Neurocomputing 154 (2015) 200–207

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A geometric semantic genetic programming system for the electoral redistricting problem

Mauro Castelli *, Roberto Henriques, Leonardo Vanneschi

ISEGI, Universidade Nova de Lisboa. 1070-312 Lisboa, Portugal

ARTICLE INFO

Article history:
Received 24 October 2013
Received in revised form
30 October 2014
Accepted 2 December 2014
Communicated by M. Bianchini
Available online 15 December 2014

Keywords:
Electoral redistricting
Genetic Programming
Semantics
Search space

ABSTRACT

Redistricting consists in dividing a geographic space or region of spatial units into smaller subregions or districts. In this paper, a Genetic Programming framework that addresses the electoral redistricting problem is proposed. The method uses new genetic operators, called geometric semantic genetic operators, that employ semantic information directly in the evolutionary search process with the objective of improving its optimization ability. The system is compared to several different redistricting techniques, including evolutionary and non-evolutionary methods. The simulations were made on ten real data-sets and, even though the studied problem does not belong to the classes of problems for which geometric semantic operators induce a unimodal fitness landscape, the results we present demonstrate the effectiveness of the proposed technique.

1. Introduction

The zone design problem (also known as redistricting) is the process of dividing a geographic space or region of spatial units into smaller subregions or districts. Probably, the most well-known instance of the zone design problem is the electoral redistricting problem. As reported by Bação et al. [1], electoral redistricting consists in the partitioning of areal units, generally administrative units, into a predetermined number of zones (districts) such that the units in each zone are contiguous, each zone is geographically compact and the sum of the populations of the areal units in any district is as similar as possible in all the districts or lies within a predetermined range [2]. Because of the spatial nature involved in constraints, redistricting is usually seen as a type of spatial clustering. Due to its NP-completeness, the electoral redistricting problem is considered a complex problem, and heuristic techniques seem to provide the best solutions. In this paper, we propose the use of Genetic Programming (GP) [5] to address the electoral redistricting problem. In particular, we use recently defined genetic operators, called geometric semantic genetic operators [11], that allow us to integrate semantic awareness in the evolutionary process.

One of the strongest points of geometric semantic operators [11] is that, using semantic information, they are able to induce, by construction, a unimodal fitness landscape on all problems consisting in matching sets of input data into known targets (like supervised learning problems, such as regression or classification). Geometric semantic operators have been used so far on many different symbolic regression problems (including several complex real-life applications [8,19–21]), generally with excellent results, and the fact that the fitness landscape is unimodal is often used as an argument to justify those results [8,19–21]. In this paper, for the first time, we apply geometric semantic operators to a problem that does *not* belong to that class: the objective of the application studied here, in fact, is *not* matching sets of input data into known targets. Thus, nothing can make us believe that the fitness landscape induced by those operators is unimodal in our case. Interestingly, the results we present demonstrate that geometric semantic operators have a beneficial effect on the search process also in this case. This fact hints that geometric semantic operators may be useful not only in single-objective supervised learning problems, but also in a larger class of problems, and possibly the justification for their effectiveness goes beyond the fact that they may induce unimodal fitness landscapes. This issue opens the door to future investigation and deserves to be deepened in the future.

The paper is organized as follows: Section 2 describes the redistricting problem and its constraints; Section 3 presents the standard GP algorithm and the canonical (syntax-based) genetic operators; Section 4 defines the geometric semantic operators used in this paper. Section 5 describes the representation of the candidate solutions, the fitness function and how the geometric semantic operators have been used in this work. Section 6 presents the experimental settings and the obtained results. Here, a comparison between the proposed framework and several existing redistricting

* Corresponding author. Tel.: +35 1213828610; fax: +35 1213872140
E-mail addresses: mcastelli@isegi.unl.pt (M. Castelli), roberto@isegi.unl.pt (R. Henriques), lvanneschi@isegi.unl.pt (L. Vanneschi).

http://dx.doi.org/10.1016/j.neucom.2014.12.003
0925-2312/

techniques is also presented. Finally, Section 7 concludes the paper and suggests ideas for possible future research.

2. Redistricting problem: Definition and constraints

In redistricting problems, the aim is to aggregate n geo-spatial regions into c partitions (or districts) subject to some constraints. The most well-known application of the redistricting problem is the electoral redistricting problem. Here, the objective is to create districts usually by grouping smaller administrative units.

The constraints that define a "good" electoral redistricting plan are as follows: 1. All the districts should be equal in population, 2. Each district should be a single continuous territory, 3. Districts should be compact.

The first constraint is known as population equality and it is particularly important for an electoral redistricting plan: since each district elects the same number of assembly members, they should have approximately the same number of voters. To evaluate the population equality, we use the sum of absolute deviations of the districts populations from the average (or ideal) population of a district:

$$\sum_{j=1}^{c} |p_j - \mu| \tag{1}$$

where p_j is the population of district j, c is the total number of districts (or clusters) and μ is the average of population of all districts.

The second constraint is the spatial contiguity constraint. A district of regions is spatially contiguous when every region within the cluster shares at least a part of its boundary with at least one other region within the cluster and the number of its connected components is equal to 1. Solutions that do not satisfy this constraint are not considered as acceptable solutions by our proposed algorithm.

The third constraint is known as compactness. Compactness is an attribute of the shape of the cluster. We define a compact cluster as a cluster that has a shape very close to that of a simple geometric shape. Examples of simple geometric shapes are circle, rectangle, and square. To evaluate the compactness of the obtained clusters we consider the following compactness measure:

$$\text{radial compactness} = \sum_{j=1}^{c} \sum_{i \in C_j} d_{ij} \tag{2}$$

where d_{ij} is the Euclidean distance between the ith region and its jth district center, while C_j denotes the cluster j.

The problem statement is to divide a geographic area that consists of n regions into c districts such that the total population within each district is as equal as possible. Each of these c districts must be spatially contiguous. Finally, all of the k districts must be as compact as possible.

While the previous constraints are related to the electoral redistricting problem, other constraints that are common in clustering problems should be considered. Let the set of initial regions be $R = \{r_1, r_2, \ldots, r_n\}$, where r_i is the ith region. Let c the number of clusters and M_i the set of all the regions that belong to cluster i. Then:

$$M_i \neq \varnothing \text{ for } i = 1, \ldots, c$$
$$M_i \cap M_j = \varnothing \text{ for } i \neq j$$
$$\bigcup_{i=1}^{c} M_i = R \tag{3}$$

Regardless of the formulation, the redistricting problem is formally computationally intractable (it is NP-complete) [3]. Hence, as Bação et al. suggest [1], heuristic techniques seem to be the best road available to produce good solutions to the problem in reasonable computational time. This is certainly a compromise but guaranteed optimality, independently of the problem's dimension, seems at this stage simply too difficult.

3. Genetic programming

In this work, the electoral redistricting problem has been addressed using a GP based system. GP [5] is one of the youngest paradigms inside the computational intelligence research area called Evolutionary Computation (EC) and consists in the automated learning of computer programs by means of a process mimicking Darwinian evolution. GP evolves computer programs, traditionally represented as tree structures. Trees represent candidate solutions for the problem at hand and they can be easily evaluated in a recursive manner. Every tree node has an operator function and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. In GP, a population of computer programs (i.e. a set of candidate solutions) is evolved. That is, generation by generation, GP stochastically transforms populations of programs into new populations of, hopefully better, programs. The search is performed as follows:

- Choose a representation space in which candidate solutions can be specified. This consists in deciding on the primitives of the programming language that will be used to construct programs. A program is built up from a terminal set (the variables in the problem, and eventually a set of constants) and a function set (the basic operators).
- Design the fitness criteria (i.e. one or more objective functions) for evaluating the quality of a solution.
- Design a selection and replacement policy. Central to every EA is the concept of fitness-driven selection in order to exert an evolutionary pressure towards promising areas of the program space. The replacement policy determines the way in which newly created candidate solutions replace their parents (i.e. existing candidate solutions) in the population.
- Design a variation mechanism for generating new solutions from existing ones. Standard GP uses two main variation operators: crossover and mutation. Crossover recombines parts of the structure of two individuals (trees representing candidate solutions), whereas mutation stochastically alters a portion of the structure of an individual.

After a random initialization of a set (population) of candidate solutions (computer programs), an iterative application of selection-variation-replacement is employed to improve the programs quality in a stepwise refinement way. For a complete introduction to GP the reader is referred to [4–6].

In this work, we used genetic operators that, diversely from the canonical ones, are based on the concept of semantics that will be introduced in the next section. To understand the differences between the genetic operators used in this work (described in Section 4) and the ones used in the standard GP algorithm, the latter are briefly described.

"*Standard*" *Crossover*: The crossover operator is traditionally used to combine the genetic material of two candidate solutions (called parents) by swapping a part of one parent with a part of the other. More specifically, having two parent individuals, sub-tree swapping crossover (also known as "standard" GP crossover) proceeds by the following steps:

- Select a random subtree in each parent. According to the situations, the selection of subtrees can be, or not be, biased so that some subtrees are selected with lower probability than others.