An Island Model for High-Dimensional Genomes using Phylogenetic Speciation and Species Barcoding

Paul Grouchy  
Institute for Aerospace Studies  
University of Toronto  
Toronto, Ontario, Canada  
paul.grouchy@utoronto.ca

Jekanthan Thangavelautham  
Institute for Aerospace Studies  
University of Toronto  
Toronto, Ontario, Canada  
thangav@ecf.utoronto.ca

Gabriele M.T. D’Eleuterio  
Institute for Aerospace Studies  
University of Toronto  
Toronto, Ontario, Canada  
gde@utias.utoronto.ca

ABSTRACT
A new speciation method for parallel evolutionary computation is presented, designed specifically to handle high-dimensional data. Taking inspiration from the natural sciences, the Phylogenetic Relations Island Speciation Model (PRISM) uses common ancestry and a novel species barcoding system to detect new species and move them to separate islands. Simulation experiments were performed on Multi-dimensional Knapsack Problems with different fitness landscapes requiring 100-dimensional genomes. PRISM’s performance with various parameter settings and on the various landscapes is analyzed and preliminary results show that PRISM can consistently produce optimal or near-optimal solutions, outperforming the standard Genetic Algorithm and Island Model in all the performed experiments.

Categories and Subject Descriptors
I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—heuristic methods; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—heuristic methods; G.1.6 [Numerical Analysis]: Optimization—global optimization

General Terms
Algorithms

Keywords
speciation, island model, species barcoding, phylogenetic species concept, parallel evolutionary algorithms, genetic algorithms

1. INTRODUCTION
Genetic Algorithms (GAs) have been enjoying much success in an ever-expanding range of applications, as the recent explosion in computational power has revealed just how powerful Holland’s original idea [11] can be. Recently, new innovations in processing power have been coming in the form of parallel computing where multiple CPU cores work together on a given task. A simple way for GAs to take advantage of this layout is to run the same GA with different initial conditions on different cores. It was found that better results could be achieved by using the Island Model (IM) [4], where migration occurs regularly between the various populations being evolved. However, Evolutionary Computation methods, including GAs and IMs, often prematurely converge to local optima [17]. This has inspired research into speciation (e.g. [8, 1]), which aims to maintain diversity by preserving multiple areas of exploration during a single run of a GA or IM.

As GAs become more popular, they are being applied to more complex real-world problems, such as Evolutionary Robotics. Such problems often require large genomes and have computationally expensive fitness function evaluations. Furthermore, owing to intricate fitness landscapes, solutions can be highly sensitive to initial conditions, necessitating many runs to produce acceptable results. This paper introduces the Phylogenetic Relations Island Speciation Model (PRISM), an Island Model with speciation based on ancestry, designed to be able to handle high-dimensional genomes and to utilize parallel architectures, while reducing the number of required function evaluations. Our focus here is not parallel implementation but the structure of the algorithm itself. The rest of the paper is organized as follows. Current speciation methods are discussed in Section 2; PRISM is presented in Section 3; the experiments and parameters used to test PRISM are listed in Section 4; the results of these experiments can be found in Section 5, and finally conclusions and future work are discussed in Section 6.

2. RELATED WORK
There are already several speciation methods designed for evolving subpopulations in parallel. Li et al. [12] developed the GA using Species Conservation (SCGA) which defines a set of species “seeds” in the current generation based on fitness and a user-defined species distance. In the next generation, the population is searched for members of the species defined by the seeds from the previous generation, again using the species distance. If a species is not represented in ancestry, designed to be able to handle high-dimensional genomes and to utilize parallel architectures, while reducing the number of required function evaluations. Our focus here is not parallel implementation but the structure of the algorithm itself. The rest of the paper is organized as follows. Current speciation methods are discussed in Section 2; PRISM is presented in Section 3; the experiments and parameters used to test PRISM are listed in Section 4; the results of these experiments can be found in Section 5, and finally conclusions and future work are discussed in Section 6.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO’09, July 8–12, 2009, Montréal Québec, Canada.
Copyright 2009 ACM 978-1-60558-325-9/09/07 ...$5.00.
belonging to any species might be overwritten by incoming species seeds being conserved. Another issue arises when attempting to use this method on high-dimensional data. This nearest neighbor clustering of species will become less relevant in higher-dimensional problems, because the distance to the nearest seed approaches the distance to the farthest seed as dimensionality is increased. The effects of this can occur for as few as 10 dimensions [2].

A method similar to the SCGA was introduced by Bessaou et al. [1], where the system alternates between “exploration” and “exploitation” at a given frequency. The system resembles a standard Island Model during the “exploitation” phase, with several subpopulations evolving on separate islands with periodic migrations between them. During the “exploration” step, all the subpopulations are merged and then speciated using classification tree speciation [14]. Each species is then assigned an island and the process repeats. This method will also be affected by dimensionality issues, as the classification tree partitions the population into subpopulations using a clustering method with randomly initialized centers. The results will be highly sensitive to these centers [5], thus rendering the species definition somewhat arbitrary.

The Speciating Island Model (SIM) of Gustafson and Burke [8] is a method that does not rely on clustering. Instead, it detects “outliers” defined by genetic dissimilarity with respect to the rest of the population and high relative fitness levels. These “outliers” are speciated by moving them to a new island, where they can evolve independent of the main population. At specified intervals, each individual is given a score based on its dissimilarity and fitness rank. A user-defined number of the top scoring individuals are then speciated. While the original algorithm was designed for solving the Tree-String Problem [9] using Genetic Programming, the applications to other types of Evolutionary Computation are obvious. However, this method does not incorporate any type of migration nor does it provide any guidelines for deciding how often to search for “outliers” and how many to speciate at a time. Again, the species definition is somewhat arbitrary, as the algorithm only checks for species at certain generations and will always detect the same number of species at these intervals regardless of the current genetic distribution within the population.

3. PRISM

3.1 Phylogenetic Speciation

The Phylogenetic Relations Island Speciation Model (PRISM) aims to detect speciation events within an evolving population and isolate the new species from one another by moving them to separate islands. The algorithm is looking for sympatric speciation [16] which occurs within a single reproducing population without reproductive barriers. To detect when one population has split into two different species, a definition of species is required. The Phylogenetic Species Concept [6], a species definition used in biology which defines species through common ancestry, was chosen for this purpose.

Species detection in PRISM, which is performed on every island after every generation, requires that a distance metric d be defined, which will differ depending on the type of genome used in the problem. In all the experiments reported in this paper, the Hamming distance was used as all genomes were binary. To perform species detection, the population is first bifurcated (see Algorithm 1 for pseudocode). To do this, we calculate all pairwise distances between individuals, and take the individual with the lowest sum of pairwise distances as center c1. Then we calculate μ1 as the mean pairwise distance from c1 to the other individuals in the population. Center c2 is then taken as the individual with the lowest sum of pairwise distances between individuals whose distance from c1 is greater than μ1. Finally, each individual is assigned to a center based on which one they are closest to. As the centers are chosen deterministically using the pairwise distances themselves, and are not dependent on user-defined parameters or fitness information, this approach will remain valid as dimensionality increases.

Algorithm 1 Population Partitioning Pseudocode

```plaintext
for i = 1 . . . popSize do
    d_i = \sum_{j} d(x_i, x_j)
end for

for i = 1 . . . popSize do
    if d(c_1, x_i) > \mu_1 then
        X_2 = X_2 \cup \{i\}
    end if
end for

for \forall i \in X_2 do
    d_i = \sum_{j \in X_2} d(x_i, x_j)
end for

for i = 1 . . . popSize do
    if d(c_1, x_i) < d(c_2, x_i) then
        C_1 = C_1 \cup \{x_i\}
    else
        C_2 = C_2 \cup \{x_i\}
    end if
end for

return C_1, C_2
```

Once the population has been divided in two, the set of common ancestors for each of the two subpopulations are found and compared (see Algorithm 2). This is done by finding the parents of the current individuals (a single parent if no crossover occurred during reproduction, two parents otherwise), then finding the parents of those parents, and so on until one or more individuals from a past generation are discovered that are the ancestors of all the individuals in the current subpopulation. This requires that the reproductive history be preserved, which is done by maintaining a 2D array indexed by generation and individual. Thus, in an array location for a given individual from a given generation, there are indices representing individuals in the previous generation who contributed genetic material to the individual in question. If there are no common ancestors for either of the two subpopulations, or if their common ancestors overlap, the two subpopulations are not considered separate species and the full population continues onto the next generation. If the two subpopulations have two distinct sets of common ancestors, then they are considered different species and are
evolved separately. This type of speciation seems to occur very frequently in practice and thus a requirement of dissimilarity (described below) is added to the species definition.

Algorithm 2 Common Ancestor Pseudocode

```plaintext
found = false
genFound = -1
A = ∅
let S_1 be the set of indices of given subpop
let S_0 be the set of unique indices of parents of subpop
let X be a numUniqueParents by subpopSize array
let X_{ij} = 0, ∀i, j
for i = 1 . . . numUniqueParents do
  for j = 1 . . . subpopSize do
    if S_0 is a parent of S_1 then
      X_{ij} = 1
    end if
  end for
end for

gen = currentGen - 1
while gen ≥ 0 AND not found do
  for i = 1 . . . numUniqueParents do
    if \( \sum_{j=1}^{subpopSize} X_{ij} = \text{subpopSize} \) then
      genFound = gen
      found = true
      A = A ∪ {i}
    end if
  end for
end if
if not found then
  S_1 = S_0
  prevNumParents = numUniqueParents
  let S_0 be the set of unique indices of parents of S_1
  numUniqueParents = size(S_0)
  Y = X
  let X_{ij} = 0, ∀i, j
  for i = 1 . . . numUniqueParents do
    for j = 1 . . . prevNumParents do
      if S_0 is a parent of S_1 then
        for k = 1 . . . subpopSize do
          if \( Y_{ik} = 1 \) then
            X_{ik} = 1
          end if
        end for
      end if
    end for
  end for
  gen = gen + 1
end if
return A, found, genFound
```

3.2 Species Barcoding

To avoid duplicate species and to provide a measure of dissimilarity for speciation detection, a novel species barcoding method was devised. Inspired by DNA barcoding used for species identification in the natural world [10], a Universal Species Code (USC) is generated for each species based on the genome sections that are identical between individuals within the species. With strong selection methods, a small high-fitness minority become the parents of a large majority of the offspring in the following generation. It is this minority that the USC generator compares, as they are the ones that have the most influence over the evolution of the species. The parameter \( \phi \) limits the number of top individuals used when producing a USC and should be selected based on selection strength and population size. For all experiments in this paper, \( \phi \) was selected by experimentally determining the average number of top individuals that produced 75% of the subsequent generation. Generating and comparing USCs for integer genomes are described below. USCs for other types of genomes are outside the scope of this paper, however the strategy described below can be extended to real-valued genomes.

3.2.1 USC for Integer Genomes

A USC for an integer genome has the same number of positions as the genome itself. To generate a USC in this case, we examine each position in each genome of the \( \phi \) individuals selected to represent the species. If at a given genome location all \( \phi \) individuals have the same value, that value is recorded in the same position in the USC. Otherwise, the USC should mark this lack of homogeneity in some fashion. See Table 1 for an example. When examining the similarity of two USCs, we simply observe how many values they have in common, ignoring all sections marked as not having a fixed value.

<table>
<thead>
<tr>
<th>Genome 1</th>
<th>Genome 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3 3 1 2</td>
<td>1 2 3 2 2</td>
</tr>
<tr>
<td>1 * 3 * 2 USC</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Simple USC example for an integer genome case where \( \phi = 2 \)

3.3 The Algorithm

Pseudocode for PRISM is presented in Algorithm 3. The algorithm begins by randomly initializing a user-defined \( I_{\text{init}} \) number of islands, each with the same population size. The populations are evolved in parallel, with each population being checked for speciation events after each generation. If a speciation event is detected, the USCs of the two subpopulations are calculated and compared. The USC is calculated using all the individuals of a subpopulation if it has fewer than \( \phi \) individuals, allowing for species of any size. Defining the length of a USC as the number of fixed-value positions it contains, the algorithm takes the USC with the smaller length and determines how many positions it has in common with the other USC. If they have more than \( \kappa \cdot I \) positions in common, where \( \kappa \in [0, 1] \) controls the definition of species and \( I \) is the length of the smaller USC, the two subpopulations are considered too similar and the entire population continues unaltered to the next generation. Otherwise, one subpopulation stays on the current island, while the other is moved to a new one. This allows both species to evolve separately. This process continues until a stopping condition is reached, usually a maximum number of generations.

3.3.1 Using \( \kappa \) to Control Speciation

As mentioned previously, dissimilarity checking is needed to control the speciation rate. By adjusting \( \kappa \), one can modify the definition of “species” used by the system. Small values of \( \kappa \) would require that subpopulation USCs differ greatly for a speciation event to occur; values closer to 1
Algorithm 3 PRISM Structure
\[
I = \emptyset \{ \text{This is the collection of islands} \}
\]
for \(i = 1 \ldots I_{\text{init}} \) do
\[
I = I \cup \{ \text{randomly initialized population} \}
\]
end for
while stopping condition not met do
\[
\text{for } \forall P_i \in I \text{ do } \{ \text{This loop can be run in parallel} \}
\]
selection, mutation and crossover on \(P_i\)
partition \(P_i\) into \(C_1 \& C_2\) \{see Alg. 1\}
let \(A_1\) be common ancestors for \(C_1\) \{see Alg. 2\}
let \(A_2\) be common ancestors for \(C_2\) \{see Alg. 2\}
if \(A_1 \cap A_2 = \emptyset \text{ AND } \text{found} = \text{true} \) then
let \(\text{USC}_1\) be USC for \(C_1\)
let \(\text{USC}_2\) be USC for \(C_2\)
let \(l\) be the size of the smaller USC
\[
s = 0
\]
for \(j = 1 \ldots \text{sizeGenome} \) do
\[
\text{if } \text{USC}_{1_j} \neq \text{USC}_{2_j} \text{ AND both defined} \text{ then }
\]
s = \(s + 1\)
end if
end for
if \(s < \kappa \cdot l\) then
\[
I = \cap \{P_i\}
I = I \cup \{C_1\}
I = I \cup \{C_2\}
\]
end if
end if
check for extinctions \{see Section 3.3.3\}
if \(\text{mod}([\text{generation}, \text{migration interval}]) = 0\) then
perform migrations \{see Section 3.3.2\}
end if
end while

lessen the required dissimilarity and a value of 1 ignores the dissimilarity checking. In PRISM however, \(\kappa\) is kept in the range \([0, 0.95]\). With \(\kappa > 0.95\), subpopulations can be practically identical and still be speciated. Not only will this incur unnecessary additional computational costs, it could also have a detrimental effect on the final results by allowing species that have already converged to continue speciating, producing similar high-fitness species which can force branches that have not yet converged to extinction (see Section 3.3.3).

When using a relatively small \(\kappa\) for the duration of a run, a few speciation events will occur in the first generations. As soon as the islands begin converging, however, the amount of difference seen in the subpopulations that arise diminishes. Thus the number of speciation events will trail off to zero long before the algorithm has fully converged, with the rest of the run being identical to a standard IM. Conversely, with large \(\kappa\), there will be many speciation events at the beginning of the run, with this number eventually starting to decrease as the algorithm approaches convergence. The problem with this scenario is that one or two dominant phylogenetic branches can flood the environment with species, quickly forcing less phylogenetic branches to extinction. This will rapidly reduce the diversity of the system, often causing premature convergence.

To address these issues, each island has its own adaptive \(\kappa\). The initial islands all start with \(\kappa_{\text{init}}\) and new species inherit the \(\kappa\) of the species from which they branched. With a small \(\kappa_{\text{init}}\), only major speciation events occur initially. If a specific island has not had a speciation event in 25 generations, its \(\kappa\) is increased by \(\beta_1\). Conversely, if an island has a speciation event and one of its new species has a speciation event in less than 10 subsequent generations, the \(\kappa\) values of the two species created by the second event are decreased by \(\beta_2\). Thus, phylogenetic branches will continue to speciate as they converge while those branches with high speciation rates will be kept from flooding the environment.

3.3.2 Migration
Migration occurs after certain generations, specified by the migration interval. During a migration event, each island has a given number of its individuals, excluding the top individual, randomly chosen to be replaced by randomly chosen individuals from a randomly chosen island, again excluding the top individual. The number of migrants is specified by the migration size. Only fully populated islands (i.e. those who did not have a speciation event in the previous generation) participate in the migration event.

3.3.3 Extinction
There is an extinction phase after every generation in PRISM, and in this phase a species can go extinct in one of two ways. To prevent single species from dominating the system, if two species are found to have the same USC, the one with the lower best fitness goes extinct. If they have the same best fitness, one is chosen at random for extinction.

The second type of extinction is purely based on fitness. The system allows a user-defined maximum of \(I_{\text{max}}\) species at any one time. If this value is exceeded, the species with the least fit top individual will go extinct, bringing the total number of species back down to \(I_{\text{max}}\). The number of species in the system can remain above this value, however, as new species are protected from fitness-based extinction for \(\tau\) generations to allow for initial development. Thus, if the least fit species are being protected when \(I_{\text{max}}\) is exceeded, there will be no extinctions until their protection has expired. If a species currently under protection has a speciation event, the new species do not get a full \(\tau\) generations of protection. Instead, they are covered by the remainder of the previous protection phase. Otherwise, low-fitness species could continue to survive through speciation. All experiments reported in this paper used \(\tau = 5\).

4. EXPERIMENTS
Experiments were run to compare the Standard Genetic Algorithm (SGA), the Island Model (IM), and the Phylogenetic Relations Island Speciation Model (PRISM) with various \(I_{\text{init}}\) settings.

4.1 Multidimensional Knapsack Problem
The Multidimensional Knapsack Problem (MKP) is a NP-hard problem with many real-world applications. We are given \(n\) objects, each with a value \(p_j\) and \(m\) resource usage parameters \(r_{ij}\). The goal is to maximize the value of the objects inside the knapsack without violating any of the resource constraints. The problem can be formally described as follows: Maximize

\[
f = \sum_{j=1}^{n} p_j x_j
\]
subject to
\[ \sum_{j=1}^{n} r_{ij} x_j \leq b_i, \quad i = 1, \ldots, m \] (2)

where \( x_j \in \{0, 1\}, \quad j = 1, \ldots, n \), indicating whether the \( j \)th object is to be placed in the knapsack (1) or not (0). The \( b_i \) values are called the knapsack constraints. The three problems used here were “5.100-00”, “5.100-10” and “5.100-20” from the OR-Library (http://people.brunel.ac.uk/~mastjjb/jeb/info.html), each having \( m = 5 \) knapsack constraints and providing all the \( r_{ij} \) and \( p_j \) values, as well as the best known solutions. The knapsack constraints were determined using
\[ b_i = \alpha \sum_{j=1}^{n} r_{ij}, \quad i = 1, \ldots, m \] (3)

where \( \alpha \) is known as the tightness ratio [7]. The three problems chosen had tightness ratios of \( \alpha = 0.25 \), \( \alpha = 0.50 \) and \( \alpha = 0.75 \), respectively. To solve these problems with a GA, binary genomes of length \( n \) are used, with each location \( j \) on the genome representing \( x_j \). In all three problems, \( n = 100 \). A fixed exterior penalty function [13] was used to penalize solutions that violated the knapsack constraints (i.e., infeasible solutions). Therefore the fitness function used is
\[ f(x) = \sum_{j=1}^{n} p_j x_j - 10 \left( \sum_{i=1}^{\min(5, n)} \max(0, \sum_{j=1}^{n} r_{ij} x_j - b_i) \right)^2 \] (4)

4.2 Experiment Parameters

Table 2 contains the parameter settings used. All runs were terminated after 600 generations and used an elitist policy where the top individual from the previous generation replaces the worst individual in the current generation if no better solutions were discovered. The distance metric used in PRISM was the Hamming distance. Migration parameters only concern IM and PRISM and were selected based on the results in [15]. The migration style used in the IM tests was the same as for PRISM (see Section 3.3.2). The \( \phi \), \( \kappa_{init} \) and \( I_{init} \) (initial number of islands) parameters apply only to PRISM. For IM and PRISM, population size refers to the population of each island. All experiments were run with two different \( I_{max} \) settings, 50 and 75. The number of islands in the IM runs were also defined by \( I_{max} \). The \( \beta_1 \) and \( \beta_2 \) values were chosen experimentally. Each SGA experiment was actually \( I_{max} \) parallel runs (i.e., IM with no migration) and used the same initial populations as the IM experiments, allowing for better comparisons. Other SGA experiments were considered, such as using one massive population or a smaller population size with more generations of evolution; however, preliminary tests suggested poorer performance than the SGA results reported here.

5. RESULTS

The results for all experiments are presented in Table 3. It is generally accepted that the smaller the \( \alpha \) value, the harder the MKP [7], and all performed experiments seem to confirm this. While PRISM generally has a significantly higher number of successful runs than the IM and SGA, it was not able to match Chu and Beasley’s algorithm [3], which achieved 100% success rates on the three problems used in this paper. However, every effort was made to keep PRISM as generic as possible. No MKP-specific heuristics were used and the fitness landscape was kept static by using a fixed penalty function. In contrast, Chu and Beasley’s method utilizes problem-specific heuristics to generate initial populations containing only feasible solutions, as well as to convert infeasible individuals to feasible ones between generations. This limits the applicability of their algorithm to MKPs only, whereas PRISM can be applied to a wide variety of problems although it too can easily be modified to use problem-specific heuristics for improved performance on a given task.

5.1 Number of Runs to Find Global Optimum

The main metric used to compare the various methods and parameter settings was the expected number of 600-generation runs required to find the global optimum for the first time. What is apparent from this metric is that PRISM can deliver significant performance improvements in all the scenarios tested. Compared to the IM, PRISM is able to roughly halve the expected number of required runs to find the global optimum, usually reducing this value by a factor of 3 or more. These reductions are especially important when the number of fitness function evaluations per run in these systems is considered. The number of function evaluations per generation required by PRISM, the IM and the SGA in all experiments were on the order of \( 10^7 \). Therefore, because in all experiments PRISM has been capable of reducing the expected number of required generations by a minimum of 1000, at the very least PRISM will provide savings on the order of \( 10^7 \) in the expected number of required function evaluations. In practice, these savings are usually much higher. One could argue that with massively parallel systems, these savings are not important. However, with only a limited number of processors available and a computationally expensive fitness function, these improvements will undoubtedly translate to significantly better performance.

5.2 Mean Best Solution

Strong performance improvements were also observed on a run-by-run basis as PRISM produced higher mean solutions than the other methods, usually accompanied by a lower standard deviation. By examining \( \%_{gap} \), which shows how close the mean solution is to the global optimum, it can
be observed that PRISM always has a mean best solution that is closer to the global optimum than those of the other methods, no matter what the $I_{init}$ setting. Furthermore, the difference between the optimum and PRISM’s mean solution is often less than half the difference between the optimum and the IM’s mean solution. Finally, PRISM usually demonstrates a lower standard deviation than the other methods, even in those scenarios that achieved the highest mean solutions. Thus, PRISM has demonstrated better and more consistent performance per run than the IM and SGA. This suggests that even with only a single run of PRISM, one can be confident in the quality of the results, whereas traditionally it requires several runs of the IM or the SGA to achieve this level of confidence.

Increasing $I_{max}$ from 50 to 75 improves mean solutions for all methods while often improving the standard deviation significantly as well. The only exception occurs for PRISM with $I_{init} = 5$ in the $\alpha = 0.75$ case. Adding species to the system could potentially reduce the benefits of migration by allowing lesser-fit species to avoid extinction. Migrants from these species would not be able to reproduce in higher-fitness populations, thus wasting a certain number of migration events. Experiments with more complex migration schemes are necessary to confirm this theory.

### 5.3 Improving PRISM

Intuitively, the first few generations of evolving solutions to the MKP will be different depending on the tightness of the constraints. When $\alpha = 0.25$, many of the initial individuals in a population will severely violate the knapsack constraints and thus have very low fitness. Therefore, under strong selection, only a few of the initial individuals will end up reproducing. Considering that they were randomly initialized, these reproducing individuals will most
likely have very different genomes, and thus each one will become the ancestor of a different subpopulation. However, as $\alpha$ is increased, one would expect more reproducing individuals in the initial generation and accordingly more distinct subpopulations in subsequent generations. One would expect PRISM to have the least initial speciation events for $\alpha = 0.25$ and that this value should increase as $\alpha$ is increased to some maximum amount dictated by factors such as the population size and adaptive $\kappa$ settings. The number of speciation events per species for a typical run of PRISM is shown in Figure 1(a). When discussing initial speciation events, it is the spike seen within the first 50 generations that is under consideration. The “Fixed Tournament Size” data in Figure 1(b) suggests that PRISM is not behaving as expected, because the peak number of splits detected decreases when $\alpha$ goes from 50 to 75. This can be attributed to the fact that PRISM is unable to detect speciation events during the first few generations of a run as the initial generation has no ancestors. An ancestral history must be built up before common ancestors of subpopulations can be found, an essential part of species detection in PRISM. During these initial generations, some of the weaker initial subpopulations may disappear due to strong selection.

To test this conjecture, as well as to improve the performance of PRISM, a set of experiments was performed where the tournament size was initially 2 and was increased by 2 every 5 generations until it reached 12. This should preserve the initial subpopulations while building the ancestral history. The results of these experiments are presented in Table 4. Not only does PRISM detect more initial speciation events in all cases but the number of speciation events per species follows a more expected trend with respect to $\alpha$ (see Figure 1(b), “Initially Increasing Tournament Size”). The relative equality of the peak number of speciations for $\alpha = 0.75$ and $\alpha = 0.50$ suggests that both cases have reached the maximum peak that the parameter settings will allow. The experiments show significant performance improvements in the $\alpha = 0.25$ case, minor improvements in the $\alpha = 0.50$ case, and no improvement in the $\alpha = 0.75$ case. These results can be explained by the fact that with tight knapsack constraints, different subpopulations may be genetically very dissimilar, whereas with looser constraints, different subpopulations will have more in common with one another. Therefore, detecting initial subpopulations is very important when the constraints are tight, but much less so when the constraints are loose as PRISM is able to rediscover lost subpopulations through speciation if subpopulations have a certain amount of genetic similarity.

6. CONCLUSIONS AND FUTURE WORK

When dealing with complex, high-dimensional problems, parallel processor systems can significantly reduce the time needed to evolve solutions in Evolutionary Computation. These parallel systems can also produce superior results, as demonstrated by the Island Model [4]. Speciation was incorporated into these systems in an effort to evolve multiple solutions in parallel. Previous speciation methods were not designed for high-dimensional genomes or had user-defined species definitions. The Phylogenetic Relations Island Speciation Model performs well in high dimensions and uses a species definition taken directly from the natural sciences.

Experiments to date have shown that PRISM has the ability to produce consistent, optimal or near-optimal solutions over several different fitness landscape scenarios while providing significant computational savings compared to the SGA and IM methods. While PRISM does require additional computational overhead, this will be overshadowed by the savings provided by the reduced number of fitness evaluations when working with computationally expensive fitness functions. Thus, by evolving different species, or groups of species, in parallel on separate CPU cores, with migration between them allowing solutions to benefit from each other’s innovations, PRISM will very likely discover better solutions to complex problems than SGAs or IMs.

PRISM does introduce several new tunable parameters, and experiments are required to determine heuristics for choosing $I_{\text{init}}$, $I_{\text{max}}$, $\kappa_{\text{init}}$, and $\phi$. While a detailed sensitivity analysis on these new parameters has not yet been done, we do note that PRISM performed well on three versions of the Multidimensional Knapsack Problem that exhibited rather different fitness landscapes using a constant parameter configuration.

<table>
<thead>
<tr>
<th>Method</th>
<th>$E$</th>
<th>$N_{\text{opt}}$</th>
<th>$%_{\text{gap}}$</th>
<th>$\mu_{\text{best}}$</th>
<th>$\sigma_{\text{best}}$</th>
<th>$\mu_{\text{gen}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRISM - $I_{\text{init}} = 2$</td>
<td>5.0</td>
<td>6</td>
<td>0.21</td>
<td>24329.9</td>
<td>31.4</td>
<td>387.4</td>
</tr>
<tr>
<td>IM</td>
<td>15.0</td>
<td>2</td>
<td>0.32</td>
<td>24304.0</td>
<td>33.3</td>
<td>350.3</td>
</tr>
</tbody>
</table>

(b) Results for Problem “5.100-10,” $\alpha = 0.50$, Best Known Solution 42757

<table>
<thead>
<tr>
<th>Method</th>
<th>$E$</th>
<th>$N_{\text{opt}}$</th>
<th>$%_{\text{gap}}$</th>
<th>$\mu_{\text{best}}$</th>
<th>$\sigma_{\text{best}}$</th>
<th>$\mu_{\text{gen}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRISM - $I_{\text{init}} = 2$</td>
<td>2.5</td>
<td>12</td>
<td>0.09</td>
<td>42717.8</td>
<td>38.4</td>
<td>383.5</td>
</tr>
<tr>
<td>IM</td>
<td>15.0</td>
<td>2</td>
<td>0.24</td>
<td>42634.0</td>
<td>52.7</td>
<td>299.4</td>
</tr>
</tbody>
</table>

(c) Results for Problem “5.100-20,” $\alpha = 0.75$, Best Known Solution 59822

<table>
<thead>
<tr>
<th>Method</th>
<th>$E$</th>
<th>$N_{\text{opt}}$</th>
<th>$%_{\text{gap}}$</th>
<th>$\mu_{\text{best}}$</th>
<th>$\sigma_{\text{best}}$</th>
<th>$\mu_{\text{gen}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRISM - $I_{\text{init}} = 2$</td>
<td>3.8</td>
<td>8</td>
<td>0.03</td>
<td>59802.7</td>
<td>14.1</td>
<td>343.3</td>
</tr>
<tr>
<td>IM</td>
<td>3.8</td>
<td>8</td>
<td>0.07</td>
<td>59779.4</td>
<td>42.7</td>
<td>263.0</td>
</tr>
</tbody>
</table>
Figure 1: Data averaged over all runs from PRISM with $I_{\text{init}} = 2$ and $I_{\text{max}} = 75$

(a) Average number of speciation events per species per 10 generations, fixed tournament size, $\alpha = 0.50$

(b) Peak average number of speciation events per species per 10 generations for both the fixed tournament size experiment and the experiment where the tournament size was initially 2 and increased by 2 every 5 generations until the normal tournament size of 12 was reached

7. ACKNOWLEDGMENTS

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada for supporting this research and the reviewers for their helpful comments.

8. REFERENCES


