Complex Clustering Using Constraint Programming: Modelling Electoral Map Creation

Lars Kotthoff[2], Barry O'Sullivan[3], S. S. Ravi[3], and Ian Davidson[1]

[1] University of California at Davis, USA
[2] University of British Columbia, Canada
[3] Insight Centre for Data Analytics, University College Cork, Ireland
[4] State University of New York at Albany, USA

Abstract. Traditional clustering is limited to a single collection of objects, described by a set of features under simple objectives and constraints. Though this setting can scale to huge data sets, many real world problems do not fit it. Consider the problem motivating this work: creating electoral district maps. Not only are two sets of objects (electoral districts and elected officials) separately clustered simultaneously under complex constraints, the clusters must be matched and it is required to find a global optimum. Existing formulations of clustering such as those using procedural languages or convex programming cannot handle such complex settings. In this paper we explore clustering this complex setting using constraint programming. We implement our methods in the Numberjack language and make use of large-scale solvers such as Gurobi which exploit multi-core architectures.

1 Introduction and Motivation

Big data has dominated the research landscape for several years as faster processors and larger storage devices allow for more data to be analyzed and stored. Big data problems allow leveraging huge amounts of *related data* such as making predictions for a given patient using similar patients' drug prescriptions / regimes / prognoses. However, many problems of interest are inherently **not** big data problems due to the intrinsic nature of the problem. Consider the motivating example in this paper of creating electoral maps in Ireland which is a *unique* problem to each country as leveraging other electoral map solutions will not help.

We term such problems *small data problems* with a further critical difference between small data and big data problems being unlike large data problems such as predicting the most effective drug using evidence based medicine, small data problems do not need to be solved frequently and hence investing a lot of resources to find an optimal solution is acceptable and expected. For example in our motivating example of creating electoral district maps, these maps are typically redrawn every census which is every ten years. This makes small data clustering problems an ideal fit for the constraint programming paradigm.

In particular we focus on small data problems that involve complex *objects*, complex *constraints* and complex *objective functions*. Each set of objects has a considerable number of constraints which will determine a feasible clustering. An example of such a constraint is that the diameter of each district cluster (called a constituency) should be at most a specified value. Finally, the required output is not one clustering of all objects [11] or two disparate clusterings [13]. Instead, we wish to find two *separate* clusterings for the different object types and an appropriate matching which assigns a cluster of elected officials to each constituency. This results in a *multi-level optimization problem* for which a globally optimal solution must be found.

In this paper, we present a multi-level clustering formulation of the electoral district map creation problem. We view this as an example of complex clustering involving complex objects, constraints and objective functions which cannot be solved with machine learning and data mining methods. Our model is succinct and captures all the core properties of the problem. Initial experiments show the usefulness of the approach but the remaining challenge is scalability. Currently, our method can solve problems involving under 1 million people on inexpensive multi-core workstations but solving the problem for the Republic of Ireland requires scaling to nearly 5 million people.

2 Motivating Problem

We now discuss the electoral map problem which is the focus application area for this paper. A simplified pictorial representation of the electoral map problem is shown in Figure 1. Electoral districts are clustered into *constituencies* which are contiguous. For each constituency, a group of *representatives* must be chosen. That is, the entire collection of representatives is broken down into clusters which are then allocated to constituencies. The population of a constituency is determined by its electoral districts.

We assume the case in which we are creating an electoral map from scratch. Changing an existing map where changes should be minimized is easily accommodated by our model through additional constraints and augmenting the objective function.

It is important to realize that electoral district maps are created (approximately) once every decade, and in many countries this is not a political process but rather an optimization process. The problem has many complexities such as (i) simultaneously clustering the electoral districts into constituencies and the elected officials into groups and then finding a bijective matching that assigns each cluster of elected officials to a cluster of electoral districts (i.e. a constituency), (ii) each elected official must effectively be in charge of an approximately equal sub-population and (iii) each constituency should contain districts which are geographically contiguous and convex-like.

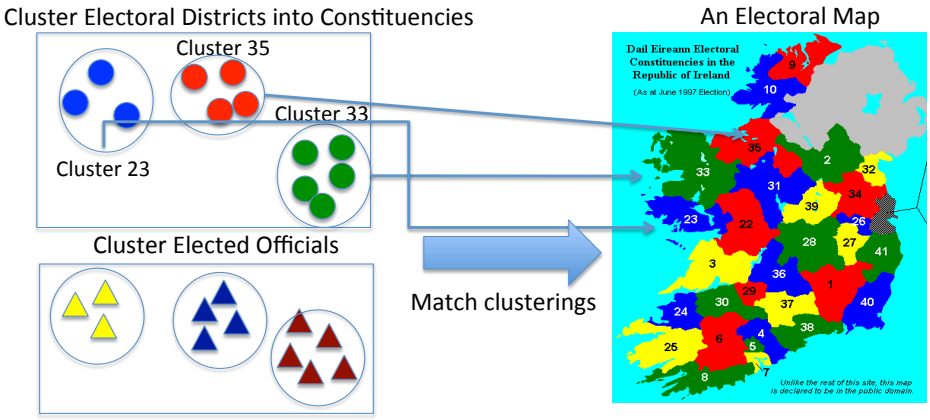

Fig. 1. An overview of our proposed application task on electoral district map creation.

3 Related work

The work related to our algorithmic research can be divided into three areas: clustering problems that search for global optima, clustering complex data, and clustering using constraint programming.

3.1 Global Optimum Clustering Methods

There exist several formulations of finding the global optimum that is suited for small data sets. Charikar et al. [4] study correlation clustering where instances are described only by their positive and negative relationships with each other; they present factor 3 approximation algorithms based on semi-definite programming formulations. Similarly, the k-medoid problem can be solved exactly using integer linear programming formulations [3]. However, both of these formulations are limited to simple objective functions. Other work [8, 10] focuses on finding global optima for partitional and hierarchical clustering with new objective functions. However, this work is only for a single collection of objects, unlike the work discussed in the current paper.

3.2 Clustering complex data

There exist formulations of clustering for heterogeneous data (multiple object types) [11, 14]. However, this work implicitly tries to find a **single** clustering of different object types. For example our own work [11] attempts to cluster images, video, text and locations so that each cluster is a heterogeneous collection of objects. This is achieved by having together and apart relations between objects of different types and projecting the data to a common low dimensional space. Similarly there has been work using complex **combinations** of together/apart

constraints [19]. In that work, the underlying constraints are modeled as relations that can be expressed in conjunctive normal form.

3.3 Clustering using declarative languages

There have been declarative approaches to solving clustering problems. For example, [8] propose a SAT-based framework for efficient clustering with constraints. SAT solvers have been used to model constraints [9] but the underlying clustering algorithm is still procedural. A declarative approach to clustering is discussed in [6]; however, this work models traditional clustering using a CP.

3.4 The Electoral Map Creation problem

To our knowledge, this is the first attempt to formulate the problem as a global optimization problem. Earlier work creates easier versions of the problem as discussed below and all use heuristics with no guarantee of finding a global optimum.

Ricca and Simeone [16] consider electoral districting for regions of Italy. They acknowledge that the optimization problem is very hard (but do not prove NP-hardness) and employ heuristics. Similarly, Photis [15] proposes SPiRAL and also uses hand crafted heuristics. Guo and Jin [12] present iRedistrict, a software platform that allows the user to interactively optimise a clustering of electoral districts. The authors apply their approach to clustering counties in the US states of Iowa and South Carolina and employ a heuristic based on Tabu search.

4 Constraint Model

An overview of our clustering model is shown in Figure 2. It is important to realize that there are multiple object types and that objects of each type are described by both features and relationships among them. We wish to cluster each object type **separately** and construct a matching between the clusters in different clusterings.

4.1 Objects

Let n_1 be the number of objects of the first type (O_1) which we shall call electoral districts (ED) and n_2 be the number of objects of the second type (O_2) which we call elected officials (EO). Let k_1 and k_2 be the number of clusters for the first and second object types respectively. Since in this problem we require a bijective matching, we have $k_1 = k_2$; we will denote this common value as k. In our example, the ED-clusters are termed constituencies and the EO-clusters are termed packs of representatives.

Let X be the $n_1 \times k$ indicator matrix that determines the cluster assignments for the type one objects. This is achieved by each column in X being a 0/1 indicator vector for membership in a cluster. In our example X indicates which

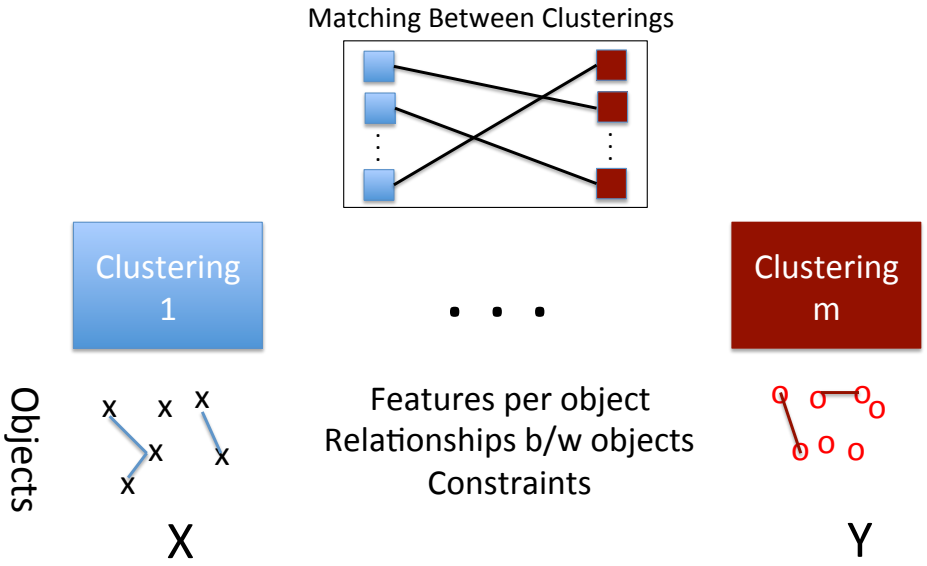

Fig. 2. An overview of our proposed setting for m different types of objects.

electoral district is assigned to which constituency. Further let Y be the $n_2 \times k$ indicator matrix that determines the object assignments for the second object type. In our example, Y indicates which official is assigned to which pack-of-representatives. Formally:

$$\begin{aligned} x_{i,j} &= 1, \text{ if electoral district } i \text{ is part of constituency } j \\ &= 0, \text{ otherwise.} \end{aligned}$$

$$\begin{aligned} y_{i,j} &= 1, \text{ if official } i \text{ is part of group } j \\ &= 0, \text{ otherwise.} \end{aligned}$$

We note that for our application, the actual values for the $y_{i,j}$ are not of interest as we only require the number of representatives for a particular district. We nevertheless model them explicitly for clarity. The additional variables do not affect solving performance in practice.

4.2 Properties/Features of Objects

We define properties P_1 and P_2 for object types one and two which can be used as the basis of the objective function and constraints. In our example $P_1 = (p_1, p_2, \ldots, p_{n_1})$ is a vector with n_1 elements, containing the populations of the electoral districts whilst P_2 is a vector with n_2 elements specifying how many people an official represents. An important benefit of the constraint programming (CP) formulation is that features need not be directly given; instead, they can be functions of other features even from other object sets. For example, P_2 cannot

be directly given since it depends on the ED-clustering and the EO-clustering. However, we can use the auxiliary vector Z defined in Equation 5 to calculate P_2.

4.3 Objective function

Whereas most clustering formulations have simple objective functions and finding a local optimum is sufficient, here we focus on complex objective functions and finding a global optimum.

In our example, we aim to make the number of people each official represents as equal as possible across all officials. In other words, the difference in the number of people represented between an official that represents most people and an official that represents fewest people should be minimized. This is an example of an objective function based on cardinality and differs from most clustering objectives which are geometric in nature. Formally, our optimization problem can be represented as follows:

$$min\left(max(P_2) - min(P_2)\right) \equiv \min\left(max(Z) - min(Z)\right) \tag{1}$$

4.4 Composition Constraints

Here we outline three types of composition constraints that are easily enforceable in a constraint model, (i) basic composition, (ii) cardinality constraints and (iii) complex composition constraints.

First we provide basic constraints to ensure that X and Y are legitimate set partitions. We require that each object is assigned to exactly one cluster ($min_{k1} = max_{k1} = 1$ and $min_{k2} = max_{k2} = 1$) which in our example means each electoral district is assigned to exactly one constituency and each constituency is assigned at least one electoral district. Formally:

$$\begin{aligned} \forall i \in \{1 \dots n_1\} :\ & min_{k1} \leq \sum_{j=1}^{k} x_{i,j} \leq max_{k1} \\ \forall i \in \{1 \dots n_2\} :\ & min_{k2} \leq \sum_{j=1}^{k} y_{i,j} \leq max_{k2} \end{aligned} \tag{2}$$

However, we could just as easily specify constraints that an object could be assigned to a minimum and/or maximum number of clusters by adjusting the values of min_{k1}, min_{k2} and max_{k1}, max_{k2} respectively. This will allow overlapping clusters and explicitly control the amount of overlap.

We require each cluster to contain at least one point. The corresponding constraints are:

$$\begin{aligned} \forall j \in \{1 \dots k\} :\ & \sum_{i=1}^{n_1} x_{i,j} \geq 1 \\ \forall j \in \{1 \dots k\} :\ & \sum_{i=1}^{n_2} y_{i,j} \geq 1 \end{aligned} \tag{3}$$

Again by changing the bounds on this summation we allow more complex constraints which bound the cardinality of cluster size. We note this is a trivial exercise in CP but is challenging in procedural languages [1].

In CP, we can easily create new auxiliary variables and impose constraints on them. We channel the number of officials per constituency into an auxiliary variable vector $A = (a_1, a_2, \ldots, a_k)$ and require that each constituency be assigned between EO_{min} and EO_{max} officials.

$$\begin{aligned} \forall j \in \{1 \ldots k\} : \text{ Let } a_j &= \sum_{i=1}^{n_2} y_{i,j} \\ \forall j \in \{1 \ldots k\} : \; EO_{min} &\leq a_j \leq EO_{max} \end{aligned} \tag{4}$$

Similarly, we can calculate the number of people each official represents (z_i) by the summing the populations of the electoral districts (given by $P_1 = (p_1 \ldots p_{n_1})$) and dividing by the number of officials assigned to the respective constituency. This gives us the vector Z. Formally:

$$\forall j \in \{1 \ldots k\} : \text{ Let } z_j = \frac{\sum_{i=1}^{n_1} x_{i,j} p_i}{a_j} \tag{5}$$

It is trivial to encode instance level composition constraints such as must-link and cannot-link that others have extensively studied (see e.g. [7, 18]) by noting the following: if ED i and j must be together then $\sum_{r=1}^{k} x_{i,r} \times \; x_{j,r} = 1$, and if they must not be together then the sum should equal 0. A particular strength of CP is that composition constraints can be used in logical combinations; something that is challenging to do in other formulations [19].

Our constraint model is very flexible and allows for arbitrary additional constraints. In our application, the Irish constitution stipulates that no official may represent more than 30,000 people. This is easily added to the model with the following constraint.

$$\forall j \in \{1 \ldots k\} : \; z_j \leq 30,000 \tag{6}$$

4.5 Relational Continuity Constraints

In addition to the requirements encoded above, we need to post constraints enforcing the connectivity of the electoral districts in a constituency. The closest approach to our work has been presented recently in [5], but the authors consider only a less complex setting – here, we do not limit the number of subsets to one and we do not know the terminal nodes a priori.

The global constraint catalog (`http://sofdem.github.io/gccat/`) is a collection of over 500 complex constraints covering a range of different application areas like graphs, networks and even SAT problems. We leverage the tree constraint [2] to ensure that the underlying ED-clusters are contiguous. That is, there may be no ED that is clustered into a constituency, but is disconnected

from the rest of the electoral districts in the same constituency. The same applies to disconnected groups of electoral districts. Formally:

$$\begin{aligned}
&\forall a, b \in \{1 \dots n_1\}, a \neq b, j \in \{1 \dots k\} : \\
&\quad (x_{a,j} = 1 \wedge x_{b,j} = 1) \implies \\
&\quad (\exists C = \{c : c \in \{1 \dots n\}, x_{c,j} = 1, c \neq a, c \neq b\}) : \\
&\quad adjacent(a, c_1) \wedge \dots \wedge adjacent(c_k, b)
\end{aligned} \tag{7}$$

In other words, Equation (7) requires that for each pair of electoral districts a and b, if they belong to the same constituency, there must exist a set C of electoral districts that are not a or b, but belong to the same constituency. Furthermore, there exists an ordering of all $c \in C$ such that the ordered set forms a path from a to b, i.e. each element in the path is adjacent to its predecessor and successor.

In practice, it is not efficient to encode this constraint in this general form – the model would need to consider all pairs of electoral districts, all possible assignments to constituencies for them, and all possible paths between them. In particular, the number of possible paths between a pair of electoral districts is far too large to encode, even for this small data problem.

We therefore express this requirement in a different, semantically equivalent form. Using the tree constraint, we require electoral districts in each constituency to form a tree. That is, one district is arbitrarily designated the root and all other districts must be connected to it or one of its children. This means each district must be adjacent to at least one other district in the same constituency, which is in turn connected to other districts. This way, we ensure that each district is connected to each other district in the same constituency, because each node in a tree is connected to each other node in the same tree, ignoring the direction of the links.

To encode a tree for each electoral district cluster, we introduce three new variables for *each node* – the rank in the tree (corresponding to the distance from the root), $rank$, the index of the cluster (i.e. constituency) the electoral district is assigned to, $cidx$, and the index of the electoral district that is its parent in the tree (i.e. one of the districts it is adjacent to on the electoral map), adj. Electoral districts can be their own parents in the tree, indicating that they are at the root of the respective tree. We then add the following constraints, according to the decomposition of the constraint as specified in the global constraint catalogue.

$$\begin{aligned}
&\forall i \in \{1 \dots n_1\}, adjacent(j, i), i \neq j : \\
&\quad (adj_i = j) \implies ((rank_i > rank_j) \wedge (adj_j \neq i) \\
&\qquad\qquad \wedge (cidx_i = cidx_j))
\end{aligned} \tag{8}$$

The intuition behind the model is that districts in a tree are ordered according to their rank, which is derived from the parent information. Each district must have a parent, parent districts must be in the same tree (i.e. constituencies), and the constraints on the ranks ensure that there are no cycles. Each tree is

characterized by a root node (which is a parent of itself) and root nodes must be in different trees, i.e. different constituencies. The number of trees must be equal to the number of constituencies (clusters).

If district j is i's parent in the same tree, then the rank of i is strictly greater than that of j, i is not j's parent, and both districts are in the same constituency.

$$\begin{aligned} &\forall i, j \in \{1 \dots n\}, i \neq j : \\ &\quad ((rank_i = 0) \wedge (rank_j = 0)) \implies (cidx_i \neq cidx_j) \end{aligned} \tag{9}$$

If two districts both have a rank of 0, they cannot be in the same constituency. That is, they are both root nodes of trees. Finally, we require the number of districts with rank 0 to be equal to the number of constituencies.

$$card(rank, 0) = k \tag{10}$$

In addition, we post the following channelling constraints.

$$\forall i \in \{1 \dots n\} : x_{i,cidx_i} = 1 \tag{11}$$

$$\forall i \in \{1 \dots n\} : adj_i = i \iff rank_i = 0 \tag{12}$$

The former constraint ensures that the cluster index $cidx$ is set according to the cluster assignments and the latter that an electoral district has rank 0 iff it is connected to itself.

4.6 Putting It All Together: The Complete Formulation

For completeness, we show in Figure 4.6 our entire CP formulation for the electoral map creation problem. In that figure, each element of the vector Z (which has n_2 entries) corresponds to the number of people represented by an official. Also, an informal description of each constraint is provided before its formal specification.

In the case of the Irish electoral map creation problem, we need each object to appear in exactly one cluster (i.e. we need a partitional clustering of both types of objects). Thus, each of the four parameters min_{k1}, max_{k1}, min_{k2} and max_{k2} is set to 1. Additionally, we set $EO_{min} = 3$, $EO_{max} = 5$ and $POP_{max} = 30,000$.

5 Experiments

We modeled the problem of finding constituencies and packs-of-representatives for Ireland as outlined above. The data on electoral districts and their populations is available on the website of the Central Statistics Office of Ireland[5].

In this preliminary evaluation, we consider only Galway City with just 22 EDs representing a population of 75529 people which must be allocated to 8 elected officials (TDs). The optimal solution is shown in Figure 4. Table 1 shows the results and that the optimum value for the parameter of Equation (6) is 2218.3.

[5] `http://census.cso.ie/censusasp/saps/boundaries/eds_bound.htm`

Objective:

$$min\,(max(Z) - min(Z))$$

$X = [x_{i,j}]_{n_1 \times k}$ and $Y = [y_{i,j}]_{n_2 \times k}$ are matrices with $\{0, 1\}$ entries.

Subject To:

I. Bounds on the number of clusters in which an object may appear.

$$\forall i \in \{1 \dots n_1\} : \; min_{k1} \leq \sum_{j=1}^{k} x_{i,j} \leq max_{k1}$$

$$\forall i \in \{1 \dots n_2\} : \; min_{k2} \leq \sum_{j=1}^{k} y_{i,j} \leq max_{k2}$$

II. Ensuring that each cluster is non-empty.

$$\forall j \in \{1 \dots k\} : \; \sum_{i=1}^{n_1} x_{i,j} \geq 1$$

$$\forall j \in \{1 \dots k\} : \; \sum_{i=1}^{n_2} y_{i,j} \geq 1$$

III. The number of elected officials per cluster is between EO_{min} and EO_{max}.

$$\forall j \in \{1 \dots k\} : \text{ Let } a_j = \sum_{i=1}^{n_2} y_{i,j}$$

$$\forall j \in \{1 \dots k\} : \; EO_{min} \leq a_j \leq EO_{max}$$

IV. Bound on the population represented by each elected official POP_{max}.

$$\forall j \in \{1 \dots k\} : \; z_j \leq POP_{max}$$

V. Ensuring that each cluster is connected.

$$\begin{aligned} &\forall i \in \{1 \dots n_1\}, j \text{ is adjacent to } i, i \neq j : \\ &\quad (adj_i = j) \implies ((rank_i > rank_j) \wedge (adj_j \neq i) \\ &\qquad\qquad \wedge (cidx_i = cidx_j)) \end{aligned}$$

Definitions of channel variables used above:

(a) Individuals allocated to each elected official:

$$\forall j \in \{1 \dots k\} : \text{ Let } z_j = \frac{\sum_{i=1}^{n_1} x_{i,j} p_i}{a_j}$$

(b) Tree to ensure cluster connectivity.

$$\forall i \in \{1 \dots n_1\} : x_{i,cidx_i} = 1$$

$$\forall i \in \{1 \dots n_1\} : adj_i = i \iff rank_i = 0$$

$$\forall i, j \in \{1 \dots n\}, i \neq j : ((rank_i = 0) \wedge (rank_j = 0)) \implies (cidx_i \neq cidx_j)$$

$$card(rank, 0) = k$$

Fig. 3. The objective and constraints used to encode our formulation.

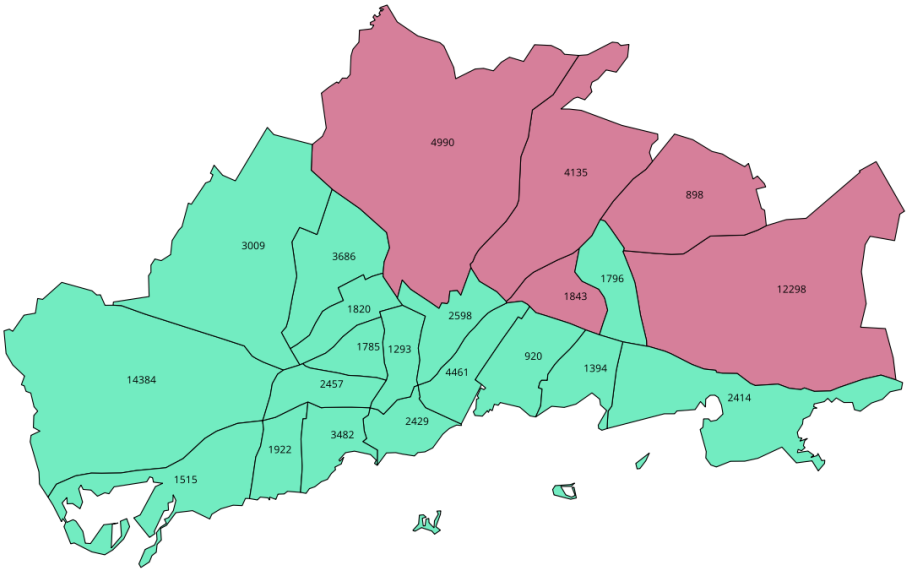

Fig. 4. Solution for Galway City. The colors denote the constituencies, the numbers show the populations for the districts.

cluster	electoral districts	population	representatives	population per representative
1	5	24164	3	8054.667
2	17	51365	5	10273

Table 1. Clusters in the optimal solution for Galway City.

6 Discussion and Challenges

The results presented above only represent a small fraction of the full problem we are interested in. Unfortunately, even this result took significant computational resource to obtain; in the order of 100 CPU hours. As the full problem is exponentially larger, we cannot hope to solve the current formulation for the full problem. This is the main challenge that we face here and it is due to two main reasons. Firstly, the optimization problem has an exponentially large number of solutions and secondly the continuity constraint is difficult to satisfy. There are several directions we believe are worth exploring which we now briefly outline.

6.1 Reducing Problem Size

Though the optimization is over the binary matrices X and Y the possible range of the objective function is large. In our work, the range of the objective function in Equation (6) is $0 \ldots 30,000$. We can use the pigeon hole principle to make this range much smaller and find another alternative but equivalent objective function.

Let t be the total population of all districts to be clustered. Then there must exist one cluster whose total population is at least t/k by the pigeon hole principle [17]. Therefore, we can set a lower bound on this variable as t/k.

6.2 More Efficient Encoding of Constraints

One way of potentially improving performance is by encoding the connectivity constraints in a better way. The encoding of the connectivity constraints provides only very weak propagation throughout search, leaving potential for additional pruning of the search tree. We have investigated a number of alternative formulations that achieve better propagation, but require exponential space to encode. This limits the applicability for our problem.

In addition, there are symmetries inherent in the formulation that we currently do not consider, for example the cluster assignments. Breaking these symmetries is not straightforward however, as we potentially need to take both clusterings into account.

7 Conclusions and Future Work

The topic of big data has dominated the research landscape and generated many promising results for clustering problems that consist of simple objective functions, constraints and object descriptions. Though these results scale to large data sets they are not suited for problems with complex objectives, constraints and object descriptions. We term these problems complex clustering problems and explore one in particular: finding electoral district maps for Ireland.

The problem involves clustering two sets of objects (electoral districts and elected officials) separately whilst simultaneously matching clusters from one clustering to the other clustering. It inherently involves two layers of optimization: clustering and bijective mapping under complex constraints including cardinality, continuity and diameter. Such problems are not easily formulated in the traditional convex, spectral or even procedural formulations used extensively in clustering.

Instead we show that constraint programming is a useful paradigm to address these challenging problems. Not only is the problem formulation in CPs easy to encode, it offers flexibility such as allowing overlapping clustering, cardinality restrictions and constraints on auxiliary variables that are challenging in other formulations. Our preliminary experimental evaluation shows the promise of the approach.

The major challenge of using our approach in practice is that the search space is very large and it takes significant computational resources to find the optimal solution and that the continuity constraint is challenging to satisfy. In future work, we will investigate ways of making the model and the solving more efficient.

Acknowledgments

We thank Nicolas Beldiceanu for suggesting the tree constraint to encode the connectivity requirement and the anonymous reviewers for their feedback. Part of this work was supported by the European Commission as part of the "ICON - Inductive Constraint Programming" project (contract number FP7-284715). The Insight Centre for Data Analytics is supported by Science Foundation Ireland under grant number SFI/12/RC/2289.

References

1. Banerjee, A., Ghosh, J.: Scalable clustering algorithms with balancing constraints. Data Mining and Knowledge Discovery 13(3), 365–395 (2006)
2. Beldiceanu, N., Flener, P., Lorca, X.: The tree constraint. In: Bartk, R., Milano, M. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Lecture Notes in Computer Science, vol. 3524, pp. 64–78. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/11493853_7
3. Charikar, M., Guha, S., Tardos, É., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem (extended abstract). In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA. pp. 1–10 (1999)
4. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. In: 44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings. pp. 524–533 (2003)
5. Christian Bessiere, Emmanuel Hebrard, G.K., Walsh, T.: Reasoning about Connectivity Constraints. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15) (July 2015)
6. Dao, T., Duong, K., Vrain, C.: A declarative framework for constrained clustering. In: Machine Learning and Knowledge Discovery in Databases, pp. 419–434. Springer (2013)
7. Davidson, I., Ravi, S.S.: The complexity of non-hierarchical clustering with instance and cluster level constraints. Data Min. Knowl. Discov. 14(1), 25–61 (2007)
8. Davidson, I., Ravi, S.S., Shamis, L.: A SAT-based framework for efficient constrained clustering. In: SIAM International Conference on Data Mining. pp. 94–105 (2010)
9. Gilpin, S., Davidson, I.: Incorporating sat solvers into hierarchical clustering algorithms: an efficient and flexible approach. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1136–1144. ACM (2011)
10. Gilpin, S., Nijssen, S., Davidson, I.: Formalizing hierarchical clustering as integer linear programming. Proceedings of the twenty-seventh AAAI conference on artificial intelligence pp. 372–378 (2013)

11. Gress, A., Davidson, I.: A flexible framework for projecting heterogeneous data. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 1169–1178. ACM (2014)
12. Guo, D., Jin, H.: iRedistrict: Geovisual analytics for redistricting optimization. J. Vis. Lang. Comput. 22(4), 279–289 (Aug 2011)
13. Hossain, M.S., Tadepalli, S., Watson, L.T., Davidson, I., Helm, R.F., Ramakrishnan, N.: Unifying dependent clustering and disparate clustering for non-homogeneous data. In: proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. pp. 593–602. ACM (2010)
14. Long, B., Zhang, Z.M., Wu, X., Yu, P.S.: Spectral clustering for multi-type relational data. In: Proceedings of the 23rd international conference on Machine learning. pp. 585–592. ACM (2006)
15. Photis, Y.N.: Redefinition of the greek electoral districts through the application of a region-building algorithm. European Journal of Geography 3(2), 73–83 (Oct 2012)
16. Ricca, F., Simeone, B.: Local search algorithms for political districting. European Journal of Operational Research 189(3), 1409–1426 (2008)
17. Rosen, K.: Discrete Mathematics with Applications to Computer Science. McGraw-Hill, New York, NY (2012)
18. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: Proc. International Conference on Machine Learning. pp. 1103–1110 (2000)
19. Zhi, W., Wang, X., Qian, B., Butler, P., Ramakrishnan, N., Davidson, I.: Clustering with complex constraints-algorithms and applications. In: AAAI (2013)