Abstract
In this article we present work on chromosome structures for genetic algorithms (GAs) based on biological principles. Mainly, the influence of noncoding segments on GA behavior and performance is investigated. We compare representations with noncoding sequences at predefined, fixed locations with "junk" code induced by the use of promoter/terminator sequences (ptGAs) that define start and end of a coding sequence, respectively. As one of the advantages of noncoding segments a few researchers have identified the reduction of the disruptive effects of crossover, and we solidify this argument by a formal analysis of crossover disruption probabilities for noncoding segments at fixed locations. The additional use of promoter/terminator sequences not only enables evolution of parameter values, but also allows for adaptation of number, size, and location of genes (problem parameters) on an artificial chromosome. Randomly generated chromosomes of fixed length carry different numbers of promoter/terminator sequences resulting in genes of varying size and location. Evolution of these ptGA chromosomes drives the number of parameters and their values to (sub)optimal solutions. Moreover, the formation of tightly linked building blocks is enhanced by self-organization of gene locations. We also introduce a new, nondisruptive crossover operator emerging from the ptGA gene structure with adaptive crossover rate, location, and number of crossover sites. For experimental comparisons of this genetic operator to conventional crossover in GAs, as well as properties of different ptGA chromosome structures, an artificial problem from the literature is utilized. Finally, the potential of ptGA is demonstrated on an NP-complete combinatorial optimization problem.

Keywords
Genetic algorithms, chromosome structures, promoter/terminator sequences, noncoding segments, spontaneous crossover, combinatorial optimization.

1. Introduction

Genetic algorithms (GAs) are a class of evolutionary algorithms (EAs) modeling the principles of evolution in order to solve complex optimization problems in science and engineering. Different solutions to a problem form a population of fixed-length bit strings (chromosomes), whose fitness (quality of solution) determines the probability of passing on its genetic material to the next generation. Random mutation of specific bits of chromosomes alter the solution (individual) so as to explore the search space for possibly better individuals. Crossover between two (or more) individuals at one or more randomly chosen sites exchanges parts of
genetic material, potentially combining building blocks of fit individuals to a solution closer to the optimum. This is the essence of the building block hypothesis in Holland (1975) and Goldberg (1989).

The problem with this intuitively convincing concept is that the GA "knows" nothing about the building blocks' structure, as no part of a conventional GA chromosome is more salient than any other, i.e., the chromosome has no explicit structure. Specifically, essential building blocks have the same probability to be destroyed as less important ones (provided they are of the same length). Thus, a few researchers have begun to take a closer look at genetic structures in biological systems, specifically, the genetic information encoded on DNA (deoxyribonucleic acid) molecules.

In nature, structure (in terms of information content) is primarily induced by coding and noncoding sequences demarcated by promoter and terminator sequences. Within the coding sequences, genetic material can be categorized as (coding) exons or (noncoding) introns. A striking, natural analog to the building block hypothesis is the exon shuffling hypothesis (Gilbert, 1978, 1987; Doolittle, 1978). It is hypothesized that exons code for functional subunits of biological structures, and shuffling of already existing partial solutions (exons) could accelerate evolution toward more complex structures.

We loosely transfer these biological principles to the chromosome structures for GAs and term this extension ptGA, where p stands for promoter and t for terminator. In a strict sense, the GA still has no "knowledge" of this new structure, as it treats all sequences the same. However, the disruptive effects of crossover with respect to building blocks are reduced with the altered chromosome structure. Moreover, we make the knowledge of this structure available to a new, nondisruptive crossover operator emerging from the chromosome structure containing noncoding sequences. As a consequence, the ptGA utilizing this kind of crossover only combines building blocks (shuffles exons) without destroying them. Still, mutation is able to destroy building blocks and to act more aggressively than with conventional GA chromosomes, specifically, when mutating a p/t-sequence.

We could additionally limit mutation to coding sequences, but this would contradict our initial nonbiological motivation for the use of noncoding sequences. The starting point of this work was the idea of a GA not only optimizing problem parameter (hereafter, parameter) values, but also the number of parameters (specifically, the number of hidden neurons in an artificial neural network). This optimization can be achieved by mutation of p/t-sequences or noncoding sequences, resulting in possible destruction or generation of a coding sequence (parameter), respectively. There are a variety of real-world problems to which the ptGA could be applied and for which the (sub)optimal number of parameters is not known in advance.

This article is organized as follows:

In Section 2 the basic information processing aspects of molecular genetics are summarized with specific emphasis on gene structure. We discuss the gene structures in basic groups of biological organisms and the way they are processed in order to build higher-level biological building blocks. We review theories on the purpose and origin of noncoding segments in the genetic code, and we conclude with the presentation of the basic principles transferred from biology to machines, specifically, to the proposed ptGA.

In Section 3 we review GA instances allowing for flexible representation of parameters and the development of the use of noncoding regions on GA chromosomes in the past decade.

In Section 4 we conduct experiments with ptGA chromosomes employing noncoding regions at fixed locations by means of a simple example taken from literature (Levenick, 1987).
Moreover, we give the results of a formal analysis of the disruption-reducing effects of noncoding sequences at fixed locations with respect to crossover.

In Section 5 we utilize p/t-sequences and we allow not only for evolution of parameter values, but also for adaptation of number, size, and location of parameters on randomly generated fixed-length strings. The distribution of the number of coding segments and their length on randomly generated ptGA chromosomes is analyzed. We introduce a new, nondisruptive crossover operator emerging from the ptGA gene structure in a self-organizing manner and compare this mechanism to conventional crossover in GAs.

In Section 6 we employ the ptGA for the successful treatment of three instances of an NP-complete combinatorial optimization problem. The results are compared to conventional GA performance described in the literature (Khuri, Bäck, & Heitkötter, 1994).

In Section 7 we conclude this article by summarizing the results. We refer to partially existing ptGA extensions, reflect on additional ptGA features to be adopted, and identify possible problem domains for future ptGA applications.

2. Molecular Genetics

We briefly summarize the stages of DNA processing by representing the most essential information-processing aspects of biological systems at the molecular level. Before delving into details, we introduce the two most fundamental types of organisms. Prokaryotes (viruses, bacteria, and blue-green algae) lack a membrane-bound, structurally discrete nucleus and other subcellular compartments. The term eukaryotes refers to all other organisms, those consisting of cells with a discrete nucleus. Prokaryotes have nearly no noncoding DNA sequences, whereas eukaryotes have many of these segments.

The majority of the information required to develop a biological system and keep it alive is encoded in genes that are concatenated on DNA molecules—the codebook of life. Two strands of nucleotides form an antiparallel double helical structure comprising the DNA. The strands contain four different deoxyribonucleotides (bases) building pairs, namely, adenosine (A), thymidine (T), and cytidine (C), guanosine (G). These pairs are joined through hydrogen bonds across the strands, resulting in a complementary structure of the two DNA strands—a duplex (Adams, 1991). Sequences of these base pairs (bp) (typically in the range of $10^3$ to $10^4$ bp) constitute a gene containing the assembly instructions for a specific protein. Proteins are the basic functional biological subunits: they enable and regulate all processes. For example, the transportation of oxygen in blood vessels is accomplished by the iron-containing protein hemoglobin, and the production of proteins itself is supported by proteins.

In prokaryotes, DNA forms a single, circular strand, whereas in eukaryotes the complete genome (all genetic information encoded in DNA) is split into packages called chromosomes. The genes needed for the production of cell-specific proteins are transcribed into messenger ribonucleic acid (mRNA). Transcription encompasses copying and processing of a specific DNA sequence. This piece of code is then transferred to a ribosome—the protein factory of a cell—where the genetic blueprint is used to construct a protein.

At a ribosome the information encoded in the mRNA is interpreted in chunks (codons) of three bases (triplets), each of the triplets representing a specific amino acid. As the 20 different amino acids can be encoded in 64 ways (4^3 different triplets out of four nucleotides), various triplets code for identical amino acids. The specific amino acid is carried by transfer RNA (tRNA), which has a complementary triplet (Anticodon) that enables the docking of the tRNA to the mRNA codon (triplet). The amino acids carried to the ribosome by tRNA
interact and combine under the influence of mechanical and electromagnetic forces to the encoded protein (protein folding). Termination of translation of mRNA code into proteins is signalled by specific triplets (stop codons) not coding for an amino acid.

The complete genome of the bacterium *Escherichia coli* (*E. coli*)—the workhorse of prokaryotic geneticists—consists of $4.5 \times 10^6$ bp, whereas the human genome contains $4.0 \times 10^9$ bp, which approximately translates to a storage capacity of 953 MB in terms of computer science. This is roughly the amount of data stored on a hard disk of today’s personal computers. It is extremely fascinating (at least for a nonbiologist) that we carry this huge amount of information in every single cell. Even more interesting is that as little as 9% of the human genome might be coding DNA (Smith, 1989).

2.1 Gene Structure and Transcription

For brevity and less complexity we will only focus on prokaryotes. *E. coli* is a prokaryote with a genome of approximately 5,000 genes arranged along a single, circular DNA molecule. Most genes are present as single copies, and the genes for proteins used for a specific biological process are clustered on the DNA. This arrangement simplifies the transcription process, as these genes can be transcribed at once. Prokaryotic genes are colinear, which means that the genetic code in a gene on the DNA is identical to the code processed by the ribosome, i.e., there are no intervening, noncoding sequences or introns, as in eukaryotes. Transcription is achieved by RNA polymerase, a single, very complex protein consisting of several functional subunits.

Promoter and terminator regions on the DNA enable RNA polymerase to detect the beginnings and ends of genes to transcribe. The canonical promoter region in *E. coli* is defined by a sequence pattern called *Pribnow box*, two chains of six nucleotides at positions $-35$, TTGACA and $-10$, TATAAT (Figure 1). The position indices refer to the first nucleotide transcribed from DNA to mRNA, which by definition is +1. As with most nucleotide sequences, these promoter sequences are not uniquely defined. When a promoter is defined as a combination of position and sequence, the redundancy of the start information is increased and the detection of the promoter region becomes more reliable, that is, becomes more fault-tolerant.

The encoding of the terminator region is accomplished by two schemes. Either the terminators are certain nucleotide sequences at the end of the gene (Downstream), causing RNA polymerase to dissociate from the DNA strand, or the termination of transcription is modulated by a protein factor known as Rho ($\rho$) factor (Figure 1). Basically, three types of regulation at the transcription level can be discerned, namely, repression of transcription (negative control), activation of transcription (positive control), and attenuation of transcription (complex translation–transcription interaction).

The gene structure of eukaryotes is similar to that of prokaryotes, but the location and structure of promoters and terminators are of greater variety and complexity. In terms of genetic code, eukaryotes differ from prokaryotes primarily in their introns, i.e., noncoding sequences that are found within and separate the parts of a single gene.

2.2 On the Role of Introns

Although to some extent introns can be also found in prokaryotes and viruses, they are typical for eukaryotes. This seeming waste of resources must have some benefits, assuming that evolution tends to favor processes consuming less energy. As intron processing and

---

2 We will also transfer this terminology to prGA chromosomes; it is the more correct biological term.

---

364 Evolutionary Computation Volume 6, Number 4
splicing of the exons requires additional energy and the permanent replication of "junk" (noncoding) DNA requires even more, there have to be advantages to compensate for these drawbacks. Not only do introns play a role in regulation of gene expression by containing promoter and enhancer sequences, but they also enable a variable combination of exons they separate, a process called exon shuffling (Gilbert, 1978, 1987; Doolittle, 1978). If the exons are thought of as gene modules encoding functional or structural subunits of proteins (this does not seem to be true for all eukaryotic genes), special modules could be used in a variety of genes and could be inserted by recombination processes (Figure 2). Alternative splicing, a process in which only subsets of exons are transcribed, is another advantage, since a single gene can encode a variety of proteins. One of the central questions in evolutionary genetics is whether prokaryotes lost introns (introns-early hypothesis), or eukaryotes acquired introns (introns-late hypothesis) in the course of evolutionary development of life (Cavalier-Smith, 1991).

The introns-early hypothesis assumes that introns could have been found in the very first gene structures and separated the code for structural and functional domains of proteins. Following these ideas, prokaryotes have optimized their genome by eliminating introns in order to replicate efficiently (high growth rates) for the price of losing flexibility. The main mechanism of intron evolution in eukaryotes is intron loss, which could be explained with retroposition, a process in which parts of spliced mRNA are re-transcribed into complementary DNA (cDNA) by reverse transcriptase (Fabry, 1995). Another convincing argument for the introns-early hypothesis is based on the likely theory that DNA is an evolutionary product of RNA (deDuve, 1994). Thus, in early cell structures the genetic information was encoded in rather short RNA strands. These old RNA forms seem to have had self-splicing capabilities (Sharp, 1987) in order to cut out introns, a process found with many modern
Table 1. Coding DNA content per haploid genome. Basic data is from Cavalier-Smith (1985), modified with data from Lodish et al. (1995).

<table>
<thead>
<tr>
<th>Species</th>
<th>Genome Size (bp)</th>
<th>Protein Coding DNA (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Escherichia coli</em></td>
<td>$4.5 \times 10^6$</td>
<td>$\sim 100$</td>
</tr>
<tr>
<td><em>Drosophila melanogaster</em></td>
<td>$1.5 \times 10^8$</td>
<td>33</td>
</tr>
<tr>
<td><em>Homo sapiens</em></td>
<td>$4.0 \times 10^9$</td>
<td>9 – 27</td>
</tr>
<tr>
<td><em>Prototopeterus aethiopicus</em></td>
<td>$1.42 \times 10^{11}$</td>
<td>0.4 – 1.2</td>
</tr>
<tr>
<td><em>Fritillaria assyriaca</em></td>
<td>$1.27 \times 10^{11}$</td>
<td>0.02</td>
</tr>
</tbody>
</table>

RNA types. Moreover, the complex protein factors involved in splicing would have had to be present before the insertion of introns (assuming introns-late), but it is extremely unlikely that this machinery evolved just by chance (deDuve, 1994).

One of the problems with the introns-early hypothesis is the position of introns in homologous eukaryotic genes. The small variance of intron locations is difficult to explain with loss of introns, as different species would have had to lose the same introns at identical positions, and this process is of extremely small probability. Similarly, simultaneous mutation of exon and intron bases could have shifted the intron positions (intron-sliding) by the observed magnitude but, again, the probability for these slidings to the correct positions tends toward zero (Rogers, 1989).

The intron-late hypothesis states that ancient gene structures contained no introns, and that these have been inserted in the course of evolution. Studies on a set of well-conserved genes of higher eukaryotes showed no correlation between intron positions and structural or functional properties of the encoded protein, but these studies could identify identical intron positions in homologous genes of related organisms. These results favor the idea of evolutionary insertion of introns, an idea that could be explained by reverse splicing of mRNA; in this process sequences are inserted and re-transcribed by reverse transcriptase (Fabry, 1995).

Just to give an idea of the quantitative proportions of coding to noncoding DNA, we present the percent of genome coding for proteins for some typical species in Table 1 (Smith, 1989).

As outlined above, the only prokaryote listed in Table 1 uses all of its DNA to encode proteins, whereas the eukaryotes have a surprisingly high content of noncoding DNA in their genome. Following all the debates of geneticists about the evolutionary advantage of noncoding DNA regions, we will investigate in detail in subsequent sections the usefulness.
of noncoding regions with artificial chromosomes for GAs. Evidently, when we are looking at the evolution of GAs, the intron-late hypothesis applies.

2.3 ptGA Chromosome Structures

We present the main genetic mechanisms modeled by ptGA, and clear up some issues concerning terminology. The biologically motivated extensions to conventional GAs are discussed briefly.

The main difference between ptGAs and standard GAs is the representation of parameters on the chromosome. Usually, the parameters are encoded at fixed locations on the chromosome, and the number and size are defined by the user. This is certainly sufficient for problems in which the number and resolution of parameters is known in advance. However, for a variety of real-world problems, this information is unknown. It might be argued that natural evolution faced the same problem, and thus it evolved a mechanism in which the number of genes could be adapted to specific environments. Hence, we transfer the concept of promoter/terminator sequences (regions) to GAs to allow for the evolution of number, size, and location of parameters. Throughout this article, we will usually use the abbreviation p/t-sequences.

Frequently, a specific bit of a chromosome is termed “gene” in GA literature. However, a wild type gene is a complex structure, and we think that the term “base” would be more appropriate, as it reflects the atomic unit of both the genetic and computer code.

On a ptGA chromosome a sequence of bases \( p \), of a specific length \( p_l \) is defined to be the \( p \)-sequence. Similarly, the \( t \)-sequence is specified by \( t_s \) and \( t_r \). The p/t-sequence bases are given in the following notation (e.g., \( p_i = \text{"010"} \)). The bases downstream from a promoter to the terminator represent a gene. The bases upstream from a promoter to the next terminator are a noncoding sequence. A gene is identical to its only exon, which is associated with one parameter; thus, we will use the terms “gene,” “exon,” and “(problem) parameter” alternatively. Evidently, the usage of the terms “introns” and “exons” for noncoding and coding sequences, respectively, does not conform to exact genetic definitions. However, by now a ptGA chromosome contains a single, unambiguous instance of noncoding segment, and we will use the terms “exon” to refer to coding regions and the term “intron” to refer to noncoding regions of a chromosome. We hope that geneticists will forgive this simplification.\(^3\) The base sequences of exons or introns will often be abbreviated as e/i-sequences.

In Figure 3 an example chromosome with exons and introns induced by p/t-sequences is depicted.

The only limit to the number of exons being possibly evolved by the use of p/t-sequences is the chromosome length \( l \). Obviously, nature is also evolving the length of chromosomes (genome sizes), and we could incorporate that into ptGA, too. However, in this work we restricted experiments to chromosomes of fixed length. The process of searching for p/t-sequences and location of exons is termed transcription. More ideas could be borrowed from nature by implementing a simple form of repression and activation of gene (exon) expression.

---

\(^3\) See also the differences between GA chromosomes and natural chromosomes.
(Section 2.1). For example, a single base upstream the promoter could indicate whether the exon is to be transcribed. This mechanism might create an even more flexible representation of possible solutions.

3. Related Work

In this section we briefly survey some ideas and suggestions of specific GA research dealing with parameter location on chromosomes, adapting gene size, or changing the number of parameters during the course of algorithmic evolution. These contributions can be seen as steps toward the use of noncoding base sequences (introns) in GAs, and we review results of the first researchers having proposed artificial chromosomes containing introns.

3.1 Pre-intron GAs

The adaptive representation genetic optimizer technique (ARGOT strategy; (Shaefer, 1987)) employs a form of Lamarckian evolution, as specific knowledge about the population is exploited to change the representation of parameters. In this scheme, “roving boundaries” are used to define the space (the number of bases) for a parameter on a chromosome. These roving boundaries are constrained by “brick wall” boundaries limiting the size of a parameter. One of the main ARGOT specific operators is the increase/decrease resolution. If the population has converged for a parameter, i.e., if the base values (alleles) are nearly identical across the population, the resolution of that parameter is increased by adding bases to its representation on the chromosome. Thus, ARGOT performs a finer search of the parameter space, coming closer and closer to the optimal (real) value. If the parameter exhibits a highly random representation, the resolution is decreased so as to locate regions of optimal parameter values. The roving boundaries can be interpreted as promoters and terminators defining the size of a gene and the resolution of a parameter value.

In Bui and Moon (1994) the notion of C-schemata was introduced. The authors observed that the position of defined bases in a schema of defining length have an impact on the destructiveness of k-point crossover (multipoint crossover). If the defined bases are clustered, the length of the sequence of don't-care symbols (nondefined bases) increases and makes it more likely that crossover sites fall in these regions without destroying the clustered bases. In Bui and Moon (1994) a schema preprocessing heuristic (graph reordering) was used to construct chromosomes with clustered schemata. The results for graph bisection problems have been superior to standard GA results. Again, this is a predecessor of GAs with intron base sequences, where the clustered bases can be loosely interpreted as exons and the bases between clusters as introns. However, there are no noncoding sequences used, but building blocks are clustered by heuristics prior to GA runs dividing the chromosome into sequences with higher and lower density of building blocks.

Messy GAs (Goldberg, Deb, Kargupta, & Harik, 1993) allow variable-length chromosomes and relax the fixed-locus assumption of conventional GAs. An illustrative messy chromosome of length $l = 3$ has the following structure: $(10)(21)(11)$, where the first number of each pair of values indicates the specific base and the second number its value. In this example, base 1 occurs twice with two different values; thus, the messy chromosome is overspecified. This conflict is resolved by a first-come-first-served rule scanning the genotype from left to right. Hence, in the above example, the value 0 would be assigned to base 1. A more severe problem arises with underspecification, especially for parameter optimization problems over some fixed number of parameters. This problem can be solved by using competitive templates representing a previously found solution. The unspecified bases are filled.
in by this template (Goldberg et al., 1993). The recombination operator with messy GAs is splice and cut. The cut operator surprisingly cuts a chromosome at a randomly chosen position. The splice operator concatenates two genotypes that do not necessarily have to be cut. At the beginning of messy GA runs, the splice operator is assigned a higher priority to increase chromosome lengths, whereas in later stages of evolution the cut operator becomes more likely. Messy GAs have been applied successfully to the evolution of fuzzy controllers (Hoffmann & Pfister, 1995). This GA variant allows for the evolution of both the number of parameters and their location, whereas the length of parameters on the chromosome is fixed. Messy chromosomes also contain a form of introns, as unexpressed overspecified genes can be viewed as noncoding regions.

3.2 First Intron GAs
Levenick (1991) reported on experiments with binary strings containing five genes with a length of six bases ($e_i = 6$), each being separated by introns of length seven ($i_l = 7$). He used a fitness function that gave increasing credit for discovery of single genes and specific combinations of them. A credit is assigned only if all the gene's bases have the correct value (Levenick, 1991). This procedure is very similar to the credit assignment for building blocks in the royal road functions (Mitchell, Forrest, & Holland, 1991). A comparison of GA runs with and without insertion of introns showed a nearly tenfold increase in successful evolution of optimal individuals. Levenick (1991) concluded that this improvement can be credited to the increased probability of a crossover site (he used one-point crossover) located in an intron region, where a gene cannot be broken by crossover.

The royal road functions (Mitchell et al., 1991) are specifically designed functions that should lay out a "royal road" for a GA, as the function values are hierarchically organized and lead the way to the optimum. This assumes the building block hypothesis (Holland, 1975; Goldberg, 1989), that new schemata are discovered via crossover that combines low-order schemata (building blocks) to higher-order schemata of increased fitness (composite solutions).

In her dissertation Wu (1996) centers on experiments with royal road functions with chromosomes containing noncoding regions (Wu, 1996). The use of introns at fixed locations resulted in only slight (or no) increase of GA performance on the royal road functions. She also investigated the use of start sequences (besides many other biologically motivated ideas such as multiple gene copies, reading frames, overlapping genes) in order to freely position the basic building blocks of the royal road function on the chromosome. This approach increased GA performance remarkably, and her thesis is, to our knowledge, the first work proposing what we call promoter sequences.

In genetic programming (GP) (Koza, 1992) "junk" code (or bloat) emerges intrinsically. A program statement like $a := a + 0$; can be destroyed by crossover without changing the fitness of a program. In Nordin, Francone, & Banzhof (1995), it is demonstrated that GP introns (bloat) play substantial roles in preserving building blocks and in reducing the disruptive effects of crossover.

4. Introns at Fixed Locations
So far, it has been implied that a main benefit of introns in GAs is the reduction of the disruptive effects of the crossover operator. If the selected crossover site(s) fall into an intron region, the exchange of genetic material cannot break up a schema in a gene, unless one gene coding for a single value is separated by intron regions. In the latter case the intron region
serves as a crossover hot spot, increasing the probability for crossover to occur in that region and splitting the schema preferably at this site. In order to analyze the effects of intron base insertion more formally, let us give some definitions.

The *exon base ratio* $e'$ and the *intron base ratio* $i'$ are given by

$$ e' = \frac{e}{I}, \quad i' = \frac{i}{I}, \quad e' + i' = 1 \tag{1} $$

where $e$ is the number of exon bases, $i$ the number of intron bases, and $I = e + i$ the chromosome length. It should be noted here that we will use $e$ and $i$ in a different context in following sections of this work, namely $e$ as the number of exons (encoded parameters) and $i$ as the number of introns (noncoding base sequences). We unify these definitions by thinking of the number of exon (intron) bases as exons (introns) of constant length $e_l = i_l = 1$. Hence, by using that definition in this section, the number of exons is equal to the number of exon bases.

### 4.1 Crossover Disruption Probabilities

Let us start with simple one-point crossover, in which the probability of destroying a schema of defining length $\delta$ is given by

$$ p_d^{(1,0)} = p_e \frac{\delta}{I - 1} \tag{2} $$

where $p_d^{(1,0)}$ stands for the disruption probability of one-point crossover without intron bases. This probability is only an approximation, as a broken schema can be repaired by genetic material from the other parent. (Bridges and Goldberg (1987) give a more detailed analysis of the crossover disruption probability.) A chromosome without introns has a length $I = e$; in other words, the genotype only contains exon bases. With the insertion of introns the length of the chromosome is increased to $I' = e + i$; thus, we get disruption probabilities without and with intron bases

$$ p_d^{(1,0)} = p_e \frac{\delta}{e - 1}, \quad p_d^{(1,0)} = p_e \frac{\delta}{e + i - 1} \tag{3} $$

and by building the ratio of disruption probabilities we get

$$ \frac{p_d^{(1,0)}}{p_d^{(1,0)}} = 1 + \frac{i}{e - 1} \tag{4} $$

and further transformation using $p^{(0)}$ yields

$$ \frac{p_d^{(1,0)}}{p_d^{(1,0)}} = 1 + \frac{i}{e' - \frac{1}{p^{(0)}}} \tag{5} $$

With $e' \gg 1/p^{(0)}$, which is a reasonable approximation for most chromosomes containing at least some exon bases, we arrive at a ratio of

$$ \frac{p_d^{(1,0)}}{p_d^{(1,0)}} = e' = 1 - i' \tag{6} $$

Equation 6 simply states that for a given defining length $\delta < p^{(0)}$ the disruption probability of one-point crossover applied to chromosomes with noncoding segments decreases.
linearly with the exon base ratio \( e' \), reducing the disruptive effects of crossover. With respect to mutation, the insertion of intron bases does not influence the expected number of mutated exon bases when the mutation operator is the standard bit-flip with fixed mutation rate \( p_m \). It should be noted that we were not concerned with the specific distribution of intron bases on the chromosome in the above equations. Obviously, a schema containing intron bases increases the defining length \( \delta \) and the probability of crossover within it. Thus, the building blocks should be tightly packed and secured by surrounding sequences of intron bases to minimize the probability of disruption. As a matter of fact, this is exactly how the gene structure of biological systems is organized, and we mimic this chromosome structure with ptGA.

Next, the disruption probability for chromosomes with fixed intron bases should be generalized to \( k \)-point crossover. For a schema to be preserved, all \( k \) crossover sites must not be in the base sequence defining the schema. Hence, for the disruption probability of \( k \)-point crossover we get

\[
p_d^{(k)} = 1 - (1 - p_d^{(1)})^k
\]  

and the ratio of the disruption probability for a chromosome with intron bases \( p_d^{(k,i)} \) to \( p_d^{(k,0)} \) without intron bases using Equation 6 becomes

\[
\frac{p_d^{(k,i)}}{p_d^{(k,0)}} = \frac{1 - (1 - e'p_d^{(1,i)})^k}{1 - (1 - p_d^{(1,0)})^k}
\]

Assuming \( p_d^{(1,i)} \ll 1 \) (implying \( \delta \ll \delta_0 \)), we may use the first-order Taylor series expansion \((1 - x)^n \approx 1 - nx\), which simplifies Equation 8 to

\[
\frac{p_d^{(k,i)}}{p_d^{(k,0)}} = e' = 1 - i'
\]

Hence, the disruption probability of building blocks is (approximately) \(^4\) linearly decreasing, with \( e' \) independent of the number of crossover sites \( k \), when we compared chromosomes with and without intron bases.

Nice as this may be, for most real-world problems the number, size, and location of building blocks is not known in advance; hence, the position of intron bases cannot be determined. Therefore, it would be desirable to additionally evolve location of intron and exon bases. In analogy to biological systems and their genetic code packed in DNA strands, we will induce this building block evolution by promoter/terminator sequences, defining begin and end of exons and interspersed introns. But before we analyze and discuss this more complex instance of the ptGA, we review an experiment from Levenick (1991).

4.2 ptGA Parameters
The following ptGA parameters remain unchanged for all experiments in this article:

- Population size: \( n = 100 \).
- Mutation rate: \( p_m = \frac{1}{2} \).
- Selection method: Binary tournament without replacement.

\(^4\) Numerical investigations of the function in Equation 8 exhibited a very good approximation by Equation 9 for a wide range of values.
Table 2. Fitness function values for various gene combinations for the L-bird problem. (A fitness of 1000 indicates the optimum.) From Levenick (1991).

<table>
<thead>
<tr>
<th>Fitness</th>
<th>Genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>5</td>
<td>A ∨ B ∨ C ∨ D ∨ E</td>
</tr>
<tr>
<td>50</td>
<td>D ∧ E</td>
</tr>
<tr>
<td>100</td>
<td>A ∧ B ∧ C</td>
</tr>
<tr>
<td>1000</td>
<td>A ∧ B ∧ C ∧ D ∧ E</td>
</tr>
</tbody>
</table>

The specific setting of the mutation rate is based on theoretical work (Bäck, 1992; Mühlenbein, 1992) giving strong evidence that this rate is an appropriate choice. Other parameters vary and will be given explicitly.

4.3 Evolution of L-birds

In Levenick (1991) the author experimented with a genotype containing five genes (exons) of length $e_1 = 6$ that were separated by a sequence of intron bases with length $i_1 = 7$. In order to illustrate this example genotype and to explain the specific fitness function used, Levenick (1991) chose the evolution of a bird with rather simplified gene structure. He writes:

Imagine the following: $A$ is a gene for elongated finger bones (advantageous for extracting food from crevices), $B$ is a gene for feathers (for warmth), $C$ is for skin flaps between fingers (again for warmth — widespread fingers with skin flaps could be used to better cover young), $D$ is for hind legs capable of a few good jumps (for catching prey), and $E$ is for clutching feet (for holding prey).

The existence of any of these genes is somewhat advantageous, the coexistence of genes $D$ and $E$ is more advantageous (can jump from branch to branch and hang on), genes for $A$, $B$, and $C$ together is even more advantageous (can wrap self or young in a feathered cover; can flap and so jump farther), and all five attributes together allow the creature to fly!

A gene is only said to be developed, if all the gene’s bases have the correct value (of 1). This procedure is very similar to the credit assignment for building blocks in the royal road functions (Mitchell et al., 1991; Wu, 1996). Comparing GA runs with and without insertion of introns showed a nearly tenfold increase in successful evolution of optimal individuals (Levenick, 1991). In the following we will refer to that problem as the $L$-bird (Levenick bird) problem. The specific fitness function is given in Table 2.

Each developed gene contributes to the L-bird fitness, and the nonlinear credit for coexistence of specific genes introduces epistasis to the problem. In Figure 4 the average number of generations $\bar{g}$ to evolve to a fully developed L-bird is shown over the intron base ratio $i'$. The introns are inserted between genes and each sequence of intron bases has the same length. For example, insertion of 10 intron bases in between the five genes (six bases each) gives $i' = 0.57 = (4 \times 10)/(4 \times 10 + 5 \times 6)$. The order of genes (gene structure) is $ABCDE$. The crossover rate is $p_c = 1.0$. The mutation of any (fixed) intron base has no influence on the fitness; hence the mutation rate is based on the number of exon bases (30),
Figure 4. Average number of generations $g^*$ to evolve to an L-bird with gene structure $ABCDE$ for various number of crossover sites $k = 1, 2, 4, 8$ over intron base ratio $i'$ (averaged on 50 runs, $i'$ recorded in steps of $\approx 0.1$).

yielding $p_m = (l/e) = 0.0333$. This result gives a constant mutation probability of an exon base and allows comparisons independent of intron base ratios.

As the gene structure seems to be amenable to the mechanism of one-point crossover (the two development stages $ABC$ and $DE$ can be directly combined by recombination at one crossover site), we also investigated 2-, 4-, and 8-point crossover for comparisons. Generally, all four crossover operators exhibit a decrease of average generations when the intron base ratio increases. Also, for constant intron base ratios the number of average generations decreases with the number of crossover sites. In the light of the previous analysis (Section 4.1) this behavior is reasonable, as a lower number of crossover sites increases the probability $p(k)$ that none of the sites fall into exon base regions ($p(k) = i'^k$). With high intron base ratios, the difference of $p(k)$ for variable $k$ becomes smaller (most crossover sites will be in intron base regions) as does the difference in average generations in Figure 4. Nevertheless, one-point crossover still improves performance at $i' > 0.8$. This improvement could be credited to the assumed synergy of this operator and the specific gene structure.

Following these arguments, it could be expected that an L-bird gene structure with nonadjacent genes of a specific development stage exhibits very similar behavior with respect to the number of crossover sites. In Figure 5 the previous experiment has been run with a different arrangement of genes ($ABDCE$).

Though cooperating genes are spread out on the chromosome, the average number of generations to reach the optimum is comparable to the gene structure $ABCDE$. Varying intron base ratios and different crossover operators have the same general influence on the optimum finding capabilities. The performance of the investigated number of crossover sites again converges at high intron base ratios including one-point crossover, as the modified gene structure does not favor this specific operator.

Based on these experiments it can be stated that the insertion of introns at predefined, fixed locations is able to accelerate evolution. However, considering more realistic problems, the insertion of noncoding sequences at predefined sites separating building blocks (exons) is hardly possible, as number, size, and location of building blocks are mostly not known in advance. For this reason we will make use of p/t-sequences for the evolution of L-birds in
5. Evolution of Introns

First, we introduce some more definitions. Let us begin with the marker length, $m_i$, given by

$$m_i = p_i + t_i$$  \hspace{1cm} \text{(10)}

with the p-sequence length $p_i$, and the t-sequence length $t_i$. The normalized chromosome length

$$\bar{l} = \frac{l}{m_i + c_i}$$  \hspace{1cm} \text{(11)}

should serve as an estimator for the maximum possible number of exons for a given chromosome length $l$, an average marker length $m_i$ with an average exon length $c_i$. As we will not be concerned with varying marker lengths on the same chromosome, we set $\bar{m}_i = m_i$. In the specific case of the L-bird problem the exons (genes) are of constant length. Hence, we may write $c_i = c$, and the normalized chromosome length becomes

$$\bar{l} = \frac{l}{m_i + c}$$  \hspace{1cm} \text{(12)}

Equation 12 implies that a p/t-sequence without any exon bases in between is also counted as an exon. For a problem in which the number of exons (parameters) is known in advance, the minimal length of a chromosome is given by

$$l_{\text{min}} = l(m_i + c_i)$$  \hspace{1cm} \text{(13)}

As the chromosomes should offer more space than the minimal length, we further introduce a space factor $s$

$$s = \frac{l}{l_{\text{min}}}$$  \hspace{1cm} \text{(14)}

Figure 5. Average number of generations $\bar{g}$ to evolve to an L-bird with gene structure $ADBEC$ for various numbers of crossover sites $k = 1, 2, 4, 8$ over intron base ratio $\bar{i}$ (averaged on 50 runs, $\bar{i}$ recorded in steps of $\approx 0.1$).
which simply gives the ratio of actual to minimal chromosome length. Intuitively, it should be easier for the ptGA to find the optimum if the chromosomes are longer than the minimal length required.

Looking back to Equation 1 for the intron and exon base ratios (where $i$ and $e$ are the number of intron/exon bases), and counting p/t-sequences as exon bases (as they are an essential part of an exon), we may write

$$ e + i = l \Rightarrow l_{\text{min}} + i = s l_{\text{min}} $$

(15)

assuming that $l_{\text{min}}$ is the space needed to encode the solution. Dividing Equation 15 by $l$ gives the intron base ratio

$$ i' = 1 - \frac{1}{s} \quad s \geq 1.0 $$

(16)

Again, it should be emphasized that these definitions can be made only when we know the optimum (optimal number of exons and their length) in advance. This is true only for the simple test problems used to analyze various effects of the ptGA chromosome structure but not for the more complex problems ptGA is laid out for.

### 5.1 Randomly Generated ptGA Chromosomes

Before we continue with our test problem, we present empirically found distributions of the number of exons and the exon length (e-sequence without p/t-sequences) in randomly generated start populations. Each base has been generated with equal probability $p(0) = p(1) = \frac{1}{2}$. We used the pseudo-random number generator (PRNG) implemented with SGA-C (Smith, Goldberg, & Earickson, 1991) employing a subtractive method taken from Knuth (1969). This PRNG displayed a departure from randomness for specific choices of p/t-sequences. Basically, the probability of two adjacent bases of different values was higher than that of identical values. However, other commonly used PRNGs exhibited a very similar tendency (Mayer, 1997).

Figure 6 shows exon (length) statistics extracted from randomly generated ptGA chromosomes with p/t-sequence lengths $p_1 = t_1 = 2$.

The maximum-likelihood estimators of this data for mean and standard deviation support evidence that the number of exons obeys a Gaussian distribution and the exon length an exponential distribution. The latter corresponds very well with a result in genetics (Gilbert & Glynias, 1993), where the authors state that the exon length is exponentially distributed when introns are inserted randomly into a gene. We did essentially the same, as the random generation of p/t-sequences induces a (random) separation of exons (coding sequences) and introns (noncoding sequences) on the chromosome.
Table 3. Gene tags specifying L-bird genes.

<table>
<thead>
<tr>
<th>Gene</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>001</td>
</tr>
<tr>
<td>B</td>
<td>010</td>
</tr>
<tr>
<td>C</td>
<td>011</td>
</tr>
<tr>
<td>D</td>
<td>100</td>
</tr>
<tr>
<td>E</td>
<td>101</td>
</tr>
</tbody>
</table>

Figure 7. Average number of generations $\bar{g}$ to evolve to an L-bird for various number of crossover sites over chromosome length, $p_e = 1.0$, $p_{in} = \frac{1}{2}$, $p_t = 0.0$, and $t_f = 0$ (averaged on 50 runs, space factor $s$ recorded in steps of 0.5).

5.2 Chromosomes with Fixed p-Sequences

We will now analyze the behavior of ptGA gene structures with p-sequences inducing evolution of noncoding sequences. As in the problem statement of L-bird evolution in Section 4.3, the fully developed L-bird genotype contains the five genes (exons) $ABCDE$. The tag of the exon has been determined by its location with the experiments in Section 4.3, and location is now subjected to evolution. The following experiments require an additional exon tag next to the exon’s p-sequence. This tag determines which of the five genes is expressed by a specific exon.

By now, we will focus on the mere use of p-sequences, as the exon length is constant with the L-bird problem (three tag bases plus six parameter bases). Table 3 gives the exon tags for the L-bird problem, thus specifying the five different genes.

With a p-sequence $p_e = 00$, a marker length $m_i = p_t = 2$, an exon length $e_t = 9$, and $l = 5$ L-bird genes, we get a minimal length of a chromosome $l_{min} = 55$. Figure 7 shows the average number of generations $\bar{g}$ to evolve a fully developed L-bird for various numbers of crossover sites, and space factors $s$ ranging from 1.5 to 6.0. The fitness function (Table 2) remains unchanged. Multiple occurrence of identical genes is possible, but the fitness is determined only by the (non)existence of a specific gene.

Consistently, the average number of generations $\bar{g}$ decreases with increasing space factor $s$, and at $s \geq 4.0$ (corresponding to an $r' \geq 0.75$, Equation 16) $\bar{g}$ is even less than with introns at
fixed locations (Figure 4). This behavior is remarkable, because the discussed gene structure requires 55 bases to evolve to the correct value, whereas it takes only 30 with the simpler structure. As with fixed intron bases, the weaker performance of crossover operators with increasing number of crossover sites seems to stem from the increased probability of crossover sites falling into p- or e-sequences, where the disruption probabilities are higher than in i-sequences. Again, this difference in performance is reduced with increasing space factor.

As mutation on its own is more aggressive than with conventional gene structures (mutation of a p-sequence base destroys the complete exon, and mutation of an i-sequence base might create a new exon), we introduce a crossover operator without disruptive effects by exploiting the knowledge on the position of e/i-sequences on a ptGA chromosome.

5.3 Spontaneous Crossover
Whereas conventional k-point crossover selects crossover sites randomly with equal probabilities for all sites on the chromosome, thereby possibly destroying p/t-sequences, spontaneous crossover selects only sites between intron bases or sites between a p/t-base and an intron base. Thus, the probability of an exon being split or even destroyed by crossover is reduced to exactly zero. The generation of additional exons in intron regions by this operator is dependent on the intron base values. Therefore, introns could also be evolved to base sequences activating or suppressing the generation of exons. For example, for a p-sequence \( p = 00 \) the creation of exons could be activated by zero-rich introns, whereas one-rich introns would suppress the formation of new exons (also depending on the t-sequence). As this crossover operator does not destroy building blocks, but has the ability to create additional exons, we call it \textit{conservative-generative}.

The main benefit of this scheme is the preservation of building blocks (exons). Once a building block has emerged, it can only be destroyed by mutation. Spontaneous crossover could propagate good building blocks in the population and combine various exons on chromosomes to a better solution. The specific operations associated with spontaneous crossover can be seen in Figure 8.

Technically, for each genotype a \textit{crossover mask} of equal length is stored, and the possible crossover sites are marked with 0, whereas sites in between exon bases or p/t-bases are marked with 1, indicating that the chromosome cannot be broken at these spots. Instead of externally defining a crossover rate \( p_c \), the occurrence of crossover is based on homologous introns in both parents. The detection of crossover regions is achieved by the logical OR of the crossover masks of the two parents. The resulting \textit{crossover template} is then scanned for strings of 0s representing possible crossover regions. In the following we will just consider the case of using the midpoint of each zero-string as a crossover site. Thus, the crossover
sites are defined as
\[ c_{\text{str},i} = \frac{b_{\text{start},i} - b_{\text{end},i}}{2} \]

where \( b_{\text{start},i} \) and \( b_{\text{end},i} \) are the first and last base positions of the \( i \)th zero-string on the chromosome. The crossover site \( c_{\text{str},i} \) is the "gap" upstream from the corresponding base position. Thus, \( c_{\text{str},i} = 2 \) denotes the position between the first and second base on the chromosome.

As this crossover procedure is solely governed by the internal chromosome structure and does not rely on a preset crossover rate, we have named it spontaneous crossover. The number of crossover sites is variable with spontaneous crossover and can be 0, i.e., no crossover, if the crossover template contains no single 0. It should be noted that this crossover type has (to our knowledge) no biological analog, and is mainly motivated by the properties discussed above, namely self-organization of position and number of crossover sites, and preservation of exons.

Figure 9 compares spontaneous crossover with the previously well-performing two-point crossover (Figure 7). For all space factors \( s \), the conservative-generative mechanism of spontaneous crossover outperforms two-point crossover on the L-bird problem. In order to obtain a more detailed picture of the internal evolutionary processes associated with spontaneous crossover, we look at the number of crossover sites emerging from the pGA gene structure using different space factors. Figure 10 depicts two typical runs searching and finding the optimal L-bird. The effective crossover rate (observed crossover events / possible crossover events) was \( p_c = 0.9924 \) and \( p_r = 1.0 \) for the left and right graph, respectively.

In the left graph (Figure 10) we can clearly observe self-organization of the gene structure, as the number of crossover sites levels off at four at \( g \approx 40 \). At this time nearly all chromosomes have the same basic structure of five genes (exons). However, the small space factor \( s = 1.5 \) does not allow for multiple copies of a specific gene on a chromosome.

With \( s = 3.0 \) (right graph of Figure 10) the picture changes. Here, the early discovery of the optimum (generation 12) leaves too little time for emergence of a common gene structure. This structure is indicated by the varying number of crossover sites (ranging from 3 to 9) with which the best individual per generation has been generated. However, the doubled
space increases the total number of developed L-bird genes in the population and allows for gene copies on a chromosome. Due to the relatively high number of crossover sites utilized by spontaneous crossover, the genes are rapidly spread in the population without destroying them. Basically, the synergy of these effects, namely, large space factor, multiple genes, and nondisruptive crossover at numerous sites, enable the good performance of spontaneous crossover.

5.4 Chromosomes with Fixed p/t-Sequences

We now deliberately increase the search space for the L-bird problem by the use of t-sequences and by increasing p/t-sequence lengths. We increase the search space in order to investigate ptGA performance toward more complex, realistic problems, where the size of parameters (genes) might vary and be evolved to (sub)optimal lengths, too. It should be stressed that this increase is not necessary with the L-bird problem, as we know the size of genes (exons) in advance (three tag bases plus six parameter bases), but meanwhile we pretend to forget.

Though it might seem to be a very simple task to assign tags to the various genes (Table 3), the interactions of tag sequences, t-sequences, and sequence lengths have to be observed carefully. For example, a tag of $tag = 000$ could be chosen for gene $A$, but this excludes all t-sequences contained in $tag$. With $ts = 00$, an exon with an exon tag $tag = 000$ could never evolve using this specific t-sequence. On the other hand, it would be suitable to employ a t-sequence with mostly 0s, as the parameter bases in the L-bird exons should evolve to 1s by definition. The tag sequences in Table 3 have been “optimized” for a p-sequence $pl = 111$ and a t-sequence $ts = 000$, resulting in a minimal chromosome length $l_{min} = 75$.

Evidently, for a more complex problem the p/t-sequences cannot be determined, as the optimal solution is unknown. That is, with a t-sequence being identical to an optimal exon base sequence, the optimum cannot be discovered. A possible solution to that problem is the evolution of p/t-sequences we proposed in Mayer (1997) where only the p/t-sequence length has to be set in advance, and the p/t-sequence bases are evolved by ptGA.

Figure 11 shows the average number of generations to find the optimum when employing both spontaneous and two-point crossover.

Again, the influence of the increasing space factor is the most notable characteristic exhibited by the ptGA runs with $pl = ts = 3$ ($g$ decreasing from $\approx 500 - 100$). However, spontaneous crossover improves only ptGA performance with $s < 2.5$. One reason might lie in the simple fitness function used. As only exons containing nine bases have a chance to evolve to the correct L-bird genes, all other exons not only do not need to be preserved, but should be split to make room for creation of exons of correct length. Intrinsically,
spontaneous crossover does not break exons, which in this case may reduce the benefits suggested above.

With a slightly changed fitness function, where exons of correct length (nine bases) receive a small credit regardless of their value, the difference between spontaneous and two-point crossover was much more pronounced (in favor of spontaneous crossover) (Mayer, 1997). Still, with the simple fitness function and a space factor $s = 6.0$, spontaneous crossover took 87 generations on average to find the optimum (Figure 11). With the corresponding $r' = 0.83$ (Equation 16) for a chromosome with introns at fixed locations it took 25 generations, because only 30 bases had to evolve to the correct value, but 75 when using $p/t$-sequences.

6. The Zero/One Multiple Knapsack Problem

After having analyzed the ptGA in the context of the simple L-bird problem, we next challenge the ptGA with an NP-complete problem. The 0/1 multiple knapsack problem is a combinatorial optimization problem that is also of practical relevance, as it can be used to model industrial opportunities like cutting stock, cargo loading, and the capital budget. We experiment with three problem instances of increasing complexity and compare ptGA performance to results employing conventional GAs (Khuri et al., 1994). The various instances of 0/1 multiple knapsack problems have been obtained from the OR-library\(^5\) (Beasley, 1990).

The 0/1 multiple knapsack problem is a variant of increased complexity of the 0/1 simple knapsack problem. For the latter, there are only two algorithms besides GAs delivering optimum solutions, i.e., dynamic programming and the branch-and-bound approaches. Only one of them (branch-and-bound) can be used to solve multiple knapsack problems (Khuri et al., 1994).

The less complex, but still NP-complete, 0/1 simple knapsack problem can be visualized by imagining a knapsack of capacity $C$ and $n$ candidate objects to be put into the knapsack.

---

\(^5\) [https://www.ms.ic.ac.uk/info.html](https://www.ms.ic.ac.uk/info.html)
Each object has a weight $w_i$ and a profit $p_i$. The goal is to fill the knapsack with objects that yield maximum profit under the constraint that the overall weight of the objects must not exceed the capacity of the knapsack. To put it more formally, we are searching a vector $\vec{x} = (x_1, x_2, \ldots, x_n)$ with $x_i \in \{0, 1\}$ that satisfies the constraint $\sum_{i=1}^{n} w_i x_i < C$, and maximizes the profit $P(\vec{x}) = \sum_{i=1}^{n} p_i x_i$. This definition leads to a straightforward encoding of the problem suitable for a conventional GA. The vector of objects is simply represented as a binary string, where the 1s and their positions on the chromosome correspond to objects in the knapsack, and the 0s denote all objects not to be packed into the knapsack.

The generalization of the 0/1 simple knapsack problem is the 0/1 multiple knapsack problem. We again have $n$ objects, but this time we imagine $m$ knapsacks. If we decide to put one object in a knapsack, it is virtually put in all $m$ knapsacks each of them having a specific capacity $c_1, c_2, \ldots, c_m$. An object's weight is dependent on the knapsack it is put in, thus, an object $i$ in knapsack $j$ has the weight $w_{ij}$. This models the consumption of resource $j$ by object $i$ within a certain budget for each resource (knapsack capacities). The profit should be maximized obeying the constraints of limited resources.

Khuri et al. (1994) used a fitness function with penalty terms (Richardson, Palmer, Liepins, & Hilliard, 1989) enabling survival of infeasible solutions in the population, especially when they overfill only a few of the $m$ knapsacks. The fitness function (Khuri et al., 1994) is given by

$$f(\vec{x}) = \sum_{i=1}^{n} p_i x_i - o p_{i,\text{max}}$$

where $p_{i,\text{max}}$ is the most expensive of the $n$ objects, and $o$ the number of overfilled knapsacks.

### 6.1 Interpretation of Exons

An evolved exon on the ptGA chromosome represents the binary-coded index of an object $i$. If an exon generates an index $i > n$ (the number of objects), this parameter is discarded (invalid object). If two or more exons (multiple exons) yield the same index, this index with the corresponding object is put only once into the $m$ knapsacks (according to the problem statement). All the valid objects encoded in one ptGA chromosome are then used to evaluate the fitness function.

As the maximum number of objects is known, we employ a p-sequence with an exon of constant length. By setting the exon length to the minimal value sufficient to encode all object indices $e_l = \lceil \log_{2} n \rceil$ ($n$ is the number of objects), we incorporate problem knowledge on the encoding level, as the search space is restricted to object indices $1 - 2^n$. Still, invalid objects can be generated (unless the number of objects is a power of 2), and the number of exons (objects) varies.

The problem under discussion is a good test for the ptGA, because the number of objects for the optimal solution is unknown, and the ptGA has to evolve the correct number of exons. Additionally, it has to discover valid objects that can be put into the knapsacks, in other words, the range of object indices. We will use the results of the conventional GA runs to determine the set of optimal object indices. From this we can calculate the minimal length $l_{\text{min}}$ (Equation 13), and use a space factor $s = 3.0$ for the experiments. Thus, problem knowledge is also incorporated via chromosome length, but the number of exons can still vary from 0 to the triple number of optimal object indices. It is essential to note that the search space that the conventional GA faces is smaller than that generated by the ptGA encoding, for the standard GA only operates with valid objects and does not have to bother with discovery of valid object indices and their number.
6.2 Experiments with Three Problem Instances

In the following we compare ptGA runs with the above encoding to the GA runs recorded in Khuri et al. (1994). We will label their results as KBH (for Khuri, Bäck, Heitkötter).

Khuri et al. (1994) used a conventional generational GA with population size \( n = 50 \), a mutation rate \( p_m = \frac{1}{n} \), crossover rate \( p_c = 0.6 \), proportional selection, and one-point crossover. For comparisons, ptGA is also run in conventional mode and labeled as (pt)GA using two-point crossover and binary tournament selection. The ptGA parameters are set to the default values used with most experiments in this article (\( p_c = 0.0 \), two-point \( p_c = 0.6 \) and spontaneous crossover).

The number of objects \( n \) is given by the specific problem instance label, the number of knapsacks is always \( m = 10 \), and the number of fitness function evaluations is identical to Khuri et al. (1994). The evolutionary search is performed 100 times. The results are presented in Table 4.

Generally, the results clearly demonstrate the capabilities of ptGA to solve difficult problems. However, the occasional large performance differences might be an indication of complex interactions of ptGA encoding and parameters. An example of this observation are the results of the Knap20 problem; here an exon length \( e_1 = 5 \) results in the possible expression of 12 invalid objects (exons). Although only four invalid objects can be generated with Knap28 by using the same exon length, the Knap20 results are significantly superior to conventional GA results, while the Knap28 results are not. After increasing the exon length to \( e_1 = 6 \) (36 invalid objects) with Knap28, the ptGA with spontaneous crossover found the optimum 14 times with a mean fitness \( \bar{f} = 12385.6 \) and a standard deviation \( \sigma = 10.67 \). Thus, despite enlarging the search space, in this specific case ptGA performance improved.

A possible explanation for this behavior is that, with increasing exon length, specific bases can be intrinsically used as a switch, i.e., as an activator or repressor of an exon. For instance, mutation of the most significant base in the above example \( e_1 = 6 \) primarily alters a valid to an invalid object or vice versa. This creates a form of memory and stores previously successful objects in the chromosome while evaluating other solutions. Further evidence along these lines is given by Mayer (1997) in the analysis of ptGA populations in which the optimum has been found. Often a great number of specific exons coding for an invalid object

<table>
<thead>
<tr>
<th>Problem</th>
<th>KBH</th>
<th>(pt)GA</th>
<th>ptGA(2-pt)</th>
<th>ptGA(SpontX)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knap15</td>
<td>83</td>
<td>93</td>
<td>91</td>
<td>95</td>
</tr>
<tr>
<td>( f_{opt} = 4015 )</td>
<td>( \bar{f} )</td>
<td>4012.7</td>
<td>4014.3</td>
<td>4014.1</td>
</tr>
<tr>
<td>5 ( \times ) 10^3 eval.</td>
<td>( \sigma )</td>
<td>-</td>
<td>2.56</td>
<td>2.88</td>
</tr>
<tr>
<td>Knap20</td>
<td>33</td>
<td>21</td>
<td>81</td>
<td>97</td>
</tr>
<tr>
<td>( f_{opt} = 6120 )</td>
<td>( \bar{f} )</td>
<td>6102.3</td>
<td>6099.5</td>
<td>6115.8</td>
</tr>
<tr>
<td>10^4 eval.</td>
<td>( \sigma )</td>
<td>-</td>
<td>20.07</td>
<td>10.75</td>
</tr>
<tr>
<td>Knap28</td>
<td>33</td>
<td>36</td>
<td>39</td>
<td>7</td>
</tr>
<tr>
<td>( f_{opt} = 12400 )</td>
<td>( \bar{f} )</td>
<td>12374.7</td>
<td>12382.9</td>
<td>12368.8</td>
</tr>
<tr>
<td>5 ( \times ) 10^4 eval.</td>
<td>( \sigma )</td>
<td>-</td>
<td>19.03</td>
<td>54.54</td>
</tr>
</tbody>
</table>

In the following we compare ptGA runs with the above encoding to the GA runs recorded in Khuri et al. (1994). We will label their results as KBH (for Khuri, Bäck, Heitkötter).
index can be observed. A single mutation at various loci of these exons—termed joker exons in Mayer (1997)—changes the nonvalid object to a valid object.

7. Summary

In this article we have reported on experiments with noncoding segments on artificial chromosomes for GAs. It has been formally shown that the insertion of "junk" code at fixed locations decreases the disruptive effects of conventional k-point crossover proportionally to the content of noncoding GA bases. Experiments on the simple, epistatic L-bird problem revealed that the insertion of noncoding segments at fixed locations improves GA performance as measured by the average number of generations to find an optimal solution. This is departing from previous work on nonepistatic problems (Wu, 1996; Wu & Lindsay, 1996), in which noncoding regions reduced the average number of generations only slightly (if at all). However, with evolution of the location of noncoding sequences, Wu (1996) reported a remarkable decrease in the number of generations used to find an optimum when compared with conventional GAs.

With the use of p/t-sequences defining beginning and end of a gene, respectively, the location, number, and length of coding and noncoding segments is subjected to evolution. Although the gene structure becomes more complex, ptGA performance is comparable to conventional encoding with problem parameters at fixed locations, and performance increases with ptGA chromosome length. Larger space on the chromosome allows for a greater number of genes (building blocks) and multiple copies of identical genes. In order to distribute building blocks efficiently, we devised a new crossover operator called spontaneous crossover. This is a conservative-generative operator that preserves existing building blocks, while possibly creating new ones. Crossover sites emerge due to size and location of introns on the chromosome.

However, the generally similar behavior (finding optima more rapidly with increasing ptGA chromosome length) of the (also) destructive k-point crossover and the non-destructive spontaneous crossover suggest that noncoding segments' main benefit is the locus-independent evolution of building blocks on separate (or even identical) ptGA chromosomes. Once a building block has evolved, it can be quickly distributed in the population, and the occurrence of identical building blocks at different locations (depending on chromosome space) ensures more likely survival of parts of the solution.

Another advantage of the proposed chromosome structure might be the self-organization of gene locations, possibly clustering cooperating genes and reducing epistasis. However, this effect remains to be studied more closely, and it should be noted that cooperating, natural genes in eukaryotes often are not even located on the same chromosome (Beebee & Burke, 1992).

The choice of p/t-sequences enabling evolution to an optimal solution will be extremely difficult to make for realistic problems. Specifically, the t-sequence heavily interacts with the exon bases representing the value of a problem parameter. Selecting a t-sequence that is part of an exon base sequence required for the (sub)optimal solution of a problem makes it impossible to find that optimum. Either we do not utilize a t-sequence at all by defining a p-sequence (causing fewer difficulties) and a constant length of exons, or we change the representation of exon values.

6 This also requires a specific encoding in which an exon is divided into subunits.
In Mayer (1997) we began to experiment with another idea borrowed from genetics, specifically, the translation of genetic code to proteins. The exon is split into tuplets of equal length (triplets in nature), each tuplet encoding a part of the exon value (amino acid in nature). Combination of tuplets allows for a very flexible and redundant encoding and significantly reduces the interactions of exon bases and t-sequence. A similar approach has been proposed in Yoshikawa, Furuhashi, & Uchikawa (1997).

Thinking of future applications, we expect ptGA to be a good choice when the number of parameters of a (sub)optimal solution is not known in advance. Problems in numerics (such as optimal number and distance of grid lines), statistics (such as optimal number and location of clusters), image processing (such as optimal number and location of segments), computer graphics (triangulation), or computer networks (flow control) are examples for the potential use of ptGA. Recently, the use of gene locators (p-sequences in our notion) with fixed-length bit strings (exon bases) encoding the parameter value have been used with IRR (implicit redundant representation) GAS (Raich & Ghaboussi, 1997). IRR GA outperformed conventional GAS remarkably when compared on a mechanical structure optimization problem. Some results of experiments on the 0/1 multiple knapsack problem (Section 6) confirm this observation.

Investigations of the applicability of these ideas, comparisons of ptGA performance with other algorithms on the class of problems indicated above, further variants of spontaneous crossover, and theoretical work on tuplet encoding will be among our future research activities.

Acknowledgments

I would like to thank Prof. Horst D. Clausen and Prof. Jochen Pfalzgraf for supporting this work and that of my "evolutionary" colleagues in a field new to the department. Special thanks to Hannes Flockner (Center for Applied Molecular Engineering, University of Salzburg) for long hours of assistance in molecular genetics, and Christian Blaschke (Department of Plant Physiology) for many discussions on evolutionary genetics. I thankfully appreciate the very constructive and detailed reviews of two anonymous referees, which helped to improve the article in many aspects.

References


ptGAs—Promoter/Terminator Sequences in GAs


