

# A Description of Holland's Royal Road Function

Terry Jones  
Santa Fe Institute  
1660 Old Pecos Trail, Suite A  
Santa Fe, NM 87505  
terry@santafe.edu

June 30, 1994

## Abstract

This note contains a description of John Holland's Royal Road function, which was presented at the Fifth International Conference on Genetic Algorithms in July 1993, and posted to the Internet Genetic Algorithms mailing list in August 1993 [2].

## 1 Introduction

The Royal Road functions were introduced in [5]. They were designed as functions that would be simple for a genetic algorithm to optimize, but difficult for a hillclimber. However, one form of hillclimbing [6], easily outperformed a genetic algorithm on what had been expected to be the simplest Royal Road function for that algorithm. Recently, Holland [2] presented a revised class of Royal Road functions that were designed to create insurmountable difficulties for a wider class of hillclimbers, and yet still be admissible to optimization by a genetic algorithm.

Holland used seven variables in the description he posted to the genetic algorithms community. In describing the class of functions, I will use his variable names where reasonable, and indicate deviations. Holland suggested a default value for each of these variables and in what follows I will use these values to provide illustration. A summary of these variables and their default values can be found in section 3.

The main description here is of the function posted to the genetic algorithms community. Holland has also suggested further generalizations, which are discussed in section 6.

The following terms will be used consistently in what follows: *block*, *complete*, *gap*, and *region*. These are introduced below and can be relied on to always mean exactly the same thing.

## 2 Description

The function takes a binary string as input and produces a real value. The function is used to define a search task in which one wants to locate strings that produce high function values. Section 5 provides some more information on fitness values.

The string is composed of  $2^k$  non-overlapping contiguous regions, each of length  $b + g$ . With Holland’s defaults,  $k = 4$ ,  $b = 8$ ,  $g = 7$ , there are 16 regions of length 15, giving an overall string length of 240. We will number the regions, from the left, as  $0, 1, \dots, 2^k - 1$ .

Each region is divided into two non-overlapping pieces. The first, of length  $b$ , will be called the block. The second, of length  $g$ , will be called the gap<sup>1</sup>. In the fitness calculation, only the bits in the block part of each region are considered. The bits in the gap part of each region are completely ignored during fitness calculation. Holland’s posting called the block part of each region by various names: “building blocks”, “elementary building blocks”, “schemata”, “lower level target schemata”, and “elementary (lowest-level) building blocks (schemata)”.

The fitness calculation proceeds in two steps. Firstly, there is what Holland calls the PART calculation. Then follows the BONUS calculation. The overall fitness assigned to the string is the sum of these two calculations.

### 2.1 The PART Calculation

The PART calculation considers each block individually, and each in exactly the same fashion. Each block receives some fitness score, and these are added to produce the total PART contribution to the overall fitness.

The fitness of each block is based entirely on the number of 1 bits it contains<sup>2</sup>. The idea is to reward 1’s up to a certain point. Every 1 up to (and including) a limit of  $m^*$  adds to the block’s fitness by  $v$ . Thus, with the default settings, a block with three 1’s would have fitness  $3 * 0.02 = 0.06$ . If a block contains more than  $m^*$  1’s, but less than  $b$  1’s, it receives  $-v$  for each 1 over the limit. With the default settings  $m^* = 4$ , and so a block with six 1’s is assigned a fitness of  $(6 - 4) * -0.02 = -0.04$ .

Finally, if a block consists entirely of 1’s (i.e. it has  $b$  1’s), it receives *nothing* from the PART calculation. Such a block will be rewarded in the BONUS calculation. If a block consists entirely of 1 bits, it will be said to be complete.

From the above, we can construct a table of fitness values based on the number of ones in a block. Table 1 gives the values for the default settings.

---

<sup>1</sup>Holland did not give a name for this part of the regions, or give a variable name for its length. These gaps have been called “introns” (borrowing from biology) and have been used with varying degrees of success [1, 4] to alter the effects of crossover in genetic algorithms

<sup>2</sup>Holland used  $m(i)$  to denote the number of 1’s in block  $i$ . I will not use a variable.

1's in block	0	1	2	3	4	5	6	7	8
Block fitness	0.00	0.02	0.04	0.06	0.08	-0.02	-0.04	-0.06	0.00

Table 1: PART Fitness Values For Holland's Default Settings

## 2.2 The BONUS Calculation

The idea of the BONUS calculation is to reward completed blocks and some combinations of completed blocks. Holland gives rewards for attaining what he calls "levels". At the lowest level<sup>3</sup>, 0, rewards are given for complete blocks (i.e blocks that consist entirely of 1's). If such a block exists, it receives  $u^*$  as its fitness. Any additional complete blocks receive a fitness of  $u$ . Thus, with the default settings, a string that contained three complete blocks would receive a BONUS fitness of  $1.0 + (2 * 0.3) = 1.6$  for level 0.

But that is not the end of the BONUS story. At the next level, *pairs* of blocks are rewarded. Suppose we label the blocks from left to right as  $B_0, B_1, \dots, B_{2^k-1}$ . The function then rewards any completed pair  $(B_{2i}, B_{2i+1})$  for  $0 \leq i < 2^{k-1}$ . The rewards are calculated in the same way as they were done at level 0. The first completed pair of blocks receives  $u^*$ , and additional completed pairs receive  $u$ .

Notice that not all pairs of completed blocks are so rewarded. Only those whose indices are of the form  $(2i, 2i + 1)$ .

We can now calculate rewards for the next level, that of quadruplets of completed blocks, and this is done in the same fashion. In general, there are  $k + 1$  levels, and at level  $0 \leq l \leq k$ , we are looking for contiguous sets of  $2^l$  complete blocks, whose first block is labeled  $B_{i2^l}$  with  $0 \leq i < 2^{k-l}$ . At all levels, the first such set of complete blocks receives fitness  $u^*$ , and additional sets of completed blocks receive fitness  $u$ . The total fitness for the level is the sum of these fitnesses.

To make this more concrete, consider the function with Holland's default values. With  $k = 4$  we have 16 regions and each contains a block of length  $b = 8$ . The BONUS fitness calculation rewards completed single blocks (this is level 0), and rewards the completion of the sets  $\{B_0, B_1\}, \{B_2, B_3\}, \{B_4, B_5\}, \{B_6, B_7\}, \{B_8, B_9\}, \{B_{10}, B_{11}\}, \{B_{12}, B_{13}\}, \{B_{14}, B_{15}\}$ , (level 1)  $\{B_0, B_1, B_2, B_3\}, \{B_4, B_5, B_6, B_7\}, \{B_8, B_9, B_{10}, B_{11}\}, \{B_{12}, B_{13}, B_{14}, B_{15}\}$  (level 2)  $\{B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7\}, \{B_8, B_9, B_{10}, B_{11}, B_{12}, B_{13}, B_{14}, B_{15}\}$  (level 3) and  $\{B_0, B_1, B_2, B_3, B_4, B_5, B_6, B_7, B_8, B_9, B_{10}, B_{11}, B_{12}, B_{13}, B_{14}, B_{15}\}$  (level 4) of completed blocks.

The total BONUS contribution to the fitness is computed by adding the fitness at each of the  $k + 1$  levels.

---

<sup>3</sup>Holland numbered his lowest level 1 not 0.