Implicit Alternative Splicing for Genetic Algorithms

Philipp Rohlfshagen and John A. Bullinaria

Abstract— **In this paper we present a new nature-inspired variation operator for binary encodings in genetic algorithms (GAs). Our method, called implicit alternative splicing (iAS), is repeatedly applied to the individual encodings in the algorithm's population and inverts randomly chosen segments of decreasing size in a systematic fashion. Its goal is to determine the largest possible segment the inversion of which yields no loss in the encoding's quality. The application of iAS potentially produces a new encoding of equal or greater quality that is maximum possible Hamming distance from its source. This allows iAS to uphold the diversity of the population even if a minimal population size is chosen. This significantly improves the performance of an otherwise standard GA on a representative set of three different optimisation problems. Empirical results are compared and analysed and future work prospects are considered.**

I. Introduction

Genetic algorithms (GAs, see [12], [17]) are abstract implementations of evolutionary systems inspired by the field of population genetics: GAs maintain a population of individuals representing potential solutions to the problem of interest. Individuals of higher fitness are, on average, chosen more frequently for reproduction. Offspring, generated by means of crossover and mutation are placed back into the population allowing the population's average fitness to increase from one generation to the next. In other words, GAs evolve increasingly better solutions over time by means of an artificial equivalent to natural selection. Evolutionary algorithms have become a popular choice of algorithm for a diverse set of different optimisation problems, especially those where traditional methods tend to fail (e.g. highly multi-modal search spaces). Numerous extensions have been suggested over recent years that improve the GA's general performance in different domains. Lately, an increasing number of such extensions have been inspired by the field of molecular genetics. The approach presented in this paper, implicit alternative splicing (iAS), is an example of such approach as it has been inspired by the post-transcriptional process of alternative splicing that takes place in the majority of human cells.

Alternative splicing is a major source of diversity in natural systems allowing the production of numerous different proteins from the same underlying strand of DNA. Interestingly, diversity plays a major role in the success of GAs and the premature loss of the population's diversity is a major factor preventing the algorithm from finding globally optimum solutions. As the population's average fitness increases over time, the differences amongst the individuals in the population have a tendency to diminish. The total number of individuals in the population (i.e. the population size) is thus a crucial factor in the overall success of the algorithm. If the population converges too quickly, the region of the search space containing the global optimum may be missed. This phenomena is known as premature convergence and is discussed in further detail in section II.

Several different techniques have been presented in the past to combat the phenomenon of premature convergence and to encourage a more thorough exploration of the search space. These techniques, for example, may alter fitness values (see [6]), randomise encodings [19], dynamically adapt the population size [8] or vary the selection pressure [1]. In either case, such methods must ensure that the generation of diversity does not interfere too much with the ongoing search process. Randomising large parts of the population, for example, certainly generates diversity, but it also limits the algorithm's ability to exploit promising regions of the search space. The method presented in this paper, iAS, generates a significant amount of diversity allowing the use a minimal population and thereby reducing computational overhead. Our results indicate that it is possible to use a population of only 2 individuals without getting trapped in local optima: iAS does not disrupt the ongoing search process but instead significantly improves the algorithm's performance in almost all cases considered here.

The rest of this paper is structured as follows: Section II discusses the issue of diversity in GAs. Section III briefly reviews the existing approaches in the literature that are related to the nature-inspired technique presented here. Section IV describes in detail the workings and background of iAS followed by an outline of the three test problems considered in section V. The experimental setup is presented in section VI followed by the results and analysis in sections VII and VIII respectively. The paper is concluded in section IX including a brief summary of potential future work.

II. Diversity

In the field of computer science, the term "nature inspired paradigm" essentially describes the abstraction of a natural process that fits a particular computational framework. In the case of evolutionary algorithms, inspiration is derived from evolutionary systems that progress by means of natural selection. However, upon close inspection, evolutionary algorithms often bear little resemblance to their natural counterparts. Such restriction of detail is required to meet the goals of evolutionary algorithms which are reliability and efficiency in terms of problem solving. Unlike models or simulations of natural processes, evolutionary algorithms are not primarily concerned with being an accurate emulation

Philipp Rohlfshagen is with the School of Computer Science, University of Birmingham, UK (email: P.Rohlfshagen@cs.bham.ac.uk).

John A. Bullinaria is with the School of Computer Science, University of Birmingham, UK (email: J.A.Bullinaria@cs.bham.ac.uk).

of a natural phenomena. Natural processes have evolved over considerable periods of time, depend on numerous accidents throughout their evolutionary history, and often differ significantly in their design from the blueprint that an external designer might develop when creating an equivalent process from scratch (see [3]). In particular, many processes in nature rely upon massive parallelism and require considerable periods of time to make progress. Such luxuries are usually unavailable to the computer scientist whose aim is to solve difficult optimisation problems in the shortest time possible using only limited resources.

One consequence of this dilemma is that the average population size used in GAs is typically as small as 100 individuals[1]. Given the usually immense size of the problem space being explored, the population's sampling power is necessarily restricted to a tiny fraction of the search space. Choosing an appropriate population size is crucial and numerous theoretical studies exist that attempt to determine optimal population sizes (see, for example, [10], [21]). In general, a large population samples a greater fraction of the search space and subsequently delays premature convergence. If the size of the population is too large, however, convergence may be slowed down too much preventing the algorithm to locate the global optimum given the allocated resources. A small population size on the other hand is computationally more efficient but very prone to get trapped in local optima. It is therefore extremely important to maintain good levels of diversity when using small populations to ensure an efficient exploration of the search space. The ability of the crossover operator to explore novel parts of the search space diminishes as the diversity in the population decreases. In the simple GA, the mutation operator, which is used to inflict spontaneous change upon newly generated offspring, is the only source of diversity. Mutations alone are often insufficient to maintain diversity in the face of selection pressure, resulting in the loss of diversity in the algorithm's population. Such premature convergence is a very common problem in the field of evolutionary computation and results in the algorithm's inability to escape local optima.

The importance of this problem is reflected in the number of publications attempting to address it. The wealth of different approaches also highlights the difficulty in maintaining high levels of diversity without disrupting the ongoing search process. Convergence, of course, is a vital part of the algorithm's success allowing the algorithm to focus on the most promising regions in the search space. Without any convergence, the search would be random. However, given the usual lack of problem specific knowledge, it is impossible to determine a priori which regions of the search space are the most promising and the algorithm typically converges to regions that *seem* the most promising but may turn out to be misleading. This phenomena is highlighted by fully deceptive problems that gradually lead the population to a local optimum that is located the maximum possible Hamming distance from the global optimum.

A good indicator of diversity is the distribution of alleles at each locus across the population over time. A randomly initialised population will have an even distribution of alleles (1's and 0's) at each locus and the average numerical value will thus be close to 0.5. The average distance to 0.5 across all loci over time is being used as a measure of diversity. The maximum possible degree of diversity is 1 meaning each locus has a perfectly even distribution of alleles.

III. Related Work

This section briefly reviews previous work related to the concepts explored here. Approaches dealing with premature convergence are too numerous to be fully reviewed in this paper. This review restricts itself to techniques presented elsewhere that draw inspiration from biological processes identical or similar to alternative splicing.

Genetic algorithms (GAs) were originally inspired by the field of population genetics and most notably Fisher's genetical theory of natural selection [9]. In recent years, however, advances in molecular genetics, including the genome sequencing projects, have triggered an interest in abstractions that are more directly inspired by the biochemical information processing architecture of eukaryotic cells. For example, properties of the genetic code have been exploited successfully by Karuptga and Gosh [14] and a simple implementation of RNA editing, a post-transcriptional process that selectively modifies individual nucleotides, has been presented by Huang and Rocha [13].

It appears that there have been no direct implementations of alternative splicing in the literature except for our own previous work on this subject (see below). There are, however, some approaches that explore the phenomena of alternative expression: One of the earliest of those being Levenick's Swappers [16]. Swappers are very simple encodings that consist of two parts, one of which is expressive (active) at any one time. This is somewhat similar to a dynamic exon-intron structure and Levenick showed how such dynamic expression may be useful in accelerating the algorithm's rate of adaptation. A similar approach that utilises the alternative expression of individuals is due to Yang [22]: Yang uses the concept of duals, encodings that are maximum Hamming distance from their source. A clever triggering machinisms stochastically creates duals of weak individuals in the population to increase their fitness. Yang's work, however, has been inspired by the complementary nature of chromosomes found in diploid organisms and not the molecular process described in the next section. Indeed, there is a whole body of research investigating the potential benefits of diploid encodings in artificial evolution. Diploidy is, however, conceptually very different to alternative splicing and usually requires a dominance scheme that defines which chromosome is active at any one time.

The Structured GA due to Dasgupta and McGregor [7] is another example of an approach that is similar in intent, but different in conception, to the approach presented here: The Structured GA uses a control sequence to determine

[1] An informal review of population sizes used in the literature suggests that the most common value is approximately 100 individuals.

which designated segment of the encoding is active. Only one segment may be active at any one time, allowing the remaining segments to accumulate neutral mutations. The authors suggested this approach to deal with dynamic environments. Dynamic environments have received special attention when it comes to the issue of maintaining diversity as it is highly important to maintain high levels of diversity in the face of an ever changing fitness landscape, possibly more so than in the static cases. Numerous approaches have been suggested for this type of domain, but only one is directly related to the phenomenon of alternative splicing: We previously presented a modular encoding the phenotype of which is determined by a series of control sequences. The problem's variables are stored as dynamic elements that may move from one segment to another. The encoding allowed the identification of linkages amongst a succession of finite states. This implicit memory, in turn, allowed an accelerated rate of adaptation [20]. As segments are predetermined, we labelled that approach as explicit. The approach presented here is implicit as segments are determined on-the-fly.

IV. Implicit Alternative Splicing

The technique presented in this paper is called implicit alternative splicing (iAS). It has been inspired by the post-transcriptional process of alternative splicing that frequently occurs in eukaryotic cells. DNA, the hereditary carrier of information, contains well defined regions, called genes, that consist of instructions to produce proteins. Genes are usually composed of exons and introns in an alternating fashion. The usual pathway of expression first transcribes DNA to RNA and subsequently modifies the RNA strand in an event called RNA processing. The typical scenario is to remove all introns and to retain all exons. However, there are numerous exceptions to this and frequently introns are retained or exons are skipped during RNA processing, allowing the production of a wide range of different proteins from the same underlying strand of DNA. Alternative splicing was thought to occur in 40-60% of all human genes [11] but it is now estimated that it may affect a far larger proportion of genes. The significance of this process is best highlighted by the case of drosophila where the fly's gender is determined by alternative splicing: The sex-specific splicing pattern for females expresses exon 4 while the male one does not [4]. Alternative splicing, often in conjunction with RNA editing, is thought to be highly relevant to the vast proteomic diversity that is evident in higher organisms. There are numerous additional benefits of alternative splicing, including temporary suspension of selection pressure on individual segments, which are highly interesting but not explicitly considered within this paper.

Our implementation of alternative splicing is necessarily highly abstract. A schematic view is shown in Figure 2: The search of an alternative expression is essentially reduced to a search for an 'exon'. An exon in terms of the binary encoding is any segment that may be expressed alternatively without degrading the encoding's quality. By an alternative expression here we mean the inversion of a series of bits. In order to find the largest possible segment suitable for inversion we repeatedly apply our method in a top-down fashion. Initially, we randomly divide the encoding into two halves and invert either one of them and evaluate the resulting encodings (that is, the inverted half together with the original and unaltered half). We then proceed by taking the 'path of least resistance' and reapply the method to the segment that caused the lesser decline in fitness. In other words, we divide the half that proved more successful and repeat the evaluation (a quarter of the encoding is now being inverted). At any stage, once we have found an inversion that produces a fitness value greater to the original fitness, we terminate the process. If the inversion is neutral in regard to fitness, we terminate with some predetermined probability p. On exit, we immediately apply the successful inversion to the original encoding.

It is debatable whether our implementation of iAS bears a significant relationship to its natural counterpart as there are numerous important differences. For example, alternative splicing in nature works by temporarily suspending selection pressure for certain splice forms which are subsequently free to accumulate mutations. This accelerated rate of change may ultimately produce a new protein (see [18]). A purely neutral approach is too costly for our purposes and we instead uphold selection pressure while allowing a temporary decrease in fitness. However, given the intrinsic differences between natural and artificial evolution, we deem the lack of biological plausibility of little significance in the light of the utility afforded by iAS (see section VII). Although the field of molecular genetics provides fruitful inspiration, biological plausibility will often need to be traded for computational efficiency.

The probability of finding large segments that may be inverted successfully diminishes as the encoding approaches its globally optimum state. In order to increase the chance of finding a large segment, we test q different divisions at each stage of iAS and proceed with the most successful one. In other words, when splitting the encoding in half, we test several possible divisions and choose the one with the least decline in quality. There are thus a total of two parameters that control iAS: The probability p determines whether iAS terminates as soon as a segment has been found the inversion of which is at least neutral in terms of quality. The parameter q, on the other hand, determines the number of trials devoted to finding the best possible division at each stage. The total number of possible divisions clearly depends on the number of bits currently being considered and we reduce q linearly, if possible, such that q is never greater than the total possible number of unique divisions. If, for example, we are considering 100 bits, there are $0.5 \times (100!/(50! \times 50!))$ unique ways to create two mutually exclusive segments of size 50. We test q randomly chosen division and proceed with the most promising one at which stage q will be reduced by a constant factor.

iAS obviously bears a significant cost in terms of function evaluations, and we reduce this cost by potentially (depending on p) terminating the process as soon as an encoding

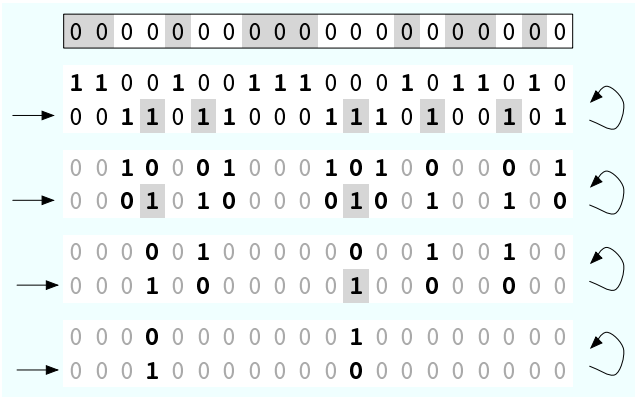

Fig. 1. A schematic view of implicit alternative splicing: The original encoding in this simple example is all zeros. The encoding is randomly divided into two halves (one half highlighted by grey squares), each of which is inverted in turn (bold numbers). At each stage, q different divisions are tested and the inversion with the least loss of fitness (highlighted by arrow) is used to repeatedly apply the technique. If at any stage, the fitness of the modified encoding is better than the original one, the method terminates. If the inversion is neutral, we terminate with some probability p. Otherwise, execution proceeds until the length of the segment is 1 bit.

has been found that is at least as good as the original one. This equality criterion is also essential for the generation of diversity as it increases the chances of success in finding a segment for inversion. The very small population size is also crucial in limiting the number of function evaluations consumed by iAS although depending on the total number of function evaluations allowed, larger populations may be used to address problems of increasing difficulty. Using a population of size 2 is very uncommon for GAs for reasons outlined in the previous section. However, other evolutionary algorithms such as evolution strategies (ES) often use very small populations. In fact, the approach presented here may be compared to $(1 + \lambda)$ ES where only a single individual is kept at any one time: At each stage, λ offspring are generated which then compete with their parent to be in the next generation. This is somewhat similar to iAS which tries numerous different encodings before competing to replace one of its parents.

V. Optimisation Problems

We tested iAS on three different optimisation problems. The first one, an artificially created 'toy'-problem allows us to investigate the characteristics of iAS in different search spaces. The second problem is Kauffman's NK-fitness landscape which is designed to allow easy tuning of the degree of ruggedness of the search space. The third and final problem is the well known Boolean satisfiability problem. Each of these problems is described in more detail below.

A. Problem 1

The first problem is artificially created to generate search spaces with different attributes in a controllable manner. It has been inspired by a very similar approach due to Yang [23]: The function to be optimised is constructed as a series of 4-bit segments. The value returned by each segment depends upon the chosen cost matrix and the number of ones within that segment. The cost matrix $[4, 3, 2, 1, 0]$, for example, returns a score of 4, 3, 2, 1 or 0 given 0, 1, 2, 3 or 4 ones within that segment (i. e. the number of ones is used as an index). This problem enables efficient modelling of different search spaces, some of which are known to be difficult for GAs. We use a target function that consists of 25 segments (100 bits) and has an optimum value of (25×4) 100. The cost matrices used are as follows: $[0, 1, 2, 3, 4]$ (all ones), $[0, 2, 2, 4, 4]$ (neutrality) and $[3, 2, 1, 0, 4]$ (fully deceptive). Each segment is distributed equally across the encoding. In order to make the problem slightly more realistic, we generate search spaces as a mixture of different cost matrices. We denote the type of the search space as a series of three values, indicating the percentage of the search space each cost matrix occupies. The type 0.33 - 0.33 - 0.33, for example, indicates that the search space is composed of 1/3 of each cost matrix and thus combines all three attributes. A type 1 - 0 - 0 would be the classical all-ones (or max-ones) problem.

B. Problem 2

The second problem is Kauffmann's NK-fitness model [15]. Kauffmann developed this problem to study the effects of genetic interactions (epistasis): Each bit contributes towards the encoding's fitness. The contribution depends upon the state of the bit itself and the state of all k bits that are linked to it. The more the bits are dependant on one another, the more rugged the search space is. The fitness values for each bit are generated randomly in the range $(0, 1)$ for each possible state and are stored in a look-up table. The final fitness is the average contribution of all bits. For random neighbourhoods, this problem has been shown NP-complete for values of $k \geq 2$ (see [2]).

C. Problem 3

The third problem under consideration is the well-known Boolean satisfiability problem, that underlies many important applications, such as computer hardware design. The instances used here are NP-complete [5]. Furthermore, this problem is an ideal binary search problem highly suitable for the technique presented here. In this paper we will use benchmark problems found online[2]: These instances have exactly three variables per clause (3-SAT) and range in size from $(n = 20, m = 91)$ to $(n = 175, m = 645)$ where n is the total number of variables and m is the number of clauses. All instances are uniformly random and completely satisfiable. There are numerous ways to represent this particular problem (e.g. path representation, graded fitness). We use the probably simplest representation and assign a fitness value to the encoding that is directly equivalent to the number of clauses that are being satisfied.

VI. Experimental Setup

The GA used to test iAS has a population size of 2: The two individuals are crossed over using uniform crossover

[2] www.cs.ubc.ca/~hoos/SATLIB/benchm.html

TABLE I

EMPIRICAL RESULTS FOR PROBLEM 1: FOR EACH DIFFERENT TYPE OF SEARCH SPACE, THE PERCENTAGE OF SOLVED TRIALS IS SHOWN (%) WITH THE AVERAGE NUMBER OF FITNESS EVALUATIONS (FE AVG I) REQUIRED TO FIND THE GLOBAL OPTIMUM (EXCLUDING UNSUCCESSFUL TRIALS), AND THE FINAL AVERAGE FITNESS AFTER 20,000 FUNCTION EVALUATIONS (FIT AVG) WITH THE AVERAGE NUMBER OF TOTAL FUNCTION EVALUATIONS REQUIRED (FE AVG II). BOLD NUMBERS INDICATE THE SEEMINGLY BEST RESULTS WITH STATISTICAL SIGNIFICANCE BEING INDICATED BY THE STAR SYMBOL.

		SGA				iAS				S
Case	Type	%	FE Avg I	Fit Avg	FE Avg II	%	FE Avg I	Fit Avg	FE Avg II	
1	1.00 0.00 0.00	100	**1648.27**	100.00	1648.27	100	2689.27	100.00	2689.27	*
2	0.00 1.00 0.00	100	**2797.00**	100.00	2797	100	3064.60	100.00	3064.60	
3	0.00 0.00 1.00	0	-	82.47	20000	0	-	**87.10**	20000	*
4	0.60 0.20 0.20	0	-	96.30	20000	20	11673.33	**98.83**	18334.67	*
5	0.20 0.60 0.20	0	-	96.40	20000	16.67	9763.60	**98.57**	18293.93	*
6	0.20 0.20 0.60	0	-	89.67	20000	0	-	**93.07**	20000	*
7	0.33 0.33 0.33	0	-	93.80	20000	0	-	**96.13**	20000	*

followed by single bit mutation. The resulting offspring then compete with their parents for a place in the next generation whereby the fitter offspring competes with the fitter parent. The performance of iAS is compared to a standard steady state GA (SGA) that uses parameter settings depending on the problem (see below). The parameter settings were established systematically for each type of problem in order to maximise the performance of the SGA. For the purpose of comparison, we also tried to use the standard GA with a population size of 2. The results were significantly worse than those produced by more traditional parameter settings and are thus omitted from the results section. Results are obtained by executing the algorithm multiple times on the same problem using different seeds for the pseudo-random number generator for each trial. Depending on the type and complexity of the problem, we chose parameter settings as follows:

A. Problem 1

The algorithms are executed 30 times for each problem with a maximum of 20,000 function evaluations allowed for each trial. Whenever the global optimum is found within the limit allowed, the algorithm's run is terminated and the number of function evaluations required is noted. The SGA uses a population size of 50, binary tournament selection and binary tournament replacement (with a randomly chosen individual). The probability of the uniform crossover operator is 0.8 and the mutation probability is $1/n$. iAS also uses a crossover probability of 0.8 and a mutation probability of 1/n. Furthermore, the value for the parameters are: $p = 0.5$ and $q = 14$.

B. Problem 2

The algorithm is executed 30 times for each problem of size 100 bits with a maximum of 20,000 function evaluations each time. We explore levels of epistasis $k \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. The higher the value of k, the higher the ruggedness of the search space. The parameter settings for SGA and iAS are identical to problem 1.

C. Problem 3

The benchmark problems being used come in groups of 100 to 1000 instances for each value of $n \in \{20, 50, 75, 100, 125, 150, 175\}$. In order to reduce the computational costs of conducting the experiments, we execute the algorithm 30 times on each of the first 20 instances of each group and do not consider instances with values of $n \geq 200$. We execute the algorithms for $n * 250$ and $n * 500$ function evaluations. The SGA uses a population of size 100, random selection and binary tournament replacement. For both approaches, the crossover probability is 0.8 and the mutation probability is $1/n$. For iAS, $p = 0.5$ and $q = 1$.

VII. RESULTS AND ANALYSIS

This section presents the results for each of the three problems being investigated. The results are compared and analysed with further discussion to be found in the next section. Statistical significance has been established using t-tests and a significance value of 0.005.

A. Results Problem 1

The results for the first problem are shown in Table I. The SGA manages to solve 2 of the cases while iAS solves at least some instances in 4 of the 7 cases. In the two cases where all instances are solved (1 and 2), the performance of either technique may be judged by the number of function evaluations required to reach the global optimum. In the other 5 cases, the final accuracy is used as the measure for comparison. The table shows that iAS is significantly better in 5 of the 7 cases and significantly worse in 1 case. iAS requires more function evaluations for the simplest case (all-ones problem) which seems to suggest that no advanced techniques are required for such simple uni-modal search space where conventional approaches fare better.

Figure 2 shows the average level of diversity across fixed intervals for both the SGA and iAS algorithms for case 7. As expected, the final average diversity for SGA approaches zero quickly and remains in that state until the end of the run. In other words, all individuals in the final population are

TABLE II

THE FREQUENCY OF SUCCESSFUL INVERSIONS BY SEGMENT SIZE FOR PROBLEM 1. ALL POSSIBLE SEGMENT SIZES ARE SHOWN IN THE LEFTMOST COLUMN WITH THE DIFFERENT CASES ACROSS THE SECOND ROW FROM THE TOP. THE VALUE 0 DENOTES AN UNSUCCESSFUL ATTEMPT TO FIND A SEGMENT.

Size	Frequency						
Case→	1	2	3	4	5	6	7
50	4.87	4.60	5.73	5.33	5.43	5.90	5.07
25	9.03	7.37	3.47	7.73	8.37	5.23	6.87
12/13	10.97	12.17	8.27	12.17	11.50	10.90	11.80
6/7	12.93	13.03	12.77	12.57	16.17	15.63	14.30
3/4	7.10	8.83	16.00	13.00	13.13	14.90	13.67
1/2	1.30	2.83	14.40	9.07	9.43	12.63	10.33
0	0.00	0.30	139.27	129.17	125.70	137.97	141.50

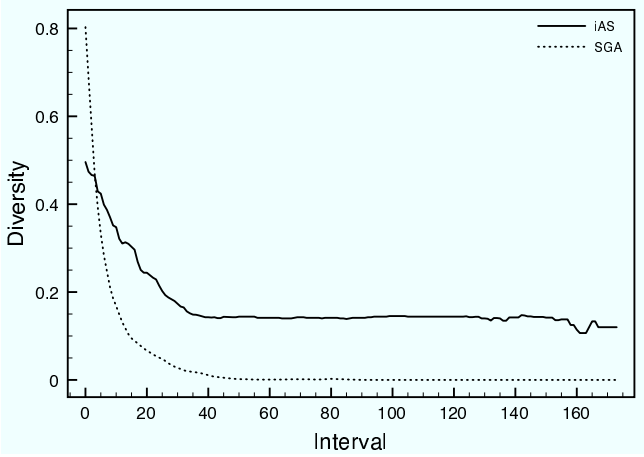

Fig. 2. The convergence over time of SGA and iAS: The SGA quickly loses all its diversity and remains in that homologous state until the run terminates. iAS, on the other hand, also loses diversity initially but retains a certain level of diversity throughout the entire run despite having a population size of only 2.

identical to one another and the variation operators are unable to lift the population from the corresponding local optimum. On the other hand, iAS maintains moderate levels of diversity throughout the entire run and subsequently produces superior results to the SGA. The behaviour of iAS is best highlighted by looking at the kinds of inversions that occur throughout the algorithm's run: Table II shows the average frequency of the various segment sizes successfully inverted using iAS. The data is averaged over all 30 runs for the 0.33 - 0.33 - 0.33 problem instance. As the encoding consists of 100 bits, the following segment sizes are possible: 50 - 25 - 12/13 - 6/7 - 3/4 - 1/2. All attempts that do not locate a segment suitable for inversion are marked as 0. The table shows that unsuccessful attempts do indeed constitute the majority of cases. However, larger segment inversion do occur with regular frequency allowing the algorithm to escape from local optima and to generate diversity for the crossover operator to act upon.

TABLE III

EMPIRICAL RESULTS FOR PROBLEM 2 AFTER 20,000 FUNCTION EVALUATIONS FOR SGA AND IAS GIVEN DIFFERENT VALUES OF k. THE SEEMINGLY BEST OUTCOMES ARE IN BOLD WITH STATISTICAL SIGNIFICANCE INDICATED BY THE STAR SYMBOL.

k	SGA	iAS	S
1	0.7119	**0.7137**	
2	0.7352	**0.7412**	
3	0.7459	**0.7561**	
4	0.7499	**0.7622**	*
5	0.7489	**0.7586**	*
6	0.7410	**0.7557**	*
7	0.7399	**0.7482**	
8	0.7294	**0.7412**	*
9	0.7235	**0.7365**	*
10	0.7202	**0.7297**	

B. Results Problem 2

The NK-fitness landscape allows one to study the behaviour of algorithms on search spaces with varying degrees of epistasis. The results from this experiment are shown in Table III. As the value for k increases, iAS outperforms the SGA and is significantly superior in 5 of the 10 cases. Again, to understand the operation of the algorithm it is best to look at the average frequencies of segment sizes that are being inverted successfully, as shown in Table IV. Unlike with Problem 1, there is a tendency towards segments of size 1/2: Although larger inversions do occur at any value of k, there is a clear tendency towards inversions of smaller sizes as k increases. The number of unsuccessful attempts increases also. This seems to indicate that, although higher levels of epistasis prevent larger invertable segments from being found, the inversion of only a few bits still yields significant improvements.

C. Results Problem 3

Problem 3 is the only problem under consideration that has real-world applications, and is therefore of special interest. The results for this case are shown in Table V, and once again iAS outperforms the SGA, both in terms of reliability and accuracy. No statistical significance has been established in this case as the differences in success rate between the

TABLE IV

THE FREQUENCIES OF SEGMENT-SIZES SUCCESSFULLY IDENTIFIED BY IAS AS SUITABLE FOR INVERSION IN THE CASE OF PROBLEM 2. ALL POSSIBLE SEGMENT SIZES ARE SHOWN ON THE LEFTMOST COLUMN WITH THE DIFFERENT VALUES OF k ACROSS THE SECOND ROW FROM THE TOP. THE VALUE 0 DENOTES AN UNSUCCESSFUL ATTEMPT TO FIND A SEGMENT.

Size	Frequency									
k→	1	2	3	4	5	6	7	8	9	10
50	5.90	5.37	5.67	5.97	6.17	6.83	7.67	6.67	5.40	6.63
25	7.77	6.40	4.90	4.27	3.07	1.93	2.33	2.40	1.70	1.50
12/13	13.63	13.20	9.63	8.53	7.07	5.20	4.20	3.10	2.87	2.40
6/7	17.70	19.73	18.73	16.57	13.80	10.93	9.00	8.30	6.37	6.43
3/4	23.23	25.20	27.83	26.33	23.33	22.03	19.93	18.33	16.43	15.17
1/2	19.77	25.47	28.93	30.77	29.80	31.53	30.50	29.30	31.10	28.33
0	119.70	111.70	108.97	110.80	117.47	120.53	124.93	128.63	130.23	134.30

TABLE V

RESULTS FOR THE 3-SAT PROBLEM: THE AVERAGE PERFORMANCE OF EACH ALGORITHM IS SHOWN AS THE AVERAGE NUMBER OF SOLVED TRIALS ACROSS THE 20 DIFFERENT INSTANCES TESTED FOR EACH VALUE OF n (%). THE FINAL AVERAGE VALUE IS SHOWN ALSO (AVG). THE TOP ROW SHOWS THE NUMBER OF FUNCTION EVALUATIONS THE ALGORITHM WAS EXECUTED FOR. THE RIGHTMOST PAIR OF COLUMNS SHOW THE LEVEL OF IMPROVEMENT DUE TO THE INCREASE IN THAT LIMIT.

		$n \times 250$				$n \times 500$				Change	
		SGA		iAS		SGA		iAS			
n	m	%	Avg	%	Avg	%	Avg	%	Avg	SGA	iAS
20	91	83.00	90.82	**89.50**	90.89	84.17	90.83	**91.17**	90.91	1.17	1.67
50	218	31.17	216.83	**60.17**	217.50	34.17	216.94	**65.50**	217.58	3.00	5.33
75	325	6.33	322.82	**38.00**	324.11	9.17	323.03	**45.67**	324.29	2.83	7.67
100	430	5.67	426.98	**23.67**	428.84	7.00	427.34	**29.50**	429.05	1.33	5.83
125	538	1.17	533.93	**15.17**	536.49	2.17	534.36	**23.00**	536.80	1.00	7.83
150	645	0.50	640.33	**15.17**	643.42	1.33	640.89	**25.00**	643.81	0.83	9.83
175	753	0.00	747.30	**3.83**	750.93	0.00	748.03	**9.00**	751.38	0	5.17

two approaches are very large. It is interesting to note that once the limit on the number of function evaluations allowed is doubled, iAS manages a considerable higher increase in performance in almost all cases while the SGA improves its performance only marginally. Again, this is due to the loss of diversity: Once the SGA converges prematurely, the variation operators alone are insufficient to escape the local optimum, and further improvements are rarely possible.

VIII. DISCUSSION

We have tested our new diversifying technique, iAS, on three different optimisation problems and found it to increase significantly the algorithm's performance in almost all instances. The rise in performance is due to a constructive increase in the population's diversity despite a minimal population size. By constructive we mean that not only does the generated diversity not interfere with the ongoing search process, but indeed speeds it up by increasing the fitness values of the underlying encodings. As mentioned earlier, the algorithm's population is expected to converge towards promising regions of the search space over time. If, however, convergence happens too quickly, the algorithm may lose its ability to escape from local optima. As it is sufficient that a single individual in the final population encodes the global optimum, it is important that the population as a whole is partly converged towards promising regions of the search space while maintaining sufficient diversity to escape from that region in the event that it does not contain the global optimum. This is what we seem to observe for iAS.

The design of the proposed GA is different from the conventional approach as it uses a population of only 2 individuals. The reason for this is two-fold: iAS requires a significant number of function evaluations and having a small population allows to fully exploit the benefits of this method without exceeding the number of function evaluations allowed. A larger population would require a considerable number of function evaluations given iAS is applied to every single individual which would prevent a sufficient number of generations to be processed prior termination of the algorithm. If, however, further computational resources are allocated to solve a certain problem, larger population sizes should be investigated. Secondly, small populations are generally computationally efficient as long as diversity may be maintained: The ability of iAS to invert segments of large size allow it to escape from local optima and to maintain a constant level of diversity. This effect is subsequently utilised by the crossover event allowing to combine superior sub-

solutions in a single individual. We also tested iAS with a population of a single individual using only mutations as additional source of variation. The performance was significantly worse indicating the benefit afforded by the crossover operator.

IX. Conclusion

In this paper we have presented a new technique, implicit alternative splicing (iAS), that significantly increases the population's diversity in a genetic algorithm (GA). This technique allows the use of a minimal population size and thereby reducing computational overhead. iAS is inspired by the post-transcriptional process of alternative splicing that is commonly found in eukaryotic cells and is thought to play a major role in the generation of proteomic diversity evident in higher organisms. We have previously demonstrated how an abstract interpretation of this biochemical process may be utilised successfully in dynamic domains [20]. In this paper, we have proposed a different implementation to deal with with three different kinds of static optimisation problem. Each of these problems has been chosen to identify different characteristics of iAS. We have shown that iAS is able to significantly increase the population's diversity and to enhance the overall quality of the solutions found. iAS is a top-down technique that aims to invert the largest possible segment of a binary encoding without loss of the encoding's fitness. Therefore, unlike local search, it is possible to escape local optima by inverting multiple bits simultaneously.

The results presented in this paper are promising, and future work will involve identifying further real-world domains that may benefit from this technique. Furthermore, our current form of iAS only works with binary encodings, and it will be interesting to determine whether an equivalent approach may be developed for permutation-type problems. It might also prove interesting to investigate whether the use of specialised variation operators could exploit the additional diversity afforded by iAS to increase the algorithm's performance even further.

Finally, it is worth noting that an important feature demonstrated by this study is the rich source of inspiration that molecular genetics can be for the field of evolutionary computation. Although iAS is highly abstract, and possibly bears only minimal resemblance to its natural counterpart, the inspiration was drawn from the study of the information processing architecture of the cell, which has much more to offer than the field of evolutionary computation currently exploits. This is an exciting prospect, and as our understanding of molecular systems increases over time, we believe future work in the field of evolutionary computation should attempt to identify further potential benefits of such phenomena.

X. Acknowledgments

This work was supported by a Paul and Yuanbi Ramsay Scholarship.

References

[1] M. Affenzeller and S. Wagner. The influence of population genetics for the redesign of genetic algorithms. *Journal of Systems Science*, 30:41–49, 2004.

[2] L. Altenberg. NK fitness landscapes. In *The handbook of evolutionary computation*, page B2.7.2. Oxford University Press, 1997.

[3] H. Atlan. The living cell as a paradigm for complex natural systems. *ComPlexUs*, 1:1–3, 2003.

[4] B. S. Baker. Sex in flies: The splice of life. *Nature*, 340:521–524, 1989.

[5] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971. Association for Computing Machinery.

[6] P. Darwen and X. Yao. Every niching method has its niche: Fitness sharing and implicit sharing compared. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 398–407, Berlin, 1996. Springer.

[7] D. Dasgupta and D. R. McGregor. Nonstationary function optimization using the structured genetic algorithm. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 145–154, Amsterdam, 1992. Elsevier.

[8] A. E. Eiben, E. Marchiori, , and V. A. Valko. Evolutionary algorithms with on-the-fly population size adjustment. In X. Y. et al., editor, *Parallel Problem Solving from Nature PPSN VIII*, pages 41–50. Springer, 2004.

[9] R. A. Fisher. *The Genetical Theory of Natural Selection*. Oxford: Clarendon Press, 1930.

[10] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. L. Miller. The gamblerś ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.

[11] E. D. Harrington, S. Boue, J. Valcarcel, J. G. Reich, and P. Bork. Estimating rates of alternative splicing in mammals and invertebrates. *Nature Genetics*, 36(9):915–917, 2004.

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

[13] C.-F. Huang and L. M. Rocha. Exploration of RNA editing and design of robust genetic algorithms. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*. IEEE Press, 2003.

[14] H. Karuptga and S. Gosh. Towards machine learning through genetic code-like transformations. *Genetic Programming and Evolvable Machine Journal*, 3(3):231–258, 2002.

[15] S. A. Kauffman. *The origins of order*. Oxford University Press, 1993.

[16] J. R. Levenick. Swappers; introns promote flexibility, diversity and invention. In F. Orlando, W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. M. Honavar. Jakiela, and R. Smith, editors, *GECCO-99: Proceeding of Genetic and Evolutionary Computation Conference*, volume 1, pages 361–368, San Francisco, CA,, July 1999. Morgan Kaufmann.

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

[18] B. Modrek and C. J. Lee. Alternative splicing in the human, mouse and rat genomes is associated with an increased frequency of exon creation and/or loss. *Nature Genetics*, 34(2):177–180, 2003.

[19] M. Rocha and J. Neves. *Lecture Notes in Computer Science: Multiple Approaches to Intelligent Systems*, volume 1611/2004, chapter Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation, pages 127–136. Springer, 1999.

[20] P. Rohlfshagen and J. A. Bullinaria. Alternative splicing in evolutionary computation: Adaptation in dynamic environments. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computing (CEC 2006)*, Piscataway, NJ, 2006. IEEE.

[21] R. E. Smith. Population size. In T. Baeck, D. B. Fogel, and Z. Michalewiczs, editors, *Handbook of Evolutionary Computation*, pages E1.1:1–E1.1:5. Institute of Physics Publishing and Oxford University Press, 1997.

[22] S. Yang. PDGA: the primal-dual genetic algorithm. In A. Abraham, M. Koppen, and K. Franke, editors, *Design and Application of Hybrid Intelligent Systems*, pages 214–223. IOS Press, 2003.

[23] S. Yang. Memory-based immigrants for genetic algorithms in dynamic environments. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, volume 2, pages 1115–1122. ACM Press, 2005.