Evolving modular behaviours

CSCI6506 Sandbox 3(a)

Assignment due date: 10th February, 2015

Task definition

- We are now going to try to introduce a mechanism for evolving 'teams' of programs without loosing any of the functionality we have gained during Sandbox 2. Success in this respect will enable GP to decompose a task into a set of program modules without having to define the number of modules a priori.
- To reach this goal we will introduce two concepts:
  1. Each GP program will specify a discrete 'action', a and a program p; hereafter BidGP. From an implementation perspective one approach would be to define the first instruction of any individual to be an integer over the range of integers associated with the task's actions. Thus, for a classification task consisting of 3 classes, then the legal set of actions from which an action can be selected are $a \in \{1, 2, 3\}$. Each individual within the BidGP population can only have a *single* action associated with it. The role of a BidGP program is to identify the circumstances under which the BidGP individual's action is deployed. This will be resolved within the context of a process of competitive bidding (described below) relative to some subset of other BidGP individuals or the 'team'.
  2. We now need some concept of how to define a 'team' of BidGP individuals such that both the number of BidGP individuals per team and the mix of individuals are identified during evolution. To do so, we make the following observations. The minimal criteria for a legal team are:
     (a) there must be at least two BidGP individuals;
     (b) the same BidGP individual can only appear within the same team once, but may appear in more than one team;
     (c) across the set of BidGP individuals associated with the same team, there must be at least two different actions present.
- With this in mind, we define an independent GA population to specify which BidGP individuals appear in each 'team' (Figure 1). That is to say, each individual from such a **team population** specifies a unique combination of BidGP individuals forming a team. Fitness will only be evaluated at the level of the team. ***Each individual in the team***

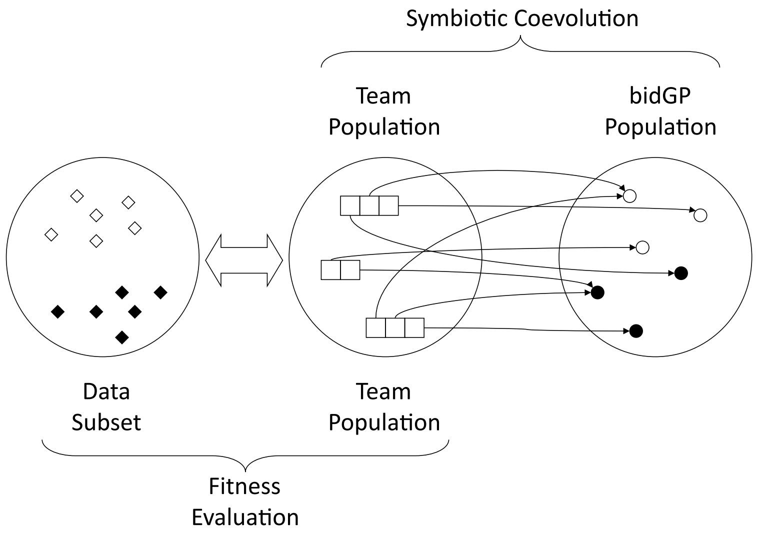

Figure 1: Basic structure for coevolving team composition and team content under a 'bidding' formulation.

population is merely a set of indexes to individuals in the BidGP population. Consider team h_i from the team population. Fitness evaluation w.r.t. exemplar p_k from the data subset now has the following form:

1. Let bGP_j denote a BidGP individual; the set of BidGP individuals from the same team is denoted $bGP_j \in h_i$
2. $\forall bGP_j \in h_i$
   (a) Evaluate bGP_j on exemplar p_k, resulting in some numerical value in the output register of the BidGP individual. Let us refer to this as $bGP_j.(out)$
3. Identify the BidGP individual from team h_i that has the largest output and record it's index, or

$$j^* = \arg(\max_{bGP_j \in h_i} (bGP_j.(out))) \tag{1}$$

   j^* defines the 'winning' BidGP individual within team bGP_j for exemplar p_k. BidGP individual bGP_{j^*} has therefore 'won' the right to suggest it's action, $bGP_{j^*}.(a)$, where this is then compared against the label specified for exemplar p_k and the outcome of this directs the estimation of fitness.

- Naturally, the above process needs repeating for all exemplars within the data subset in order to define the fitness for team h_i. The entire process being repeated for all teams.
- In the case of a generational model for the **selection operator**, the following process is recommended:

1. Identify the worst Gap% team individuals and delete them;

2. Test for any BidGP individuals that are not used by the remaining $(100 - Gap)\%$ teams and delete them;
3. Create $Gap\%$ new team individuals using the variation operators (example definition below).

- Initialization of team and BidGP populations assumes the following form:

  1. Initialize a BidGP population of size $2 \times H$ in which each action are equally probable (program initialization follows the same heuristic as you have previously assumed).
  2. Initialize a team population of size H with each team, h_i, consisting of 2 to ω team members (such that the above conditions on actions per team are satisfied). Naturally, from an implementation perspective, assuming a fixed length representation would imply that all individuals consist of ω genes. In the case of individuals with less than ω team members, it would then be necessary to 'pad' the genome with integers that do not represent a legal BidGP index (say, an index of -1).

- **Variation operators** need to maintain diversity in the team and BidGP populations. It is suggested that, given a fixed size population for team and BidGP populations, a set of mutation operators are applied to create $Gap\%$ new teams at each generation by,

  1. Choosing a team currently in the team population to clone (say with uniform p.d.f.);
  2. Identify one or more BidGP individuals to remove and / or add (from the team) while enforcing the criteria for a legal team (see first bullet point);

- If BidGP population size $< 2 \times H$, then you can also consider the case of adding new individuals to the BidGP population. Such a process would begin by cloning a BidGP individual that one of the *new* $Gap\%$ teams indexes. Only after cloning a BidGP individual and updating the team pointer to index the cloned BidGP individual would you then apply your mutation operators.

- An overview of the entire cycle of your algorithm would have the following form:

```
1. Initialize BidGP population with $2 \times H$ individuals;
2. Initialize Team population with $H$ individuals;;
3. FOR $(g = 0; g! = $ STOP$; g++)$
   (a) Sample $\tau$ training exemplars;[1]
   (b) FOR $(k = 0; k < \tau; k++)$
        i. FOR $(i = 0; i < H; i++)$
           A. Evaluate $(h_i, p_k)$;
       ii. Update FITNESS $(h_i)$;
   (c) Rank All teams
   (d) DELETE worst $H \times Gap$ hosts;
```

[1]Using preferred approach from Sandbox 2.

   (e) DELETE any BidGP without a host reference;[2]
   (f) ADD BidGP individuals to make population back up to $2 \times H$;
   (g) ADD $H \times Gap$ new hosts, making use of new BidGP individuals;[3]

- Further details for this style of process for evolving teams of programs can be found in [1]. Note, however, the variant outlined above represents a subset of the capabilities described in the reference work.

Reporting

- Benchmark your resulting teaming formulation for GP using the two data sets you experimented with in Sandbox 2.

- Write a short 2 page summary answering the following questions:

  - How does the classification performance compare to the GP solution from Sandbox 2 (assuming the same process for identifying a champion classifier for deploying under test conditions)?
  - How many BidGP individuals are used to express a solution in your champion individual?
  - Is there just one BidGP individual per action or do you find that for some classes more BidGP individuals are utilized?
  - How many input attributes does a BidGP individual employ? Is GP able to associate specific attributes with specific actions?
  - What can you say about the types of classes that teams learn to classify first during evolution?
  - Identify 2 other approaches for constructing teams of programs. What are their respective advantages / disadvantages compared to the framework adopted in this sandbox?

- Your report should be emailed to `mheywood@cs.dal.ca` before midnight on the sandbox due date.

References

[1] P. Lichodzijewski and M. I. Heywood. Symbiosis, complexification and simplicity under GP. In *ACM Proceedings of the Genetic and Evolutionary Computation Conference*, pages 853–860, 2010. `http://dl.acm.org/citation.cfm?doid=1830483.1830640`.

[2]Count the number of deleted BidGP individuals.

[3]A new host should not consist of BidGP individuals created at step 3.(f) alone. See comments regarding **variation operators**.