Chapter 8: Result discussion

73-77

Empirical Results

To verify the effect these disruptions might have on solutions found by evolutionary computation, I performed the optimization of topologies for the natural language problem outlined previously under the three genome coding schemes discussed above. In order to better measure the effect of crossover, computations were performed with no mutation and population size of 21 elements. Network fitness values were computed by taking the sum of square errors for all output nodes throughout the presentation of a language of 413 sentences. Training is performed on 20% of this same language. In order to verify consistent performance, all evolved networks were validated by performing 144 bootstrap validation runs per network (Weiss and Kulikowski (1991)). Evolutionary runs were repeated 48 times, and the graphs presented here, although taken from particular runs, are typical of results throughout all runs.

Figures 8-1 and 8-2 show the fitness for the best, average, and worst individuals in a run of 40 generations. Figure 8-1 plots the values for SYSTEM-A, while figure 8 shows the values for SYSTEM-C. It quickly becomes evident that average and even worst element evolve towards the best solution faster under SYSTEM-C. This is in accordance with the prediction made by TDI computations for these two coding schemes; the traits that make the best individual so good can be transferred to other members of the population more efficiently under SYSTEM-C.

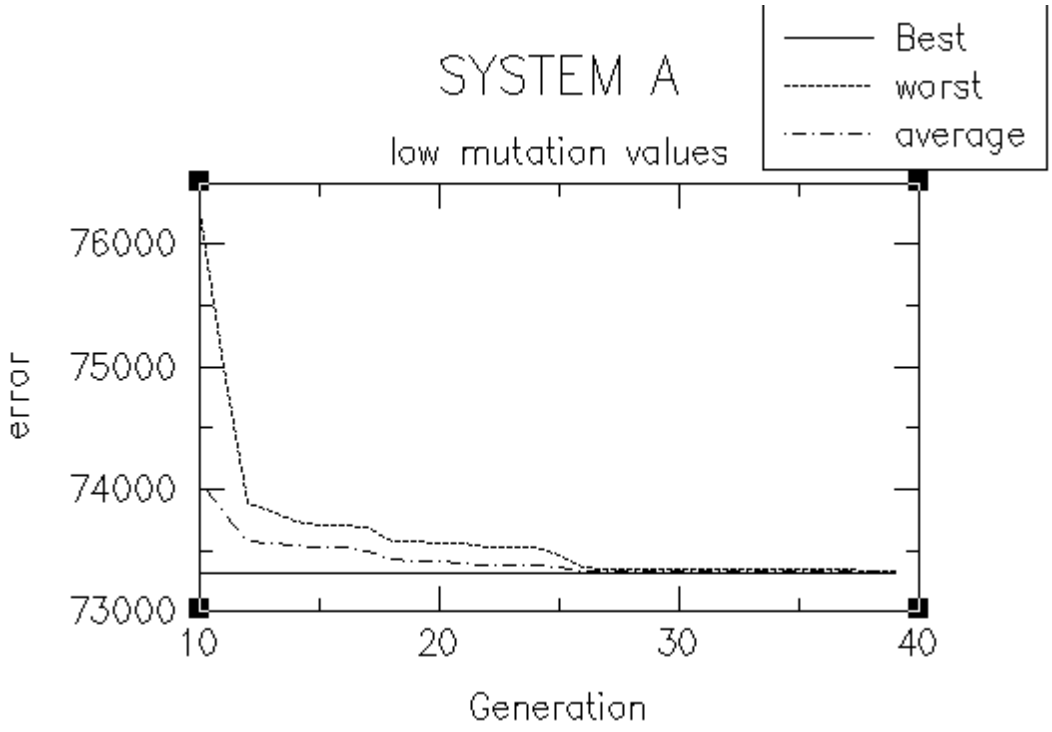

Figure 8-1: fitness values for SYSTEM-A

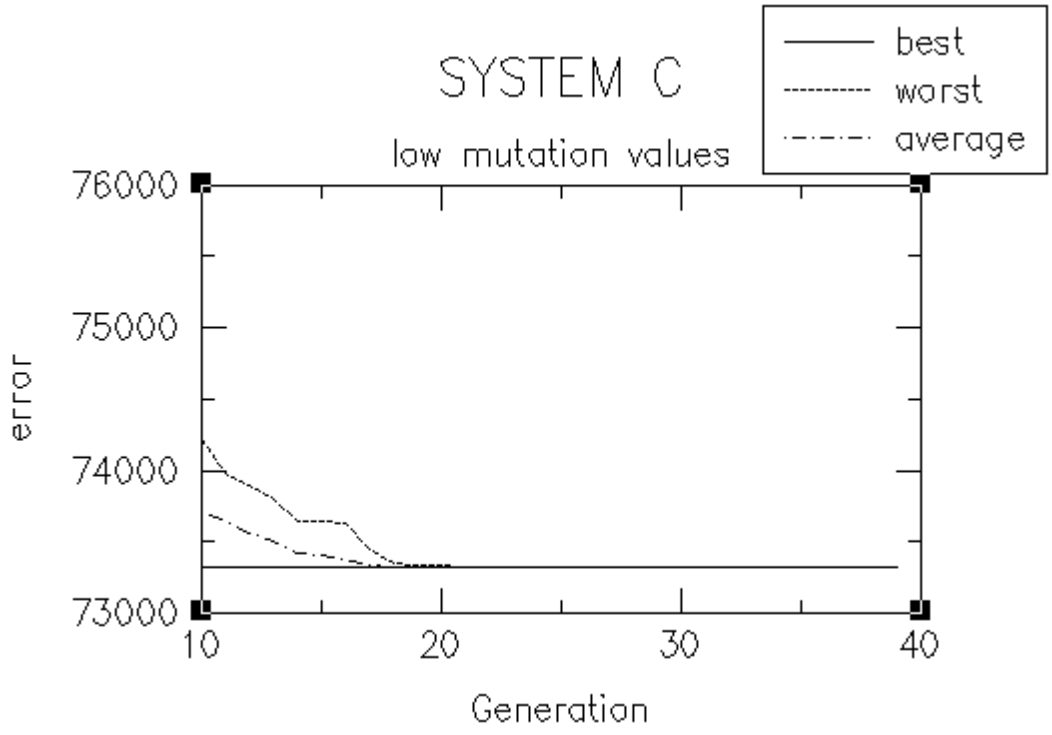

Figure 8-2: fitness values for SYSTEM-C

In order to verify that it is in fact the ability to efficiently process GA schemata that causes this performance difference, runs with SYSTEM-C were repeated with higher mutation values. A sample run of this type of experiment is plotted in figure 8-3. Notice how it now takes longer for the GA to process and disperse good schemata through the population.

It is important to notice that what is being computed, predicted, and verified here is simply the ability of a GA system to process the schemata that already exist in a population.

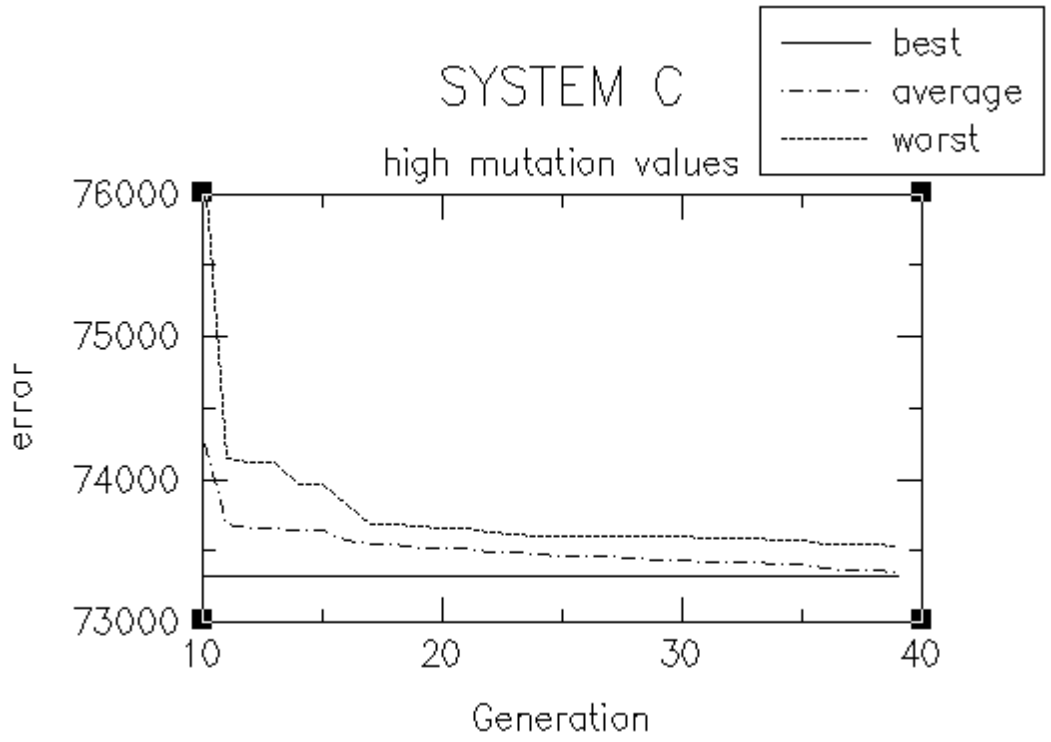

Figure 8-3: fitness values for SYSTEM-C under higher mutation values

These computations do not talk directly about the ability of the system to discover new schemata. In fact, random mutation in some cases could generate an individual which is better than anything currently in the population, or anything that could be evolved from the population by schema recombination. An example of this is shown in figure 8-4. Notice, however, that although the best elements in this population are better than the

ones obtained in previous experiments, the system is still slower to process the schemata. This is shown by the bigger difference between better and worse elements within this particular run.

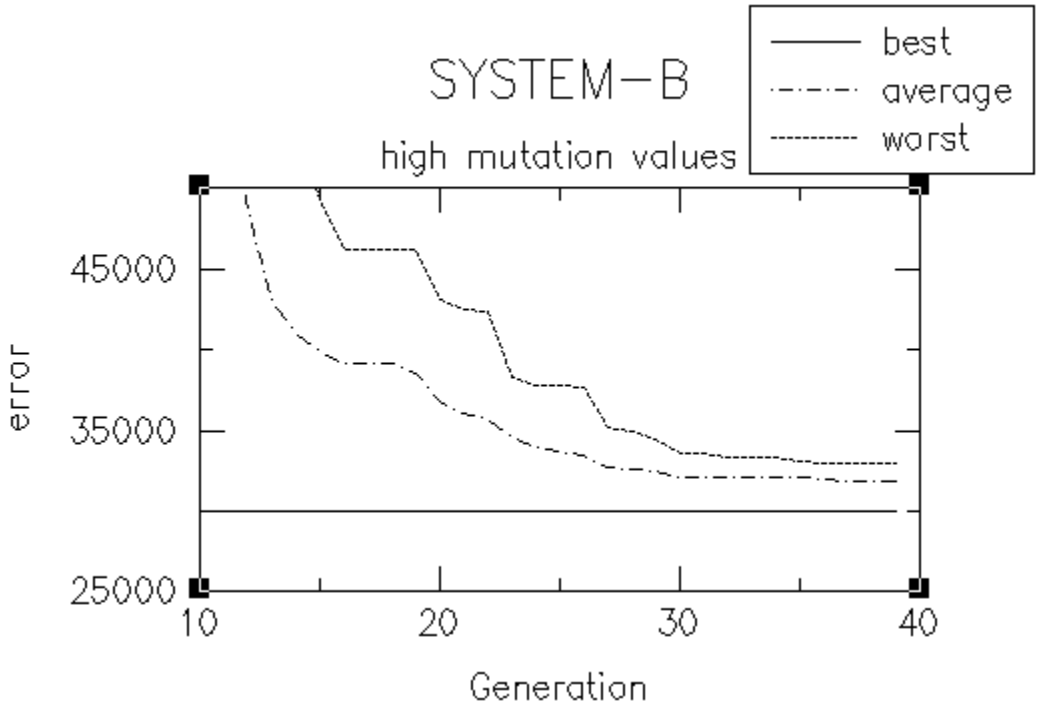

Figure 8-4: fitness values for SYSTEM-B under higher mutation values

Algorithm Category	Strengths	Weaknesses	Predominant Algorithm(s)
AI Methods	Some ideas formed the basis of all further work on the subject e.g. word window	Very domain specific	Expert Systems, as described by Small (1981) Semantic Networks, as described by Dahlgren (1988)
Knowledge Based	Accuracy	Rely on precompiled lexical knowledge resources	The Lesk algorithm, as described in Lesk (1986) Yarowsky's algorithm, as described in Yarowsky (1992)
Supervised	Accuracy	Dependent on pre-annotated corpora for training data	Naïve Bayesian Classifier, as described by Gale et al. (1993)
Unsupervised	No pre-training necessary Works on multiple languages with no modification to the algorithm	Merely discriminates between word senses; not disambiguate word senses	Yarowsky's algorithm, as described in Yarowsky (1995)

Summary of Word Sense Disambiguation Approaches