A Genetic Algorithm for Unit Selection based Speech Synthesis

Rohit Kumar

Language Technologies Research Center
International Institute of Information Technology, Hyderabad INDIA
rohit @ iiit.net

Abstract

We describe the use of a Genetic Algorithm (GA) for the Unit Selection problem, which is essentially a search/optimization problem. The various operators for the GA have been defined and comparison with optimization reached by hill climbing approaches is presented.

1. Introduction

Speech Synthesizers in current state of art text to speech systems [1] are based on Data Driven Concatenative Synthesis. Required sequence of basic units is synthesized by selecting appropriate unit instances from a huge database consisting of multiple instances of basic units with varying prosodic properties. The crux of this approach is the unit selection problem which involves selecting the appropriate instance of required units from the database.

1.1. The Unit Selection Problem

Unit Selection involves finding the best sequence of unit instances which is closely matching with a given target specification of the required unit sequence in terms of features. The best instance is decided by minimizing target cost between features specification of desired unit and available instances of the unit and the joining costs between the selected instances in the sequence of units [2].

In order to synthesize a sequence of 15 units (nearly 5 words) by selecting units from a typical database having on an average 10 occurrences of each basic unit, we have 10^{15} possible sequences. As we move towards bigger and bigger databases for natural sounding and emotional synthesis, efficient algorithms for unit selection need to be explored.

Essentially the Unit Selection problem is a heuristic search problem involving optimization of selection costs of each unit across the sequence of units to reach a minimum. We can either reach the optimization by doing a local optimization or by attempting a global optimization. A deeper analysis of the nature of the Unit Selection problem can let us understand which approach is better in reaching optimal results efficiently.

We have experimented with local optimization in [3]. Global optimization is a search problem in very large high dimensionality search space. Enumerative/Brute force search is not suggested considering that Speech Synthesis would often want a real time speed performance. Genetic Algorithms (GA) have been widely successful for solving global optimization problems in huge search spaces [4]. Through this paper, we describe a GA implemented by us for the Unit Selection problem.

Before we proceed, we look at some of the unit selection algorithms being used in popular systems. [2] and [5] describes a unit clustering and a pruned viterbi search based unit selection that chooses a best path through a state transition network so as to minimize unit distortion (i.e. target cost) as well as concatenation distortion (i.e. join cost). This algorithm is used in Festival and CHATR systems. A 3-Tier Non-Uniform Unit Selection Algorithm used in the Microsoft China Mandarin TTS is described in [6] which reduces the choices for each unit in the sequence at each tier based on feature distances and concatenation costs.

Further this paper is organized as follows: Section II introduces the basic Genetic Algorithm. In section III we describe the implementation of the various genetic operators for the Unit Selection problem. Section IV presents the perceptibility comparison between local optimization approach and the GA approach.

2. A Genetic Algorithm

The basic Genetic Algorithm searches in the large search space by searching in parallel at multiple locations in the search space rather than at just one location. It does this by producing a population of various possible solutions distributed throughout the search space and then operates on these solutions using basic operations to create newer generation of better solutions. The operators correspond closely with the operations of nature as pertains to survival of species [7]. A basic GA is described ahead [8].

[Step 1]. An initial population consisting of several possible solutions in the search space is created. The issues to be addressed at this step are the size of the initial population and the criteria for choosing the initial population so as to distribute the search at the various locations in the search space.

[Step 2]. The fitness (or mis-fitness) for each solution in the current population is evaluated using a fitness function. The fitness function is a measure of the suitability of the solution to survive into the next generation.

[Step 3]. A new generation of the population is created by applying the genetic operators of selection, crossover, mutation and elitism on individuals of the current generation.

Selection operation involves survival of some of the individuals of the current populations into the next generation. The probability of a element's selection is dependent upon its fitness.

Crossover operation combines 2 individuals of the current generation (chosen with equal probability) into offspring solutions for the next generation.

Mutation operation alters some of the individuals at some points to create new solutions and is a mechanism to

extend search to unexplored domains in the search space.

Elitism ensures that the fittest individuals of each generation always propagate into next generation.

[Step 4]. The Steps 2 and 3 are repeated for several generations to allow evolution of very stable and fit population (solutions). Generally some termination criteria are used to stop the evolution after some iterations of this loop. The so resulting best solutions are then used appropriately.

Use of a GA for Speech Synthesis is advantageous as we can terminate evolution at any step and pick the best solution. Also as size of databases grow, GA based algorithm could be more and more efficient than local optimization approaches.

3. Genetic Algorithm for Unit Selection

Before we proceed to describe the various parameters and operations we have used at each step of the unit selection GA, brief description of the speech database is given.

3.1. Unit Selection Speech Database

The speech database we are using is developed from the Hindi databases described in [3]. It consists of basic units of varying sizes at syllable and phone levels. Each instance of the units is stored along with prosodic and linguistic features like pitch, duration, energy, phonetic context and syllable position in the word.

The instances of each unit are further stored in the increasing order of their global prosodic mismatch function (GPMF) [9] value. The GPMF is an objective function that we use to describe the suitability of an instance of a unit in the most probable situations the unit might be used.

3.2. The various steps of GA for Unit Selection

The various steps of a basic GA for Unit Selection problem are described. It must be mentioned that we are not doing any mutation in the population currently.

Given the sequence of basic units (syllables and phones) to be synthesized, the solution consists of the sequence of appropriate instances of each. As each individual in the population is a possible solution, it is implemented as a list of instances of units in the desired sequence. The representation of each individual in the population is shown in figure 1.

Instance of U^1	Instance of U^2	Instance of U^3	…	Instance of U^{n-1}	Instance of U^n

Figure 1. Representation of each Individual in the population for synthesis of a *n* - Unit Sequence

3.2.1. Creating the Initial Population

The size of the population is taken to be a multiple of number of instances of the unit that has the maximum number of instances in the database of all the units in the sequence to be synthesized. The maximum number of instances of a unit in the database is restricted to a small number (100) while creation of the database [9]. Also a lower threshold on the population size (30) is kept.

The initial population is created by initializing the required number of individuals with instances of units as per the sequence to be synthesized. This population initialization is done so as to distribute the individuals throughout the search space. We see each unit required in the sequence as a dimension in the search space with *n* - discrete possible choices on it.

If we have in all *m* individuals in the population and a particular unit has *n* instances, then forming a circular list of the *n* instances, every $\frac{n}{m}$ *th* (rounded to the nearest greater integer) is filled into the *m* individual while rotating in the list of instances. Following similarly for all the units we come up with an initial population.

Here we are trying to create a population which has the instance of the units evenly distributed among the individuals. The combination and permutations amongst the instances will happen as we proceed through the generations to evolve better solutions.

3.2.2. Prosodic Mismatch as measure of unfitness

Each individual in the population is evaluated using a measure of its unsuitability for use as the required sequence. The measure incorporates the unsuitability due to distortion between adjacent instances in the selected sequence. Let the sequence of units in the *i* th individual be represented by

$$U_i^1 \ , \ U_i^2 \ , \ U_i^3 \ .... \ U_i^k \ , \ U_i^{k+1} \ , ... \ U_i^{n-1} \ , \ U_i^n$$

The unfitness of the *i* th individual is calculated as *Score(i)*.

$$Score(i) = \sum_{k=1}^{n} \left(Mismatch\left(U_i^{k-1}, U_i^k\right) + Mismatch\left(U_i^k, U_i^{k+1}\right)\right)$$

…… Equation (1)

Mismatch function measures the mismatch between 2 units. It is measured as the weighted sum of the feature mismatch between the 2 units. Currently we are using prosodic features (pitch, duration and energy) and linguistic features (neighboring phonemes and nuclei of neighboring syllables). Mismatch of Linguistic features is Boolean i.e. 0 (matching) or 1(not matching). So mismatch is calculated as

$$Mismatch\ \left(U_i^x, U_i^y\right) = \left\{ \begin{matrix} PMF\ \left(U_i^x, U_i^y\right) + \\ PM\ \left(U_i^x, U_i^y\right) + NM\ \left(U_i^x, U_i^y\right) \end{matrix} \right\}$$

…… Equation (2)

PMF is the prosodic mismatch function as described in [3]. We have used the following weights for the each prosodic feature. W_{Pitch} = 2.5, $W_{Duration}$ = 2.5, W_{Energy} = 1.0. PM and NM are Boolean function to compare linguistic features.

P M (A, B) = 0, if last phone of A is same as first phone of B
= $W_{PhonemeMatch}$, otherwise

N M (A, B) = 0, if nuclei of A is same as nuclei of B
= $W_{NucleiMatch}$, otherwise

We have set $W_{PhonemeMatch}$ = 3.0 and $W_{NuclieMatch}$ = 2.0. The score of each individual in the population is calculated and the individuals are ranked such that the individual with minimum score has the best rank as it is the most suitable solution currently in the population.

3.2.3. *Evolution of better solutions*

Now the next generation is created by applying the genetic operations on the population. An elite section of population is transferred to the next generation without modification. The size of elite section is set to 10% of the population size with a minimum threshold of 5.

The selection operator selects individuals from the current population and passes them to the next generation without modification. The individual with best rank is most probable to be selected. If ranks are such that bigger ranks are better, a roulette wheel selection is used. The section of circumference on the wheel for each individual is proportional to $rank^p$, such that p = 2 to 3 typically. Hence an individual with rank 10 is a hundred or a thousand times more probable to be selected than an individual with rank 1.

Crossover operation involves selection of 2 individuals from the population and creation of a new individual which is a result of the combination of the 2 original individuals. The crossover operation we have combines individuals A and B of the population consisting of sequence of units

$$U_A^1,\ U_A^2,\ldots.\ U_A^k,\ U_A^{k+1},\ldots U_A^{n-1},\ U_A^n$$

and

$$U_B^1,\ U_B^2,\ldots.\ U_B^k,\ U_B^{k+1},\ldots U_B^{n-1},\ U_B^n$$

respectively. They combine to create an individual

$$U_{AB}^1,\ U_{AB}^2,\ldots U_{AB}^k,\ U_{AB}^{k+1},\ldots U_{AB}^{n-1},\ U_{AB}^n$$

It must be noted that U_A^k and U_B^k are instances of the same unit. Let us break up the recombination problem and suppose that we have already selected the first *k - 1* units in the sequence, and we must choose amongst U_A^k and U_B^k for the k^{th} element of the combined sequence. We calculate score of both the choices as

$$Score\left(U_A^k\right)=\begin{bmatrix} Mismatch(U_{AB}^{k-1},U_A^k)+\\ Mismatch(U_A^k,U_A^{k+1})+\\ Mismatch(U_A^k,U_B^{k+1}) \end{bmatrix} \quad \ldots Equation\ (3)$$

The unit with lesser score is chosen for U_{AB}^k. For the first unit the first component in the scoring formula is ignored and for the last component the later 2 components are ignored.

Our implementation of Selection and Crossover is complimentary such that Probability of Selection plus Probability of Crossover is unity. We select 2 individuals using the selection operation and then randomly generate a number between 0 and 100. If the number overshoots $P_{Crossover}$ we select the better of the individuals and pass it to the next generation. Otherwise we cross the 2 individuals and pass the resultant into the next generation. This is repeated to regenerate the next generation of size equal to the initial population size. As mentioned earlier, no mutation is applied.

3.2.4. *Termination Criteria*

The evolution of newer solutions continues iteratively till one of the termination criteria is met. Evolution terminates if the change in fitness of the best individual in the population is less than 0.1% in consecutive generations subject to condition that atleast 5 generations have evolved. Atmost 50 generations are allowed to evolve. We have observed that this second criteria never comes into play during synthesis of meaningful short sentences.

It must be mentioned that we synthesize the utterance phrase by phrase limiting the maximum length of a sequence going to be synthesized.

Fig 2 shows the plot of fitness of best individual in the population as we progress through an evolution during the synthesis of a sequence of 57 basic units. Also shown is the plot of sequence mismatch covariance between best and worst solutions in each generation. The purpose is to show the justification for the termination criterion that has come into play after 14 evolution cycles. We can observe that the termination has happened at the right point when the best solution seems to be stabilized and not evolving any further.

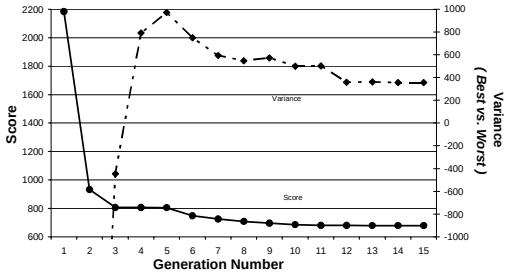

Fig 2. Plot of Score of Best Individual and Variance between Best and Worst Individual as Evolution happen

Figure 3 shows the plots of join-to-join mismatch sequence for the best 200 individuals (out of a population of 300) in the 1st, 7th and 14th generations. We can observe that as we progress through evolutions the randomness in the population converges towards the optimal solution, which shows the effectiveness of the crossover and other operations.

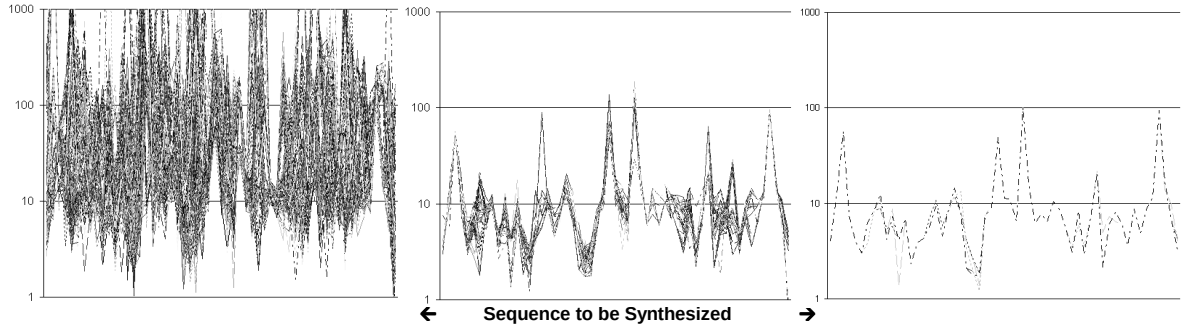

Figure 3. Join-to-Join Mismatch Sequence for Best 200 Individuals in 1st, 7th and 14th Generations (Left to Right)

3.3. Effect of various parameters

In order to explore the optimal set of parameter values for the Genetic Algorithm, we have conducted some experiments.

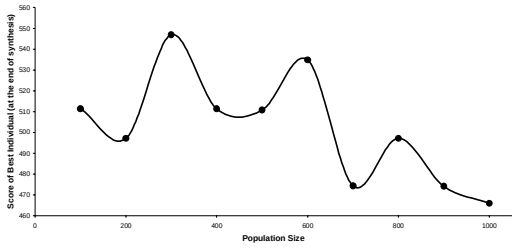

Fig 4. Plot of Score of Best Individual after termination of evolution for different Population Sizes

Figure 4 shows the plot of score of best individual in the population at the end of synthesis for a typical sentence against various values of Population Size. Broadly, the score will be lesser (i.e. better) as the size of population increases, as more and more areas of search space will be explored. But it also depends on the algorithm used for creating the initial population. There is scope for improvement here. Further, higher size of population amounts to more computation time, which is undesirable.

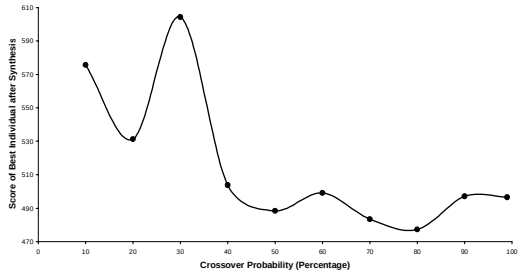

Fig 5. Plot of Score of Best Individual after termination of evolution for different Crossover Probabilities

Figure 5 shows the same plot for different values of crossover probabilities. We can observe that high probability of crossover is better. $P_{Crossover}$=0.8 should be optimal.

4. Evaluation and Comparison

Table1: Comparison of GA and Local Optimization based Synthesizers

Sentence Number	GA based synthesizer		Local Optimization based synthesizer	
	Score	**Variance**	**Score**	**Variance**
1	2.500	0.5714	3.000	0.5714
2	2.250	0.2143	2.875	0.6964
3	3.875	0.4107	3.500	2.0000
4	3.000	0.2857	3.500	0.8571
5	2.250	0.5000	2.625	1.4107
6	3.500	0.5714	3.875	0.9821
7	3.250	0.2143	2.750	1.3571
8	3.375	0.5535	3.125	0.6964
9	2.875	1.5535	3.125	1.5536
10	3.250	0.5000	2.875	1.2679
Average	**3.013**	**0.5375**	**3.125**	**1.1392**

The GA based Hindi Unit Selection Speech Synthesizer was evaluated and compared with the Local Optimization based Speech Synthesizer described in [4]. 8 native Hindi speaking subjects (2 females and 6 males) evaluated a set of 10 sentences synthesized individually from both the synthesizers by grading the output on a scale of 0 (worst) to 5 (Best). The results of the evaluation are shown in table 1. The scores (averaged across 8 subjects) and variance of each sentence for both synthesizers is shown.

We can observe that the GA based Unit Selection synthesizer gets an overall score less than the Local Optimization based Unit Selection synthesizer. We can also see that the scores are very close to each other and further that the scores of GA is more consistent (low variance) as compared to the hill climbing based synthesizer.

5. Conclusion

An Evolutionary Unit Selection Algorithm is proposed and various details of its implementation are mentioned. Several parameters for the GA are experimented with and termination criteria are justified. On comparison with Local Optimization, we observe that GA scores less but very close with more consistency. Further there is scope of improvement in the GA based Unit Selection during the creation of Initial Population to cover wider search space. GA based Unit Selection can offer faster results as size of Unit Selection databases grows.

Acknowledgement

I would like to acknowledge the contributions of Prof. Rajeev Sangal and Mr. S. P. Kishore for their inputs and discussions from time to time. Also I am thankful to Volunteers for perceptual tests for their contribution.

6. References

[1] AT & T Natural Voices™ Text to Speech System, http://www.naturalvoices.att.com/

[2] Alan W. Black and Nick Campbell, "Optimizing selection of Units from Speech Databases for Concatenative Synthesis", In *Proceedings of Eurospeech 95, vol 1., pp. 581 – 584, Madrid, Spain, 1995*

[3] S P Kishore, Rohit Kumar and Rajeev Sangal, "A Data Driven Synthesis Approach For Indian Languages using Syllables as Basic Unit", *in Proceedings of Intl. Conf. on NLP (ICON) 2002, pp. 311-316, Mumbai, India, 200*

[4] Lawrence Davis ed., "Handbook of Genetic Algorithms", *Van Nostrand Reinhold, New York, 1991*

[5] Alistair Conkie, "A robust unit selection system for speech synthesis", i*n Joint Meeting of ASA/EAA/DAGA, Berlin, Germany, March 1999*

[6] Min Chu, Hu Peng, Hong-yun Yang, Eric Chang, "Selecting Non-Uniform Units from a very large corpus for Concatenative Speech Synthesizer", *in Proceedings of ICASSP, Salt Lake City, 2001*

[7] David E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", *Addison Wesley Publishing Company, 1989*

[8] Xavier Hue, "Genetic Algorithms for Optimization", *Edinburgh, 1997*

[9] Rohit Kumar, S. P. Kishore, "Automatic Pruning of Unit Selection Speech Databases for Synthesis without loss of Naturalness", *submitted to ICSLP, Jeju Island, Korea, 2004*