# Counting Solutions in Reduced Boolean Parity.

M. Collins

CISA, Edinburgh University, Scotland
mcc@dcs.ed.ac.uk

**Abstract.** Motivated by the interest in the boolean parity problem, this paper examines the structure of solutions to a simple form of the boolean parity problem and provides efficient methods for counting them. Understanding the structures which control solution distribution is important for understanding the performance of search mechanisms on the reduced boolean parity problem. EA search mechanisms have reportedly performed better than random search in solving the simplified form of the boolean parity problem, despite fitness neutrality leading to an absence of a search gradient, renewing interest in understanding the solution distribution.

## 1 Introduction

Using the Cartesian GP [3, 5] Yu and Miller have shown limited success in solving a simplified form of the boolean parity problem [4]. The form of the boolean parity problem examined uses only the Boolean EQ and XOR operators, and shall be referred to as the *reduced Boolean parity problem*.

This paper examines the properties of solutions to the reduced Boolean parity problem, and provides methods for counting solutions within the representation space. Section 2 defines the reduced boolean parity problem and the properties of a solution. Section 3 describes the representation used in this paper. Section 4 identifies structures which are present in all solutions to the reduced Boolean parity problem. These structures are then exploited to efficiently count the possible solution arrangements, greatly simplifying the counting process. Section 5 presents results for some common parameters and Section 6 concludes this work and suggests how this knowledge could be used in future work.

## 2 Boolean parity

The Boolean parity problem is the task of identifying whether a Boolean input sequence contains an even (and equivalently odd) number of TRUE inputs. As a benchmark problem in EA, it is equivalent to choosing a sequence of functions from a set $F$, and input assignments from a set of possible inputs $I$, such that the result of evaluating the function over all possible inputs is to correctly classify the parity of the inputs. It is recognised that the problem difficulty is highly dependent upon the function set used [1].

A solution to the Boolean parity problem is a function which, when evaluated over all possible inputs, correctly distinguishes the odd parity inputs from the even parity inputs. This can be achieved in two ways, by identifying the odd parity inputs, or by identifying the even parity inputs, and one form can be converted to the other by negation of the output. Irrespective of the nature of the parity measured, both methods correctly partition the input space.

When the output of the tested function is compared to the output of an odd (or even) parity function there are three possible situations: the sequences are the complement of each other, half the sequences are coincident, or the sequences are identical. These cases correspond to the case where the function evaluated is the inverse parity of the contrasted parity function, the case where the function is not a solution to either of the parity problems and the case where the parity of the function evaluated is precisely the same as the contrasted parity function. The first and last cases are acceptable as solutions to the Boolean parity problem.

## 3 Representation

In general, the input alphabet $I$ of size $|I|$ is a set of distinct Boolean values $\{I_1, ..., I_{|I|}\}$, corresponding to the parity of the problem. In the reduced Boolean $p$-parity problem, the input alphabet is $p$ elements, and the function set is {XOR,EQ}. The number of possible arrangements using from 1 up to $F_{Max}$ functions is:

$$\sum_{i=1}^{F_{Max}} 2^i p^{i+1} \tag{1}$$

Where an example arrangement using 2 functions and an alphabet of 3 inputs is: $I_1$ EQ $I_2$ EQ $I_3$ and is a solution to the 3 parity problem. Another arrangement but this time with 3 functions, $I_1$ EQ $I_2$ EQ $I_3$ EQ $I_1$ is not a solution to the 3 parity problem. In this work, the total number of arrangements of functions and inputs is referred to as the total number of arrangements.

## 4 Solution structure

Restricting the available operator set to {XOR, EQ} has been shown [1] to restrict the function space. The commutative and associative properties of the XOR, EQ operators create equivalence relations between candidate solutions and a set of functionally equivalent representations. For instance $I_3$ XOR $I_1$ EQ $I_2$ is rearranged to give $I_1$ EQ $I_2$ XOR $I_3$, which whilst potential distinct entities in the representation space, are identical in functional terms. Table 1 shows some simple candidate functions and their simplified equivalent functions.

Solutions occur for this form of the Boolean parity problem iff all Boolean inputs are referenced once in the simplified equivalent solution. A simple proof of this, if fewer inputs are referenced then the function is ignorant of some parity altering input state changes and can not be a solution to the parity problem.

| | Function | Simplified equivalent | Solution? |
|---|---|---|---|
| 1 | $I_3$ XOR $I_1$ EQ $I_2$ | $I_1$ EQ $I_2$ XOR $I_3$ | Even 3-parity. |
| 2 | $I_1$ XOR $I_3$ XOR $I_1$ EQ $I_2$ | FALSE EQ $I_2$ XOR $I_3$ | Non solution. |
| 3 | $I_1$ EQ $I_1$ XOR $I_3$ XOR $I_1$ EQ $I_2$ | TRUE XOR $I_1$ EQ $I_2$ XOR $I_3$ | Odd 3-parity. |

**Table 1.** Examples of simple functions and their simplified equivalent expressions, constructed from an input alphabet of size 3.

Duplicate references to inputs result in introns, and introns vanish from the simplified form. XOR and EQ are both commutative and since for any Boolean input $I_x$, $I_x$ XOR $I_x$ = FALSE and $I_x$ EQ $I_x$ = TRUE. Even numbers of references to the same input either invert or leave intact the exact functionality of the other functions; arrangements of this type can not alter the acceptability of the function in terms of distinguishing even and odd parity. Finally choosing between the XOR and EQ functions is actually irrelevant to the acceptability of the solution. XOR is simply the negation of EQ and instances of it can be replaced by NOT EQ without altering the function of the expression. Negation of a result is irrelevant to the identification of parity, since it simply changes an even parity function into an odd parity function and vice versa.

Table 1 shows three example representations of potential solutions to the reduced 3 parity problem. The functions are numbered on the first column and shows a solution to the 3 parity problem and two cases where an intron is created. The first function contains references to all the possible inputs with no repeated input references, it is thus a solution to the 3 parity problem. The second function shows the effective loss of the input reference $I_1$ due to it being referenced twice. The third function shows the case where the simplified function maintains references to all the inputs, though through the effect of the intron the basic solution has had its polarity reversed (it is now an odd parity solution).

Using the fact that even numbers of references to the same input are effectively neutral to the output validity, the number of solutions possible to the $p$-parity problem which reference $n$ inputs, $S(p, n)$, can be counted by separately calculating the number of arrangements of the $n-1$ functions used and multiplying this by the number of input sequences which meet the requirement of referencing each input once and only once in the simplified representation. There are $n-1$ functions, each of which can be one of the 2 function types, the number of function arrangements is then $2^{(n-1)}$. The number of input sequences which generate a reference to each input only once in the simplified representation is denoted by $I(p, n)$.

$$S(p, n) = 2^{(n-1)} I(p, n) \tag{2}$$

The number of input arrangements which can be simplified to reference each input only once is best counted by ignoring the $p$ inputs which must be distinct for a solution, and counting the number of ways the remaining $n-p$ inputs can be arranged to be introns with no effect. Evidently the introns can only exist if

there are an even number of the $n-p$ remaining references, and further each of the inputs referenced by this surplus must be referenced an even number of times by the surplus.

The number of ways the $n$ input references can make a solution to the $p$ parity problem is then given in two steps. The first step calculates the ways introns can be made from the surplus inputs. This is the number of ways that pairs of the $n-p$ surplus input references can be assigned to groups where all pairs in the group have the same input reference and no two groups have the same input reference. The second step involves multiplying the number of intron arrangements by the number of distinct permutations of $n$ input references.

Enumerating the groups is performed by generating the set of integer partitions of the $(n-p)/2$ pairs. As an example the integer partitions of four are: {{4},{3,1},{2,2},{2,1,1},{1,1,1,1}} and represent the number of ways intron arrangements can be made from the XOR EQ operators with 8 input references; as an octuple {4}, a hextuple and a pair {3,1}, two quadruples {2,2},a quadruple and two pairs{2,1,1}, and as four pairs {1,1,1,1} respectively.

The number of solutions which are present in a partition $\pi$, denoted $N(\pi)$ is the number of arrangements of groups to the partition, $A(\pi)$, multiplied by the number of distinguishable input assignments to the $n$ inputs which have this partition arrangement $D(\pi)$.

$$N(\pi) = A(\pi)D(\pi) \tag{3}$$

Continuing with 3-parity and using the intron partition {2,1,1} as an example, the number of ways of choosing the input for the first element of the partition is simply $\binom{3}{1}$. The second and third elements of the partition are identically sized and can not be distinguished, also one input reference has been used in choosing the assignment for the first partition, so the number of ways of assigning inputs to the second and third elements is the unassigned input alphabet choose two: $\binom{2}{2}$. The total number of arrangements of the partition is $A(\{2,1,1\}) = \binom{3}{1}\binom{2}{2} = 3$. To be specific, the basic labellings for the 3-parity 2,1,1 intron partition (those made from one quadruple and two pairs using an alphabet of three) are {{1111,22,33},{2222,11,33},{3333,11,22}}. The possible permutations of the partition labeling are considered in equation 6. In order to generalise this process we require a utility function to count the number of distinct elements of a particular size in the partition: Let the function $C(\pi, s)$ represent the number of partition elements of size $0 < s \leq I/2$ in the partition $\pi$, and let $|\pi|$ represent the number of elements and $\pi_i$ represent the $i^{th}$ element in the partition $\pi$.

$$C(\pi, s) = \sum_{i=1}^{|\pi|} (\pi_i = s) \tag{4}$$

$$A(\pi) = \prod_{i=1}^{|\pi|} \binom{p - \sum_{j=0}^{i-1} C(\pi, j)}{C(\pi, i)} \tag{5}$$

Any solution containing the introns represented by the partition $\pi$ thus contains $\pi_i + 1$ references to the input assigned to $\pi_i$. These $\pi_i + 1$ assignments are indistinguishable giving:

$$D(\pi) = \frac{n!}{\prod_{i=1}^{|\pi|} (2\pi_i + 1)!} \tag{6}$$

Continuing with the 3 parity example, and using 9 input references for demonstration purposes, intron partition arrangements and the number of distinguishable input arrangements possible are shown in Table 2.

| **Integer partition** $\pi$ | **Num. assignments to the I.P,** $A(\pi)$ | **Num. of distinct permutations,** $D(\pi)$ | **Total** $N(\pi)$ |
|---|---|---|---|
| {3} | $\binom{3}{1}$ | 9!/7! | 216 |
| {2,1} | $\binom{3}{1}\binom{2}{1}$ | 9!/5!3! | 3024 |
| {1,1,1} | $\binom{3}{3}$ | 9!/3!3!3! | 1680 |
| **Total** | | | 4920 |

**Table 2.** The 3-parity problem with 9 inputs. Intron arrangements (as represented by integer partitions) are shown in column one, and the number of ways of assigning inputs to the introns is shown in column two and the third column shows the number of ways such an arrangement could be produced.

And so the total number of intron arrangements possible for the $p$-parity problem using the XOR,EQ operators is:

$$I(p, n) = \sum^{\forall \pi} N(\pi) \tag{7}$$

The number of solutions possible for all possible functions referencing up to a maximum of $n$ inputs, is then:

$$\sum_{i=p}^{n} S(p, i) \tag{8}$$

## 5 Results

Table 3 shows the relationship between the parity, the number of inputs referenced and the number of solutions in the space. For comparison the number of possible arrangements of input references and function assignments is given, and the amount of the space which is occupied by solutions is given as a percentage.

Previous work [4] examined the 5, 8 10 and 12 even parities and permitted size of up to 100 functions (101 inputs). In [4] the 5-parity results used XOR and EQ functions, whereas the 8, 10 and 12 parity used only EQ. For comparison,

| Parity | Num. inputs | Num. solutions | Num. possible | Percent |
|---|---|---|---|---|
| 4 | 4 | 192 | 2048 | 9.4 |
| | 5 | 0 | 16384 | 0 |
| | 6 | 15360 | 131072 | 11.7 |
| | 7 | 0 | 1048576 | 0 |
| | 8 | 1032192 | 8388608 | 12.3 |
| 5 | 5 | 1920 | 50000 | 3.84 |
| | 6 | 0 | 500000 | 0 |
| | 7 | 268800 | 5000000 | 5.38 |
| | 8 | 0 | 50000000 | 0 |
| | 9 | 29675520 | 500000000 | 5.94 |

**Table 3.** Examples of the number of solutions and the number of possible arrangements against the parity and the size of the function (measured in terms of the number of input references made by the function)

| Parity | Num. solutions | Num. possible | Percent |
|---|---|---|---|
| 5 | $3.157\ 10^{99}$ | $5.556\ 10^{100}$ | 5.68181 |
| 6 | $1.303\ 10^{106}$ | $5.421\ 10^{108}$ | 0.24038 |
| 7 | $4.508\ 10^{113}$ | $3.091\ 10^{115}$ | 1.45833 |
| 8 | $1.013\ 10^{118}$ | $2.204\ 10^{121}$ | 0.04596 |
| 9 | $1.187\ 10^{124}$ | $3.209\ 10^{126}$ | 0.37007 |
| 10 | $1.241\ 10^{127}$ | $1.334\ 10^{131}$ | 0.00930 |
| 11 | $1.880\ 10^{132}$ | $2.013\ 10^{135}$ | 0.09341 |
| 12 | $2.568\ 10^{134}$ | $1.315\ 10^{139}$ | 0.00195 |
| 13 | $9.989\ 10^{138}$ | $4.249\ 10^{142}$ | 0.02351 |
| 14 | $3.176\ 10^{140}$ | $7.546\ 10^{145}$ | 0.00042 |
| 15 | $4.724\ 10^{144}$ | $7.997\ 10^{148}$ | 0.00591 |

**Table 4.** The number of solutions and the number of possible arrangements for the 5 to 15 parity problems using up to 100 functions.

Table 4 shows the $\sum_{i=p}^{100} S(p,i)$ result for the first 10 non-trivial parity problems; from 5-parity through to 15-parity. It should be noted that for all experiments in this work both XOR and EQ functions are used.

However, the choice of the maximum number of functions used in the representation is significant. If the parity problem is even and the maximum number of functions is even, then all the arrangements which use the maximum number of functions — by far the majority of the space — are non-solutions. The same effect occurs when the problem to be solved is odd party and the maximum number of functions is odd. This sampling effect can be seen in Table 4, which shows distinct bias towards solving odd parity problems as a consequence of having an even maximum number of functions (100 in this case).

## 6 Conclusion

This paper presents efficient methods for counting the solutions to the reduced Boolean parity problem, and provides example results for some common parameters. The structure of the solution space; distinct solutions permuted by various intron assignments, indicates the space is regularly populated with elementary solutions which have functionally identical alternative representations at distances governed by the possible permutations of the intron groups. This suggests that the space may be better explored by moving between equivalence classes; a promising topic for future work.

## 7 Thanks

Thanks to all those who helped with this paper, not least Michelle Galea and Henrik Westerberg, whose insight with combinatorics allowed me to make the deadline. Thanks also to John Levine and Jacques Fleuriot for reading this paper at extremely short notice. Also thanks to the anonymous reviewers, whose contributions are gratefully received.

## References

1. W. Langdon. R. Poli. *Boolean Functions Fitness Spaces* 1997. University of Birmingham Technical Report CSRP-98-16.
2. J. Miller. *What Bloat? Cartesian Genetic Programming on Boolean problems* E. Goodman (Ed.) 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers, pp 295-302.
3. J. Miller. *An empirical study of the efficiency of learning boolean functions using a Cartesian Genetic Programming approach* R. Poli et al (Eds.) Proceedings of the Third European Conference on Genetic Programming. 2000. pp. 121-132.
4. T. Yu. J. Miller. *Finding Needles in Haystacks Is Not Hard with Neutrality* J. Foster et al (Eds.) EuroGP 2002, LNCS 2278, pp. 13-25.
5. T. Yu. J. Miller. *Neutrality and the Evolvabilty of Boolean Function Landscapes* Proceedings of the Fourth European Conference on Genetic Programming. 2001.
6. W. Langdon. R. Poli. *Foundations of Genetic Programming* Springer-Verlag. 2002. ISBN 3-540-42451-2. pp 145-150.