Searching for Better Performance on the King-Rook-King Chess Endgame Problem

Wayne Iba
Department of Mathematics and Computer Science
Westmont College
955 La Paz Road
Santa Barbara, CA 93108
IBA@WESTMONT.EDU

Abstract

For many classification problems, genetic algorithms prove to be effective without extensive domain engineering. However, the chess King-Rook-King endgame problem appears to be an exception. We explore whether modifications to a baseline parallel genetic algorithm can improve the accuracy on this particular problem. After describing the problem domain and our implementation of a parallel genetic algorithm, we present an empirical evaluation of several approaches intended to improve overall performance. Our results confirm the challenging nature of this domain. We describe several directions that may yet deliver significant improvements.

Introduction

Genetic Algorithms (GA) manage to learn highly accurate models in a surprisingly wide variety of domains. In cases where predictive accuracy initially suffers under a GA, a moderate level of parameter tuning and/or representation engineering will typically yield satisfactory results. We have encountered a domain, the King-Rook-King chess endgame problem (KRK), that appears to be highly resistant to solution by a GA. In this paper, we describe our experience with this problem and our attempts to solve it using a parallel GA. Although the immediate goal is to improve the accuracy of a parallel GA on one particular problem, we hope that in the process we can gain some insight into what makes this problem challenging. Such insight could then guide the selection of learning methods for other domains.

Related Work

First described by Clarke (1977), Bain and Hoff generated the KRK database and made it available through the UCI Machine Learning Repository (Asuncion & Newman, 2007). Bain and colleagues used the database in their studies of inductive logic programming (ILP). An ILP model forms logic programs that describe classes of instances; the logic may consist of Boolean operators and relations, arithmetic operators and relations, and operators and relations defined by the program itself. Bain and Srinivasan (1994) point to the utility of the KRK domain as a benchmark domain. Subsequently, others applied a genetic programming (GP) approach that successfully evolved strategies to play winning endgames for these problems (Lassabe et al., 2006).

Reportedly, the ILP and GP approaches worked reasonably well at the level of selecting moves to complete an endgame. However, Lassabe et al. (2006) do not report classification results and Bain & Srinivasan (1994) report an accuracy of 16% over the entire dataset. In light of the results we present below, this confirms a consistent pattern throughout machine learning where attribute-value representations match the performance of relational methods. However, this matching of performance relies on an appropriate domain theory or on an expert's representation engineering. On the other hand, GA methods are well suited to problems where a domain theory does not exist. Thus, it seems reasonable to expect success on the KRK problem based on a combination of a GA and a moderate level of representation engineering.

In the present work, we take direction from previous studies of parallel genetic algorithms. Cantu-Paz (2001) and Belding (1995) both performed extensive parametric studies of different strategies for selecting individuals to migrate between islands and different frequencies for such migration. Based on what their results established, we hold migration strategy and frequency constant. However, questions remained regarding island connectivity; Belding reports no differences between two particular topologies without addressing this factor's significance, or lack thereof. Thus, we vary the island connectivity in our experiments below.

The King-Rook-King Endgame Problem

When a game of chess gets down to three pieces, a king and a rook for one player and just the king for the other, the endgame problem of checkmating the isolated king is known as the King-Rook-King problem. Because such endgames present significant challenges for humans and computer programs alike, we would like to apply machine learning methods in order to determine which moves to play in a given situation. Providing the basis for such an endeavor, the King-Rook-King dataset consists of 28,056 board positions with their respective classification. A board position is given as six features: rank and file for each of the three pieces. The classification, the attribute of ultimate interest, is given either as the minimum number of moves for White to win (0 to 16 moves), or alternately as "draw." The board positions are not uniformly distributed across these 18 classes. The

distribution ranges from 27 board positions for the smallest class up to 4,553 for the most frequent class. This suggests a baseline accuracy of 16.2% by guessing the most frequent class[1]. However, our implementation does not take advantage of this fact and starts with a random accuracy on the order of 5.5%.

One of the strengths of ILP systems is their ability to effectively create features that are derived from the raw attribute values. This is also a reason that such methods have been successfully applied to the KRK domain. Assuming that some feature engineering would be necessary, we added six additional attributes representing the absolute distance between each pair of pieces in terms of rank and file (Iba, Marshman & Fisk, 2008). Note, we represent the distance between two pieces on the same rank or file as a 1 so that our rule-matching scheme can identify such conditions. Thus, these values range from 1 to 8 for differences of 0 to 7. Below, we describe additional features we added in search of improved performance in this problem domain.

Baseline Implementation

For this study, we use an extension of the island, or coarse-grained, GA model for parallel genetic algorithms (Iba et al., 2008). In this standard approach, we partition the population of individuals over a collection of semi-isolated islands, which map to separate processors. On each island, the local population evolves according to the traditional GA. Periodically, a relatively small number of selected individuals migrate between islands, thereby infusing their genetic material into another island's population. In this manner, we obtain much of the benefit of a larger population at a fraction of the cost; we generally observe savings proportional to the number of islands, which are simulated on separate processors. In a previous study, we varied migrant selection strategy and migration topology (Iba et al., 2008). Finding no significant differences between migrant selection strategies, here we consistently use the fitness-proportionate migration strategy (Cantu-Paz, 2001). In our current search for improved accuracy, we do vary the island connectivity and several other system parameters described in the next section. However, we are not focused on the parallel aspects of the implementation. Rather, we rely on parallelism in order to reduce the overall run-time of our experiments.

An individual's genome consists of 18 groups of multiple patterns, one for each possible classification. Each of the patterns within a group matches against one corresponding feature of a board position. The group having the most pattern matches determines the prediction made for a given endgame position. We compute an individual's fitness as the fraction of board positions in the training set that it correctly classifies. Given that we represent the features of endgame instances by setting the bit to 1 in the corresponding position of each feature, we treat the patterns in an individual's genome as bit-masks. A positive match for a feature is said to occur when an individual's feature pattern includes a 1-bit in the position of the instance's feature. For example, if an individual has a pattern [01101100] for the white king's rank, and an endgame problem has the white king in the third rank – represented as [00000100] – we find a match when we mask these together. These bit-mask patterns (as well as our representation of the attributes of an instance) require one byte each. In the baseline implementation, an individual's genetic code consists of 1,728 bits; that is, a byte for each of the 12 features multiplied by 18 classes.

Our experience with this system on the KRK problem motivated us to explore ways to improve the overall accuracy. In the next section we describe the steps we implemented toward this end and in the following section we present our results from those changes.

Attempts to Improve Performance

Our efforts to improve the performance of the system on the KRK problem involved two categories of issues. The first addressed a more comprehensive evaluation of the full range of system parameters. The second explored additional features with which to represent instances.

System parameters

We evaluated a wide range of values for the mutation parameter. These tests did not reveal any significant differences or improvements. Therefore, we do not show results for these variations. Subsequent tests hold the mutation parameter constant at 0.01 (i.e., approximately one individual out of one-hundred will experience a single point-mutation).

After our previous study, we implemented additional crossover reproduction settings. Based on the observed improvement from one to two crossover points and realizing the genetic code of each individual consists of well over 1,500 bits, we implemented a variable crossover strategy to increase genetic mixing. Two individuals that are selected to reproduce will have anywhere from one crossover point up to a maximum value, which we vary between 5 and 40. We select the actual number of crossover points for a particular pairing of two individuals from a uniform distribution between 1 and the maximum number.

We also considered the role of migration strategy and island connectivity in our previous study. Here, we do not vary the method for selecting individuals to migrate, always using the fitness-proportionate selection (Cantu-Paz, 2001). However, we do explore how the migrants that are selected get distributed to other islands. For a fixed number of migrants, we evaluate dividing them among any number of neighbors between one and four.

Numerous researchers have established the similarity between one large population with n individuals and k smaller populations, each with n/k individuals. Here, we vary the value of k (islands), partly in order to confirm that the performance from the parallel GA corresponds to that of the canonical method, but also partly to decrease the time needed to complete experiments. We want to use as many islands as possible in order to maximize the speedup from our parallel implementation without losing predictive accu-

[1]Note that the similarity of this number to the accuracy reported by Bain & Srinivasan (1994) is completely circumstantial. Their method makes reasonable classifications and does not rely on frequency weighting.

racy. Lastly, we also vary population size with the expectation that accuracy will increase as we increase total number of individuals.

Additional attributes

Unsatisfied with overall improvements from system tuning, we looked for alternative features that we could add to the representations of instances and individuals. As described earlier, we already added six features representing the distance between pairs of pieces, bringing the total to twelve features. We also created several additional features. Because checkmating the black king relies on trapping it against the edge or corner, we include features for the Manhattan distance to the nearest corner and the straight-line distance to the nearest edge. In addition, we included a feature that captures the Manhattan distance between the kings. Finally, we created features that combine the distance of the black king to the nearest corner with the distance between the kings (separately for rank and file).

Empirical Results

We ran the following experiments on a 32-node cluster computer consisting of 2.8 Ghz Pentium 4 processors, communicating with each other over a gigabit switch. We have written our code in C++ using the MPI library. For all experiments, our training set consists of 1000 random samples from the full dataset of 28K instances. We use this sample to determine the fitness of individuals for the purposes of survival, breeding, and migration. However, we periodically evaluated the accuracy of the population over the entire dataset. For these experiments, we hold the mutation rate constant at 0.01. Unless otherwise noted, we used eight islands with populations of 400 individuals on each island, for a total population of 3,200 individuals. The migration interval after which selected migrants move between islands is held fixed at 20 generations.

Crossover breeding

In our previous study (Iba et al., 2008), we considered only single and double crossover reproduction. We hypothesized that the length of the genome in our individuals resulted in these crossover values taking insufficiently large steps through the space of models and that higher values would improve learning. To measure the potential benefit from an increased crossover rate, we tested the system with four values of maximum crossover. Figure 1 shows the fitness values for the most fit individuals (across all islands) under maximum crossover values of 5, 10, 20 and 40. We compute these fitness values over the training set of 1,000 instances; for the most fit individuals, Figure 2 reports their accuracy over the entire dataset.

From the figures, we see that increasing crossover rate improves performance up to a point. However, continued increase from 20 to 40 does not provide a benefit and may deteriorate performance. For subsequent experiments, we hold the maximum crossover rate constant at 20.

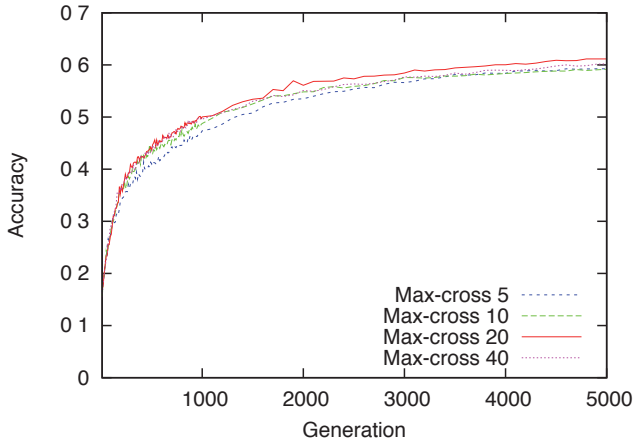

Figure 1: Fitness values for the best individuals over the training set in the Chess Endgame problem comparing maximum crossover rates of 5, 10, 20 and 40 for a population of 3,200 individuals spread over eight islands.

Island topology

Conventional wisdom establishes that only a small number of migrants are necessary to accrue the benefit from the parallel island model. In addition, limited or local connectivity has been shown to approach the performance of fully connected topologies. Thus, we face a tradeoff in terms of the number of migrants moving between islands, the number of immediately adjacent islands, and the degree to which we approach the performance of a single monolithic population. As an attempt to ensure that the number of neighboring islands had no significant impact on learning behavior, we tested a number of different values of this parameter. We expected to find comparable results over a range of connectivity values.

Figure 3 displays the fitness values of the best individuals for five levels of island interconnectedness. In each case, 24 individuals are selected to migrate and are divided evenly between either one, two, three, four or six neighbors. As before, we measure the fitness of individuals over the training set of 1,000 instances; Figure 4 reports the predictive accuracy for the best individuals measured over the entire dataset.

In agreement with our hypothesis, these results show that the dispersal of migrants has no significant impact on accuracy. Seeing no difference between the conditions, we use only two neighbors for our subsequent experiments unless noted otherwise.

Population size and distribution

To ensure that we were not sacrificing performance by resorting to a parallel GA, we varied the distribution of the population over differing numbers of islands. Figure 5 shows the best individuals from a total population when distributed over 8, 16, and 32 islands. Figure 6 shows the best individuals over the entire dataset.

Although the accuracies for 8 and 16 islands are quite similar, we see an unusual result for 32 islands. On the

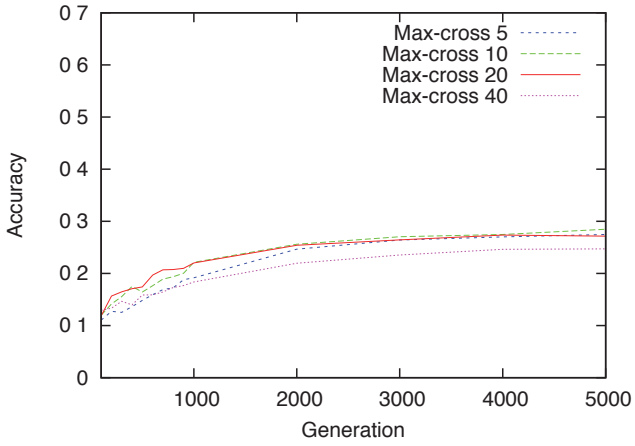

Figure 2: Classification accuracy of the best individuals over the entire KRK dataset comparing maximum crossover rates of 5, 10, 20 and 40 for a population of 3200 individuals spread over eight islands.

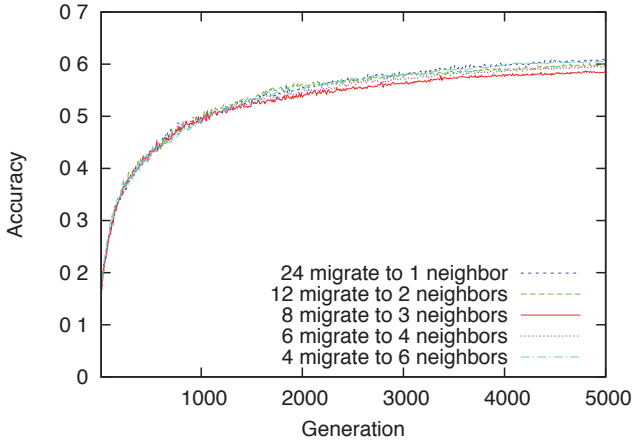

Figure 3: Fitness values for the best individuals over the training set in KRK domain shown for island connectivity values of 1, 2, 3, 4 and 6.

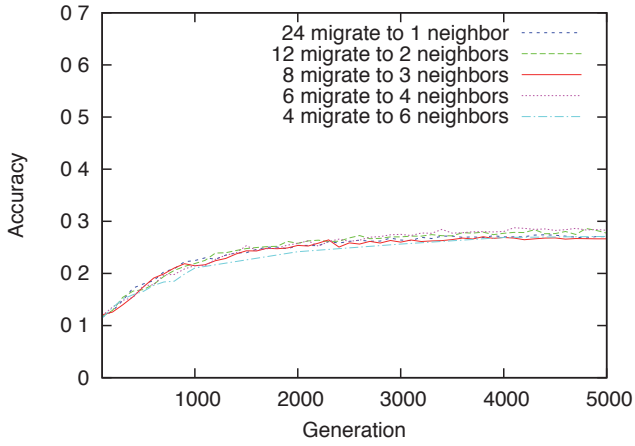

Figure 4: Classification accuracy of the best individuals over the entire KRK dataset for island connectivity values of 1, 2, 3, 4 and 6.

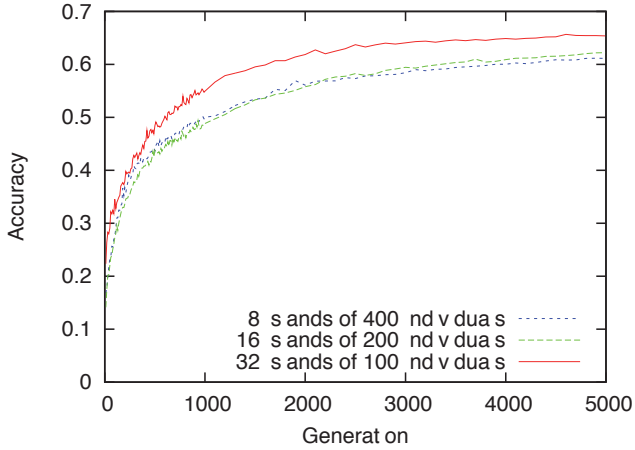

Figure 5: Fitness values for the best individuals over the training set in KRK domain shown for distributions of 3,200 individuals across 8, 16 and 32 islands.

sample set of 1,000 instances, the 32 island condition surpasses the other conditions by approximately five percentage points. However, with respect to the full dataset, the same condition performs about seven percentage points worse than the others. We attribute this somewhat surprising result to overfitting. Because no other condition has given indications of overfitting, we did not expect this result. One explanation points to the relationship between the migration interval and the smaller population size in the 32-island condition. Perhaps with only 100 individuals, the local populations converge on local optima between migration intervals, thereby discouraging search for more accurate and robust models. However, this issue requires further investigation.

Finally, we also ran a test with a significantly larger total population. Figure 7 compares the previous results with a similar run where each of the 32 islands has a population of 400 giving a total population of 12,800 individuals. In this case, we see that the larger population, when tested on the full dataset, is not subject to the overfitting that we observe in the condition with 100 individuals on 32 islands. Furthermore, we note that in the case of the larger population performance on the sample training set is still improving after 5,000 generations whereas the condition with 32 islands of 100 individuals has plateaued. Although the accuracy on the full dataset improved slightly, we also note that the results are not drastically better than the 16 island condition with 200 individuals on each.

Additional features

In all of the previous experiments, we represented board positions with twelve features as described above. For this experiment, we evaluated the contribution of the six features we originally added as well as a number of other features we are considering. To the extent that additional features capture distinctive characteristics of the classes, we hypoth-

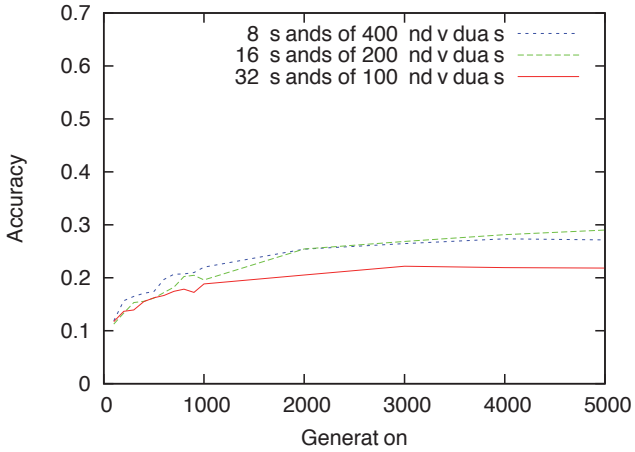

Figure 6: Classification accuracy of the best individuals over the entire KRK dataset for distributions of the total population among either 8, 16, or 32 islands.

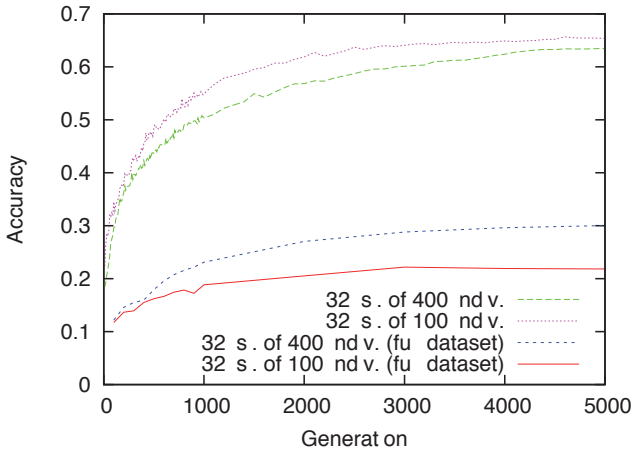

Figure 7: Fitness values of the best individuals over the training set and classification accuracy over the entire KRK dataset for total populations of 3,200 and 12,800 spread over 32 islands.

esized that adding informative features would provide representational leverage and improve classification accuracy.

Figure 8 compares runs using only the six features found in the dataset, 12 features including the pairwise distances between pieces, 15 features including the distance of the black king to the nearest corner, and 19 features including the combinations of distance to corner and distance between kings. As before, we show the results over the entire dataset in Figure 9. The curves reveal a significant benefit over the original six features, but clearly diminishing returns as additional features are added. Here again, we may be glimpsing a mild case of overfitting where the condition with 19 features does slightly better than the others with respect to the sampled training set, but does not perform as well as the 12 or 15 feature conditions when measured against the entire dataset. Additional features may provide the expressive ability to learn more effective models, but they require more

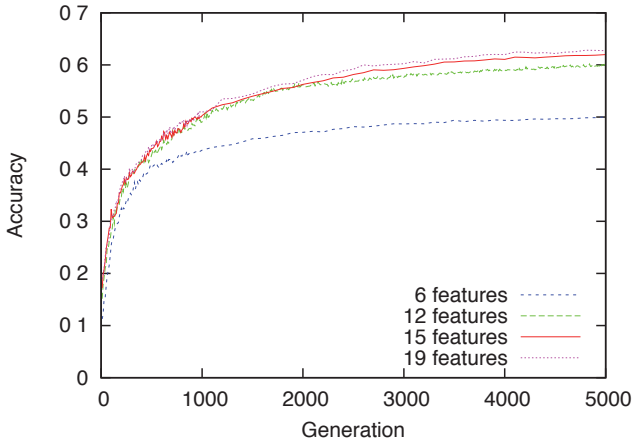

Figure 8: Fitness values for the best individuals over the training set in KRK domain shown for sets of 6, 12, 15 and 19 features.

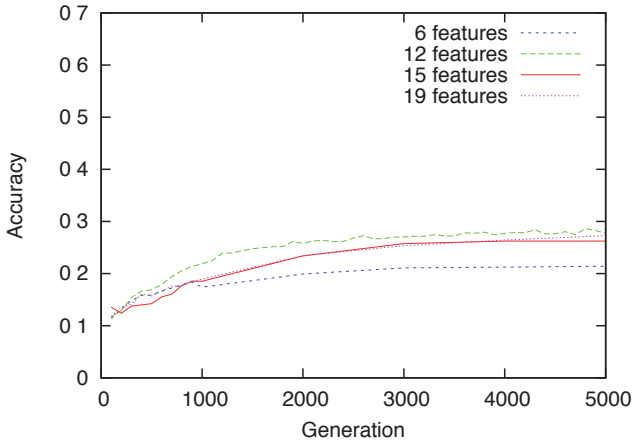

Figure 9: Classification accuracy of the best individuals over the entire KRK dataset for sets of 6, 12, 15 and 19 features.

data and training to exercise this quality; likewise, providing more degrees of freedom invites overfitting.

Discussion

We set out to improve the performance of a traditional GA on the King-Rook-King chess endgame problem in order to better understand the characteristics that make domains amenable to solution by genetic algorithms. This problem domain is known to provide challenges even for methods that exploit structural relationships between instance features (Bain & Srinivasan, 1994). Although we increased accuracy with respect to our previous study (Iba et al., 2008) and we surpassed the results of Bain and Srinivasan (1994) by a significant margin, we are far from having "solved" this problem.

We might conclude that the difficulty encountered by the GA stems from disjunctions in the target concepts. The bit-pattern of features in the genomes of our individuals can represent disjunctions in feature values but cannot represent

disjunctions of combinations of different features. For example, we can represent rank 1 or 8 for the position of the white king, but cannot represent the disjunctive combination of rank 1 and file 1 or rank 8 and file 8. However, a comparison to Quinlan's ID3 algorithm (Quinlan, 1986) – a method that does handle disjunctive concepts – achieves only 26.8% accuracy given a training set of 1,000 instances corresponding to our results prior to introducing additional features (Iba et al., 2008). The modest improvement relative to our previous study together with the relative lack of published studies elsewhere, underline the difficulty of this problem domain. However, the current results do suggest several directions for continued exploration.

Certainly, designing and adding new features may provide significant improvements. By discovering such features, we may gain a better understanding of the characteristics that make the KRK domain such a challenging task. Toward this end, we would like to explore automated feature construction approaches. If done carefully, we hope to identify the key interactions between the domain representation and the genetic algorithm approach that we employ here.

In future work, we need more extensive analysis of the classification errors made by the evolving populations. For example, is it the case that win-in-three problems are being classified as twelves moves to checkmate, or are they getting classified as either two or four moves to checkmate? This analysis anticipates a modified classification task. Instead of discrete classes that must be learned, we could try to learn to predict a numeric value that is nearer or farther from the actual number of moves to checkmate. It would be interesting to evaluate this alternative task both as a fitness measure and as a performance metric.

Another approach we intend to explore bears some similarity to alternate easier versions of the problem. Instead of a single classification task with 18 values, some have treated the KRK problem as 18 different classification problems (Gleich, 2003). However, we would not want to go that far. Instead, it may be fruitful to explore a hybrid approach where island populations evolve specialized classification models, but where the overall problem remains that of distinguishing 18 classes and the disparate models must be unified in some manner.

These are just a few of the possibilities that hold promise for achieving better accuracy on the KRK problem.

Acknowledgments

Former students, Kelsey Marshman, Benjamin Fisk, Joel Stewart and Daniel Rufener contributed to earlier stages of this work. Morgan Vigil provided helpful comments on a draft of the current paper. We also thank the originators and maintainers of the UCI Machine Learning Repository, as well as those who have contributed data sets. Finally, we thank the anonymous reviewers for their helpful comments that have improved the current version.

References

Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository http://www.ics.uci.edu/˜mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science.

Bain, M. & Srinivasan, A. (1994). Inductive logic programming with large-scale unstructured data. *Machine Intelligence 14*, Furukawa, K., Michie, D., Muggleton, S. (Eds). Oxford University Press

Belding, T. C. (1995). The distributed genetic algorithm revisited. In *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Cantu-Paz, E. (2001). Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics*, 7, 311–334.

Clarke, M. R. B. (1977). A Quantitative Study of King and Pawn Against King. *Advances in Computer Chess*, 1, 108-110. M. R. B. Clarke, ed. Edinburgh: Edinburgh University Press.

Gleich, D. (2003). Machine learning in computer chess: Genetic programming and KRK. Independent Study Report, Harvey Mudd College.

Iba, W., Marshman, K. & Fisk, B. (2008). Evaluating a parallel evolutionary algorithm on the chess endgame problem. In *Proceedings of the 2008 International Conference on Genetic and Evolutionary Methods* (GEM08).

Lassabe, N., Sanchez, S., Luga, H., & Duthen, Y. (2006). Genetically programmed strategies for chess endgame. In *Proceedings of GECCO-2006.*

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1: 81-106.