and population sizes, be treated as 1-bit units in the PSI measure. That may help to explain in part why the introduction of second-level fitness bonuses in hierarchically structured BB problems do not greatly speed problem solution (i.e., the "Royal Road" is not taken).

It is argued that a GA features Building Block sampling, rather than individual sampling, so the population size should

not be smaller than what is needed to sample the space of the largest Building Block, for crossover-dominated GAs. Premature convergence is regarded as evidence that the population size was not large enough to discover all the BBs. At the same time, too large a population results in too many wasted evaluations in each generation, resulting in a high mixing cost. Therefore, the appropriate population size should balance the BB sampling and mixing costs.

7. ACKNOWLEDGEMENT

We wish to acknowledge the support of the Provincial Natural Science Foundation of Hubei (No. 2011CDB341), the Fundamental Research Funds for the Central Universities (No. CUG120109) and the Michigan State University High Performance Computing Center.

8. REFERENCES

[1] P. Collard, A. Gaspar, M. Clergue, and C. Escazut. Fitness distance correlation, as statistical measure of genetic algorithm difficulty, revisited. In *ECAI*, pages 650–654, 1998.

[2] Y. Davidor. Epistasis variance: A viewpoint on GA-hardness. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*. Morgan Kaufmann Publishers, 1991.

[3] R. Fisher. *Genetical Theory of Natural Selection* - 2^{nd} *edition*. Dover, New York, 1958.
A reprint of Fisher's classic 1929 mathematical analysis of the process of evolution by natural selection.

[4] S. Forrest and M. Mitchell. Relative building-block fitness and the building-block hypothesis. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 109–126. Morgan Kaufmann, San Mateo, CA, 1993.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1989.

[6] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In S. F. et al., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64, San Mateo, CA, 1993. Morgan Kaufman.

[7] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

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

[9] J. H. Holland. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. *Evolutionary Computation*, 8(4):373–391, 2000.

[10] B. Howard and J. Sheppard. The royal road not taken: A re-examination of the reasons for GA failure on R1. In K. D. et al., editor, *Genetic and Evolutionary Computation – GECCO-2004, Part I*, volume 3102 of *Lecture Notes in Computer Science*, pages 1208–1219, Seattle, WA, USA, 26-30 June 2004. Springer-Verlag.

[11] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In L. J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[12] H. Kargupta. The gene expression messy genetic algorithm. In *International Conference on Evolutionary Computation*, pages 814–819, 1996.

[13] Z. Li and E. D. Goodman. Learning building block structure from crossover failure. In D. T. et al., editor, *Genetic and Evolutionary Computation Conference, GECCO 2007, Proceedings, London, England, UK, July 7-11, 2007*, pages 1280–1287. ACM, 2007.

[14] Z. Li and E. D. Goodman. A practical search index and population size analysis based on the building block hypothesis. In *GECCO*, pages 1123–1124, 2008.

[15] Q. Liu, T. Wu, and X. Luo. Improved P-hub network model and GA solution based on rough set theory. *Journal of Computers*, 7(5):1191–1195, 2012.

[16] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

[17] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In F. J. Varela and P. Bourgine, editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1991*, pages 245–254, Paris, 11–13 1992. A Bradford book, The MIT Press.

[18] B. Naudts. *Measuring GA-hardness*. PhD thesis, University of Antwerpen, Antwerpen, Netherlands, 1998.

[19] R. Poli and E. G. López. On the effects of bit-wise neutrality on fitness distance correlation, phenotypic mutation rates and problem hardness. In *FOGA*, pages 138–164, 2007.

[20] H. Shi and W. Li. Evolving artificial neural networks using simulated annealing-based hybrid genetic algorithms. *Journal of Software*, 5(4):353–360, 2010.

[21] R. A. Watson, G. S. Hornby, and J. B. Pollack. Modeling building-block interdependency. In *Proceedings of the Third Conference on Parallel Problem Solving from Nature(PPSN V)*, volume 1498, pages 97–106, 1998.

[22] R. A. Watson and J. B. Pollack. Hierarchically consistent test problems for genetic algorithms: Summary and additional results. In S. Brave and A. S. Wu, editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 292–297, Orlando, Florida, USA, 13 July 1999.

[23] T.-L. Yu and D. E. Goldberg. Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In M. K. et al., editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 1385–1392. ACM, 2006.