The Proportional Genetic Algorithm Representation

Annie S. Wu
School of EECS
University of Central Florida
Orlando, FL 32816-2362
aswu@cs.ucf.edu

Ivan Garibay
School of EECS
University of Central Florida
Orlando, FL 32816-2362
igaribay@cs.ucf.edu

1 Introduction

We have developed a genetic algorithm (GA) with a new representation method which we call the proportional GA (PGA). The PGA representation focuses on the idea that it is the content rather than the order of the encoded information that matters. As a result, the PGA representation is based on multisets rather than permutations. We extend the idea of location independent representations to a generally usable encoding for integer and floating point numbers. Specifically, the PGA assigns one or more unique characters to each parameter or component of a solution. The value of a parameter is determined from the proportion of the characters assigned to that parameter as compared with the total number of characters in the individual, or from the relative proportions of the assigned characters of that parameter. Thus, characters that exist are "expressed" and, consequently, interact with other expressed characters. Characters that do not exist are "not expressed" and simply do not participate in the interactions of the expressed characters. This representation may be further extended with the addition of *non-coding* characters that are not associated with any parameters. These non-coding regions are beneficial for fine tuning purposes.

We tested three variations of the PGA in our experiments. PGA1 is specialized for resource allocation problems. PGA2 and PGA3 may be applied to the general class of problems in which one is searching for a vector of values. Length restrictions limit our discussion here to PGA2 and PGA3. Full details about the PGA are available in [1].

2 Representation details

A consequence of the PGA's set-based representation is that the two following example individuals encode the same information.

```
AccBdDeeEbAbBDEccaAAAEebbEEECCDbbbABCDEedcbaAAddbA
AAAAAAAAAaaBBBbbbbbbbbbCCCcccccDDDDddddEEEEEEEeeee
```

The following two tables show how PGA2 and PGA3, respectively, would decode the example individuals into expressed values. In both cases,
Expressed value $= V_{i,min} + pct(V_i) \times (V_{i,max} - V_{i,min})$.
V_{min} and V_{max} are predefined constants.

Value V	V_{min}	V_{max}	# *positive _char*(V)	# *negative _char*(V)	*pct*(V)	Expresssed value
V_1	0	10	9 A's	2 a's	$9/(9+2)$	8.18
V_2	0	10	3 B's	9 b's	$3/(3+9)$	2.5
V_3	0	10	3 C's	5 c's	$3/(3+5)$	3.75
V_4	0	10	4 D's	4 d's	$4/(4+4)$	5.0
V_5	0	10	7 E's	4 e's	$7/(7+4)$	6.36

PGA2: Character assignment and expressed values. $pct(V_i) = \frac{positive\_char(V_i)}{positive\_char(V_i)+negative\_char(V_i)}$

Value V	V_{min}	V_{max}	# *positive _char*(V)	# *negative _char*(V)	*pct*(V)	Expresssed value
V_1	0	10	9 A's	2 a's	2/9	2.22
V_2	0	10	3 B's	9 b's	3/9	3.33
V_3	0	10	3 C's	5 c's	3/5	6.0
V_4	0	10	4 D's	4 d's	4/4	1.0
V_5	0	10	7 E's	4 e's	4/7	5.71

PGA3: Character assignment and expressed values.

$$pct(V_i) = \begin{cases} \frac{positive\_char(V_i)}{negative\_char(V_i)} & \text{if } positive\_char(V_i) < negative\_char(V_i) \\ \frac{negative\_char(V_i)}{positive\_char(V_i)} & \text{otherwise} \end{cases}$$

References

[1] A. S. Wu and I. Garibay. The proportional genetic algorithm: Gene expression in a genetic algorithm. *Genetic Programming and Evolvable Hardware*, 2002. In press.