Duplicate Genotypes in a Genetic Algorithm

Simon Ronald
Program and Data Optimisation Group
National Key Centre for Social Applications of GIS
University of Adelaide, Adelaide, SA, 5005, Australia
dna@gisca.adelaide.edu.au

Abstract—
In a simple genetic algorithm, building-block accumulation stops when the population converges. Premature convergence is a clear indication that this accumulation process has occurred insufficiently. In such a case the final population members will not contain the best combination of building blocks. It is common practice to remove duplicate-population genotypes in a steady-state GA. This allows the process of building-block accumulation to continue for a greater period of time compared with a regular GA and typically results in better final-population solutions. A striking observation is the fact that duplicates degrade GA quality early in a GA evolution run. The results show that diversity loss through duplicates is a serious weakness in the steady-state GA model.

I. Introduction

This paper presents three contributions related to duplicate prevention in a steady-state Genetic Algorithm [1].

The first contribution of this paper is to establish that the process of removing duplicates in a steady-state GA is not at odds with the schema theorem (Section III). It is established that duplicate-removal does not compromise the fundamental theorem of GAs.

The second contribution of this paper is to demonstrate that preventing duplicates results in improved performance in the order-based encoding domain (Section VI-C). Previous studies in duplicate prevention have not been used in this domain. In order to improve the efficiency of duplicate prevention, this paper used the *Hash Tagged duplicate removal algorithm* [2] as applied to steady-state GAs.

The third contribution of this paper is an exploration of the reasons why duplicate prevention is effective in an evolving population. In Section VII, an examination is made to the relative contributions of the crossover and mutation operators. It will be demonstrated that crossover plays a much more pronounced role and is effective for a greater number of generations when diversity is preserved through duplicate prevention. It will be shown that this leads to better building-block accumulation and hence better final population solutions.

II. Introduction

Fogel [3] writes ' ... a genetic algorithm that does not employ a heuristic method for preventing or postponing premature convergence ... will not tend to discover nearly globally optimal solutions in a reasonable number of generations'. This paper presents two methods for postponing premature convergence by retaining genotype diversity within the population. A duplicate removal technique is devised to prevents exact duplicate genotypes from appearing during all phases of evolution. Initial population members can be eliminated in the population using duplicate removal. Duplicate removal can be applied to ensure that created children are different from their parents. Lastly, duplicate removal can be applied to enforce that new children are also different from all other population members.

The computation complexity of duplicate removal should be considered with respect to the number of distance comparisons. A distance comparison in a simple binary domain may involve the evaluation of a genotypic metric norm such as the Hamming measure [4]. However in other domains, such as permutation domains, the distance functions can be much more complex [1]. The number of distance comparisons required to to ascertain whether a new genotype is different from the N other population genotypes is has an important impact on the efficiency of the underlying GA especially when large population sizes are involved. For traditional compare-all comparison techniques a genotype-uniqueness test requires $O(lN^2)$ (l=chromosome length, N=population size) distance comparisons. However, this complexity can be reduced to $O(l)$ using a method of genotype Hash tagging [1], [2].

This paper considers duplicate removal in the domain of permutation-encoded problems. The benefits of duplicate removal are demonstrated on the Travelling Salesperson Problem (TSP).

A. Related Work in Duplicate Removal

Mauldin [6], first observed the benefits of eliminating duplicate genotypes during a GA run. Mauldin used a uniqueness operator which acted on binary-encoded genotypes and prevented duplicate and similar genotypes in an evolving population. This operator only allowed a new child x to be inserted into the population if x was greater than a Hamming-distance threshold from all existing population genotypes. Davis, observing the number of duplicates in an evolving population, wrote in [7] that regular steady-state GAs '... create their allotted number of genotypes with a great deal of duplication.' Davis showed that with a binary encoded GA, removing duplicates in a steady-state GA resulted in superior performance in a comparable number of child evaluations [7][1]. This observation was later re-confirmed by Eshelman and Schaffer [9]. Eshelman and Schaffer tested the effect of preventing duplicates, along with new operators and selection-based innovations[2]. Their tests were conducted on thirteen mathematical test

[1] Based on a presented experiment for optimisation of the F6 test function

[2] The main focus of this paper was to discuss the technique of Incest Prevention however a study of the prevention of duplicates was performed in isolation to the technique of incest prevention.

0-7803-4869-9/98 $10.00©1998 IEEE

problems which included an epistatic problem (interrelated genes) and two deceptive problems (with sub-optimal attractors). Their results showed that the prevention of duplication of genetic strings significantly reduced the mean number of evaluations required to find the global optimum for each problem when averaged over 50 independent evolution runs[3].

B. Significance of Duplicate Removal

It will be argued and demonstrated that the technique introduces a powerful diversity preservation mechanism resulting in better final-population solutions. Simulation results are presented which illustrate the effectiveness of duplicate removal in GAs for a variety of population and problem sizes. These results are directly compared to a regular GA not employing duplicate removal. Results are presented which compare duplicate-removed GAs with other techniques of diversity preservation. These experiments show that hash tagging gives superior results for a sample 30 town TSP problem, as compared with a regular GA. It will be demonstrated that the effectiveness of duplicate prevention in permutation-type problems is in agreement with the reported success of duplicate prevention in binary-encoded domains.

III. Duplicate Prevention and the Schema Theorem

Previous work in duplicate removal has assumed that removing duplicates in a steady-state GA is theoretically sound. Whilst this is the view of the author, this section analyses duplicate removal in the context of the schema theorem. Since the schema theorem helps model why GAs perform so well as state-space searchers, it must be proved [4] that duplicate removal is not at odds with any of the basic mechanisms that accumulate good solution components.

A. Duplicates in Generational GAs

In generational GAs, duplicate genotypes are introduced into the new population during the reproduction phase [10]. During reproduction a new GA population is created from an old population. For each genotype x in the old population, the probability that it receives zero or more copies of itself in the new population can be determined by the fitness of x. This form of reproduction leads to fit schemata accumulation[5].

Duplicate schemata accumulate in the new population according to a schema recurrence relation [10]. The notation H refers to schema H. The value $m(H,t)$ refers to the number of schema H contained in the population at generation t. The value $f(H)$ refers to the average fitness of all genotypes containing schema H. The notation $\bar{f}$ refers to the average population fitness.

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}}. \quad (1)$$

From Equation 1, it follows that if a schema H has a fitness above the population fitness by a constant factor, then schema H will receive a geometrically increasing number of trials in the population.

In generational GAs, reproduction is one mechanism by which above-average schemata are propagated into subsequent generations until winning schemata are represented in all of the population members and no further growth is possible.

If duplicates were removed for the sake of maintaining diversity in a generational GA then the mechanism of reproduction would not function as a source of selection pressure, as duplicate-fit individuals would be forbidden. The mission of this paper is to determine the degradation effect of duplicates that emerge from processes other than the duplicates caused by reproduction[6]. By removing the reproduction process two complex interrelated processes would result, selection pressure would be reduced[7] as well as the effect of diversity loss cause by reproduction. By eliminating reproduction it is difficult to measure the performance contribution of selection pressure versus the performance degradation of diversity. To solve this problem, steady state GAs were considered.

B. Duplicates in Steady-State GAs

The situation is different when considering steady-state GAs as a reproduction mechanism is not used. A rank-based selection strategy is considered along with a replacement-of-the-worst child-replacement strategy. The variable c will be used to denote the number of steady-state generations that have elapsed since the initial population. The quantity $p_o(H)$ denotes the probability that schema H is destroyed by the application of all of the genetic operators acting on H in a given child generation. The growth of a schema H_a with a fitness greater than the median-population fitness value changes from generation to generation according to the recurrence relation ([1] Section 1.4.5)

$$m(H_a, c+1) > m(H_a, c)\left[1 + \frac{(1-p_o(H_a))}{N(N-1)}\right]. \quad (2)$$

From Equation 2, it can be seen that a steady-state GA achieves the goal of near-exponential proliferation of above-median schemata without needing to add duplicates with reproduction, as is done in a generational GA. Therefore, removing duplicates in a steady-state population does not interfere with the underlying mechanism of building-block propagation.

[3] Eshelman and Schaffer excluded duplicate genotypes from the count of the number of evaluations required to find the global optimum. Their final results therefore give no comparative feel for the effort involved in regenerating new non-duplicate genotypes. The results in Section VI give comparisons based on the total number of evaluations including any duplicates that were regenerated.

[4] although it is obvious

[5] The use of reproduction is only one form of selection pressure that is possible in a generational GA, fitness-based parent selection and the fulfillment of the new population by eliminating less-fit individuals are just two other additional or alternative methods of selection.

[6] duplicates arising as a result of crossover and mutation during evolution

[7] If reproduction is the only source of selection pressure, the removal of duplicates would eliminate all selection pressure and result in a non-GA.

The next point to be explored is that the largest possible schema in a GA with an encoding of l genes per genotype is a schema of order l. When duplicates are prevented in a steady-state GA, these schema of order l are not able to propagate at all. This is because more than one such schemata would represent one or more duplicate genotypes. In the next section it is shown that these schema of order l are not important building blocks and the growth of such schema can be ignored without compromising the effectiveness of the GA search.

C. The Role of the Order l Schema

The following equation shows schema growth according to the probability that an above average schema H_a is destroyed by the application of an arbitrary genetic operator o. The probability that o destroys the schemata H is given by p_o. The number of instances of H_a in the evolving population is given by

$$m(H_a, c+1) \geq m(H_a, c)\,[1 + c_1(1 - p_o(H_a))] + g \quad (3)$$

where c_1 is a constant. In binary encodings, with one point crossover and bit mutation, p_o is a monotone-increasing function of the length of schema H ($\delta(H)$) for conventional crossover. Additionally, p_o is a monotone-increasing function of the number of defined loci (the order) of schema H ($o(H)$) for conventional mutation [10]. When conventional crossover and mutation are applied together, the probability of schema destruction is given by $p_{do} = k_1\delta(H) + k_2 o(H)$. This implies that

$$m(H, c+1) \geq m(H, c)\,[1 - k_1 c_1 \delta(H) - k_2 c_1 o(H)] + g \quad (4)$$

where k_1 and k_2 are constants. This is a form of the well-known theorem of GAs [11]. The exponential index of the growth of schema H is $1 - k_3\delta(H) - k_4 o(H)$, where k_3 and k_4 are constants. Since all above average schema are competing for trials, the real winners will be those that have a small $\delta(H)$ and a small $o(H)$. This has been stated 'Short, low-order, above-average schemata receive exponentially increasing trials in subsequent generations' [10]. Given that the genetic operators destroy a schemata H with a probability proportional to its order and length it is possible to conclude that given a variety of schemata of different orders and lengths which are all above average by the same degree, short and low-order schemata will clearly dominate in the competition for reproductive trials. It has been shown that genetic operators that operate in the permutation domain, such as PMX and inversion [10], [11], do not preserve the high-length o-schemata. Therefore the o-schemata of order l, which are schemata of the highest order and longest length $(l-1)$, do not represent important building blocks during GA evolution.

One more point must be explored. The schemata of order l contain many lower-order building blocks (the set of schemata S). The schemata in S may not propagate when their containing schema of order l is a duplicate and is refused entry into a population. However, the schemata S can find many other opportunities to proliferate; they only need to be associated with other higher-order containing schemata to make up a slightly different genotype (that is unique in the population) to gain entry into the population.

In summary, preventing duplicates in steady-state GAs does not interfere with the ability of the GA to propagate schemata nor does it interfere with the proliferation of small, highly fit, building blocks.

IV. Sources of Diversity Loss in GAs

In a steady-state GA, duplicate genotypes can enter an evolving population through any of three scenarios:

1. Duplicate genotypes appearing in the initial (randomly generated) population;
2. Duplicate genotypes appearing when a newly created child is identical to one of its parents, and;
3. Duplicate genotypes appearing when an operator creates a new child x which is different from the parents of x, but the same as another existing population genotype.

The standard steady-state GA technique does not eliminate any of these sources of genotype duplication. It has become a common practice however for GA practitioners to ensure that newly created children are different from their parents. Davis, for instance, addresses scenario 2 with special genetic operators in [12]. These operators, such as guaranteed-average and guaranteed-mutation, create children which are different from the parents. If they should generate a child the same as either parent then the child is repeatedly regenerated until it is different. Using this strategy, if two children are created from two parents, then five genotype to genotype comparisons are required to determine that both children are different from their parents and different from each other.

A number of ad-hoc experiments over a variety of TSP problems revealed that even when duplicates in scenario 2 (in the list above) were eliminated, many other duplicates still crept into the population during the later stages of evolution as a result of scenario 3. A simple algorithm to preclude scenario 3 duplicates is to check each newly generated genotype against each other genotype in the population and regenerate the child in the case of a match. However, for each new child this would introduce an execution time of N genotype-to-genotype comparisons. This is the approach taken by Davis [7]. Davis argues that this time overhead is not significant in real-world problems where most of the GA execution time is taken in the fitness-evaluation module. Despite this observation, the comparison time becomes an unnecessary computational overhead in a GA with a large population and a large genotype. The efficient hash tagging duplicate removal algorithm from [2] was used to remove duplicates.

It would not be difficult to modify the duplicate removal algorithm to preclude the occurrence of scenario 1 (duplicates in initial population) as well. With the permutation encoding considered in this paper duplicates are not expected for practical population parameters. The proof in [2] showed that the probability of initial population duplicates would only become significant for very small permutation encodings (e.g. small TSP problems), and for large population sizes, an unusual combination for most problems.

V. A Sample Encoding

In this paper a GA using a fixed length genotype is considered. The experiments and examples given in this paper all refer to an order-based encoding. The choice of this order-based domain is not central to the arguments and methods presented in this this chapter, and other encodings such as binary encodings might be considered next. A permutation encoding can be represented by a list of distinct integer values, for example

$$x = [4, 3, 0, 1, 2]. \quad (5)$$

Each integer value in the list directly encodes the relative ordering of some problem-specific object. This representation was used in the TSP experiments because it

1. prohibits missing or duplicate allele values;
2. it allows high-performance genetic operators (such as the edge recombination operator [13]) to be used;
3. it facilitates a simple decoding mechanism from the genotype to the phenotype for TSPs.

In the cyclic TSP, an encoding of x represents a TSP circuit that starts with the town with the label 4, proceeds in order to towns labelled 3, 0, 1, and 2, and returns back to the town labelled 4 to make a complete circuit. The hash tagging algorithm is now discussed in relation to this encoding.

VI. Experiments with Duplicate Removal

A. Experimental Conditions

The GA used in all experiments was a steady-state GA. A replacement-of-the-worst genotype strategy was used where the population member with the worst fitness value was replaced with each new child. In every case, the initial population was created by shuffling lists of town identifiers (gene values) in each genotype to create N random Hamiltonian circuits. All of the graphs in this paper represent a simulation curve (a run) obtained by taking the average over 50 different GA sub-runs. Each GA sub-run was seeded with a unique random number seed which guaranteed a different random number stream for each sub-run. Shift-isomorphism removal was performed in all experiments including those not employing duplicate removal[8]. All of the fitness graphs show the number of created children on the abscissa. To enable a fair comparison of computational effort, the number of children produced included all children that were generated, even those not incorporated in the population due to a hash-tag clash. This is the most conservative way of presenting the duplicate removal results. If regenerated children were eliminated from the child count then all of the duplicate removal results would converge significantly faster (around 50% faster), as a large number of hash-tag clashes occur towards the end of a GA run.

The Edge Recombination Operator (ERO) [13] was used as the crossover operator. For mutation, a cyclic-inversion operator, as described in [15], was employed. The shortest chunk sizes (2 genes) were chosen 1.75 more times than the largest chunk size of $\frac{l}{2}$ genes, with a linear probability profile used for all chunk lengths in between. This probability value was nominally found to give a good balance between frequently disturbing short sub-circuits and occasionally disturbing the longer sub-circuits.

Throughout all experiments the ERO was applied with a fixed probability of 0.4, and inversion was applied with a probability of 0.6. Only a single operator was applied in one steady-state generation.

In each experiment if the population size was greater or equal to 1000, then a linear selection bias of 1.9 was used. This is a typical selection bias for populations of this size [15]. This means that the population genotypes were ranked in order of increasing fitness, and the fittest population genotype was chosen on average 1.9 time more often than the worst population genotype. If the population size was less than or equal to 1000, then a more gentle selection bias of 1.01 was used. This low, almost random, selection pressure was found to be beneficial to all experimental runs[9].

B. The Oliver 30-town TSP

The Oliver 30-town TSP problem [13] was used as the first test problem in the duplicate removal experiments. The 30 town coordinates lie within a 100x100 grid and are defined in [13].

All distances were calculated as Euclidean, rounded down to the nearest integer, as is stipulated in the TSP benchmark standard [16]. The fitness of a genotype was defined as the the length of the shortest published tour length ([13]) divided by the length of the tour resulting from that genotype. Hence the best expected fitness value was 1.

C. Duplicate Removal versus Regular GA

Three experiments were conducted to compare a duplicate removed GA with a regular GA not employing duplicate removal. Two population sizes of 50 and 200 genotypes were used. It can be seen from the average-fitness performance graph Figure 1, that for both population sizes, hash tagging converges, on average, to a better final solution.

Figure 2 shows the fitness curve for the four population sizes of 20, 50, 200 and 1200 plotted on the same graph.

In the regular GA, various degrees of premature convergence are evident, with the fitness shortfall dependant on the size of the population used. The smallest population size of 20 converges around 6% short of the best-known solution. The largest population size of 1200 does not significantly pre-converge. The duplicate removed run shows a much lesser degree of pre-convergence than a regular GA in the smaller population size of 50 individuals.

Table I indicates the number of runs in which the best-known solution was reached when the three population sizes of 50, 200, and 1200 were used. In a regular GA,

[8] Shift isomorphism removal is implemented by requiring that the gene value 0 is always encoded in the first gene position. This eliminates permutation rotations and reduces the size of the search space by a factor of $l - 1$, see [1].

[9] Note that a selection bias of 1.01 constitutes an almost random parent selection scheme, but an exponential allocation of trials to above average schemata still occurs because of the replacement-of-the-worst regime.

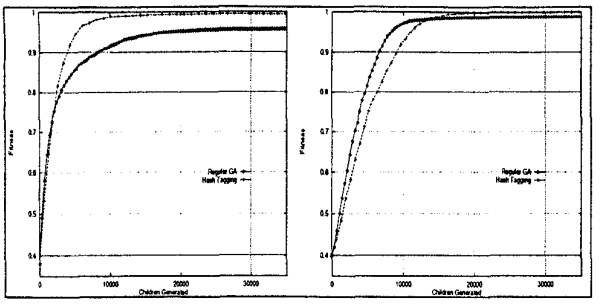

Fig. 1. Oliver 30-town TSP with and without duplicate removal averaged over 50 runs: Population size of 50 *(left)*. Population size of 200 *(right)*.

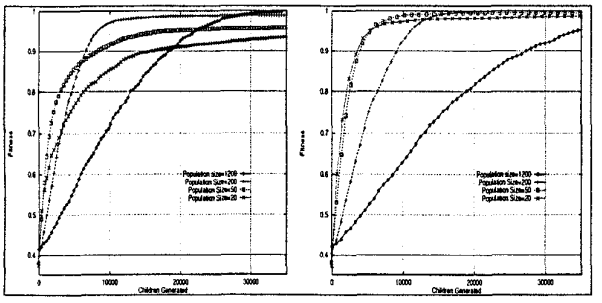

Fig. 2. Oliver 30 Town TSP with and without duplicate removal averaged over 50 runs: Without duplicate removal *(left)*, with duplicate removal *(right)*.

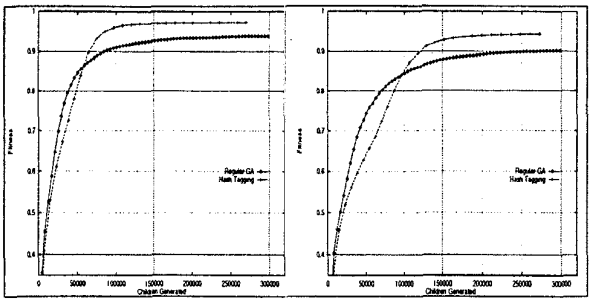

Fig. 3. Hash tag comparison for larger problem sizes, averaged over 50 runs: 76 town problem, *(left)* 101 town problem *(right)*.

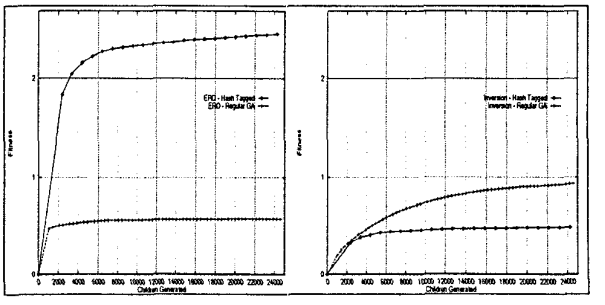

Fig. 4. Sum total of fitness contribution of genetic operator versus the number of children produced. The test problem was the Oliver 30 Town TSP with population size of 200, averaged over 50 runs: Crossover (ERO) *(left)*, Mutation (inversion) *(right)*.

it can be seen that the runs that used the larger population size of 1200 found significantly more solutions than population sizes of 50 or 200. This can be attributed to increased schemata processing in the larger population size. However, it is interesting that hash tagging in a population size of 50 finds almost twice as many of the best-known solutions than a regular GA can achieve with a population size of 1200.

C.1 Larger Sized Problems

The duplicate-removal algorithm performance was also tested on two larger sized TSP problems. The 76 town problem was chosen from the TSPLIB library (problem PR76) [16]. The 101 town problem was the EIL101 problem. The population size used in both cases was set at 200. The graphs in Figure 3 show, averaged over 50 independent runs, that duplicate removal is clearly effective in both of the larger problem cases in producing higher-fitness final population champions than is the case where duplicate removal is not used.

Method	Popsize=50	Popsize=200	Popsize=1200
Regular GA	3	3	22
Duplicate Removal	41	50	49

TABLE I

THE NUMBER OF TIMES THE BEST-KNOWN SOLUTION WAS REACHED OVER THE 50 INDEPENDENT RUNS FOR THE OLIVER 30 TOWN TSP, DUPLICATE REMOVAL VERSUS REGULAR GA.

VII. Why Duplicate Removal Works

The results in Section VI showed that duplicate removal had a significant beneficial effect in maintenance of diversity in a GA run. It was demonstrated that this lead to desirable properties, i.e. better final solutions being reached. This section presents an argument as to why preserving diversity should result in better solutions. The fundamental theorem of GAs suggests that short, low-order building blocks accumulate at an exponential rate as the GA runs [10]. However when saturation, or near-convergence occurs, this exponential growth decays to zero growth. Therefore, if building blocks are not accumulating in a near-exponential fashion, the GA process has ceased working effectively. At the point of near-convergence, late in a GA run, small gains are made when the occasional mutation (inversion) results in a better solution. However, such gains are small as the GA is reduced to a mutative-hill climber. If diversity is preserved in a GA for a longer period of time then the accumulation of good building blocks can occur for longer. The time in which implicit parallelism is at work can be extended. Since implicit parallelism usually results in a much broader search than hill climbing [10], [17], it would be expected that best-final-population genotype will contain more good building blocks and have a higher fitness than a GA with greater diversity loss.

Figure 4 shows the comparative effectiveness of crossover

and mutation throughout evolution for both a duplicate removed GA and a regular GA. The curves were based on two derived variables v *(left)* and m *(right)*. These variables were derived as follows; At the start of evolution, two variables v and m were set to zero. During evolution, when the ith child c_i was generated, the following variable increments were made:

1. If c_i was created by crossover (ERO) and if the fitness of the child was greater than the fitness of both parents that created it. That is, if $f(c_i) > f(p_1)$ and $f(c_i) > f(p_2)$, then

$$v = v + f(c_i) - \max(f(p_1), f(p_2)) . \quad (6)$$

2. If c_i was created by mutation (inversion) and if the fitness of the child is greater than the fitness of the parent $(f(c_i) > f(p))$, then

$$m = m + f(c_i) - f(p) . \quad (7)$$

Therefore, the v and m curves give a measure of the effectiveness of each operator in producing children fitter than their parents.

Figure 4 *(left)* shows that crossover is much more effective in a duplicate removed GA as compared with a regular GA. It can be seen that crossover continues to be useful right up until generation 24,000, whereas in a regular GA the effect of crossover is finished at generation 10,000. The effect of mutation in Figure 4 *(right)* is more significant in a regular GA than a duplicate removed GA. This is consistent with the idea that a regular GA prematurely converges in the early phases of evolution. At this point, crossover becomes less effective as all parents resemble each other and little genetic material can be recombined into newly-produced children. The only improvement mechanism that remains is mutation, where new edge relationships are introduced into the population through the inversion operator. The consistent growth of the inversion curve right up until generation 24,000 suggest that small improvements are being made by mutation until the end of the GA run.

VIII. Conclusions

A. Removing Duplicates

It has been shown that the idea of duplicate removal using hash tags in a steady-state GA is not at odds with the schema theorem [18] and the fundamental theorem of GAs. It was argued that the technique of duplicate removal preserves genetic diversity and prolongs the useful period in which mutation and crossover can work together to solve a particular problem. This enhanced search has been more effective in solving small-sized TSP type problems without resorting to problem-specific hybrid techniques such as 2-opt and 3-opt [19]. A number of simulation results have shown that TSP problems of various sizes have better performance in a duplicate removed GA for a comparable number of total child generations, compared to a regular GA. Hash tagging was compared with the technique of reseeding and parallel populations and superior results were observed within the stated parameters.

A number of TSP test problem sizes were chosen and the results indicated the effectiveness of duplicate removal in each. These simulation results illustrate that if diversity is preserved, it may not be necessary to use huge population sizes or other diversity-preservation techniques to find good solutions to a given problem.

A striking observation when investigating duplicate removal was that crossover (Edge Recombination) was much more effective throughout all phases of evolution when duplicates were removed (Figure 4 *left*). In contrast to this, mutation (inversion) was less effective throughout the later phases of evolution (Figure 4 *right*). This suggests that removing duplicates assists in schema-accumulation and implicit parallelism throughout all phases of a GA. This observation differs from the popular notion that duplicates result in minor GA degradation due to near-end-of-run diversity loss. More experimentation and analysis in different domains is due.

References

[1] S. Ronald, *Genetic algorithms and permutation-encoded problems. Diversity Preservation and a Study of Multi-Modality*, Ph.D. thesis, University South Australia. The Department of Computer and Information Science, 1995.

[2] S. Ronald, "Preventing diversity loss in a routing genetic algorithm with hash tagging," in *Complex Systems: Mechanism of Adaption*, R. Stonier and Xing Huo Yu, Eds. 1994, pp. 133–140, IOS Press, Amsterdam.

[3] D. Fogel, *Evolutionary computation. Towards a new philosophy of machine intelligence*, IEEE Press, Piscataway, NJ, 1995.

[4] D. Goldberg, "Genetic algorithms with sharing for multimodal function optimization," *Genetic Algorithms and their Applications. Proceedings of the Second International Conference on Genetic Algorithms*, pp. 41–49, 1987.

[5] M. Mauldin, "Maintaining diversity in genetic search," in *National Conference on Artificial Intelligence*, 1984, pp. 247–250.

[6] L. Davis, *Handbook of genetic algorithms*, Van Nostrand Reinhold, New York, 1991.

[7] L. Eshelman and J. Schaffer, "Preventing premature convergence in genetic algorithms by preventing incest," in *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. 1991, pp. 115–122, Morgan Kaufmann Publishers, San Mateo, CA.

[8] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[9] J. Holland, *Adaption in natural and artificial systems*, The University of Michigan Press, 1975.

[10] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*. 1989, pp. 61–67, Morgan Kaufmann, San Mateo, CA.

[11] D. Whitley, T. Starkweather, and D. Fuquay, "Scheduling problems and traveling salesmen: The genetic edge recombination operator," in *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, J. Schaffer, Ed. 1989, pp. 133–139, Morgan Kaufmann Publishers, CA.

[12] D. Whitley, "Getting started with GENITOR," *Genitor User Documentation*, 1993.

[13] G. Reinelt, "TSPLIB - a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–385, Fall 1991.

[14] D. Ackley, "An empirical study of bit vector function optimization," in *Genetic Algorithms and Simulated Annealing*. 1987, pp. 170–203, Pitman Publishing, London, L. Davis ed.

[15] R. Hollstien, *Artificial genetic adaption in computer control systems*, Ph.D. thesis, University of Michigan, 1971, Dissertations Abstractions International.

[16] H. Taha, *Operations Research, Fifth Edition*, Macmillan Publishing Company, 1987.