SELF-ADAPTATION IN GENETIC ALGORITHMS USING MULTIPLE GENOMIC REDUNDANT REPRESENTATIONS

Maheswara Prasad Kasinadhuni, Michael L. Gargano, Joseph DeCicco and William Edelson

Maheswara Prasad Kasinadhuni, mpkasinadhuni@hotmail.com,
Michael L. Gargano, mgargano@pace.edu,
Joseph DeCicco, email_of_Joseph @yahoo.com,
William Edelson, wedelson@pace.edu
Computer Science, Pace University, NYC, NY 10038

Abstract. We consider an extension to optimization problems [7, 13, 14] where the element costs are not fixed, but are time dependent. We propose using multiple genomic redundant representations in a self-adapting genetic algorithm (GA) employing various codes with different locality properties. These encoding schemes insure feasibility after performing the operations of crossover and mutation and also ensure the feasibility of the initial randomly generated population (i.e., generation 0). The GAs solving this class of NP hard problems, where costs are not fixed but are time dependent, employ non-locality or locality representations when appropriate (i.e., the GA adapts to its current search needs) which makes the GAs more efficient. A few applications with time dependent costs will also be presented.

1 Introduction to Problem Applications

1.1 Maximal Rooted Hamiltonian Directed Paths:

We wish to find a maximal rooted Hamiltonian directed path in a complete graph with nodes { 0, 1, 2. …, n-1, n } and root node 0 [10]. An (n+1) by n table of lists is given showing the values of a directed edge from node i to node j at step t (where for fixed i, (i, j) is the same at t). The problem is to find a maximal directed path through the graph that begins at node 0 by choosing a directed edge in the path at each step t.

1.2 Minimal Node Base - A Directed Communications Network Application:

A communications network is a digraph D = (N,A) where N, the set of nodes, is thought of as a set of message transmitters and A, the set of arcs, as the allowable directed connections between pairs of nodes. A node base B in a communications network is a minimal subset of nodes with the property that if a message is placed at each node of B, then every node of D will receive the message. Suppose one wants to build a satellite communications receiving station at each node in a node base of a communications network D. Since these receiving stations are built on a node base *every*

node in D can receive all messages. The cost of building such a station is different for each node D and is assumed to vary over time. It is desired to construct these stations one per time period so that the cost of the completed network is minimized.

1.3 Minimal Bidding (with Slack Time) Application:

A company is to construct a transportation network (graph) connecting a group of cities. Due to budget limitations only one link will be contracted for construction each month. Different contractors having different price scales, which vary over time, bid on the construction of different links of the potential network. The cities together with all the possible links can be modeled to form a multigraph H. The goal is to construct the desired network (an underlying subgraph G with q edges of the multigraph H) such that the total cost over time of the entire construction is minimized. We also assume that it takes one unit of time to construct a link and that the network must be completed in the t = q + s time periods available (q periods for actual construction and s slack periods where no construction is performed). Dummy (slack) links with zero cost for each period can be included. In this problem, we are searching for an underlying graph G whose q edges are sequenced and whose total cost is minimum. We also considered giving discounts from bidders that win more than one contract.

These problems are similar to each other. For example, the maximal rooted Hamiltonian directed path problem is equivalent to the minimal bidding problem where there is only one bidder for each edge, the costs are inverted, there are no discounts, and slack time is zero. Since the rooted Hamiltonian directed path problem is known to be NP hard, the other problems discussed above are also NP hard.

2. Genetic Algorithm Methodology

A **genetic algorithm (GA)** is a biologically inspired, highly robust heuristic search procedure which can be used to find optimal (or near optimal) solutions to NP hard problems. The GA paradigm uses an adaptive methodology based on the ideas of Darwinian natural selection and genetic inheritance on a population of potential solutions. It employs the techniques of crossover (or mating), mutation, and survival of the fittest to generate new, typically fitter members of a population over a number of generations [1, 2, 3].

We propose GAs for solving these optimal sequencing problems using novel multiple genomic redundant encoding schemes (described for each application later). Our GAs create and evolve an encoded population of potential solutions so as to facilitate the creation of new feasible members by standard mating and mutation operations. (A feasible search space contains only members which satisfy the problem constraints, that is, a sequencing [7, 8, 13,14].) When feasibility is not guaranteed, numerous methods for maintaining a feasible search space have been addressed in [11], but most are elaborate and complex. They include the use of problem-dependent genetic operators and specialized data structures, repairing or penalizing infeasible solutions, and the use of heuristics.) By making use of problem-specific encodings, each class of problem insures

a feasible search space during the classical operations of crossover and mutation and, in addition, eliminates the need to screen during the generation of the initial population.

Given a cost matrix, fitness is calculated by summing the costs of the edges of the sequence (i.e., the fitness is the total cost = T = $C(e_1,1) + C(e_2,2) + \ldots + C(e_n,n)$. Note that a rough lower bound for the fitness is the sum of the minimum costs of each column of the cost matrix). We adapted many of the standard GA techniques found in [1, 2, 3] to these specific problems. A brief description of these techniques follows. Selection of parents for mating involves randomly choosing one very fit member of the population and the other member randomly. The reproductive process is a simple crossover operation whereby two randomly selected parents are cut into sections at some randomly chosen positions and then have the parts of their encodings swapped to create two offspring (children). In our applications the crossover operation produces an encoding for the offspring that have element values that always satisfy the position bounds (i.e., range constraints). Mutation is performed by randomly choosing a member of the population, cloning it, and then changing values in its encoding at randomly chosen positions subject to the range constraints for that position. A grim reaper mechanism replaces low scoring members in the population with newly created more fit offspring and mutants. The GA is terminated when, for example, either no improvement in the best fitness value is observed for a number of generations, a certain number of generations have been examined, and/or a satisficing solution is attained (i.e., the total cost T is not necessarily optimum, but is satisfactory).

The Generic Genetic Algorithm

We can now state the generic genetic algorithm we used for each application:

1) Randomly initialize a population of multiple genomic redundantly encoded potential solutions.
2) Map each population member to its equivalent phenome.
3) Calculate the fitness of any population member not yet evaluated.
4) Sort the members of the population in order of fitness.
5) Randomly select parents for mating and generate offspring using crossover.
6) Randomly select and clone members of the population to generate mutants.
7) Sort all the members of the expanded population in order of fitness adjusting each of the multiple segments to reflect the phenome with best fit.
8) Use the grim reaper to eliminate the population members with poor fitness.
9) If (termination criteria is met) then return best population member(s)
   else go to step 5.

3. Encodings

Each of the applications we will discuss has multiple permutation encodings to identify the sequencing via different representations. Here we define the permutation code, forward code, and backward code for a permutation.

The 5-permutation 41532 or P[1] = 4, P[2] = 1, P[3] = 5, P[4] = 3, and P[5] = 2 can represent itself. This is one of multiple representations of 41532. We call this the **permutation code** and PC[1] = 4, PC[2] = 1, PC[3] = 5, PC[4] = 3, and PC[5] = 2. The permutation code typically has children that are less similar to their parents. It has low locality.

An n permutation of the integers { 1, 2, …, n } can also be encoded by an array of size n where the value of the k^{th} position can range over the values 1, 2, …, n – k + 1.

An encoding of a permutation of the elements can also be represented as an array FC (**forward coding**) where $1 \le FC[k] \le n - k + 1$ for $1 \le k \le n$. In order to decode a permutation code FC to obtain the permutation that it represents, begin with an empty array P of size n, then for $1 \le i \le n$ fill in the $FC[i]^{th}$ empty position (from left to right starting at position 1) of P with the value i. Consider an example, with n = 5 and FC[1] = 2, FC[2] = 4, FC[3] = 3, FC[4] = 1, and FC[5] = 1 (or 24311) which represents the permutation P[1] = 4, P[2] = 1, P[3] = 5, P[4] = 3, and P[5] = 2 (or 41532).

Given a permutation array P, the reverse process begins with an empty array FC of size n, then for $1 \le i \le n$ starting with i = 1 and ending with i = n fill in the i^{th} position of FC (from left to right starting at position 1) with the value k – (# of values ≤ i that occur before position i in P) where P[k] contains the value i. (Note that FC[n] will always be 1, so that, we can shorten FC to an n – 1 element array if we wish.) An ordering of a set of 5 elements { e_1, e_2, e_3, e_4, e_5 } based on the forward code (24311) would then be 5-tuple (e_4, e_1, e_5, e_3, e_2).

An encoding of a permutation of the elements can also be represented as an array BC (**backward coding**) where $1 \le BC[k] \le n - k + 1$ for $1 \le k \le n$. In order to decode a permutation code BC to obtain the permutation that it represents, begin with an empty array P of size n, then for $1 \le i \le n$ fill in the $BC[i]^{th}$ empty position (from left to right starting at position 1) of P with the value n – i + 1. Consider an example, with n = 5 and BC[1] = 3, BC[2] = 1, BC[3] = 2, BC[4] = 2, and BC[5] = 1 (or 31221) which represents the permutation P[1] = 4, P[2] = 1, P[3] = 5, P[4] = 3, and P[5] = 2 (or 41532).

Given a permutation array P, the reverse process begins with an empty array BC of size n, then for $1 \le i \le n$ starting with i = 1 and ending with i = n fill in the i^{th} position of BC (from left to right starting at position 1) with the value k – (# of values ≥ i that occur before position i in P) where P[k] contains the value n – i + 1. (Note that BC[n] will always be 1, thus, we can shorten BC to an n – 1 element array if we wish.) An ordering of a set of 5 elements { e_1, e_2, e_3, e_4, e_5 } based on the backward code (31221) would then be 5-tuple (e_4, e_1, e_5, e_3, e_2). The forward and backward codes typically have children that are more similar to their parents. They have high locality.

Next we consider a multiply redundant representation [12] that can be given by concatenating these lists. Thus a **multiply redundant representation** of the permutation 41532 would then be 243113122141532 with forward, backward, and permutation codes concatenated in that order. It is easy to mate and mutate this multiple representation scheme [6, 7, 13, 14], however the resulting list may not reflect the same phenotype in each segment of the multiple genome. In this case we simply choose a best performing segment and repair the entire multiple genome to mirror the best phenome in all of the other redundant segments. Suppose 243113122141532 is a multiple genome for 41532

and 322211431152134 is a multiple genome for 52134.

Mating on positions 010110000110100 i.e.,swap positions 2,4,5,10,11,13 to get the child, 223213122151432 after swapping in those positions.

(Notice in the last segment we get 51432 since positions 11 and 13 now reflect 4 and 5 in the first genome 41532 in the same order as the second genome 52134.)

Assuming the first segment 22321 represents the best phenome 51243, we repair the entire code and get 223211332151243 which is the multiple genome for 51243.

4. Problem Applications

A variety of sequencing applications constrained by time dependent costs will now be presented. These include rooted Hamiltonian directed path, node base communications, and bidding with slack time applications. (In [4, 5] similar applications of less complexity were solved; specifically, if C is monotonic non-decreasing function, there is a sequencing such that the maximum cost element is minimum amongst all sequencings (this problem is called the Minmax Problem and is useful for maintaining low levels of cost outlays per period over time; the Minmax Problem generalizes to Maxmin, Minmin, and Maxmax versions each of which have an efficient algorithmic solution)).

For each application, we will present a mathematical model, a feasible multiple genomic redundant encoding, and a discussion of the evaluation of fitness. Each solution will employ the generic GA discussed previously. These applications not only insure feasibility but also have fitnesses that are a function of a permutation. In fact building genomes based on these codes actually makes the GAs in these time dependent cases *more efficient* [8].

4.1 Maximal Rooted Hamiltonian Directed Paths

We wish to find a maximal rooted Hamiltonian directed path in a complete graph with nodes { 0, 1, 2. …, n-1, n } and root node 0 [10]. An (n+1) by n table of lists is given showing the values of a directed edge from node i to node j at step t where for fixed i, (i, j) is the same at t. The problem is to find a maximal directed path through the graph that begins at node 0 by choosing a directed edge in the path at each step t.

Mathematical Model Given a directed complete graph K = (V,F) with nodes { 0, 1, 2. …, n-1, n } and root node 0 and an (n+1) by n table P of lists showing the values of a directed edge from node i to node j at step t where for fixed i, (i, j) is the same at t. The problem is to find a maximal directed path through the graph that begins at node 0 by choosing a directed edge in the path at each step t.

Feasible Encodings For each of the n! ordered sequences, we can associate a unique code of length $3 \bullet n$.

Evaluation of Fitness The fitness of a population member is evaluated by simply summing the costs of each of the ordered di-edges in sequence. In this application, the fitness is related to the total value so that $T = P(e_1,1) + P(e_2,2) + \ldots + P(e_n, n)$ and thus a population member with a high value has a high fitness.

4.2 Minimal Node Base - A Directed Communications Network Application

A communications network is a digraph D = (N,A) where N, the set of nodes, is thought of as a set of message transmitters and A, the set of arcs, as the allowable directed connections between pairs of nodes. A node base B in a communications network is a minimal subset of nodes with the property that if a message is placed at each node of B, then every node of D will receive the message. Suppose one wants to build a satellite communications receiving station at each node in a node base of a communications network D. Since these receiving stations are built on a node base *every*

node in D can receive all messages. The cost of building such a station is different for each node D and is assumed to vary over time. It is desired to construct these stations one per time period so that the maximum cost of the completed network is minimized.

Mathematical Model A digraph is strongly connected if for every ordered pair of nodes the second can be reached from the first by a direct path. A strong component of D consists of a maximal strongly connected subgraph of D. The condensation of D is a digraph $D^* = (N^*,A^*)$ where N^* is the set of strong components of D and an arc (u,v) is in A^* if there exists an arc in D that joins a node in u to a node in v. The following is an important theorem for node base communication networks [8].

Theorem. The condensation D^* of a digraph D has a unique node base B^* and the node bases of D are those sets B consisting of one node from each of the s strong components of D which is in B^*. Let E be the union of the vertices contained in the strong components of D^* and β the set of node bases of D. The cardinality(β) is the product of the orders of all the strong components of D.

Feasible Encodings Let (m_k) be the number of nodes in the k^{th} strong Component of D. For each of the $(m_1)\bullet(m_2)\bullet\ldots(m_s)\bullet(s!)$ ordered node base sequences, we can associate a unique code of length $(s) + 3\bullet(s) = 4\bullet(s)$.
consisting of a prefix of length s in the form of a node base code $(v_1, v_2, \ldots, v_s)$
where $(m_0 + m_1 + m_2 + \ldots + m_{i-1} + 1) \le v_i \le (m_1 + m_2 + \ldots + m_i)$ for each $1 \le i \le s$ with $m_0 = 0$ for convenience, followed by a standard permutation multiple code of length $3\bullet s$. The first s positions represent the nodes to be used to build the base and the last $3\bullet s$ positions represent the permutation (sequencing) of these nodes.

Evaluation of Fitness The fitness of a population member is evaluated by simply summing the costs of each of the nodes in an ordered base. Moreover, in this application, given an ordering $(v_1, v_2, \ldots, v_s)$, the fitness is inversely related to the total cost $T = C(v_1,1) + C(v_2,2) + \ldots + C(v_s,s)$, that is, a population member with a low cost has a high fitness.

4.3 Minimal Bidding (with Slack Time) Application

A company is to construct a transportation network connecting a group of cities. Due to budget limitations only one link will be contracted for construction each month. Different contractors having different price scales, which vary over time, bid on the construction of different links of the potential network. The cities together with all the possible links can be modeled to form a multigraph H. The goal is to construct the desired network (an underlying subgraph G of the multigraph H with q edges) such that the total cost over time of the entire
construction is minimized. We also assume that it takes one unit of time to construct a link and that the network must be completed in the $t = q + s$ time periods available (q periods for actual construction and s slack periods where no constuction is performed). Dummy (slack) links with zero cost for each period can be included. In this problem, we are searching for an underlying graph G whose q edges are sequenced and whose total cost is minimum.

Mathematical Model Given an undirected multigraph H = (V,F) with no loops whose underlying simple graph G = (V,E) has q edges, F consists of (m_i) multiple edges for each edge e_i in G. Each of $(m_1 + m_2 + \ldots + m_q)$ edges in H can be thought of as a bid

to build edge e_i in G. Let E = all the edges in H and let $\{ B \subseteq H \mid B = \{ b_1, b_2 \ldots, b_q \}$ contains exactly one bid for each edge of G $\}$. Since the network must be completed in t = q + s time periods, s ($s \geq 0$) dummy links with zero cost for each period of t can be included to be used as slack periods where no construction is done (e.g., during winter when costs are prohibitively high). Thus we can expand the definition to include these dummy contractor bids. Thus, we consider $\{ B \subseteq H \cup D \mid$ where $B = \{ b_1, b_2 \ldots, b_{q+s} \}$ contains exactly one bid for each edge of G and also includes dummy (slack) bids $\}$. We also considered giving discounts from bidders that win more than one contract.

Feasible Encodings For each of the $(m_1)\bullet(m_2)\bullet\ldots(m_{q+s})\bullet((q+s)!)$ ordered bid sequences, we can associate a unique code of length $(q + s) + 3\bullet(q + s) = 4\bullet(q + s)$. consisting of a prefix of length q in the form of a bid code $(b_1, b_2, \ldots, b_{q+s})$ where $(m_0 + m_1 + m_2 + \ldots + m_{i-1} + 1) \leq b_i \leq (m_1 + m_2 + \ldots + m_i)$ for each $1 \leq i \leq (q + s)$ with $m_0 = 0$ for convenience, followed by a permutation code of length $3\bullet(q + s)$. The first (q + s) positions represent the bids to be used to build the edges of G and the last $3\bullet(q + s)$ positions represent the permutation or time sequencing of these bids.

Evaluation of Fitness The fitness of a population member is evaluated by simply summing the costs of each of the bids in order. That is, in this application given $(b_1, b_2, \ldots, b_{q+s})$, the fitness is inversely related to the total cost including any discounts $T = C(b_1,1) + C(b_2,2) + \ldots + C(b_{q+s},q+s) - D$, so that a population member with a low cost has a high fitness.

5. Results

The multiple genomic redundant representation **using all three segments was most efficient**. We experimented with all seven possible combinations, that is, forward only, backward only, permutation only, forward&backward, forward&permutation backward&permutation and finally all three together forward&backward&permutation. The experiments using all three were consistently most efficient even though more overhead was generated.

The multiple genomic redundant representation using all three segments was also examined as to which representations dominated at various stages of the search. The **permutation code was used extensively in the earlier generations** when the GA was searching more globally since this coding scheme does not have a good locality property. **In the later stages the GA adapted its search using mostly the forward and backward codes** which have a stronger locality property.

7. Conclusions

We considered using multiple genomic redundant representations in a self-adapting genetic algorithm to solve rooted Hamiltonian directed path, node base communications, and bidding optimization problems whereby the element costs are not fixed, but are time dependent. First we showed that these problems are NP hard. We then demonstrated that using multiple genomic redundant representations to create a self-adapting genetic algorithm by employing various codes with different locality properties that the genetic algorithm's efficiency was improved. The GAs solving this class of NP

hard problems, where costs are not fixed but are time dependent, employ non-locality or locality representations when appropriate since the GA adapts to its current search needs making the GA more efficient.

8. Ackowledgements

We wish to thank Pace University's School of Computer Science and Information Systems for partially supporting this research. We also wish to thank Dr. Fred Grossman and Dr. Joseph Malerba for their constructive comments on this research.

References

1. M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, (2001).
2. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, (1989).
3. L. Davis, Handbook of Genetic Algorithms, Van Nostrand Reinhold, (1991).
4. M. L. Gargano and S. C. Friederich, On Constructing a Spanning Tree with Optimal Sequencing, Congressus Numerantium 71, (1990) pp. 67-72.
5. M. L. Gargano, L. V. Quintas and S. C. Friederich, Matroid Bases with Optimal Sequencing, Congressus Numerantium 82, (1991) pp. 65-77.
6. M. L. Gargano and W. Edelson, A Genetic Algorithm Approach to Solving the Archaeology Seriation Problem, Congressus Numerantium 119, (1996) pp. 193-203.
7. W. Edelson and M. L. Gargano, Minimal Edge-Ordered Spanning Trees Solved By a Genetic Algorithm with Feasible Search Space, Congressus Numerantium 135, (1998) pp. 37-45.
8. F. S. Roberts, Discrete Mathematical Models, Prentice-Hall Inc., (1970).
9. F. S. Hillier and G. J. Lieberman, Introduction to Operations Research, Holden-Day Inc. (1968).
10. K. H. Rosen, Discrete Mathematics and Its Applications, Fourth Edition, Random House (1998).
11. Z. Michalewicz, Heuristics for Evolutionary Computational Techniques, Journal of Heuristics, vol. 1, no. 2, (1996) pp. 596-597.
12. F. Rothlauf and D.E.Goldberg,Redundant Representations in Evolutionary Computation, Evolutionary Computation, vol. 11, no. 4, 2003 pp.381-416.
13. J. DeCicco, M.L. Gargano, W. Edelson, A Minimal Bidding Application (with slack times) Solved by a Genetic Algorithm Where Element Costs Are Time Dependent, GECCO, (2002).
14. M.L. Gargano, W. Edelson, Optimally Sequenced Matroid Bases Solved By A Genetic Algorithm with Feasible Search Space Including a Variety of Applications, Congressus Numerantium 150, (2001) pp. 5-14.