Automatic Dwelling Segmentation of Buenos Aires Province for the 2010 Argentinian Census

February 4, 2013

Abstract

One of the logistical challenges in planning a population census is determining which dwellings each enumerator must visit within a census tract. This is known as the *dwelling segmentation problem*, which generally includes a set of constraints on the enumerators' assigned routes and various criteria regarding the homogeneity and uniformity of the segmentation solutions. In this work, we present a computational approach for this problem, which represents an improvement over manual methods and existing software. The resulting method was successfully applied to the Province of Buenos Aires in the 2010 Argentinian census.

KEYWORDS: population census, segmentation.

Introduction

A population census is a procedure for acquiring demographic information on a geographical area, generally an entire country. The information gathered typically includes data on the population, housing, educational and employment characteristics of the area's inhabitants. A census usually takes place on a single day, the actual collection of the information being carried out by census enumerators who visit each dwelling in the area covered.

To ensure the smooth and successful completion of such a major logistical operation in a few short hours, the planning for a census usually commences several years in advance.

Determining which dwelling units each enumerator must visit is crucial, and this task is called the *dwelling segmentation problem*. The dwelling segmentation problem consists of partitioning a set of dwellings into subsets that satisfy certain constraints. In its full generality, the dwelling segmentation problem generalizes the set partitioning problem, hence it is NP-complete.

A relatively high-level partition, as with the redistribution of electoral divisions, gives rise to the *redistricting problem*. There is a considerable literature devoted to this issue [2, 3, 6, 9, 10, 11, 12] and various software packages carry out redistricting either automatically or semi-automatically. In [4], for example, the authors present an open source tool implemented in the R statistical environment. But since the constraints arising in electoral redistricting generally differ from those imposed by dwelling segmentation problems, the latter require their own specifically designed algorithms. Dwelling segmentation also involves certain homogeneity and uniformity criteria that must be satisfied by the solutions, as will be detailed in *Description of the problem*. This was almost impossible to achieve with the previously-used manual solution methods, which consisted of human operators constructing segmentations in a greedy fashion, with no pre-specified rules or "algorithmic instructions" given beforehand to them. Within such a scheme, an operator could find by chance a good segmentation, but could fail if his/her first decisions led to a poor final result.

The present study analyzes the dwelling segmentation problem as it arose in the Province of Buenos Aires in the 2010 Argentine census, and reports on the development of a computational approach for solving it that we applied during the census planning process. Based on the experience from the previous census, the Province's statistical authority (*Dirección Provincial de Estadística*) decided to tackle the dwelling segmentation problem automatically with operations research techniques, and contacted the University of Buenos Aires in order to explore this issue. The present study is the result of such collaboration. We are not aware of any other studies in the operations research literature that have investigated dwelling segmentation in a census context.

The remainder of this paper is organized as follows. In *Description of the problem* we review the relevant characteristics of the 2010 census and we discuss the dwelling segmentation

problem to be addressed. In *The segmentation algorithm* we propose a solution algorithm and the main difficulties encountered during its implementation are discussed. In *Computational results* we set out the results obtained with this tool, and *Conclusions* contains some final remarks. The appendix sets out the algorithm and its parameters in detail.

Description of the problem

Argentina is divided into 23 provinces in addition to the Autonomous City of Buenos Aires (formerly known as the Federal Capital) which enjoys special administrative status. The Province of Buenos Aires (see Figure 1) is Argentina's largest province geographically (304,907 km^2) and the most highly populated, with some 15.6 million inhabitants, corresponding to the 39% of the total population in the country. It excludes the Autonomous City of Buenos Aires but does include Greater Buenos Aires, the outer belt of suburbs surrounding the Autonomous City that is home to about 9 million people. Together the two areas form a single large population nucleus. With the exception of Greater Buenos Aires, the Province is predominantly rural and agricultural, though some areas are dependent primarily on tourism (the Atlantic coast), mining (the south) or steelmaking (the northeast along the Parana River). By contrast, Greater Buenos Aires is primarily urban and industrial.

All of the provinces are subdivided into *partidos* or *departamentos*. In the case of Buenos Aires Province there are 134 of these divisions, which are further subdivided for census purposes into 19,577 *radios censales* or census tracts. Each census tract contains about 300 dwellings distributed across a set of contiguous blocks that may number anywhere from 1 to 50, depending on the population density. There are 16,216 census tracts located in urban areas, the focus of the present study. The remaining census tracts are divided into 2,886 rural tracts and 475 intermediate tracts. The rural and intermediate tracts are sparsely populated, involve long distances (the census takers cover these tracts by car), and the corresponding maps are not fully updated. The geographical information systems team considered that an automatic solution was not suitable for these tracts, as it would involve a lot of manual preprocessing. Due to these facts, the rural and intermediate tracts are not part of the present study.

Figure 1: The Province of Buenos Aires is Argentina's largest province and the most highly populated (15.6 million inhabitants).

For any given census tract, the segmentation problem involves partitioning the constituent blocks into disjoint sets of dwellings, known as *segments*, each of which is assigned to a single census enumerator. As will be seen later in more detail, the sides of a block may under certain conditions be split across different segments. The blocks in Argentinian urban centers typically have four sides, although other block morphologies are not infrequent.

A principal requirement for the feasibility of a segment is that its various blocks or block sides must be contiguous. According to criteria set by the *Dirección Provincial de Estadística*, every block or side must either directly face another block or side, or, in the case of a side only, must continue directly on from a side in an immediately adjacent block. Thus, a segment cannot "cross" from one block to another diagonally. The segment examples in Figure 2 illustrate these restrictions. The segments in panels (a), (c) and (e) are composed

by contiguous blocks or sides, whereas the segments in (b), (d) and (f) have non-contiguous ones.

The 2010 census rulebook defined a further set of constraints for a feasible segmentation of a census tract as follows:

- Each dwelling in a tract must belong to a single segment, and the complete set of segments created by the segmentation process must include all of the tract's dwellings.
- The number of dwellings contained in each segment must be within a prespecified range, and the bounds of this range depend on the census tract. For 90% of the segments, where the basic census form is utilized, each segment must contain between 32 and 40 dwellings. On the remaining census tracts, where a more detailed census form is utilized, the rulebook specifies the range $[10, 16]$ instead. Furthermore, some particular census tracts admit different values for these parameters.
- A side of a block containing no dwellings must also be contained within a segment in order to be assigned to an enumerator, since the dwellings database may be out of date, and new dwellings may be present in such a block side. If such a block side does contain dwellings then the generated segment may conflict with the upper bound for the number of dwellings per segment but, unfortunately, there is no available information in order to avoid this situation.
- The length of the route to be covered by any enumerator must not exceed an upper bound L that will depend on the population density of their assigned tract.
- Enumerators must not be required to cross major thoroughfares, railway lines or watercourses.
- Every segment must be wholly contained within a single tract.

The last of these constraints is particularly significant for our purposes. Since a segment cannot include dwellings from more than one tract, the segmentation problem addressed in this work is reduced to 16,216 separate instances, one for each urban tract.

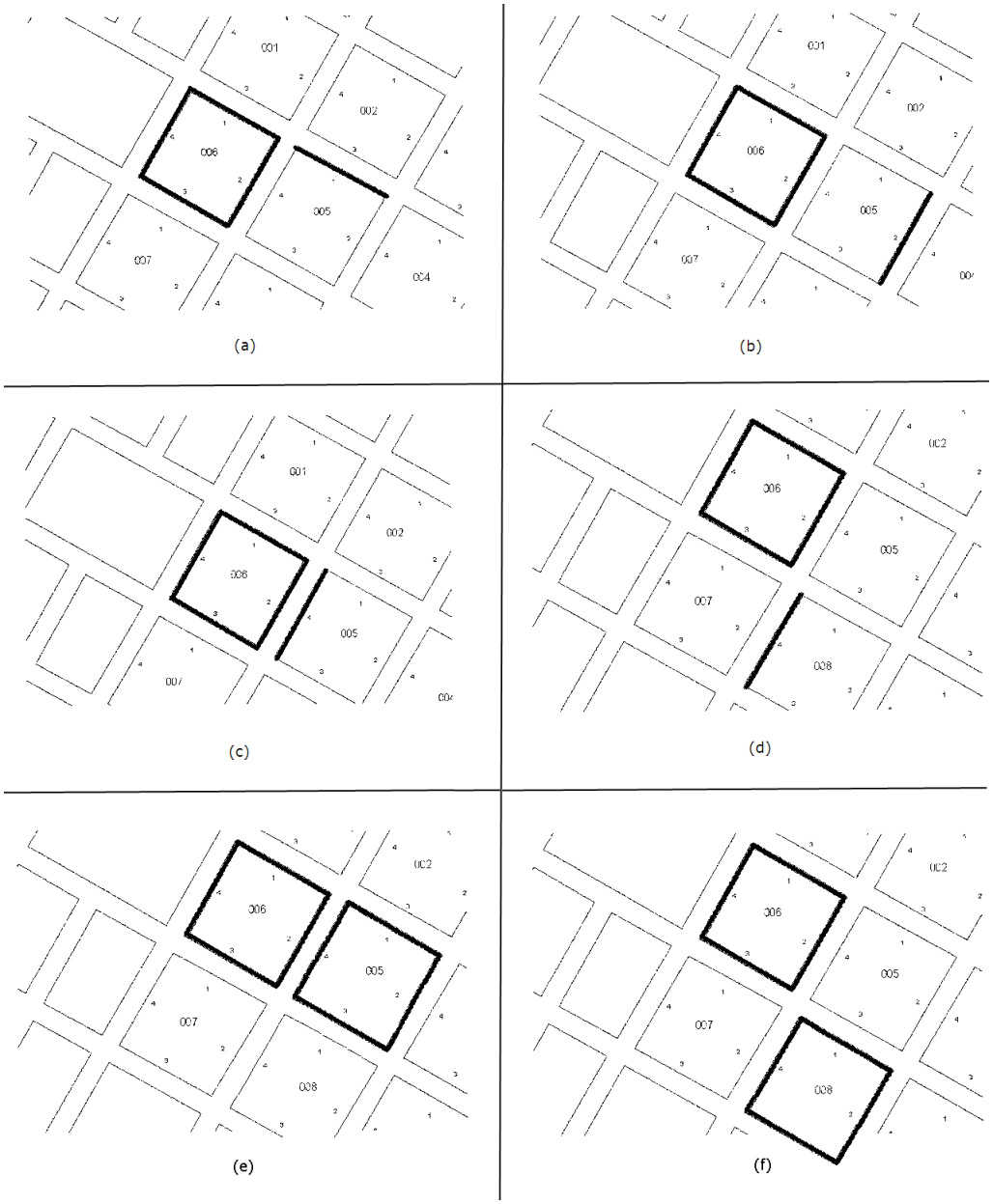

Figure 2: Feasible segments must contain contiguous block sides and cannot cross from one block to another diagonally. Segments (a), (c) and (e) are feasible since the blocks or sides marked in bold are contiguous; whereas segments (b), (d) and (f) are infeasible since their blocks or sides are not contiguous.

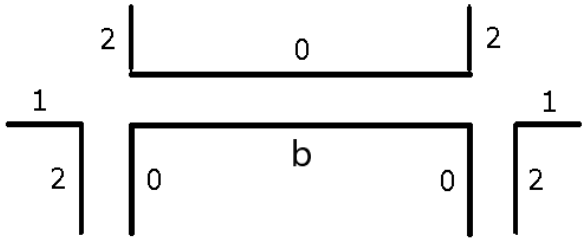

Figure 3: After enumerating block b, preference should be given to the block sides marked with a 0, then block sides marked with a 1 and, finally, block sides marked with a 2.

Although there is no a priori objective function for the segmentation problem, the segmentations which meet certain *adjacency* and *compactness* criteria are preferred. These criteria apply to each individual segment. The following *adjacency* preferences are the most important ones and relate to the routes followed by the enumerators. Upon arriving at either corner of one side of a block (say, side b in Figure 3), the enumerator, rather than crossing a street to continue, should preferably either proceed from that corner along the connecting side of the same block or cross over to the side of the adjacent block directly facing. These options are marked with a 0 in the figure. If this is not possible, the enumerator should continue in the same direction (the same side of the street) to the next block (one of the sides marked 1 in the figure). If this, too, is impossible, the enumerator should then proceed to one of the sides marked with a 2 in the adjacent blocks. The numbers 0, 1 and 2 thus define the levels of adjacency, which in turn determine the order of the preferences just described.

Additionally, the following *compactness* preferences, in order of importance, should be considered in a segmentation:

1. Preference should be given to segments consisting only of whole blocks.

2. Segments should contain complete sides and should be as "compact" as possible. This criterion is defined informally and refers to the breadth of a segment across the various blocks it contains (see Figure 4). For example, a segment consisting only of whole blocks is considered to be compact whereas one that has various blocks at least one but not all of whose sides belong(s) to some other segment is not. The reason for attempting

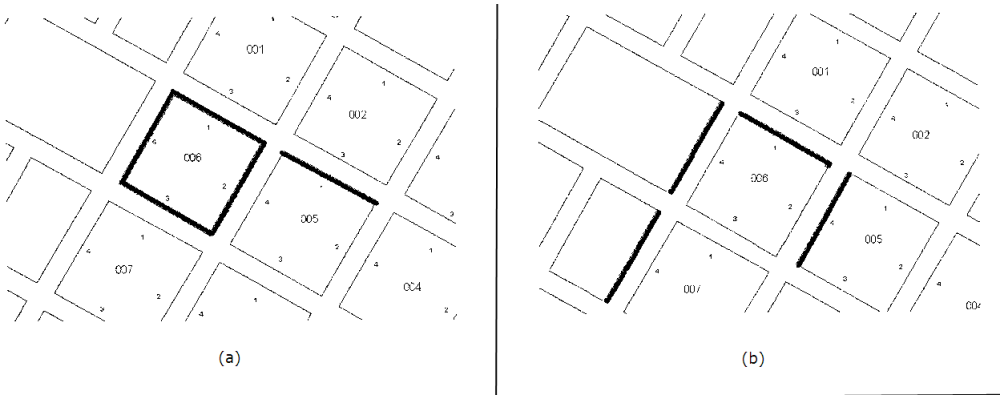

Figure 4: Segments should be as "compact" as possible. Segment (a) is considered to be compact whereas segment (b) is not, since it involves just one block side per block.

to define segments that are as compact as possible is that they tend to minimize the chance of enumerator route errors.

3. If the segmentation cannot be completed using entire block sides, some of the block sides can be divided between two or more segments. This situation may arise if the block side contains a number of dwellings exceeding the maximum allowed per segment, or contains less than the corresponding lower bound but the combination with any adjacent side exceeds the upper bound, thus violating the feasibility constraints. Typically this occurs when a block contains one or more apartment buildings, in which case preference should be given to segments that include all of the apartments in any single apartment building so that it can be covered by one enumerator.

4. If the apartments in a single building cannot all be kept together (e.g., if the building contains more dwellings than the maximum number of dwellings per segment), preference should be given to segments that do not split up the apartments of a single floor. In other words, each floor should where possible be covered by a single enumerator. Please note that apartments spanning more than one floor typically have their address allocated to a single floor, and are considered as a single dwelling for the census purposes.

The segmentation algorithm

In order to tackle the dwelling segmentation problem in each individual census tract, we developed an algorithm that tries to find a feasible segmentation. The full details of the algorithm are set out in the appendix.

The algorithm iteratively considers sets of *candidate segments*. At each iteration, the algorithmm generates a set of feasible segments and tries to find a complete segmentation involving segments from this set. To this end, we solve an integer programming model which takes as input the set of candidate segments, and seeks a feasible segmentation maximizing the global compactness. If such a segmentation exists (i.e., if the integer programming model is feasible) then the algorithm stops and returns the segmentation found by the model solution. On the other hand, if the model is infeasible, then the algorithm moves on to the next iteration, where a new (usually larger) set of candidate segments is generated and the procedure is repeated.

The first iterations consider feasible segments whose block sides are joined by the adjacencies marked with 0 in Figure 3. At the i-th iteration, all feasible segments involving adjacencies marked with 0 in Figure 3 and at most i blocks are generated, and the model takes such set of segments as input. If all these iterations fail to find a feasible segmentation, then the adjacencies marked with 1 in Figure 3 are also considered and the process is repeated. If this procedure again fails, then all feasible adjacencies from Figure 3 are taken into account when generating the set of candidate segments and the process is repeated.

If the above procedure does not find a feasible solution, we activate the option of dividing block sides into parts. To this end, we introduce a parameter that sets the maximum number of dwellings a block side can have, with all sides exceeding this number then being divided into two or more parts. The block side division algorithm (not shown here) greedily generates two or more parts from each block side, each having no more than the number of dwellings specified by the parameter. Once the block sides have been divided we repeat the overall above procedure, which now takes as input the generated parts instead of the original block sides. As long as this procedure fails to find a complete segmentation, we decrease this parameter

–so each (divided) block side will have a smaller number of dwellings– and repeat the above procedure.

In the preliminary testing and adjustment stages, this algorithm solved a large number of the highly urban census tracts but had serious difficulties segmenting tracts with relatively low population densities. An example is the tract in the city of Olavarria in Figure 5, for which the algorithm did not find a feasible solution. The main obstacle in these cases is the presence of many blocks with few if any dwellings, meaning the algorithm must execute a large number of iterations to generate enough feasible segments to cover the whole tract. Furthermore, with few dwellings per block the number of feasible segments is very high. For the tract in Figure 5, the candidate set after 7 iterations has more than 100,000 such segments, resulting in excessively high segment generation and solutions times. In order to tackle this situation, we (a) treat blocks with few dwellings as a single segment and (b) arbitrarily combine side blocks with few dwellings. The details and related parameters are described in *Improvements for low-density census tracts* in the appendix.

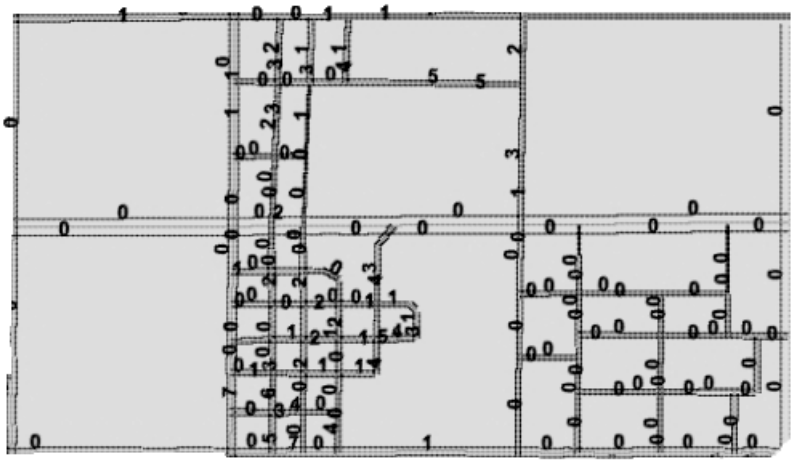

Figure 5: Medium-density tracts such as this one from Olavarria City are difficult for the algorithm to solve. The numbers indicate how many dwellings are on each block side.

Computational results

Given the number of different parameters and their significance for the algorithm's execution time, we conducted preliminary experiments with various test census tracts to identify the best set of parameter values for different tract types. In light of the results, we grouped tracts by population density into three categories: high-density (up to 10 blocks), medium-density (11 to 30 blocks) and low-density (more than 30 blocks) tracts.

We coded the algorithms in C++ and solved the integer programming models using CPLEX 12.1. The data on blocks, sides and dwellings were drawn from the geographical database maintained by the Province of Buenos Aires. We implemented the necessary interfaces in a geographic information system to export the data and import and view the segmentations generated by our algorithm.

Execution times for generation of the segments were relatively short, the worst case being about 2 minutes using the parameters specified in the appendix. More than 99% of the integer programming models for the various tracts were solved in a few seconds implying that the linear relaxation is very tight and the first feasible solution found by CPLEX was usually optimal. In a few cases the execution time reached the imposed time limit (we refer to the appendix for the actual value of this parameter depending on the tract density).

The largest number of segments for each solved census tract (i.e., the number of feasible segments in the last integer programming model in the execution of the segmentation algorithm) is on average 1,305 segments for high-density tracts (with a standard deviation of $\sigma = 4,424$). For medium-density tracts the average number of segments in the largest integer programming model is 3,826 ($\sigma = 8,709$), and for low-density tracts the average number of segments is 10,314 ($\sigma = 11,265$).

By way of comparison, the segmentation of the province for the previous (2001) census was done manually and required 25 employees working full time over 30 days (i.e., 6,000 person-hours). In this work we deal with urban tracts, and these tracts corresponded to 80% of the segmentation time for the 2001 census. For the 2010 count, automated computer tools perfomed the dwelling segmentation for the first time. The algorithm presented here delivered satisfactory results for 96% of the urban census tracts in about 320 CPU hours running on a

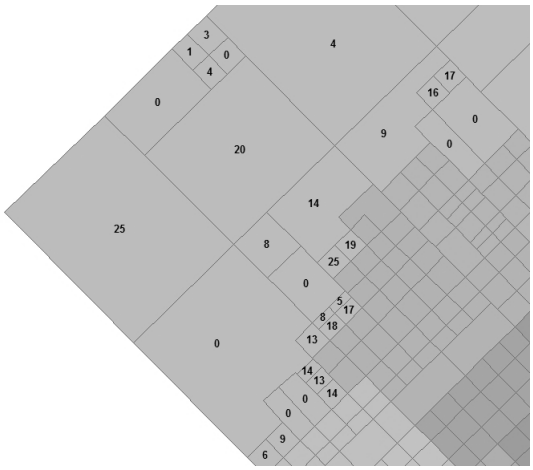

Figure 6: The segmentation algorithm could not find a feasible solution for this medium-density tract, due to the combination of sparsely-populated areas and denser blocks.

computer with a 2.4 GHz Intel Celeron© processor and 2 GB of RAM (the equivalent of less than one day on a cluster of 15 computers).

For approximately 600 census tracts (4% of the total urban tracts) the algorithm did not find a feasible solution automatically. In these cases, the segmentation problem was solved either using the tool but with slightly relaxed constraints or manually. As an example, consider the medium-density tract in Figure 6. The pre-processing stage combined the blocks with few dwellings to form indivisible groups, as described in the previous section. There are three contiguous blocks in the tract with 14, 13 and 14 dwellings, respectively (see bottom of figure), that are surrounded by very thinly populated blocks which in this case were combined in a single group and which "isolate" the aforementioned three. The latter have a total 41 dwellings, which due to the upper bound on the number of dwellings per segment meant they could not be put into a single segment, nor could two segments be constructed from them due to the lower bound on the number of dwellings per segment. This tract was solved in a few seconds by relaxing the upper bound by one unit.

Many of the census tracts not solved by the algorithm had characteristics similar to those in Figure 6. In other words, they were for the most part thinly populated but with denser areas along their boundaries (usually bordering more heavily populated tracts). When faced to these results, planning staff at the census authority asserted that the boundaries of these

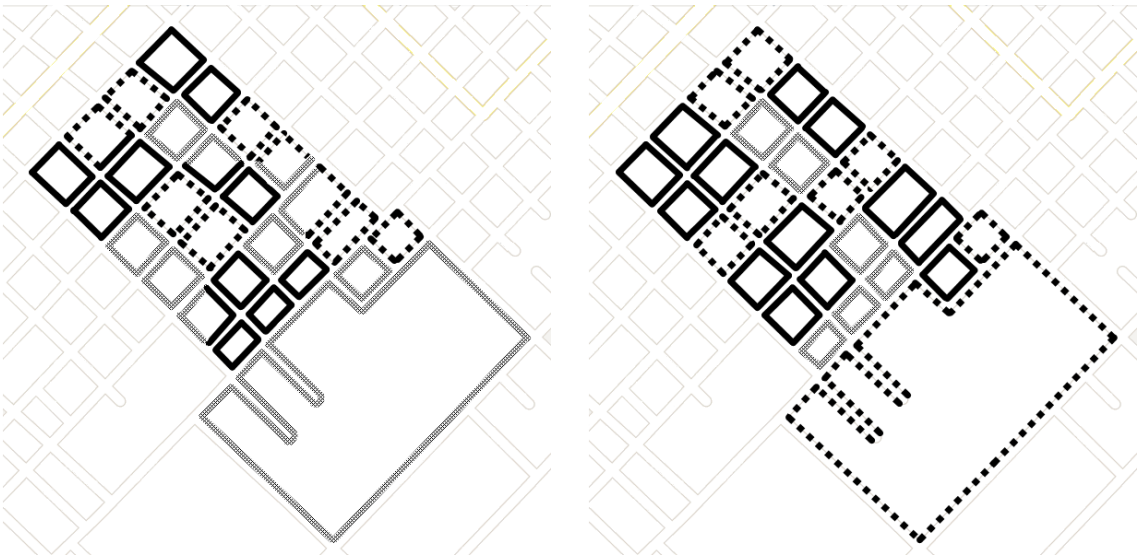

Figure 7: Segmentations generated by the procedure proposed in this work satisfy all constraints and tend to have more compact segments than manually-generated segmentations. The left figure shows the manual segmentation for a census tract in Azul City for the previous census in 2001, whereas the right figure shows the segmentation obtained by our computational procedure for the same input data.

tracts were poorly drawn in the sense that the inclusion of areas of widely varying densities should be avoided. These cases were ultimately the result of population growth.

A comparison with available data from the 2001 census shows that the manually-generated segmentations quite frequently violated the lower and upper bounds for the number of dwellings assigned to each census enumerator, a situation which did not hold for any of the automatically-solved census tracts in the 2010 census (i.e., 96 % of the census tracts). Furthermore, segments in automatically-generated segmentations tend to be more compact than those in manual segmentations. For example, Figure 7 shows both the manual segmentation for a census tract in Azul City in the 2001 census and a segmentation obtained with the tools described in this work. The automatically-generated segmentation contains much more complete blocks and satisfies all the constraints. This is a consequence of the greedy nature of the method employed by the manual operators, which tend to complete the "current segment" by including adjacent block sides without revising the previous decisions. In this sense, the lower and up-

per bounds on the number of dwellings per segment greatly constrain the operator's freedom to choose compact segments –even though they were frequently violated–, a situation that is better handled by the procedure proposed in this work.

Conclusions

We presented a computational tool that was used in the 2010 Argentinian Census to address the dwelling segmentation problem for the Province of Buenos Aires. We managed to complete the implementation in time to meet the two-month deadline established by the census planning process. Unlike the previous census, in which there were no rules or algorithmic instructions provided to the manual operators –hence different operators could generate very different segmentations for the same census tract–, the algorithm's automatic procedure generated uniform results for the entire province, thus ensuring similar workloads could be assigned to all census enumerators. Finally, processing times were also considerably reduced compared to the manual method.

The algorithm's performance proved to be very sensitive to the values chosen for its parameters. With the values presented in the appendix, the algorithm could arrive at solutions in a matter of seconds whereas poorly chosen values risked diminishing the feasibility of the model or leading to the generation of hundreds of thousands of segments, extending execution times to various hours. The categorization of the census tracts by population density greatly facilitated the identification of suitable parameter levels. The incorporation of a sequential procedure in the proposed algorithm was fundamental in guaranteeing the preferences regarding desirable segment characteristics were also met.

Despite these highly satisfactory results, it should be noted that the implemented algorithms provide a heuristic solution. An interesting task for future research would therefore be to explore an integrated integer programming model addressing all the problem characteristics. A natural generalization of the model presented in the appendix will contain a huge number of variables, hence a column generation approach should be considered. The major challenge involved in developing this approach would be to solve the column generation subproblem.

From a theoretical point of view, it would be interesting to explore the computational complexity of the segmentation problem for particular classes of instances. If the streets may define an arbitrary graph then the problem is NP-complete, but its computational complexity for more regular morphologies is, to the best of out knowledge, open.

Since the census tracts contained about 300 dwelling units and each enumerator had to be assigned a number of dwellings within a prespecified range, the quantity of enumerators per tract varied very little (from 8 to 10 approximately). The objective of minimizing their number would have therefore had little purpose and was not pursued. This issue was explored in [8], where indeed it was shown that by minimizing the number of enumerators just a 2% improvement could have been attained for high-density tracts, and a 5% improvement could have been attained for low-density tracts.

As regards the views of the segmentation algorithm's ultimate users, Mr. Fernando Aliaga, head of geographical information systems for the 2010 Census in the Province of Buenos Aires, stated that the "use of this computer tool allowed us to produce a homogeneous segmentation with uniform compactness criteria, unlike the manual segmentation method which depends in large measure on operator decisions"[1]. The census itself was conducted on October 27, 2010 and was pronounced an organizational success by the provincial authorities [13].

Acknowledgements

The authors would like to thank the *Dirección de Estadística* from the Province of Buenos Aires for the initiative of carrying out the dwelling segmentation with automatic techniques. The authors are also grateful to Mr. Fernando Aliaga, head of geographical information systems for the 2010 Census in the Province of Buenos Aires, and to his team for their collaboration in various key aspects of this study. The authors gratefully acknowledge Rodrigo Sotelo, Andrés Weintraub, Kenneth Rivkin, and Jaime Miranda for their insightful remarks on this paper. Finally, the authors would like to thank the annonymous reviewers, the Associate Editor, and the Editor-in-Chief for their constructive comments, which greatly helped to improve this work.

This project was partially funded by the Ministerio de Economía de la Provincia de Buenos

Aires. Partial funding was also provided by ANPCyT PICT-2007-00518, CONICET PIP 112-200901-00178, and UBACyT 20020090300094 and 20020100100980 (Argentina), as well as by FONDECyT 1110797 and the Millennium Science Institute "Complex Engineering Systems" (Chile).

Appendix: The segmentation algorithm

In this appendix we provide the full details of the algorithm for the dwelling segmentation problem in an individual census tract. We say that a segment is *exceeded* if it has more dwellings than the maximum allowed number of dwellings per segment or is longer than the upper bound L. The algorithm uses non-exceeded segments, but these may or may not be feasible depending on whether or not they satisfy the minimum number of dwellings per segment. When trying to construct a feasible solution only feasible segments are considered. By defining *adjacency type* $\delta \in \{0, 1, 2\}$ as in Figure 3, a segment is said to be δ-*connected* if it is connected by adjacencies of levels up to δ.

We set out the proposed segmentation procedure in Algorithm 1. The procedure uses geographical data on the census tract as inputs, and the value of $\delta \in \{0, 1, 2\}$ as a parameter. Denote as S_i the set of non-exceeded (though not necessarily feasible) segments that have no more than i blocks, $i \geq 1$. At the i-th iteration, Algorithm 1 tries to find a feasible segmentation using the subset S'_i of feasible segments from S_i. To this end, we solve an integer programming model trying to maximize the global compactness; we describe this model in the following section. The algorithm iterates until either a feasible solution is found or S_i does not vary with respect to S_{i-1}, or until a previously specified iteration limit MI is reached. As soon as it finds a feasible solution for the model for the given subset S'_i of segments, the algorithm terminates and returns the solution obtained.

We now describe the construction of the set S_i of non-exceeded feasible segments that have no more than i blocks. In lines 2–4, the algorithm generates a base set of segments S_b using complete block sides. For each block, all possible non-exceeded segments S_b contained within it are generated. If the integer programming procedure cannot find a solution with the feasible segments from S_b (i.e., the segments from S_b satisfying the lower bound on the

Algorithm 1 Segmentation algorithm for δ-connected segments.

1: $S_b \leftarrow \{\}$ *// base set*
2: **for each** block q **do**
3: $S_b \leftarrow S_b \cup$ { non-exceeded segments contained in q }
4: **end (for)**
5: $i \leftarrow 1$
6: $S_i \leftarrow S_b$
7: **repeat**
8: Run ILP segmentation model with all segments from S_i satisfying the constraint which imposes a minimum number of dwellings per segment (i.e., *feasible* segments)
9: **if** solution found **then**
10: End (with solution)
11: **end (if)**
12: $S_{i+1} \leftarrow S_i$
13: **for each** $(s_i, s_b) \in S_i \times S_b$ **do**
14: **if** $s_i \cup s_b$ is a non-exceeded δ-connected segment **then**
15: $S_{i+1} \leftarrow S_{i+1} \cup \{(s_i \cup s_b)\}$
16: **end (if)**
17: **end (for)**
18: **if** $S_{i+1} = S_i$ **then**
19: End (without solution)
20: **end (if)**
21: $i \leftarrow i + 1$
22: **until** $i > MI$
23: End (without solution)

number of dwellings per segment), then in lines 12–17 we add new segments to the current set, constructing S_{i+1}. In these lines, each existing segment $s \in S_i$ is connected with each base segment $t \in S_b$ from a neighbouring block, such that the resulting segment $s \cup t$ is a non-exceeded δ-connected segment. After line 17, the set S_{i+1} contains all non-exceeded δ-connected segments involving at most $i + 1$ blocks.

Since there is no formal objective function attached to the dwelling segmentation problem, the solution provided by Algorithm 1 may not be a good solution. However, the sequential procedure readily incorporates the compactness preferences of the selected segments, thus

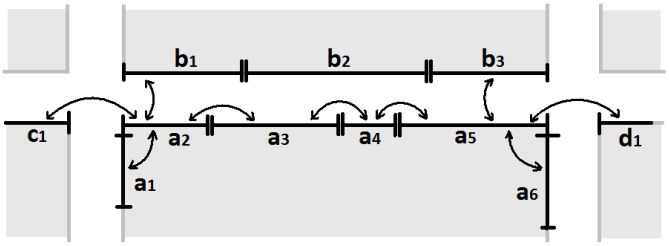

Figure 8: When there are divided block sides, a census enumerator can only cross a street at the end of a block side.

increasing the likelihood that the preferred solutions will be found first.

To find solutions with the most compact segments possible, we sequentially run Algorithm 1 for $\delta = 0, 1, 2$ and we interrupt the procedure as soon as the first feasible solution is found.

If the above process does not find a feasible solution, we activate the option of dividing block sides into parts. In this setting, the parameter P specifies the maximum number of dwellings a block side can have, with all sides exceeding this number then being divided into two or more parts. Each such part can have at most P dwellings. The block side division algorithm employs a greedy strategy, and attempts to keep all apartments in a single building together in the same part. Starting from one of the corners of the block side to be divided, this procedure constructs a part by advancing along that side until either an apartment building is encountered or P dwellings have been counted. In either case, construction of the block part is completed and a new one is started, the procedure being repeated until the algorithm obtains an entire set of parts for the divided side. The division algorithm also considers apartments included in buildings, by greedily trying to keep together apartments in the same floor. Finally, we specify new adjacencies for the divided sides as in Figure 8.

Note that the smaller is P, the greater will be both the number of block side parts in the tract and the size of the base set S_b.

Once the block sides have been divided as just described, we execute Algorithm 1 again for

$\delta = 0, 1, 2$ to find a feasible solution. If still no solution is found, the value of P is lowered and the process repeated. We set out the entire procedure in Algorithm 2. A list PL inputted to the algorithm contains the values for P, and the algorithm employs these values successively in order of decreasing size given that the greater is P, the less divided will be the solutions. If upon completion the procedure has not found a solution, it terminates and informs the user that no segmentation could be identified. Note that P is initially set at less than or equal to the maximum allowed number of dwellings per segment, but the parameter value will depend on the population density of the census tract to be segmented.

Algorithm 2 Segmentation algorithm

1: **for each** $P \in PL$ **do**
2: Divide sides with number of dwellings greater than P.
3: **for** $\delta = 0$ **to** 2 **do**
4: Run Algorithm 1 for δ
5: **if** Algorithm 1 found a solution **then**
6: Return solution and end
7: **end (if)**
8: **end (for)**
9: **end (for)**
10: End (without solution)

Improvements for low-density census tracts

As mentioned in *The segmentation algorithm*, the algorithm must be enhanced in order to successfully tackle low-density tracts. The difficulties can be sidestepped by modifying Algorithm 1 as follows:

- We treat blocks with few dwellings as a single non-exceeded segment for purposes of adding segments to the base set S_b (line 3). To this end, we add a parameter MP that sets the minimum number of dwellings a block must have before it can generate more than one base segment. If a block falls short of this number, only one segment for the

	Param.	High-density	Medium-density	Low-density
Maximum no. of iterations for generating segments	MI	4	7	9
Minimum no. of dwellings for dividing a block	MP	1	2	10
Minimum no. of dwellings in base segments (segments below minimum are grouped to form new ones)	MH	0	1	5
Maximum no. of dwellings per block side part	PL	[32, 16, 10]	[32, 16]	[40, 32, 20]
Execution time limit for ILP model (sec)	MT	60	60	120

Table 1: Algorithm parameter values by type of census tract.

complete block is added to S_b. This considerably reduces the size of S_b for low-density tracts.

- Once the base segments S_b are generated, if any of them have few dwellings they are arbitrarily combined with an adjacent segment to form a single base segment. To accomplish this, we add a parameter MH to set the minimum number of dwellings a segment must have to be added to S_b. We insert this treatment of S_b in Algorithm 1 following line 4.

The values of the various parameters for low-density tracts must be identified with particular care, as inappropriate choices may increase execution times (if P is very low, for example) or reduce the model's feasibility (if values MP and MH are too high). Table 1 gives the parameters values for each category. Recall that the categories are given by population density: high-density tracts involve up to 10 blocks, medium-density tracts involve 11 to 30 blocks, and low-density tracts involve more than 30 blocks.

The integer programming model

We now formulate an integer linear programming model for selecting a set of segments covering all the dwellings within a census tract. Let $\mathcal{R}$ be the census tract to be segmented and let S

be the (input) set of feasible segments to be considered. For each segment $s \in S$, introduce the binary variable x_s such that $x_s = 1$ if and only if segment s is included in the solution.

To maximize the compactness of the segments selected by the model, we specify the following objective function. Given a segment $s \in S$, we define its *compactness* to be $\text{comp}(s) = \frac{\text{sides}(s)}{\text{blocks}(s)}$, where sides($s$) and blocks($s$) are the number of sides and blocks, respectively, in segment s. Thus, a segment consisting of a single complete rectangular block will have a compactness of 4, while a segment containing four sides that are each from a different block will have a compactness of 1. On this definition, however, a segment comprised of one complete block will have the same compactness as a combination of two segments each made up of a half block (i.e., two sides of a block) or four segments each consisting of a quarter block (a single side). Clearly, the first of these three possibilities is the most desirable. To priorize segments that are compact in this desirable sense and avoid undesirable ones, define the *valuation* of a segment s as $\text{val}(s) = k^{\text{comp}(s)}$. Any value $k \geq 3$ is a reasonable choice for k. In this particular application $k = 10$ was chosen and provided good results, although in [8] it was shown *a posteriori* that $k = 3$ would have given better results for some census tracts.

Now let $\mathbb{V}$ be the set of dwellings in $\mathcal{R}$, and let $\mathbb{L}_0$ be the set of block sides without dwellings in $\mathcal{R}$. For each $v \in \mathbb{V}$ denote as $S_v \subseteq S$ the set of feasible segments that include dwelling v, and for each $l \in \mathbb{L}_0$ denote as $L_l \subseteq S$ the set of feasible segments that include side l. With these definitions, we can formulate a simple integer linear programming model for the segmentation problem:

$$\max \sum_{s \in S} \text{val(s)} \cdot x_s$$

$$\sum_{s \in S_v} x_s = 1 \qquad \forall v \in \mathbb{V} \tag{1}$$

$$\sum_{s \in L_l} x_s = 1 \qquad \forall l \in \mathbb{L}_0 \tag{2}$$

$$x_s \in \{0, 1\} \qquad \forall s \in S \tag{3}$$

As regards the constraints, (1) ensure that each dwelling is covered by exactly one segment while (2) guarantee that block sides with no dwellings are also covered by exactly one segment,

since sides with no dwellings must be visited by an enumerator. Recall that the model includes no preferences regarding adjacency levels for segments that cross streets (that is, that extend beyond a single block), which are addressed within the global procedure.

References

[1] Aliaga F (2010) Personal communication. Buenos Aires, Argentina (November 5, 2010).

[2] Altman M (1997) Is Automation the Answer: The Computational Complexity of Automated Redistricting. *Rutgers Computer and Law Technology Journal* **23**(1):81–141.

[3] Altman M, MacDonald K, McDonald MP (2005) From Crayons to Computers: The Evolution of Computer Use in Redistricting. *Social Science Computer Review* **23**(3):334–346.

[4] Altman, M. and McDonald, M.P. (2009) Bard: Better Automated Redistricting. *Journal of Statistical Software* **42**(4):1-28.

[5] Barnhart C, Johnson EL, Nemhauser G, Savelsbergh M, Vance P (1998) Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research* **46**(1):316–329.

[6] Bozkaya B, Erkut E, Laporte G (2003) A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research* **144**(1):12–26.

[7] IBM ILOG (2009) *User's Manual for CPLEX*.

[8] Fernández Slezak F (2012) Programación matemática para distribución eficiente de censistas en censos nacionales: el censo 2010 de la Provincia de Buenos Aires como caso de estudio (in Spanish). Degree Thesis in Mathematics, University of Buenos Aires, Argentina.

[9] Fleischmann B, Paraschis JN (1988) Solving a large scale districting problem: a case report. *Comput. Oper. Res.* **15**(6):521–533.

[10] Garfinkel RS, Nemhauser G (1970) Optimal Political Districting by Implicit Enumeration Techniques. *Management Science* **16**(8):B495–B508.

[11] Helbig RE, Orr PK, Roediger RR (1972) Political redistricting by computer. *Comm. ACM* **15**(8):735–741.

[12] Hess SW, Weaver JB, Siegfeldt HJ, Whelan JN, Zitlau PA (1965) Nonpartisan Political Redistricting by Computer. *Operations Research* **13**(6):998–1006.

[13] La voz de Tandil (2010). Se censó más del 95% de las viviendas en la provincia. Retrieved November 15^{th}, 2010.
`http://www.lavozdetandil.com.ar/ampliar_nota.php?id_n=20090`.