Math. Program., Ser. A (2015) 149:425–457
DOI 10.1007/s10107-014-0762-4

FULL LENGTH PAPER

Efficient geo-graph contiguity and hole algorithms for geographic zoning and dynamic plane graph partitioning

Douglas M. King · Sheldon H. Jacobson · Edward C. Sewell

Received: 16 January 2012 / Accepted: 3 February 2014 / Published online: 18 February 2014

Abstract Graph partitioning is an intractable problem that arises in many practical applications. Heuristics such as local search generate good (though suboptimal) solutions in limited time. Such heuristics must be able to explore the solution space quickly and, when the solution space is constrained, differentiate feasible solutions from infeasible ones. Geographic zoning problems allocate some resource (e.g., political representation, school enrollment, police patrols) to contiguous zones modeled by partitions of an embedded planar graph. Each vertex corresponds to an area of the plane (e.g., census block, town, county), and local search moves one area from its current zone to a different zone in each iteration. Enforcing contiguity constraints may require significant computation when the graph is large. While existing algorithms

This research has been supported in part by the National Science Foundation (IIS-0827540). The second author was also supported in part by the Air Force Office of Scientific Research (FA9550-10-1-0387). This material is based upon work supported in part while the second author served at the National Science Foundation. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, National Science Foundation, or the United States Government. The computational work was conducted with support from the Simulation and Optimization Laboratory at the University of Illinois.

D. M. King (✉)
Department of Industrial and Enterprise Systems Engineering,
University of Illinois at Urbana-Champaign, Urbana, IL, USA
e-mail: dmking@illinois.edu

S. H. Jacobson
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
e-mail: shj@illinois.edu

E. C. Sewell
Department of Mathematics and Statistics, Southern Illinois University Edwardsville,
Edwardsville, IL, USA
e-mail: esewell@siue.edu

require linear or polylogarithmic time (in the number of vertices) to assess contiguity in each local search iteration, the geo-graph paradigm shows how contiguity can be verified by examining only the set of vertices that border the transferred area (i.e., those areas whose boundaries share at least a single point with the boundary of the transferred area). This paper develops efficient algorithms that examine these vertices more quickly than traditional search-based methods, allowing practitioners to more fully consider their zoning options when creating zones with local search.

Keywords Graph partitioning · Planar graphs · Graph contiguity · Political districting

Mathematics Subject Classification (2000) 05C85, Graph algorithms · 05C40, Connectivity · 05C10, Planar graphs · 05A18, Partitions of sets

1 Introduction

Graph partitioning problems seek to divide the vertices of a graph, $G = (V, E)$, among disjoint subsets, $V_1, V_2, \ldots, V_k$, such that $V = \bigcup_{i=1}^{k} V_i$. Partitioning problems arise in a number of practical fields, including VLSI [1], image segmentation [29,35], and geographic districting [10,27]. Partitioning problems are typically expressed as an optimization problem, where the subsets are generated according to an objective and constraints that are relevant to the particular problem being considered. Problems in the VLSI domain typically seek to allocate the elements (vertices) of a system among clusters (subsets) while minimizing the amount of communication between clusters (i.e., edges with endpoints in different subsets) [1], while those in image segmentation may seek to group pixels (vertices) into regions (subsets) with similar characteristics [35]. Geographic districting problems seek to group small areas (vertices) into contiguous districts (subsets) that exhibit desirable properties (e.g., high election competitiveness, compactness of shape) [22].

While numerous graph partitioning problems exist, solving such problems tends to be an intractable task for large graphs. Therefore, heuristic methods are frequently used to generate solutions that are good but suboptimal. In the classical minimum-cut graph partitioning problem, the goal is to divide the vertex set into two subsets of equal size while minimizing the number of edges with endpoints in different subsets (i.e., the *cut* edges). This approach is consistent with the goals of VLSI partitioning if edges represent communication between two elements of a system. The Kernighan–Lin algorithm is a heuristic approach to finding an optimal (i.e., minimum cut) partition; it begins by dividing the vertices into two equal sized subsets, then identifies a sequence of vertex exchanges that reduce the number of cut edges [21]. The Fiduccia–Mattheyses algorithm is similar in structure, but replaces vertex exchanges with one-way migrations [14]. When vertices are transferred individually, the effect on the number of cut edges and the size of each subset can be assessed with little computation; for example, the only edges that can change from cut to uncut (or vice versa) are those with the transferred vertex as an endpoint. However, if the goals of the partitioning problem change such that each subset must induce a con-

nected subgraph in G (i.e., *contiguity constraints* are applied), assessing the effect on contiguity requires $O(|V_i|)$ time with traditional search-based methods, where V_i is the subset that is losing a vertex. When G contains a large number of vertices, the computation required to assess contiguity can be significant; more efficient algorithms are needed when contiguity constraints are applied to graph partitioning problems.

Contiguity constraints are common in graph partitioning problems when applied to geographic districting. In geographic districting problems, each vertex in the graph corresponds to a small geographic area or *unit*, and each edge links two units that share a common boundary (i.e., they are geographically adjacent). Districts are constructed through partitioning the vertex set, such that each subset corresponds to the collection of units that composes a particular district or *zone*. Typically, each zone must be a contiguous geographic area, and hence, the vertices of each subset must induce a connected subgraph. In practice, districting problems may need to consider graphs with large numbers of vertices. For example, United States Congressional Districts are constructed from census blocks whose populations are measured each decade during a national census. In the year 2000, California was divided into 533,163 census blocks, from which 53 districts were constructed, while Texas allocated its 675,062 census blocks among 32 districts. Among the 43 states apportioned multiple districts, Hawaii had both the fewest census blocks (18,990) and the fewest districts (two), still a relatively large zoning problem. Of the 43 multi-district states, all had more than 9,000 blocks per district, with an average of 22,516 blocks per district [33,34]. While these problem sizes may seem small when compared to problems in the VLSI domain, which may consider systems with millions of elements [1], the burden of requiring contiguous zones increases computation in local search approaches to geographic zoning. In these practical applications when V_i contains many vertices, assessing contiguity in $O(|V_i|)$ time can limit the decision-maker's ability to appropriately consider their options.

The *geo-graph* is a graph model specifically tailored for partitioning problems in geographic zoning that employ local search heuristics [22]. This model exploits the particular plane embedding of the graph, as prescribed by the unit boundaries. When all unit boundaries of an n-unit region are drawn on the plane, they divide the plane into $n+1$ areas, with n areas representing the units and the last area encompassing the infinite region of the plane that lies outside these units. If this drawing of the boundaries is interpreted as a plane graph, its *dual* defines the adjacencies of the units: each unit has a vertex embedded in the interior of its area, and two vertices are joined by an edge for each segment on their shared boundary [37]. In addition, the dual embeds a vertex in the infinite area outside the units, and its incident edges are similarly defined. While this final vertex does not participate directly in the zoning process (i.e., it will not be assigned to any zone), it plays an important role in the definition of the geo-graph by defining the area outside the zoned region. This vertex will be referred to as v_0; when discussing the vertex set of a geo-graph in proofs and definitions, V_0 will be used in cases that include v_0, and V will be used in cases when v_0 is excluded. For example, Fig. 1 depicts the translation of unit boundaries into the dual graph representing unit adjacencies. Figure 1a shows a drawing of unit boundaries on the plane; if this drawing is interpreted as a plane graph, then Fig. 1b superimposes its dual graph (with edges

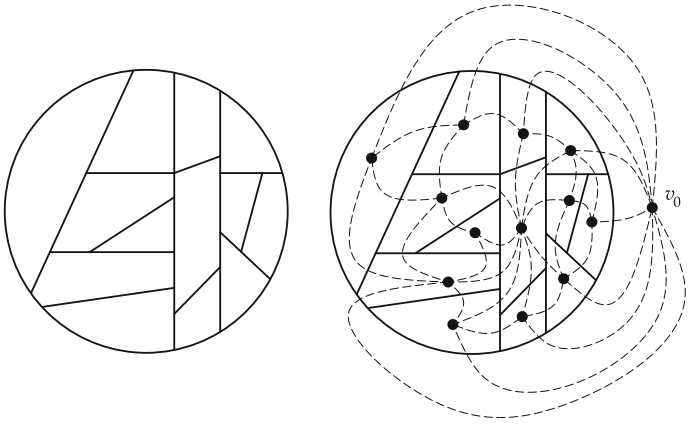

Fig. 1 A set of example unit boundaries and their associated dual graph. **a** Unit boundaries drawn on the plane. **b** Unit boundaries with their dual graph superimposed (vertex v_0 represents the infinite area outside the units)

drawn as dashed lines) on the same drawing. The edges of this dual graph connect units that share a common boundary segment, and hence, link adjacent units.

A geo-graph, $G = (V, E, B, z)$, extends the vertex and edge sets of this graph with two functions: a *boundary function*, B, and a *zoning function*, z. The boundary function associates each vertex, $v \in V_0$, with the single simple closed curve, $B(v)$, that defines the boundary of its associated unit. These boundary curves implicitly define the edges in E, which define the neighborhood of each vertex, given as $N(v) \equiv \{x \in V_0 : vx \in E\}$. However, by definition, this neighborhood excludes vertices whose units share only isolated points on $B(v)$. The *augmented neighborhood* extends the neighborhood to include these point-adjacent units, as shown in Definition 1.

Definition 1 Let $G = (V, E, B, z)$ be a geo-graph. For any $v \in V_0$, let the *augmented neighbors* of v, $R(v) \equiv \{x \in V_0 : B(v) \cap B(x) \neq \emptyset\}$, be the set of units whose boundaries share at least an isolated point with the boundary curve of unit v. A sequence of vertices, $S = (s_1, s_2, \ldots, s_k)$, is called a *strand* if $s_i \in R(s_{i-1})$ for $i = \{2, 3, \ldots, k\}$.

Consider the highlighted unit in Fig. 2a; the units contained in its augmented neighborhood are shown in Fig. 2b, such that $R(v) = \{v_A, v_B, v_C, v_D, v_E, v_F, v_G, v_H, v_I, v_J\}$, whereas $N(v) = R(v) - v_A$, since $B(v_A)$ shares only a single isolated point on $B(v)$. If the units are arranged in a two-dimensional rectangular grid, as in Fig. 3a, the highlighted unit has $R(v) = \{v_A, v_B, v_C, v_D, v_E, v_F, v_G, v_H\}$ (as shown in Fig. 3b), with $N(v) = \{v_B, v_D, v_E, v_G\}$, since the other four units in $R(v)$ only intersect $B(v)$ at one of its corner points.

While the point-adjacent units included in $R(v)$ should not be considered adjacent for the purpose of assessing zone contiguity, they contain all of the vertices visited as one travels around the boundary curve $B(v)$. Conceptually, for any two vertices, $x_1, x_2 \in R(v)$, one can travel from x_1 to x_2 around the outside of unit v along two distinct *perimeters* on v, where each perimeter is a sequence of vertices

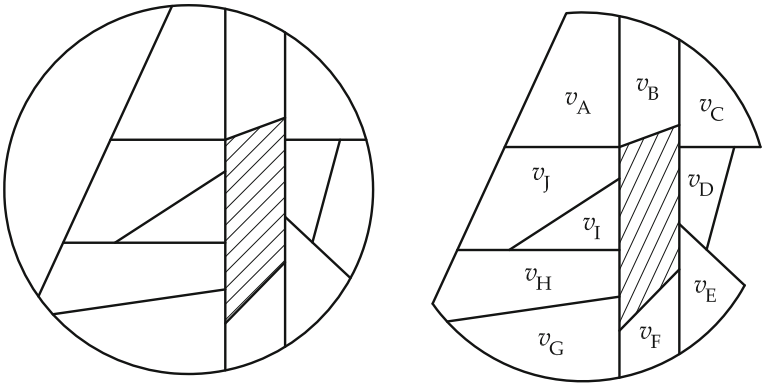

Fig. 2 The augmented neighborhood for a unit from the region depicted in Fig. 1. **a** The region from Fig. 1 with one unit highlighted. **b** Units in the augmented neighborhood, $R(v)$, of the highlighted unit

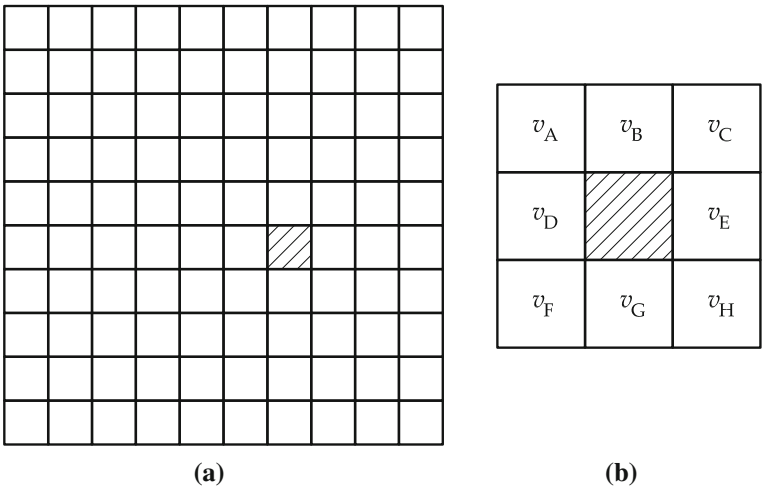

Fig. 3 The augmented neighborhood for a unit in a grid region. **a** Unit boundaries of a grid region, with one unit highlighted. **b** Units in the augmented neighborhood, $R(v)$, of the highlighted unit

encountered as one travels around the boundary curve $B(v)$. A formal definition of these perimeters is given in [22]. To demonstrate the construction of these perimeters, consider the highlighted unit depicted in Fig. 2b; the two (v_A, v_F)-perimeters on v are $(v_A, v_B, v_C, v_D, v_E, v_F)$ traveling clockwise and $(v_A, v_J, v_I, v_H, v_G, v_F)$ traveling counterclockwise.

The zoning function assigns each vertex to exactly one zone; the number of zones is assumed to be a constant positive integer $m(G)$ such that the set of zone labels is $M(G) \equiv \{1, 2, \ldots m(G)\}$ and $z(v) \in M(G)$ for each $v \in V$. Furthermore, vertex $v_0 \in V_0$ is assigned to the dummy zone 0, such that $M_0(G) \equiv M(G) \cup \{0\}$. The set of vertices in any set of zones $J \subseteq M_0(G)$ is defined by $V(J) \equiv \{v \in V_0 : z(v) \in J\}$; when

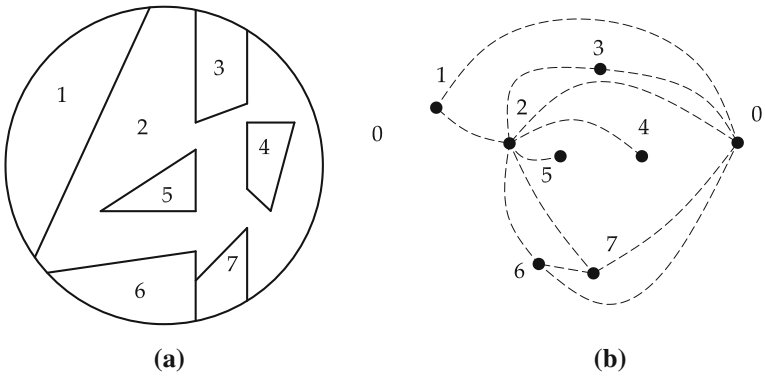

Fig. 4 A possible set of zones for the units depicted in Fig. 1. **a** A possible set of zones. **b** The auxiliary graph, $H(G)$, for this set of zones

$j \in M_0(G)$ is a singleton, $V(j)$ is used rather than $V(\{j\})$. To emphasize connectivity, a geo-graph is termed *zone-connected* if the subgraph induced by V_j is connected for every $j \in M(G)$. These zone assignments can decompose the neighbor set for $v \in V$ into the neighbors belonging to each zone, such that $N_j(v) \equiv N(v) \cap V(j)$. Each zoning function defines a particular partition of V, where $V = \bigcup_{j=1}^{m(G)} V(j)$, and $V(j_1) \cap V(j_2) = \emptyset$ for any two distinct $j_1, j_2 \in M(G)$.

The geo-graph summarizes zone-level adjacencies with an *auxiliary graph*, $H(G) = (V', E')$, whose vertex set is the set of zone labels in G (i.e., $V' \equiv M_0(G)$), and whose edges link zones with boundaries that share at least a single common point. Since the zone boundaries are composed of the boundaries of the units they encompass, these edges can be identified by unit boundary curves, as shown in Definition 2.

Definition 2 For any geo-graph, G, let $H(G) = (V', E')$ be the auxiliary graph summarizing the zone-level adjacencies of G. Each vertex in $V' \equiv M_0(G)$ corresponds to a zone of G, and the edge set $E' \equiv \{j_1 j_2 \in V' \times V' : \exists x_1, x_2 \in V_0, z(x_1) \in V(j_1), z(x_2) \in V(j_2), x_1 \in R(x_2)\}$ contains an edge between each pair of zones whose boundaries share at least one point in common.

Figure 4a demonstrates one way to partition the units in Fig. 1a into seven zones. The auxiliary graph for these zone assignments is shown in Fig. 4b. When considering the grid region depicted in Fig. 3a, one way to partition its units into eight zones is shown in Fig. 5a, with the associated auxiliary graph depicted in Fig. 5b. Note that both examples also include zone 0, which represents the infinite area that lies outside the unit boundaries.

This auxiliary graph is similar to a quotient graph [2], with the exception that its edges are defined by augmented neighbors in different zones, rather than the standard neighbors. A key result is that $H(G)$ can be used to identify any *holes* present in the zones [22]. A zone is said to have a hole if it is possible to draw a simple closed curve through the interior of the zone, such that this curve encloses one or more points that are not part of that zone. Due to the structure of the geo-graph, any such points must belong to another zone; if all zones are contiguous (i.e., G is zone-connected), then the

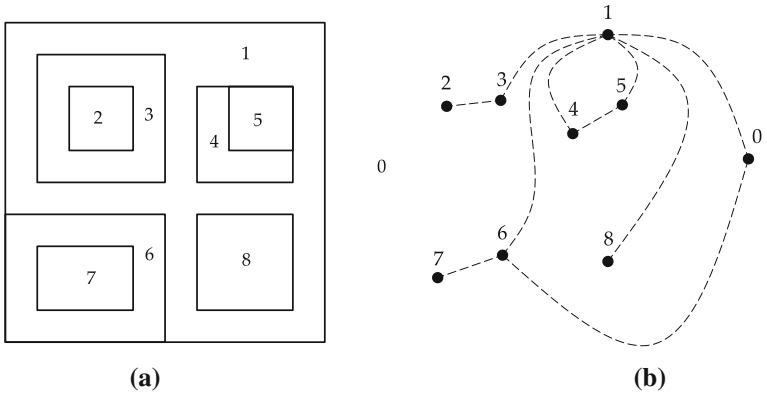

Fig. 5 A possible set of zones for the units depicted in Fig. 3. **a** A possible set of zones. **b** The auxiliary graph, $H(G)$, for this set of zones

entire other zone must be enclosed by this curve. In this case, the zone residing in this hole is *surrounded* by the zone containing the closed curve. For any zone $j \in M(G)$, the set of all zones surrounded by zone j is represented by $\Pi(j) \subseteq M(G) - j$.

Two zones in $\Pi(j)$ are said to be in the same *pocket* of zone j if any simple closed curve through the interior of zone j that encloses one of these zones must also enclose the other zone. For example, consider the zones depicted in Fig. 5a. Any curve drawn through the interior of zone 1 that encloses either zone 4 or zone 5 must enclose both, and hence, zones 4 and 5 are contained in the same pocket of zone 1. In contrast, while zones 4 and 8 are both surrounded by zone 1, it is possible to draw a curve through the interior of zone 1 that encloses zone 4 but not zone 8; therefore, these zones are not in the same pocket of zone 1. This definition of pockets allows $\Pi(j)$ to be decomposed into subsets (collectively called the *pocket set* of j) corresponding to the pockets of zone j. The number of pockets of zone j is given by $\pi(j)$, and its pocket set is defined by $\Pi_1(j), \Pi_2(j), \ldots, \Pi_{\pi(j)}(j)$. This pocket set can be derived directly from the auxiliary graph, $H(G)$, per Lemma 1. A simple procedure for determining all pockets of zone j is to employ a graph search on the subgraph induced by $V' - j$ in $H(G)$, which can be executed in $O(m(G)^2)$ time, since there can be as many as $\binom{m(G)+1}{2}$ edges in E'. By extension, assessing the holes and pockets of all zones can be done in $O(m(G)^3)$ time by executing a separate search for each of the $m(G)$ zones.

Lemma 1 ([22]) *Let $G = (V, E, B, z)$ be a zone-connected geo-graph, with associated auxiliary graph $H(G) = (V', E')$. For any zone $j \in M(G)$, let $Y_0, Y_1, Y_2, \ldots Y_k$ be the components of the subgraph induced by $V' - j$ in $H(G)$, with $0 \in Y_0$. Then zone j has $\pi(j) = k$ pockets, with the pocket set defined by $\Pi_i(j) = Y_i$ for $1 \leq i \leq \pi(j)$.*

Consider the zones depicted in Fig. 4a. One can draw a simple closed curve through the interior of zone 2 such that zone 4 is enclosed, and hence, zone 4 is surrounded by zone 2. Zone 5 is also surrounded by zone 2. These conclusions are consistent with analysis of the auxiliary graph, $H(G)$, of these zones; Fig. 6a shows the auxiliary graph of these zones with zone 2 removed. This removal splits $H(G)$ into three components: one component contains all of the zones not surrounded by zone 2 (i.e., zones 0, 1, 3, 6,

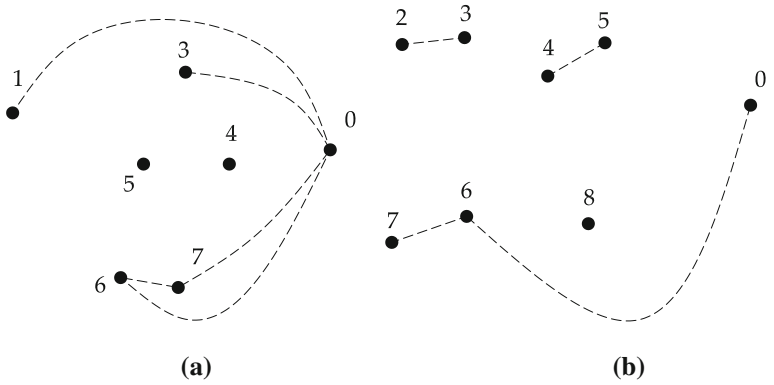

Fig. 6 Identifying holes for the zones in Figs. 4b and 5b by analyzing their auxiliary graphs. **a** The auxiliary graph in Fig. 4b with zone 2 removed. **b** The auxiliary graph in Fig. 5b with zone 1 removed

and 7), while zones 4 and 5 are each in separate components, and hence, are surrounded by zone 2 according to Lemma 1. Furthermore, Lemma 1 indicates that zones 4 and 5 are each in different pockets of zone 2 (i.e., $\Pi_1(2) = \{4\}$ and $\Pi_2(2) = \{5\}$), which is consistent with the fact that one can draw a simple closed curve through the interior of zone 2 that only encloses one of these two zones. Similarly, Fig. 6b shows the auxiliary graph of the zones in Fig. 5a when zone 1 is removed. The remaining subgraph splits the other zones into four components. Zones 0, 6, and 7 are not surrounded by zone 1, while the other zones are divided among three pockets (i.e., $\Pi_1(1) = \{2, 3\}$, $\Pi_2(1) = \{4, 5\}$, and $\Pi_3(1) = \{8\}$).

While identifying zone holes may be of interest to some zone designers, who may wish to prevent their creation [9,25,31], they can also simplify contiguity assessments during local search. Typical local search approaches to geographic zoning transfer a single unit from its current zone to a different zone in each iteration [5,26,27]. In such approaches, local search must be able to quickly determine whether this transfer causes any zone to become discontiguous. While assessing contiguity of the zone that gains the unit can be accomplished by verifying that the transferred unit has at least one neighbor in its new zone, contiguity of the zone that is losing this unit can be more difficult to assess. The geo-graph allows the impact on contiguity of both zones to be assessed by examining only the set of units in $R(v)$, per Theorem 1 [22]. One simple procedure for carrying out these examinations is to execute one simple search on $R(v)$ for each pocket of zone $z(v)$; each search removes the vertices of one pocket from $R(v)$ (i.e., $R(v) \cap V(M_0(G) - \Pi_j(z(v))) = R(v) - V(\Pi_j(z(v)))$) and determines whether $N_{z(v)}(v)$ is contained in a single component of the remaining subgraph. This simple procedure can be carried out in $O(m(G)|R(v)|)$ time.

Theorem 1 ([22]) *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$. The subgraph induced by $V(z(v)) - v$ is connected if and only if $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(\Pi(z(v)) \cup \{z(v)\})$, and $N_{z(v)}(v)$ is contained in one component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ for every $j \in \{1, 2, \ldots, \pi(z(v))\}$.*

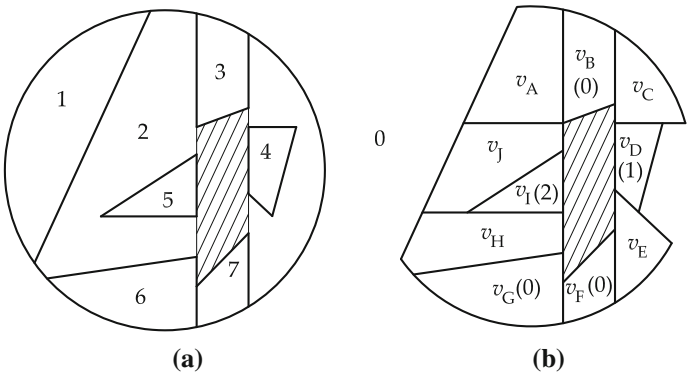

Fig. 7 Assessing contiguity of the zone 2 in Fig. 4a after the removal of a unit, v, by analyzing the augmented neighborhood, $R(v)$. **a** Removal of the highlighted unit disconnects zone 2 in Fig. 4a. **b** The augmented neighborhood, $R(v)$, of this unit; each unit is labeled with the pocket of zone 2 that contains it

Consider the highlighted unit depicted in Fig. 7a; it is clear that removing this unit from zone 2 causes this zone to become discontiguous. Theorem 1 allows one to come to this conclusion by only examining the units in the augmented neighborhood of this unit, which are shown in Fig. 7b. These units are given vertex names, and any augmented neighbor not in zone 2 is labeled with a number in parentheses; this number indicates which pocket of zone 2 contains this unit (e.g., unit v_D has the label "(1)", since it is contained in pocket 1 of zone 2), with the label "(0)" given to units that are not in any pocket of zone 2. This augmented neighborhood has $N_2(v) = \{v_C, v_E, v_H, v_J\}$, which are the neighbors of v in its own zone (i.e., zone 2). Since zone 2 has two pockets, Theorem 1 considers three conditions on $R(v)$. One condition asks whether $N_2(v)$ is contained in a single component of the subgraph induced by the augmented neighbors in $R(v)$ after all units not in a pocket of zone 2 are removed. This condition is not true, since removing the vertices labeled with "(0)" eliminates all paths between vertices $v_C, v_J \in N_2(v)$. Hence, Theorem 1 concludes that removing this unit causes zone 2 to become discontiguous.

In contrast, consider the highlighted unit depicted in Fig. 8a. This unit can be removed from zone 1 without violating a contiguity constraint. Its augmented neighborhood is shown in Fig. 8b, with notation following from the previous example. This unit has $N_1(v) = \{v_B, v_D, v_E, v_G\}$; since zone 1 has three pockets, Theorem 1 considers four conditions on $R(v)$; each asks whether $N_1(v)$ is contained in a single component of some subgraph induced by $R(v)$ after removing one of the units adjacent to a corner of $B(v)$, since units in three corners each reside in a different pocket of zone 1 (i.e., the northwest, northeast, and southeast corners), while the southwest corner unit does not reside in any pocket of zone 1. Removing any one of these corner vertices does not split $N_1(v)$ among multiple components of the remaining subgraph of $R(v)$. For example, one condition of Theorem 1 asks whether all vertices from pocket 2 can be removed from $R(v)$ without splitting $N_1(v)$ among multiple remaining components; removing the only such vertex (v_C) from $R(v)$ leaves the remainder of $R(v)$ in a single component. All other conditions are similarly satisfied, and hence, Theorem 1 indicates that the highlighted vertex can be removed from zone 1.

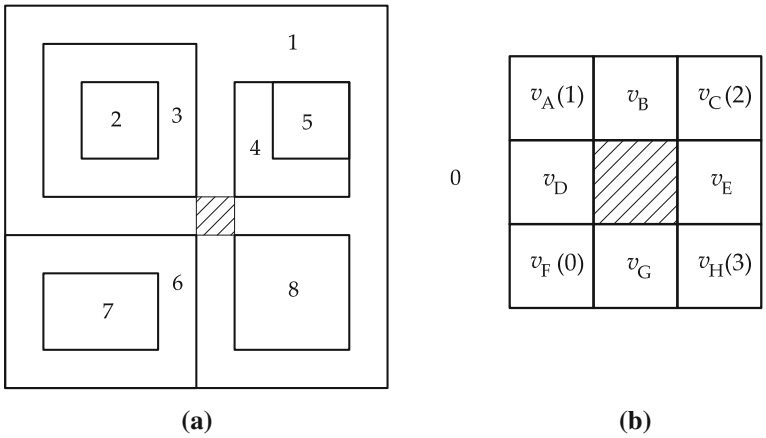

Fig. 8 Assessing contiguity of the zone 1 in Fig. 5a after the removal of a unit, v, by analyzing the augmented neighborhood, $R(v)$. **a** Removal of the highlighted unit leaves zone 1 contiguous in Fig. 5a. **b** The augmented neighborhood, $R(v)$, of this unit; each unit is labeled with the pocket of zone 1 that contains it

The value of Theorem 1 is that it examines only the vertices in $R(v)$, rather than all of the vertices in the zone $z(v)$, when assessing the contiguity of zone $z(v)$ after unit v is removed. The size of $R(v)$ will typically be small in practical applications. For example, if census blocks were arranged in a rectangular two-dimensional grid pattern, then $|R(v)| \leq 8$, independent of the grid size. Consider the process of creating United States Congressional Districts in Kansas after the 2000 census, where 173,107 census blocks (i.e., units) were grouped into four districts (i.e., zones), for an average of 43,276 census blocks per district [33,34]. On average, $R(v)$ contains only 7.09 census blocks in this graph [22]. Moreover, numerical experiments in Kansas have shown that simple geo-graph algorithms that execute a separate search on $R(v)$ for each pocket to assess the conditions proposed by Theorem 1 can reduce computation (measured by number of edges visited) and execution time by three orders of magnitude on average when compared to traditional search methods on the entire zone; the geo-graph approach required 0.00289 s to assess contiguity for 1,940 census blocks (visiting an average of 7.92 edges in each assessment), while traditional breadth-first search required 2.96 s to assess contiguity for the same census blocks (visiting an average of 11,449 edges per assessment) [22].

While simple algorithms have been proposed to identify all pockets of the zones in a geo-graph in $O(m(G)^3)$ time and assess zone contiguity during local search in $O(m(G)|R(v)|)$ time [22], this paper introduces efficient algorithms that further reduce computation by eliminating redundancies that naturally arise in these simple algorithms. For example, consider the analysis of the augmented neighborhood in Fig. 8a using the conditions established in Theorem 1. Assessing these conditions requires the investigation of four separate subgraphs of $R(v)$, each independently removing one of the corner units shown in Fig. 8a. Each removal leaves the remainder of $R(v)$ as a single path with two endpoints in $N_1(v)$; for example, removing v_C leaves a single path from v_B to v_E. Every edge in $R(v)$ will then be visited in all but one

of the subgraphs analyzed in Theorem 1. The main contributions of this paper are algorithms that identify all pockets of all zones in a geo-graph in $O(m(G)^2)$ time and assess zone contiguity during local search in $O(|R(v)|)$ time. Alternative methods involving dynamic graphs can assess contiguity in time that is polylogarithmic in $|V_i|$, where V_i is the set of units in the zone that is losing vertex v, but their time complexities remain dependent on the number of vertices in a zone (see Sect. 2). Geo-graph algorithms avoid this dependency by providing algorithms that can assess contiguity by only examining the vertices in $R(v)$. While the actual run times of these algorithms will depend on the specific problem instances to which they are applied, geo-graph algorithms are likely to perform well if $|R(v)|$ does not increase substantially with the size of the graph.

This paper is structured as follows. Section 2 discusses existing approaches to contiguity constrained graph partitioning, focusing on algorithms from dynamic graph theory. Section 3 presents an efficient algorithm for identifying pockets using an algorithm based on the dominators of the auxiliary graph $H(G)$. Section 4 discusses ways to eliminate the redundant exploration that arises when assessing each condition of Theorem 1 in a separate search, and proposes an efficient algorithm that eliminates such redundancies. Finally, Sect. 5 discusses the role that these algorithms play in practical political districting problems. An "Appendix" contains existing lemmas from King et al. [22] that are applied in the proofs presented in this paper. Where relevant, all graph definitions are taken from West [36]; for example, the term "components" refers to connected components, and the term "path" refers to a simple path where no vertex is visited more than once.

2 Background

Geo-graph partitioning is very similar to general graph partitioning with connectivity constraints. For example, Chlebíková [8] presents a 4/3-approximate algorithm for partitioning a graph into two connected subsets with an objective of balancing the weight of these subsets by maximizing the minimum weight between them; the weight of a subset is defined as the sum of the (non-negative) weights of its vertices. When the graph is partitioned into three subsets, Salgado and Wakabayashi [28] provide a 2-approximate algorithm. These partitioning problems are typically NP-hard. Becker et al. [3] restrict analysis to rectangular grids and present an approximate algorithm that runs in $O(|V|^3 p^2)$ time, where p is the desired number of partite sets; this problem remains NP-hard even when the grid has only two rows. The balance objective applied to partitioning problems is similar in form to population-balancing constraints that are often applied to political districting problems. However, districting problems often seek to optimize additional characteristics of the districts, such as compactness of their shape or competitiveness of the resulting elections. The structure of the plane graph represented by a geo-graph is more general than a grid graph, but more specific than a general graph that may not be planar. The geo-graph model is ideal for geographic zoning, as it exploits the plane nature of the graph while remaining compatible with the diverse set of objectives that may be of interest, such as election competitiveness, compactness of shape, and fairness to natural communities of interest [7].

When local search partitions a geo-graph $G = (V, E, B, z)$, Theorem 1 demonstrates that enforcing contiguity constraints requires analysis of only the vertices in $R(v)$ when a single vertex $v \in V$ is transferred from one zone to another in each iteration. This kind of approach, where vertices are transferred between zones one-at-a-time, is similar to previous work in the field of *dynamic graphs*. In general, a dynamic graph is one that evolves over time as edges and vertices are added or removed. Dynamic graph algorithms track the structure of the evolving graph so that queries about its properties can be answered more quickly than assessing them from scratch [11,17]. In general, dynamic graphs support two categories of operations: *updates* that alter the structure of the graph and *queries* that inquire about the properties of the graph.

A number of dynamic graph algorithms have been proposed in the literature; these algorithms vary in the type of graphs on which they operate (e.g., planar, directed), how these graphs can be modified (e.g., edge deletion and addition, vertex deletion and addition), and the graph properties that can be tracked and queried (e.g., connectivity, bipartiteness, minimum spanning tree). In general, algorithms that allow edges to be inserted and deleted are termed *fully dynamic*, while those that allow for edges to be inserted or deleted (but not both) are termed *partially dynamic* [11]; fully dynamic algorithms that also permit insertion and deletion of vertices are termed *completely dynamic* [16].

Geographic zoning applications emphasize zone contiguity, and hence, dynamic graphs that support connectivity queries are most relevant. For a particular pair of vertices $v_1, v_2 \in V$ in a graph $G = (V, E)$, such queries are of the form "is there a path from v_1 to v_2 in G?" Reif [24] develops a partially dynamic algorithm (supporting edge deletions) that updates G in $O(g|V| + |V| \log |V|)$ time (where g is the genus of G) and queries connectivity in $O(1)$ time; Henzinger and King [18] improve these times with a fully dynamic algorithm that requires $O(|V|^{1/3} \log |V|)$ amortized time per update and $O(1)$ time per connectivity query. Henzinger and King [17] develop a fully dynamic algorithm that updates G in $O(\log^3 |V|)$ amortized time and queries connectivity in $O(\log |V| / \log \log |V|)$ time; Holm et al. [19] improve updates to $O(\log^2 |V|)$ time, while Thorup [32] improves updates to $O(\log |V| (\log \log |V|)^3)$ time and connectivity queries to $O(\log |V| / \log \log \log |V|)$ time. By restricting analysis to planar graphs, Eppstein et al. [12] develop a fully dynamic algorithm that updates in $O(\log |V|)$ time for edge insertions, $O(\log^2 |V|)$ time for edge deletions, and $O(\log |V|)$ time for connectivity queries. For plane graphs, Eppstein et al. [13] maintain a spanning forest in $O(\log |V|)$ time for each update and query operation, where updates can change the weight of an existing edge, remove an edge, or add an edge that is compatible with the embedding.

Frigioni and Italiano [16] consider an alternative formulation for dynamic planar graphs. They consider planar graphs that are subgraphs of some global graph. Any vertices or edges that are added to the graph must appear in this global graph. For example, such graphs can represent communication networks where processors (vertices) and links (edges) can undergo failure and repair. This behavior, called *switching*, is such that operational processors and links are considered *on*, and those that have failed are considered *off*. The connectivity of the communication network is equivalent to the connectivity of the subgraph induced by the graph elements that are switched on. Connectivity information can be updated and queried in $O(\log^3 n)$ time [16].

The partitioning process modeled by a geo-graph, $G = (V, E, B, z)$ shares many similarities to the switching model, in that each zone comprises a subgraph of G. Adding or removing units from this zone is equivalent to switching those units on or off; this model can maintain connectivity about $m(G)$ zones using $m(G)$ separate subgraphs. However, these subgraphs do not evolve independently; each vertex is switched on in exactly one subgraph, so when a vertex is switched on in one subgraph it must be simultaneously switched off in another. A closer relationship is revealed by combining these subgraphs into a single switching model that allows units to be switched to more than two possible settings. For example, consider $m(G)$ possible switch settings where the units switched to setting k are those that compose zone k. The mapping of units to switch settings is equivalent to z, the zoning function of the geo-graph. Therefore, when one unit is transferred from its current zone to another zone in each iteration, zoning can be modeled as the *dynamic partitioning* of a static graph. In such a problem, a query operation would correspond to asking "can vertex v be transferred to zone j without violating contiguity constraints?" and an update operation would carry out such a transfer. Using the algorithms that will be presented in this paper, queries can be carried out in $O(|R(v)|)$ time and update operations can be executed in $O(|R(v)| + m(G)^2)$ time. An update operation comprises two steps: updating the adjacency matrix of $H(G)$ requires $O(|R(v)|)$ time, with $O(m(G)^2)$ additional time required to update the pockets of all zones if edges are added to or removed from $H(G)$.

3 Efficient pocket algorithm

In a geo-graph, G, the pockets of all zones are defined by the structure of the auxiliary graph, $H(G) = (V', E')$, as shown in Lemma 1. As vertices migrate between zones during local search, edges will be added and removed from $H(G)$, and hence, pockets must be reassessed as $H(G)$ evolves. A trivial method for reassessing the pockets of each zone would be to enumerate the components of the subgraph induced by $V' - j$ in $H(G)$ using graph search for each zone $j \in V'$. Since $|V'| = m(G) + 1$ and $|E'| \leq m(G)^2$, this approach would require $O(m(G)^3)$ time. This section uses graph *dominators* to identify the pockets of all zones in $O(m(G)^2)$ time.

For an undirected graph, $G = (V, E)$ with a specified root vertex $r \in V$, vertex $x_1 \in V$ is said to *dominate* vertex $x_2 \in V - x_1$ if every x_2, r-path must pass through x_1 [23]. These domination relationships can be efficiently expressed in a *dominator tree* rooted at r; for any distinct vertices $x_1, x_2 \in V$, x_1 dominates x_2 if and only if x_1 is an ancestor of x_2 in the dominator tree [23]. An efficient algorithm has been developed to construct the dominator tree of G in $O(|E|)$ time [6]. Dominators have been applied in a number of areas to identify graph elements whose removal breaks a component of a graph into multiple components, such as identifying the strong articulation points and strong bridges in directed graphs [15]. By Lemma 1, constructing the dominator tree of $H(G)$ (rooted at 0) can efficiently identify all surrounded zones; for any two zones $x_1, x_2 \in V'$, zone x_1 surrounds zone x_2 if and only if x_1 is an ancestor of x_2 in the dominator tree. Therefore, the pockets of zone x_1 consist entirely of its descendants in the dominator tree.

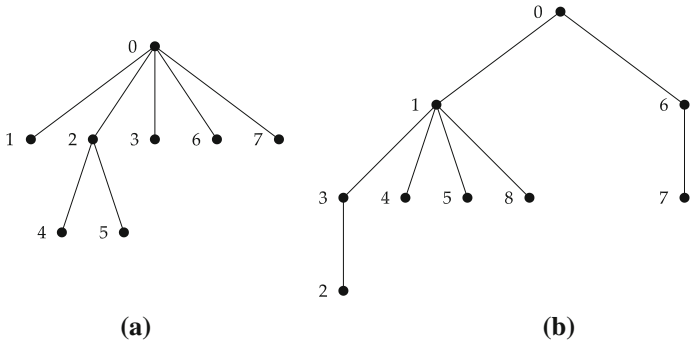

Fig. 9 Dominator trees for the auxiliary graphs shown in Figs. 4b and 5b. **a** Dominator tree for the auxiliary graph in Fig. 4b. **b** Dominator tree for the auxiliary graph in Fig. 5b

After constructing the dominator tree of $H(G)$, the only remaining task is to classify which descendants appear in each pocket of x_1. Descendants are classified in two stages, beginning with the immediate descendants of x_1, and then extending these classifications to descendants further down in the tree. First, Lemma 2 shows that identifying the components of the subgraph induced by the immediate descendants of zone $j \in V'$ can identify which of these descendants appear in each pocket of j. Once the immediate descendants of zone j have been assigned to the pockets of j, Lemma 3 shows that each remaining descendant of j can be classified based on which immediate descendant of j appears on its vertex-to-root path in the dominator tree. For example, Figs. 9a and 9b show the dominator trees of the zones depicted in Figs. 4a and 5a, respectively. Consider zone 1 in Fig. 9b, whose immediate descendants are $D = \{3, 4, 5, 8\}$. The zones in D induce three components in $H(G)$ (Fig. 5b): zones 3 and 8 each reside in their own component, while zones 4 and 5 reside in the same component. Zone 2 is a more distant descendant of zone 1; iterating up its vertex-to-root path in the dominator tree shows that it resides in the same pocket as zone 3. This analysis yields three pockets of zone 1: $\Pi_1(1) = \{2, 3\}$, $\Pi_2(1) = \{4, 5\}$, and $\Pi_3(1) = \{8\}$, which is consistent with the definition of pockets, as well as the components of the subgraph induced by $V' - 1$ in $H(G)$ as shown in Fig. 6b.

Lemma 2 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph, with associated auxiliary graph $H(G) = (V', E')$. Let D be the set of immediate descendants of zone j in the dominator tree of $H(G)$ rooted at 0. Zones $x_1, x_2 \in D$ are in the same pocket of j if and only if there is an x_1,x_2-path on the subgraph induced by D in $H(G)$.*

Proof of Lemma 2 Sufficiency: Suppose no x_1,x_2-path exists on the subgraph induced by D in $H(G)$. Let P be an x_1, x_2-path on the subgraph induced by $V' - j$ in $H(G)$. All of the zones on P are surrounded by zone j, and hence, are descendants of j in the dominator tree. Let $v \in P$ be a zone visited by P that is not an immediate descendant of j, and let v' be the ancestor of v that is an immediate descendant of j. Since v' can only appear on P a maximum of one time, then P contains either an x_1, v-path on $H(G)$ that does not pass through v' or an x_2, v-path on $H(G)$ that does not pass

through v' (or both). Assume, without loss of generality, that P contains an x_1, v-path that does not pass through v' (call it P_I). Since x_1 is not a descendant of v' in the dominator tree, then there is a $0, x_1$-path on $H(G)$ that does not pass through v' (call it P_{II}). Appending P_I to the end of P_{II} provides a $0, v$-walk on $H(G)$, which contains a $0, v$-path on $H(G)$, neither of which contain v'. This contradicts the fact that every $0, v$-path on $H(G)$ must contain v', since v' is an ancestor of v in the dominator tree. *Necessity*: Since j cannot appear on the x_1, x_2-path on the subgraph induced by D, both x_1 and x_2 are in the same component of the subgraph induced by $V' - j$ in $H(G)$. Hence, they must appear in the same pocket of zone j by Lemma 1. □

Lemma 3 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph, with associated auxiliary graph $H(G) = (V', E')$. Let u and j be ancestors of x on the dominator tree of $H(G)$, such that j is the immediate ancestor of u. Both $u, x \in \Pi_k(j)$ for some $k \in \{1, 2, \ldots, \pi(j)\}$.*

Proof of Lemma 3 Let P be an $x, 0$-path on $H(G)$. It must be that u and j appear on P, since they are dominators of x. Moreover, it must be that this path visits u before it visits j, since j is a dominator of u. Therefore, P contains an x, u-path that does not visit j, and so u and x are in the same component of the subgraph induced by $V' - j$ in $H(G)$. If k is chosen such that $u \in \Pi_k(j)$, Lemma 1 shows that $x \in \Pi_k(j)$. □

Based on Lemmas 2 and 3, Algorithm 1 depicts a procedure that can identify all pockets of all zones. The goal of this algorithm is to produce an $m(G) \times m(G)$ matrix, $pmatrix$, where element (i, j) is equal to 0 if zone j is not surrounded by zone i, and is equal to positive integer k if zone j is contained in pocket k of zone i. Storing these pocket assignments in a matrix allows them to be accessed in $O(1)$ time, which is necessary to execute the efficient contiguity algorithm discussed in Sect. 4. Lemma 4 further demonstrates that this algorithm runs in $O(m(G)^2)$ time.

Lemma 4 *Algorithm 1 runs in $O(m(G)^2)$ time.*

Proof of Lemma 4 This algorithm performs four tasks in sequence:

1. Line 1 requires $O(m(G)^2)$ time to initialize $pmatrix$, since $pmatrix$ contains $m(G)^2$ entries.
2. Line 2 requires $O(|E'|)$ time to construct the dominator tree using the algorithm presented in [6].
3. Lines 3–8 identify the connected components of the subgraph induced by the immediate descendants of each zone (except zone 0) in the dominator tree. Each zone can only be the immediate descendant of one zone in the dominator tree, and hence, identifying the components of all such sets of immediate descendants with simple graph search will not visit any edge in E' more than twice. Hence, Lines 3–8 can be executed in $O(|E'|)$ time.
4. Lines 9–12 consider each zone k_2 and iterate up its vertex-to-root path in the dominator tree and assign zone k_2 to the appropriate pocket of each zone encountered along this path. No path can contain more than $m(G) + 1$ zones, and hence, Lines 9–12 can be executed in $O(m(G)^2)$ time.

Since $|E'| \leq m(G)^2$, these four tasks can be executed in $O(m(G)^2)$ time. □

Algorithm 1: Efficient algorithm to determine the pockets of all zones in geo-graph G

Input : The auxiliary graph, $H(G) = (V', E')$, of geo-graph G
Output: Matrix $pmatrix$ of size $m(G) \times m(G)$

```
1  Initialize pmatrix to contain all zeros;
2  Construct the dominator tree of H(G);
   /* Classify immediate descendants into pockets, by Lemma 2        */
3  for k1 ← 1 to m(G) do
4      Let D ⊆ V' − k1 be the immediate descendants of k1 in the dominator tree;
5      Let {D1, D2, ..., Dπ(k1)} be the components of the subgraph induced by D in H(G);
6      for d ∈ D do
7          Let i be the index of component Di containing d;
8          pmatrix[k1, d] ← i;
   /* Classify distant descendants into pockets, by Lemma 3          */
9  for k2 ← 1 to m(G) do
10     Let P = (p1, p2, ..., pγ) be the vertex-to-root path of k2 on the dominator tree;
11     for i ← 2 to γ do
12         pmatrix[pi, k2] ← pmatrix[pi, pi−1];
```

Algorithm 1 stores the pocket assignments for each zone in an $m(G) \times m(G)$ matrix, which allows these assignments to be queried in $O(1)$ time, but also requires $O(m(G)^2)$ preprocessing time as shown in Lemma 4. An alternative approach can avoid matrix storage, instead determining pocket assignments by querying the dominator tree directly. For any two zones $j_1, j_2 \in V'$, zone j_1 is surrounded by zone j_2 if and only if zone j_2 is its ancestor in the dominator tree rooted at zone 0. Moreover, if zone j_1 is a descendant of zone j_2, its pocket assignment can be determined by identifying the pocket assignment of zone j_3, which is both an ancestor of zone j_1 and an immediate descendant of zone j_2, per Lemma 3. Therefore, once the dominator tree has been constructed and the immediate descendants of each zone have been classified into its pockets (requiring $O(|E'|)$ time, as shown in the proof of Lemma 4), the pocket assignments of all zones can be found by identifying *level ancestors* in the dominator tree. The level ancestor of zone j at depth d in a tree is denoted $LA(j, d)$, where a zone has depth d if it is a distance d from the root in the tree [4]. Hence, if zone j_2 resides at depth d_2 in the dominator tree, then zone j_1 is surrounded by zone j_2 if and only if $LA(j_1, d_2) = j_2$; moreover, if zone j_1 is surrounded by zone j_2, then the pocket of zone j_2 that contains it is the same pocket that contains zone $LA(j_1, d_2 + 1)$. Hence, querying a pocket assignment requires two level ancestor queries. Level ancestors can be queried in $O(1)$ time after $O(n)$ preprocessing time, where n is the number of vertices in the tree [4]. In the case of the dominator tree of $H(G)$, $n = m(G) + 1$, and hence, total preprocessing time for this approach is $O(|E'|)$ to construct the dominator tree of $H(G)$, classify the immediate descendants of each zone into its pockets, and perform preprocessing for the level ancestor algorithm. Querying pocket assignments can still be accomplished in $O(1)$ time. While this algorithm reduces the time complexity of the preprocessing step to $O(|E'|)$, the value of E' can change over the course of local search, as the structure of the auxiliary graph $H(G)$ evolves. There can be

as many as $\binom{m(G)+1}{2}$ edges in E', and hence, preprocessing still has time complexity $O(m(G)^2)$ in the worst case.

4 Efficient contiguity algorithm

Verifying the conditions of Theorem 1 with simple search requires searching up to $m(G)$ subgraphs of $R(v)$; since geo-graph G is planar, each search can be resolved in $O(|R(v)|)$ time, or $O(m(G)|R(v)|)$ time to complete all searches. Each of these searches begins with the vertices in $R(v)$, removes a subset of its vertices, and checks whether $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by the remaining vertices. The first condition of this theorem removes all vertices not contained in any pocket of zone $z(v)$, while the second condition separately removes all of the vertices from each pocket. These subgraphs may share many common vertices; every subgraph will contain the vertices of $R(v) \cap V(z(v))$, and any other vertex will appear in all but one of the subgraphs. Conceptually, verifying these conditions is similar to graph biconnectivity. A connected graph is called *biconnected* if and only if removing any single vertex does not split the remaining subgraph into more than one component. Similarly, the conditions of Theorem 1 are satisfied if and only if removing some subset of the vertices of $R(v)$ does not split $N_{z(v)}(v)$ among multiple components of the remaining subgraph. This section develops an algorithm to evaluate all of the conditions in Theorem 1 in $O(|R(v)|)$ time by transforming them into a single biconnectivity evaluation on a related graph called the *pocket graph*. The vertex set of this graph, $V^{\pi}(v) \equiv \{j \in \{0, 1, 2, \ldots, \pi(z(v))\} : V(\Pi_j(z(v))) \cap R(v) \neq \emptyset\}$ will be the set of pockets of zone $z(v)$ that contain vertices in $R(v)$. The structure of the pocket graph and the process by which it is constructed will also be discussed.

To simplify notation in the conditions of Theorem 1, the set of zones that are not surrounded by zone $z(v)$ are collected into a set called the *null pocket* of $z(v)$, $\Pi_0(z(v)) \equiv M_0(G) - (\Pi(z(v)) \cup \{z(v)\})$; the null pocket is equivalent to the connected component Y_0 in Lemma 1. Adopting this notation allows the two separate conditions of Theorem 1 to be rewritten as a single collection of conditions, as shown in Corollary 1; these conditions will be collectively termed the *pocket conditions*. While the null pocket is defined in opposition from the other pockets from a surroundedness perspective, considering all pockets together provides a convenient partition of $M_0(G)$ by noting that $M_0(G) = \{z(v)\} \cup (\bigcup_{j=0}^{\pi(z(v))} \Pi_j(z(v)))$ for any $v \in V$. Therefore, the term "pocket" in this section can refer to the null pocket unless explicitly stated otherwise, and the collective term "pockets" includes the null pocket.

Corollary 1 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$. The subgraph induced by $V(z(v)) - v$ is connected if and only if $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ for every $j \in \{0, 1, 2, \ldots, \pi(z(v))\}$.*

Proof of Corollary 1 Follows directly from Theorem 1 by noting that $M_0(G) - \Pi_0(z(v))$ is equivalent to $\Pi(z(v)) \cup \{z(v)\}$. □

The section discusses an efficient algorithm for testing whether removing unit $v \in V$ from its current zone, $z(v)$, causes that zone to become discontiguous. This algorithm

runs in $O(|R(v)|)$ time by assessing the conditions established in Corollary 1. This section is structured as follows. Section 4.1 discusses the process of combining the augmented neighbors of v that are in zone $z(v)$ into connected *neighbor components*. Section 4.2 makes important observations on the composition of perimeters of v, and their links to the individual pockets of $z(v)$ whose vertices appear in $R(v)$. Section 4.3 applies these links to individually examine the pockets of $z(v)$ whose vertices appear in $R(v)$ and create their associated *neighbor pockets*. Section 4.4 examines the relationships between the neighbor pockets by analyzing the pocket graph. Finally, Sect. 4.5 combines the results from the previous sections to develop the efficient contiguity algorithm. In addition to formal proofs, some lemmas in this section also include a *Proof Sketch* that discusses the main idea applied in the proof.

4.1 Neighbor components

Each pocket condition removes the vertices of one pocket from $R(v)$; any vertices in zone $z(v)$ will be present in the subgraph investigated by every pocket condition. If each pocket condition is verified by executing a simple search on the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$, then edges whose endpoints are both in $R(v) \cap V(z(v))$ may be visited during every pocket condition. To eliminate these redundant visits, the vertices of $R(v) \cap V(z(v))$ are grouped into connected *neighbor components*. Each neighbor component can then be treated as a single meta-vertex in later examinations. Therefore, the set of pockets neighboring each neighbor component (as well as the set of edges from each neighbor component into each pocket) are also needed.

Definition 3 Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$. Let $T_1(v), T_2(v), \ldots, T_{\alpha(v)}(v)$ be the components of the subgraph induced by $R(v) \cap V(z(v))$, termed the *neighbor components* of v. Furthermore, order these components such that $T_i \cap N_{z(v)}(v) \neq \emptyset$ for $i = 1, 2, \ldots, \alpha'(v)$ and $T_i \cap N_{z(v)}(v) = \emptyset$ for $i = \alpha'(v) + 1, \alpha'(v) + 2, \ldots, \alpha(v)$ for some $0 \leq \alpha'(v) \leq \alpha(v)$. For any $k \in \{1, 2, \ldots, \alpha(v)\}$, define:

1. $J_k(v) \equiv \{j \in V^\pi(v) : \exists x_1, x_2 \in R(v), x_1 \in T_k(v), x_2 \in V(\Pi_j(z(v)))\}$ as the set of pockets of zone $z(v)$ present in $R(v)$ that are adjacent to $T_k(v)$
2. For each $j \in J_k(v)$, define $E_{j,k}(v) \equiv \{xy \in E : x \in T_k(v), y \in R(v) \cap V(\Pi_j(z(v)))\}$ to be the set of edges with one endpoint in $T_k(V)$ and the other endpoint in $R(v) \cap \Pi_j(z(v))$.

Consider the augmented neighborhoods depicted in Fig. 7b. Based on the zone assignments shown in Fig. 4a, this unit resides in zone 2. Each unit in the augmented neighborhood that is not in zone 2 is labeled with the pocket that contains it (e.g., unit v_I resides in pocket 2). Therefore, $R(v) \cap V(z(v)) = \{v_A, v_C, v_E, v_H, v_J\}$, which induces three components: $T_1(v) = \{v_A, v_J, v_H\}$, $T_2(v) = \{v_C\}$, and $T_3(v) = \{v_E\}$. All three neighbor pockets contain a unit in $N_2(v)$, and hence, $\alpha(v) = \alpha'(v) = 3$. Considering $T_1(v)$, the pockets that are adjacent to this neighbor pocket are $J_1(v) = \{0, 2\}$, and the edge sets from $T_1(v)$ into each pocket are given by $E_{0,1}(v) = \{v_A v_B, v_H v_G\}$ and $E_{2,1}(v) = \{v_J v_I, v_H v_I\}$. The second neighbor component, $T_2(v)$, has $J_2(v) = \{0, 1\}$

with $E_{0,2}(v) = \{v_C v_B\}$ and $E_{1,2}(v) = \{v_C v_D\}$. The third neighbor component, $T_3(v)$, has $J_3(v) = \{0, 1\}$ with $E_{0,3}(v) = \{v_E v_F\}$ and $E_{1,3}(v) = \{v_E v_D\}$.

From Definition 3, it is clear that if $\alpha'(v)$ is equal to one, then all pocket conditions are satisfied, since $R(v) \cap V(z(v)) \subseteq R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ for every $j \in \{0, 1, 2, \ldots, \pi(z(v))\}$. Furthermore, if $\alpha'(v) = 0$, then v is the only unit in zone $z(v)$, since G is zone-connected; its removal would eliminate zone $z(v)$, and therefore such a transition should not be allowed. Hence, only cases where $\alpha'(v) \geq 2$ need to be considered.

4.2 Perimeters

The perimeters of unit v describe the sequences of units encountered while traveling around the boundary of unit v between two of its augmented neighbors. The goal of this section is to link the pocket conditions of Corollary 1 to the compositions of these perimeters. The key result (Lemma 7) is that, if the pocket condition of pocket $j \in V^{\pi}(v)$ is satisfied, then one can construct a path between any two augmented neighbors of v in pocket j of zone $z(v)$ that only passes through vertices that are either (a) in pocket j of zone $z(v)$, or (b) in zone $z(v)$. This result will be applied in the next section to individually examine the pockets of zone $z(v)$. A key observation that leads to this result is that the vertices in each pocket exhibit a transitive property, as shown in Lemma 5.

Lemma 5 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph. For any zone $k \in M(G)$, $j \in \{0, 1, 2, \ldots, \pi(k)\}$, and vertices $v_1, v_2 \in V(M(G) - k)$ such that $v_1 \in R(v_2)$, if $v_1 \in V(\Pi_j(k))$, then $v_2 \in V(\Pi_j(k))$.*

Proof of Lemma 5 If $v_2 \in V(z(v_1))$, then $v_2 \in V(\Pi_j(k))$ by definition. Otherwise, since $v_1 \in R(v_2)$, zones $z(v_1)$ and $z(v_2)$ are adjacent in the auxiliary graph $H(G)$ by Definition 2, and hence, are in the same component of the subgraph induced by $V' - k$ in $H(G)$. Therefore, $z(v_2) \in \Pi_j(k)$ by Lemma 1. Therefore, $v_2 \in V(z(v_2)) \subseteq V(\Pi_j(k))$. □

One limitation of assessing each pocket condition independently is that individual vertices of $R(v)$ may appear in many or all of the subgraphs to be searched. In fact, verifying the pocket condition corresponding to pocket $j \in \{0, 1, 2, \ldots, \pi(z(v))\}$ requires searching a subgraph containing all of the vertices in $R(v)$ *except* those in $V(\Pi_j(z(v)))$. Lemma 6 translates this pocket condition into an equivalent condition on *only* the vertices in $V(\Pi_j(z(v)))$ by examining the perimeters of v.

Lemma 6 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph. For any $v \in V$ and $j \in V^{\pi}(v)$, $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ if and only if for every pair of vertices $x_1, x_2 \in R(v) \cap V(\Pi_j(z(v)))$ (including cases when $x_1 = x_2$) there is an (x_1, x_2)-perimeter on v, call it W, such that $W \cap N(v) \subseteq V(\Pi_j(z(v)))$.*

Proof of Lemma 6 Sufficiency: Let $v \in V$ and $j \in \{1, 2, \ldots, \pi(z(v))\}$. Suppose that there are two vertices, $x_1, x_2 \in R(v) \cap V(\Pi_j(z(v)))$ whose (x_1, x_2)-perimeters on

v are W_1 and W_2, and $W_i \cap N(v) \not\subseteq V(\Pi_j(z(v)))$ for $i = 1, 2$. For $i = 1, 2$, let $W_i' \subseteq W_i$ be the subsequence of W_i consisting of only vertices in $W_i \cap N(v)$ in addition to endpoints x_1 and x_2. Each W_i' is an x_1, x_2-strand. Let $w_i \in W_i'$ be the first vertex visited by W_i' such that $w_i \notin V(\Pi_j(z(v)))$. It must be that $z(w_i) = z(v)$, otherwise $w_i \in V(\Pi_j(z(v)))$ by Lemma 5, which contradicts the definition of w_i. Let P be any w_1, w_2-path on $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$, and let C be the cycle constructed by appending v to P. By this construction, x_1 and x_2 are on different (w_1, w_2)-perimeters on v. Since neither x_1 nor x_2 can appear on C, Lemma 16B states that $x_1 \in Int(C)$ and $x_2 \in Ext(C)$, or vice versa. If $j \neq 0$, this contradicts Lemma 17F, which states that if $x_1 \in Int(C)$, then $x_2 \in V(\Pi_j(z(v))) \subseteq Int(C)$. If $j = 0$, let S be a v_0, x_1-strand on $V(\Pi_0(z(v)))$. Let x_1' be the vertex visited after v_0 in this strand; $x_1' \notin Int(C)$ by Lemma 17C. Since $x_1' \notin R(v) \cap V(M_0(G) - \Pi_0(z(v)))$ and $C \subseteq R(v) \cap V(M_0(G) - \Pi_0(z(v)))$, it must be that $x_1' \in Ext(C)$. Therefore, by Lemma 17E, there must be some vertex $s \in S$ such that $s \in C$. However, this contradicts the fact that $C \cap S = \emptyset$, since $S \subseteq V(\Pi_0(z(v)))$ and $C \subset R(v) \cap V(M_0(G) - \Pi_0(z(v)))$.
Necessity: Suppose that $N_{z(v)}(v)$ is not contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$. Let $y_1, y_2 \in N_{z(v)}(v)$ be two vertices in different components of this subgraph, and let Y_1 and Y_2 be the two (y_1, y_2)-perimeters on v. For $i = 1, 2$, let $u_i \in Y_i$ be such that $u_i \in V(\Pi_j(z(v)))$; if such a vertex does not exist, then Y_i is a y_1,y_2-walk on $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ that contains a path on the same vertex set, which contradicts the supposition that these vertices are in different components. By this construction, y_1 and y_2 appear on different (u_1, u_2)-perimeters on v, which contradicts the fact that at least one perimeter (call it W) must have $W \cap N(v) \subseteq V(\Pi_j(z(v)))$. □

While Lemma 6 relates the pocket conditions to conditions on the perimeters of v, the sequence of units encountered on these perimeters may not be known for many applications. For example, geographic information system (GIS) software such as ArcGIS may be able to extract adjacency data for the basic units (e.g., census blocks, counties) that define the region being zoned, but may not be able to extract the order in which adjacent units are visited as one travels along a unit perimeter. To analyze the conditions of Lemma 6 using graph search, the following lemma translates this condition on perimeters to a condition on paths in $R(v)$.

Lemma 7 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph. For any $v \in V$ and $j \in V^{\pi}(v)$, if $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$, then for every pair of vertices $x_1, x_2 \in R(v) \cap V(\Pi_j(z(v)))$, there is an x_1, x_2-path, P, such that $P \subseteq R(v) \cap V(\Pi_j(z(v)) \cup \{z(v)\})$.*

Proof of Lemma 7 Choose any $x_1, x_2 \in R(v) \cap V(\Pi_j(z(v)))$. By Lemma 6, one of the (x_1, x_2)-perimeters on v (call it W) has $W \cap N(v) \subseteq N(v) \cap V(\Pi_j(z(v)))$. An example is shown in Fig. 10, where W passes clockwise around the hatched unit, and $\{x_1, w_1, w_2, w_3, w_4, w_5, x_2\}$ are the units in $W \cap N(v)$. The unlabeled wedge-shaped areas are the other units that appear on W but not in $W \cap N(v)$. Let $L \subseteq B(v)$ be the portion of the closed curve $B(v)$ that corresponds to the construction of perimeter W. If L is a single point, then $W \subseteq R(x_1)$, and hence, $W - V(z(v)) \subseteq V(\Pi_j(z(v)))$ by Lemma 5. If L consists of more than a single point, choose any $w \in W - N(v)$ and

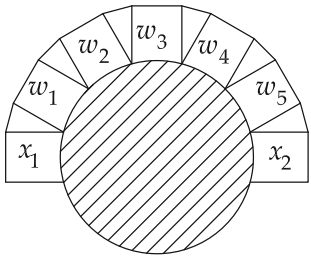

Fig. 10 Example (x_1, x_2)-perimeter on v in Lemma 7

let $b_w \in B(w) \cap L$, then choose $x' \in W \cap N(v) \subseteq N(v) \cap V(\Pi_j(z(v)))$ such that $b_w \in B(x')$; such a point b_w and vertex x' must exist by the construction of perimeter W. Therefore, $B(w) \cap B(x') \neq \emptyset$, and hence, $w \in R(x')$; if $w \notin V(z(v))$, then $w \in V(\Pi_j(z(v)))$ by Lemma 5. Therefore, either $w \in V(z(v))$ or $w \in V(\Pi_j(z(v)))$. For any $w \in W$, either $w \in W \cap N(v)$ (in which case $w \in V(\Pi_j(z(v)))$) or $w \notin W \cap N(v)$ (in which case $w \in V(z(v))$ or $w \in V(\Pi_j(z(v)))$). Since $W \subseteq R(v)$, then W is an x_1,x_2-walk on $R(v) \cap V(\Pi_j(z(v)) \cup \{z(v)\})$, which contains an x_1,x_2-path on the same vertex set. □

4.3 Neighbor pockets

This section individually examines each pocket $j \in V^\pi(v)$ by constructing the *neighbor pockets* of vertex v, defined as follows.

Definition 4 Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$. For any $j \in V^\pi(v)$, the *neighbor pocket* of v for pocket j is defined by $\Theta_j(v) = \{u \in R(v) : \text{for some } x \in R(v) \cap V(\Pi_j(z(v))) \text{ there is an } x, u\text{-path, } P, \text{ with } P \subseteq R(v) \cap V(\Pi_j(z(v)) \cup \{z(v)\})\}$.

For each $j \in V^\pi(v)$, the neighbor pocket $\Theta_j(v)$ is unique. Based on this definition, Lemma 7 immediately links the composition of a neighbor pocket to its pocket condition, as shown in Lemma 8. Moreover, this definition immediately implies that a neighbor pocket only contains vertices in pocket j of $z(v)$ and entire neighbor components of v, as shown in Lemma 9.

Lemma 8 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$. For any $j \in V^\pi(v)$, if $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$, then the subgraph induced by $\Theta_j(v)$ has a single component.*

Proof of Lemma 8 Let $x_1, x_2 \in R(v) \cap V(\Pi_j(z(v)))$. By Lemma 7, there is an x_1, x_2-path, P, such that $P \subseteq R(v) \cap V(\Pi_j(z(v)) \cup \{z(v)\})$, and therefore x_1 and x_2 are in the same component of $\Theta_j(v)$. Since this can be done for any choice of $x_1, x_2 \in R(v) \cap V(\Pi_j(z(v)))$, then $R(v) \cap V(\Pi_j(z(v)))$ must be contained in a single component of the subgraph induced by $\Theta_j(v)$. By definition, each component of the subgraph induced by $\Theta_j(v)$ must contain at least one vertex $x \in R(v) \cap V(\Pi_j(z(v)))$, and hence this subgraph cannot have more than one component. □

Lemma 9 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$. For any $k \in \{1, 2, \ldots, \alpha(v)\}$, let $t_k \in T_k(v)$. For any $j \in \{1, 2, \ldots, \pi(z(v))\}$, $t_k \in \Theta_j(v)$ if and only if $T_k(v) \subseteq \Theta_j(v)$.*

Proof Sketch: Apply the connectedness of $T_k(v)$ to extend inclusion of t_k in $\Theta_j(v)$ to the rest of $T_k(v)$.

Proof of Lemma 9 Sufficiency: Let $t' \in T_k(v)$ such that $t' \neq t_k$. Let P_1 be an x, t-path such that $x \in R(v) \cap V(\Pi_j(z(v)))$ and $P_1 \subseteq R(v) \cap V(\Pi_j(z(v)) \cup \{z(v)\})$ as provided by the definition of $\Theta_j(v)$, and let $P_2 \subseteq R(v) \cap V(z(v))$ be a t, t'-path, as provided by the definition of $T_k(v)$. Appending P_2 to the end of P_1 provides an x, t'-walk on $R(v) \cap V(\Pi_j(z(v)) \cup \{z(v)\})$, which contains an x, t'-path on the same vertex set. Therefore, $t' \in \Theta_j(v)$. Since this result holds for any $t' \in T_k(v)$, $T_k(v) \subseteq \Theta_j(v)$.
Necessity: Follows directly, since $t_k \in T_k(v)$ and $T_k(v) \subseteq \Theta_j(v)$. □

Consider the augmented neighborhood with three pockets depicted in Fig. 7b. Though only one vertex of pocket 1 (i.e, v_D) is contained in this augmented neighborhood, by Definition 4, the neighbor pocket of pocket 1 extends to include v_C and v_E, such that $\Theta_1(v) = \{v_C, v_D, v_E\}$. Consistent with Lemma 9, this neighbor pocket includes the entirety of neighbor components $T_2(v)$ and $T_3(v)$, as discussed in Sect. 4.1. Similarly, $\Theta_0(v) = \{v_A, v_B, v_C, v_E, v_F, v_G, v_H, v_J\}$, and $\Theta_2(v) = \{v_A, v_H, v_I, v_J\}$. Each of these neighbor pockets induces one component, so Lemma 8 cannot conclude that removing unit v from zone 2 (Fig. 4a) causes zone 2 to become discontiguous.

Consider the construction of $T_1(v), T_2(v), \ldots, T_{\alpha(v)}$; one straightforward way to construct these neighbor components is to choose an unvisited vertex in $R(v) \cap V(z(v))$ and execute a graph search algorithm that is allowed to visit vertices in $R(v) \cap V(z(v))$. Once this search terminates, this process repeats if there remain any unvisited vertices in $R(v) \cap V(z(v))$. Each search may encounter edges with one endpoint in the neighbor component and the other endpoint in a pocket of $z(v)$. Though the search should not visit the pocket vertex associated with this edge, the search process can store (1) the set of pockets encountered in this way, and (2) the set of edges from the neighbor component into each encountered pocket.

Maintaining the sets in Definition 3 is an important part of the efficient contiguity algorithm. For example, the Lemma 8 condition can be tested with search as follows. For any $j \in V^{\pi}(v)$, choose any vertex in $R(v) \cap V(\Pi_j(z(v)))$ and begin a search that can only visit vertices in $R(v) \cap V(\Pi_j(z(v)))$. However, the first time this search discovers an edge with one endpoint in $R(v) \cap V(\Pi_j(z(v)))$ and the other endpoint in $T_k(v)$ for some $k \in \{1, 2, \ldots, \alpha(v)\}$, this search can immediately add the edges in $E_{j,k}(v)$ to the search rather than exploring the vertices in $T_k(v)$ individually. If this search does not visit all of the vertices in $R(v) \cap V(\Pi_j(z(v)))$, then Lemma 8 implies that $N_{z(v)}(v)$ is not contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$, and Corollary 1 shows that removing v from its current zone causes that zone to become discontiguous.

From a computational perspective, one can determine how often each edge is visited while assessing Lemma 8 using graph search. When evaluating pocket $j \in V^{\pi}(v)$, this search will visit two types of edges: (1) those with both endpoints in $R(v) \cap V(\Pi_j(z(v)))$, and (2) those with one endpoint in $R(v) \cap V(\Pi_j(z(v)))$ and the other

endpoint in $T_k(v)$ for some $k \in \{1, 2, \ldots, \alpha(v)\}$. Each edge will be visited at most twice. Furthermore, each edge visited in this way will not be visited by the search for any other $j' \in V^\pi(v) - j$. Each of these edges appears in the subgraph induced by $R(v)$; by the planarity of G, the total number of edges visited will be $O(|R(v)|)$, and hence, all searches can be conducted in $O(|R(v)|)$ time. Furthermore, consider the case of using search to construct the neighbor components of v. For any $k \in \{1, 2, \ldots, \alpha(v)\}$, constructing neighbor component k will visit two types of edges: (1) those with both endpoints in $T_k(v)$, and (2) those with one endpoint in $T_k(v)$ and the other endpoint in $R(v) \cap V(\Pi_j(z(v)))$ for some $j \in V^\pi(v)$. Each edge will be visited at most twice, and any edge visited when constructing neighbor component $T_k(v)$ will not be visited when constructing any $T_{k'}(v)$ for $k' \in \{1, 2, \ldots, \alpha(v)\}$ and $k' \neq k$. Therefore, both the neighbor components and neighbor pockets can be constructed in $O(|R(v)|)$ time.

From Definition 3 and Lemma 9, each neighbor component will appear in one or more neighbor pockets; in fact, neighbor component $T_k(v)$ will appear in neighbor pocket $\Theta_j(v)$ for every $j \in J_k(v)$. Lemma 10 shows that if $\alpha'(v) \geq 2$ and neighbor component $T_k(v)$ appears in exactly one neighbor pocket for some $k \leq \alpha'(v)$, then the pocket condition corresponding to that neighbor pocket is violated in Corollary 1. Moreover, Lemma 11 shows that each neighbor pocket must contain at least one neighbor component $T_k(v)$ with $k \leq \alpha'(v)$.

Lemma 10 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$ and $\alpha'(v) \geq 2$. For any $k \in \{1, 2, \ldots, \alpha'(v)\}$, if $J_k(v) = \{j\}$ (i.e., $|J_k(v)| = 1$), then $N_{z(v)}(v)$ is not contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$.*

Proof Sketch: Every path leaving $T_k(v)$ must first enter $R(v) \cap V(\Pi_j(z(v)))$, so removing $V(\Pi_j(z(v)))$ from $R(v)$ isolates $T_k(v)$ from the other neighbor components.

Proof of Lemma 10 Let $J_k(v) = \{j\}$, and let $v_k \in N_{z(v)}(v) \cap T_k(v)$. Since $\alpha'(v) \geq 2$, choose any $k' \in \{1, 2, \ldots, \alpha'(v)\}$ such that $k' \neq k$ and let $v_{k'} \in N_{z(v)}(v) \cap T_{k'}(v)$. Let P be any $v_k, v_{k'}$-path on $R(v)$, and let x be the first vertex visited by P such that $x \notin T_k(v)$. Since $v_{k'} \notin T_k(v)$, such an x must exist, and furthermore, $x \in R(v) \cap V(\Pi_{j'}(z(v)))$ for some $j' \in V^\pi(v)$. Since x is adjacent to some vertex in $T_k(v)$, then $T_k(v) \subseteq \Theta_{j'}(v)$ by Lemma 9, and hence, $j' \in J_k(v)$. Since $J_k(v) = \{j\}$, therefore $j' = j$ and $x \in R(v) \cap V(\Pi_j(z(v)))$. Since such an $x \in R(v) \cap V(\Pi_j(z(v)))$ can be found for any $v_k, v_{k'}$-path on $R(v)$, then v_k and $v_{k'}$ cannot be in the same component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$. □

Lemma 11 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$. For any $j \in V^\pi(v)$, there is at least one $k \in \{1, 2, \ldots, \alpha'(v)\}$ such that $T_k(v) \subseteq \Theta_j(v)$.*

Proof Sketch: Beginning at any vertex in $R(v) \cap V(\Pi_j(z(v)))$, travel around the perimeter of v. The first vertex encountered in $N(v) - V(\Pi_j(z(v)))$ must appear in such a $T_k(v)$.

Proof of Lemma 11 Let $j \in V^\pi(v)$ and choose any $x_1 \in R(v) \cap V(\Pi_j(v))$ and $x_2 \in N_{z(v)}(v)$, with W being a (x_1, x_2)-perimeter on v. Let $x_2' \in W$ be the first vertex that W visits on $N_{z(v)}(v)$. Since $x_2' \in N_{z(v)}(v)$, it must be that $x_2' \in T_k(v)$ for some

$k \in \{1, 2, \ldots, \alpha'(v)\}$. Let W' be the subsequence of W that begins at x_1 and visits the vertices of $W \cap N(v)$, ending at x_2'. Note that W' is an x_1, x_2'-strand, and every vertex visited along W' is in $R(v) \cap V(\Pi_j(z(v)))$, except for x_2'. Let x_1' be the vertex that W' visits immediately before x_2', and let W'' be a shortest sequence from x_1' to x_2' on W. Let $x_2'' \in T_k(v)$ be the first vertex that W'' visits in $T_k(v)$, and let x_1'' be the vertex visited immediately before x_2''. Therefore, $x_1'' \in N(x_2'')$. By this construction, $B(x_1') \cap B(x_1'') \neq \emptyset$, and therefore $x_1'' \in V(\Pi_j(z(v)))$ by Lemma 5. By the definition of $\Theta_j(v)$, it must be that $x_1'' \in \Theta_j(v)$, and therefore $T_k(v) \in \Theta_j(v)$ by Lemma 9. □

4.4 Pocket graph

The previous section noted several cases when the pocket conditions of Corollary 1 are violated based on the composition of an individual neighbor pocket (i.e., if $\Theta_j(v)$ induces more than one component in $R(v)$ for some $j \in V^\pi(v)$) or a neighbor component (i.e., if $|J_k(v)| = 1$ for some $k \in \{1, 2, \ldots, \alpha'(v)\}$). However, these conditions do not exhaust all possible compositions of $R(v)$, since they examine each pocket in isolation. This section examines the relationships between pockets in $V^\pi(v)$ by constructing and analyzing a *pocket graph*. Its vertex set, $V^\pi(v)$, corresponds to the set of pockets with at least one vertex appearing in $R(v)$. The edges of the pocket graph correspond to pairs of neighbor pockets that contain a common neighbor component.

Definition 5 Let $G = (V, E, B, z)$ be a zone-connected geo-graph. Define the *pocket-graph* on vertex $v \in V$ as $G^\pi(v) = (V^\pi(v), E^\pi(v))$, such that $E^\pi(v) = \{j_1 j_2 \in V^\pi(v) \times V^\pi(v) : \exists k \in \{1, 2, \ldots, \alpha(v)\}, T_k(v) \subseteq \Theta_{j_1}(v)$ and $T_k(v) \subseteq \Theta_{j_2}(v)\}$.

The following two lemmas and theorem demonstrate the relationship between the pocket graph and the pocket conditions. Lemma 12 demonstrates how edges in the pocket graph are associated with paths between vertices in the associated neighbor pockets, while Lemma 13 shows how any individual pocket condition is not satisfied if and only if the vertex corresponding to its neighbor pocket is not a cut-vertex of the pocket graph. Finally, Theorem 2 shows that satisfying all pocket conditions is equivalent to the biconnectivity of the pocket graph.

Lemma 12 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$, and the subgraph induced by $\Theta_j(v)$ has one component for every $j \in V^\pi(v)$. For $j_1, j_2 \in V^\pi(v)$, $j_1 j_2 \in E^\pi(v)$ if and only if, for every $x_1 \in \Theta_{j_1}(v)$ and $x_2 \in \Theta_{j_2}(v)$, there is an x_1, x_2-path in the subgraph induced by $\Theta_{j_1}(v) \cup \Theta_{j_2}(v)$.*

Proof Sketch: Since $j_1 j_2 \in E^\pi(v)$ if and only if neighbor pockets j_1 and j_2 share a neighbor component, show that it is possible for a path to pass from $\Theta_{j_1}(v)$ directly into $\Theta_{j_2}(v)$ if and only if such a shared neighbor component exists.

Proof of Lemma 12 Sufficiency: Choose any $x_1 \in \Theta_{j_1}(v)$ and $x_2 \in \Theta_{j_2}(v)$. Let $k \in \{1, 2, \ldots, \alpha(v)\}$ be such that $T_k(v) \in \Theta_{j_1}(v)$ and $T_k(v) \in \Theta_{j_2}(v)$ by the definition of the pocket graph, and choose any $t_k \in T_k(v)$. Let P_1 be an x_1, t_k-path on the vertices of $\Theta_{j_1}(v)$, and let P_2 be a t_k, x_2-path on the vertices of $\Theta_{j_2}(v)$. Appending P_2 to the

end of P_1 provides an x_1, x_2-walk on the vertex set $\Theta_{j_1}(v) \cup \Theta_{j_2}(v)$, which contains an x_1, x_2-path that appears in the subgraph induced by the same vertex set.
Necessity: Choose $x_1 \in \Theta_{j_1}(v)$ and $x_2 \in \Theta_{j_2}(v)$, and let P be an x_1, x_2-path on the subgraph induced by $\Theta_{j_1}(v) \cup \Theta_{j_2}(v)$. Suppose that no such $x' \in P$ exists such that $x' \in \Theta_{j_1}(v) \cap \Theta_{j_2}(v)$. Let $x'_2 \in P$ be the first vertex that P visits such that $x'_2 \in \Theta_{j_2}(v)$, and let $x'_1 \in \Theta_{j_1}(v)$ be the vertex that P visits immediately before x'_2. It must be that $x'_1 \notin R(v) \cap V(z(v))$, otherwise $x'_1 \in \Theta_{j_2}(v)$ by definition. Similarly, it must be that $x'_2 \notin R(v) \cap V(z(v))$. Therefore, $x'_1 \in R(v) \cap V(\Pi_{j_1}(z(v)))$ and $x'_2 \in R(v) \cap V(\Pi_{j_2}(z(v)))$, which contradicts Lemma 5. Therefore, there is an $x' \in P$ such that $x' \in \Theta_{j_1}(v) \cap \Theta_{j_2}(v)$. It must be that $x' \in R(v) \cap V(z(v))$, since $\Theta_{j_1}(v) \subseteq R(v) \cap V(\Pi_{j_1}(z(v)) \cup \{z(v)\})$ and $\Theta_{j_2}(v) \subseteq R(v) \cap V(\Pi_{j_2}(z(v)) \cup \{z(v)\})$, and $V(\Pi_{j_1}(z(v))) \cap V(\Pi_{j_2}(z(v))) = \emptyset$. Therefore, $x' \in T_k(v)$ for some $k \in \{1, 2, \ldots, \alpha(v)\}$, which implies that $T_k(v) \subseteq \Theta_{j_1}(v)$ and $T_k(v) \subseteq \Theta_{j_2}(v)$ by Lemma 9, and therefore $j_1 j_2 \in E^\pi(v)$ by the definition of the pocket graph. □

Lemma 13 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$, $\alpha'(v) \geq 2$, and $|J_k(v)| \geq 2$ for every $k \in \{1, 2, \ldots, \alpha'(v)\}$, and the subgraph induced by $\Theta_j(v)$ has one component for every $j \in V^\pi(v)$. For any $j \in V^\pi(v)$, $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ if and only if the subgraph induced by $V^\pi(v) - j$ in $G^\pi(v)$ is connected.*

Proof Sketch: Iteratively apply the results of Lemma 12 to consider paths on $G^\pi(v)$ with length greater than one.

Proof of Lemma 13 Sufficiency: Choose any $j_1, j_2 \in V^\pi(v) - j$. By Lemma 11, there are $k_1, k_2 \in \{1, 2, \ldots, \alpha'(v)\}$ such that $T_{k_1}(v) \subseteq \Theta_{j_1}(v)$ and $T_{k_2}(v) \subseteq \Theta_{j_2}(v)$. Choose any $x_1 \in T_{k_1}(v) \cap N(v)$ and $x_2 \in T_{k_2}(v) \cap N(v)$, and let P be an x_1, x_2-path such that $P \subseteq R(v) \cap V(M_0(G) - \Pi_j(z(v)))$. Initialize $P^\pi = (j_1)$ to be the sequence of vertices in $V^\pi(v)$ visited so far along P, $a = j_1$ to be the most recent vertex of $V^\pi(v)$ in P^π, and $y = x_1$ to be the most recent vertex of V visited by P. Begin the algorithm:

1. Advance y along P until either encountering a vertex, $y' \in P$, such that $y' \notin \Theta_a(v)$ (then proceed to Step 2) or reaching the end of P (then proceed to Step 3).
2. Let $a' \in V^\pi(v) - j$ be such that $y' \in \Theta_{a'}(v)$. Since there is a y, y'-path on P that passes only through the vertex set $\Theta_a(v) \cup \Theta_{a'}(v)$, the edge $aa' \in E^\pi(v)$ by Lemma 12. Append a' to the end of P^π, let $a = a'$, let $y = y'$, and proceed to Step 1.
3. If $a = j_2$, then exit the algorithm. Otherwise, note that $x_2 \in \Theta_{j_2}(v)$, so the y, x_2-path on P contains only vertices in $\Theta_a(v) \cup \Theta_{j_2}(v)$, and therefore $aj_2 \in E^\pi(v)$ by Lemma 12. Append j_2 to the end of P^π and exit the algorithm.

This algorithm must terminate, since P has finite length, and y advances at least one vertex along P each time Step 1 executes. When the algorithm terminates, P^π contains a j_1, j_2-walk on $V^\pi(v) - j$, which contains a j_1, j_2-path on the same vertex set. Since this can be done for any $j_1, j_2 \in V^\pi(v)$, the subgraph induced by $V^\pi(v) - j$ in $G^\pi(v)$ is connected.
Necessity: Choose any $x_1, x_2 \in N_{z(v)}(v)$, let $k_1, k_2 \in \{1, 2, \ldots, \alpha'(v)\}$ be such that $x_1 \in T_{k_1}(v)$ and $x_2 \in T_{k_2}(v)$, and let $j_1, j_2 \in \{1, 2, \ldots, \pi(z(v))\}$ be such that $T_{k_1}(v) \in$

$\Theta_{j_1}(v)$ and $T_{k_2}(v) \in \Theta_{j_2}(v)$. Since $|J_k(v)| \geq 2$, j_1 and j_2 can be chosen such that $j_1 \neq j$ and $j_2 \neq j$. If $k_1 = k_2$, then there is an x_1, x_2-path on $R(v) \cap V(z(v))$, and therefore x_1 and x_2 are in the same component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$. Otherwise, let P^π be a j_1, j_2-path on $V^\pi(v) - j$. Initialize $P = (x_1)$ to be the sequence of vertices visited by the following algorithm, $a = j_1$ to be the most recent vertex of P^π visited by the algorithm, and $y = x_1$ to be the most recent vertex of V visited by P. Begin the algorithm:

1. Let a' be the vertex that P^π visits immediately after a. If $a' = j_2$, let $y' = x_2$. Otherwise, choose any $y' \in \Theta_{a'}(v)$. Noting that $y \in \Theta_a(v)$, $y' \in \Theta_{a'}(v)$, and $aa' \in E^\pi(v)$, Lemma 12, provides a y, y'-path, $P' \subseteq \Theta_a(v) \cup \Theta_{a'}(v)$. Since $a \neq j$ and $a' \neq j$, $P' \cap V(\Pi_j(z(v))) = \emptyset$. Append P' to the end of P. If $a' = j_2$, exit the algorithm. Otherwise, let $a = a'$ and $y = y'$, and repeat Step 1.

This algorithm must terminate, since P^π has finite length, and a advances one vertex along P^π each time Step 1 executes. When the algorithm terminates, P contains an x_1, x_2-walk on $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$, which contains an x_1, x_2-path on the same vertex set. This can be done for any $x_1, x_2 \in N_{z(v)}(v)$, and therefore $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$. □

Theorem 2 *Let $G = (V, E, B, z)$ be a zone-connected geo-graph with $v \in V$, $\alpha'(v) \geq 2$, and $|J_k(v)| \geq 2$ for every $k \in \{1, 2, \ldots, \alpha'(v)\}$, and the subgraph induced by $\Theta_j(v)$ has one component for every $j \in V^\pi(v)$. The vertex set $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ for every $j \in \{1, 2, \ldots, \pi(z(v))\}$ if and only if $G^\pi(v)$ is biconnected.*

Proof of Theorem 2 *Sufficiency*: Be Lemma 13, the subgraph of $G^\pi(v)$ induced by $V^\pi(v) - j$ must be connected for every $j \in V^\pi(v)$. Therefore, $G^\pi(v)$ must be biconnected.
Necessity: Choose any $j \in \{1, 2, \ldots, \pi(z(v))\}$. If $\Theta_j(v) \neq \emptyset$, then $j \in V^\pi(v)$, and the subgraph of $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ contains $N_{z(v)}(v)$ in a single component by Lemma 13, since the subgraph of $G^\pi(v)$ induced by $V^\pi(v) - j$ is connected. If $\Theta_j(v) = \emptyset$, then $R(v) \cap V(\Pi_j(z(v))) = \emptyset$, and therefore $R(v) \cap V(M_0(G) - \Pi_j(z(v))) = R(v)$. Let $x_1, x_2 \in N_{z(v)}(v)$ and let W be a (x_1, x_2)-perimeter on v. Since W is an x_1, x_2-walk on $R(v)$, it contains an x_1, x_2-path on $R(v)$. This can be done for any $x_1, x_2 \in N_{z(v)}(v)$, and therefore $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$. □

Consider the augmented neighborhood depicted in Fig. 7b. As shown in Sect. 4.1, this augmented neighborhood has three neighbor components with $\alpha'(v) = 3$, and $|J_1(v)| = |J_2(v)| = |J_3(v)| = 2$. Three pockets appear in this augmented neighborhood ($V^\pi(v) = \{0, 1, 2\}$), and the neighbor pockets of v are given by $\Theta_0(v) = \{v_A, v_B, v_C, v_E, v_F, v_G, v_H, v_J\}$, $\Theta_1(v) = \{v_C, v_D, v_E\}$, and $\Theta_2(v) = \{v_A, v_H, v_I, v_J\}$, each of which induces a single component (per Sect. 4.3). By Definition 5, the edge set of the pocket graph is $E' = \{01, 02\}$, since $\Theta_0(v)$ and $\Theta_1(v)$ share $T_2(v)$ as constructed in Sect. 4.1, and $\Theta_0(v)$ and $\Theta_2(v)$ share $T_1(v)$. Hence, the pocket graph $G^\pi(v)$ is shown in Fig. 11a. This pocket graph is not biconnected,

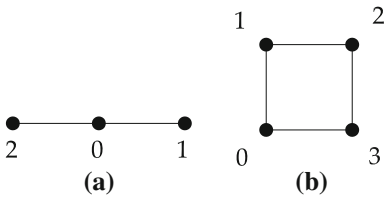

Fig. 11 Pocket graphs for the augmented neighborhoods in Figs. 7b and 8b. **a** The pocket graph for the augmented neighborhood shown in Fig. 7b. **b** The pocket graph for the augmented neighborhood shown in Fig. 8b

and hence, Theorem 2 (in conjunction with Corollary 1) can conclude that removing the highlighted unit from zone 2 (as shown in Fig. 7a) causes this zone to become discontiguous.

In contrast, consider the augmented neighborhood shown in Fig. 8b. It has four neighbor components, $T_1(v) = \{v_B\}$, $T_2(v) = \{v_D\}$, $T_3(v) = \{v_E\}$, and $T_4(v) = \{v_G\}$, with $\alpha'(v) = 4$. Each neighbor component is adjacent to two pockets (i.e., $|J_1(v)| = |J_2(v)| = |J_3(v)| = |J_4(v)| = 2$). Four pockets appear in this augmented neighborhood, with neighbor pockets given by $\Theta_0(v) = \{v_D, v_F, v_G\}$, $\Theta_1(v) = \{v_A, v_B, v_D\}$, $\Theta_2(v) = \{v_B, v_C, v_E\}$, and $\Theta_3(v) = \{v_E, v_G, v_H\}$, each of which induces a single component. By this construction, the pocket graph has edge set $E^\pi = \{01, 12, 23, 30\}$, as shown in Fig. 11b. This pocket graph is biconnected, and by Theorem 2 (in conjunction with Corollary 1), the highlighted unit can be removed from zone 1 without causing it to become discontiguous (as shown in Fig. 8a).

4.5 Algorithm

The preceding sections have established equivalent conditions for Corollary 1 based on analyzing the neighbor components, neighbor pockets, and pocket graph of v. This section demonstrates that all of these analyses can be carried out in $O(|R(v)|)$ time. Notably, Theorem 2 requires evaluation of the biconnectivity of the pocket graph, $G^\pi(v)$. Biconnectivity of $G^\pi(v)$ can be assessed in $O(|V^\pi(v)| + |E^\pi(v)|)$ time with existing search algorithms [20,30]. Clearly, $|V^\pi(v)| \leq |R(v)|$, since each pocket in $V^\pi(v)$ must contain at least one of its vertices that appears in $R(v)$. However, consider the construction of the edges of the pocket graph. For each $k \in \{1, 2, \ldots, \alpha(v)\}$, the number of pairs of neighbor pockets in which $T_k(v)$ appears is given by $\binom{|J_k(v)|}{2}$; it is not immediately clear that the number of edges in the pocket graph will be $O(|R(v)|)$.

Lemma 14 shows that biconnectivity of the entire pocket graph does not need to be evaluated. Based on the construction of the pocket graph, the subgraph induced by $J_k(v)$ will be a complete graph. Lemma 14 and Corollary 2 show that all of these edges do not need to be added to the pocket graph; adding the edges of a cycle through $J_k(v)$ is sufficient, for a total of $|J_k(v)|$ edges added. The reduced pocket graph, $\overline{G}^\pi(v) = (V^\pi(v), \overline{E}^\pi(v))$, is produced by adding only these cycles. Since there must be at least one edge connecting a vertex in $T_k(v)$ with a vertex in pocket j for $j \in J_k(v)$, the number of edges added to the reduced pocket graph is at most equal to the number of edges with exactly one endpoint in $T_k(v)$ (and the other endpoint elsewhere in $R(v)$). Since there is no edge with endpoints in $T_k(v)$ and $T_{k'}(v)$ for

Algorithm 2: Determine whether $N_{z(v)}(v)$ is contained in a single component of the subgraph induced by $R(v) \cap V(M_0(G) - \Pi_j(z(v)))$ for every $1 \leq j \leq \pi(z(v))$

Input : Zone-connected geo-graph $G = (V, E, B, z)$ with $v \in V$ such that $N_{z(v)} \neq \emptyset$
Output: $true$ or $false$

$V^\pi(v) \leftarrow \{j \in \{1, 2, \ldots, \pi(z(v))\} : R(v) \cap V(\Pi_j(z(v))) \neq \emptyset\}$;
Construct $T_1(v), T_2(v), \ldots, T_{\alpha'(v)}(v), T_{\alpha'(v)+1}(v), \ldots, T_{\alpha(v)}(v)$;
if $\alpha'(v) = 1$ **then**
    /* $N_{z(v)}(v)$ contained in one component of $R(v) \cap V(z(v))$ */
    return $true$;
else
    for $i = 1$ **to** $\alpha(v)$ **do**
        /* Store neighbor component data, per Definition 3 */
        $E_{i,j}(v) \leftarrow \{xy \in E : x \in T_i(v), y \in R(v) \cap V(\Pi_j(z(v)))\}$;
        $J_i(v) \leftarrow \{j \in V^\pi(v) : E_{i,j}(v) \neq \emptyset\}$;
        if $(|J_i(v)| < 2)$ & $(i \leq \alpha'(v))$ **then**
            /* A pocket condition fails, by Lemma 10 */
            return $false$;
/* Construct neighbor pockets, check number of components, by Lemma 8 */
for $j \in V^\pi(v)$ **do**
    Construct $\Theta_j(v)$;
    if $\Theta_j(v)$ *has more than one component* **then**
        return $false$;
/* Construct the edge set of the pocket graph, by Corollary 2 */
$\overline{E}^\pi(v) \leftarrow \emptyset$;
for $i = 1$ **to** $\alpha(v)$ **do**
    if $|J_i(v)| = 2$ **then**
        Let $J_i(v) = \{j_1, j_2\}$;
        $\overline{E}^\pi(v) \leftarrow \overline{E}^\pi(v) \cup \{j_1 j_2\}$;
    else
        Let $(j_1, j_2, \ldots, j_{|J_i(v)|})$ be any ordering of $J_i(v)$;
        $\overline{E}^\pi(v) \leftarrow \overline{E}^\pi(v) \cup \{j_1 j_{|J_i(v)|}\}$;
        for $k = 2$ **to** $|J_i(v)|$ **do**
            $\overline{E}^\pi(v) \leftarrow \overline{E}^\pi(v) \cup \{j_{k-1} j_k\}$;
Let $\overline{G}^\pi(v) = (V^\pi(v), \overline{E}^\pi(v))$;
return $IsBiConnected(\overline{G}^\pi(v))$;

$k, k' \in \{1, 2, \ldots, \alpha(v)\}$ and $k' \neq k$, the number of edges in $\overline{E}^\pi(v)$ is at most equal to the number of edges in the subgraph induced by $R(v)$, and hence, biconnectivity of the pocket graph can be assessed in $O(|V^\pi(v)| + |\overline{E}^\pi(v)|) \leq O(|R(v)|)$ time. Algorithm 2 collects all of the required analyses into one algorithm, and runs in $O(|R(v)|)$ time by Lemma 15.

Lemma 14 *Let $G = (V, E)$ be a graph, where $\overline{V} \subseteq V$ induces a complete subgraph in G, and let $\overline{G} = (V, \overline{E})$ be identical to G, except that the subgraph induced by $\overline{V}$ is*

replaced with any biconnected subgraph on $\overline{V}$ in $\overline{G}$. Then graph G is biconnected if and only if graph $\overline{G}$ is biconnected.

Proof of Lemma 14 *Sufficiency*: Choose any vertices $v_1, v_2, v_3 \in V$. Since G is biconnected, let P be a v_1, v_2-path on G that does not pass through v_3. If $P \cap \overline{V} = \emptyset$, then P appears in $\overline{G}$. Otherwise, let $v'_1, v'_2 \in \overline{V}$ be the first and last vertices, respectively, that P visits in $\overline{V}$, with P_1 and P_2 being the v_1, v'_1-path and v'_2, v_2-path, respectively, contained in P. Since the subgraph induced by $\overline{V}$ in $\overline{G}$ is biconnected, let $\overline{P}$ be a v'_1, v'_2-path on the subgraph induced by $\overline{V}$ that does not include v_3. Concatenating P_1, $\overline{P}$, and P_2 provides a v_1, v_2-walk in $\overline{G}$ that contains a v_1, v_2-path in $\overline{G}$. Since this can be done for any choice of $v_1, v_2, v_3 \in V$, $\overline{G}$ is biconnected.
Necessity: Note that $\overline{E} \subseteq E$. Adding edges to a graph cannot decrease its connectivity, and therefore the biconnectivity of $\overline{G}$ implies that G is also biconnected. □

Corollary 2 *Suppose that $G^\pi(v) = (V^\pi(v), E^\pi(v))$ is constructed by iterating over $i = 1, 2, \ldots, \alpha(v)$ and adding an edge between every pair of zones in $J_i(v)$, while $\overline{G}^\pi(v) = (V^\pi(v), \overline{E}^\pi(v))$ is constructed by adding a cycle through the set of zones in $J_i(v)$. Then graph $G^\pi(v)$ is biconnected if and only if graph $\overline{G}^\pi(v)$ is biconnected.*

Proof of Corollary 2 Define $\overline{G}^\pi_0(v), \overline{G}^\pi_1(v), \ldots, \overline{G}^\pi_{\alpha(v)}(v)$ as a sequence of graphs such that graph $\overline{G}^\pi_k(v)$ is constructed by iterating over $i = 1, 2, \ldots, \alpha(v)$. If $i \leq k$, edges are added between every pair of zones in $J_i(v)$, otherwise a cycle of edges is added through the zones in $J_i(v)$. By this construction, $\overline{G}^\pi_0(v) = \overline{G}^\pi(v)$ and $\overline{G}^\pi_{\alpha(v)}(v) = G^\pi(v)$. The proofs is by truncated backward induction.
Base Case: For $k = \alpha(v)$, $\overline{G}^\pi_k(v)$ is biconnected, since $\overline{G}^\pi_{\alpha(v)}(v) = G^\pi(v)$.
Inductive Case: By induction, assume $\overline{G}^\pi_{k+1}(v)$ is biconnected. Note that $\overline{G}^\pi_k(v)$ only differs from $\overline{G}^\pi_{k+1}(v)$ in the subgraph induced by $J_{k+1}(v)$, with $J_{k+1}(v)$ inducing a complete subgraph in $\overline{G}^\pi_{k+1}(v)$ and inducing some other biconnected subgraph in $\overline{G}^\pi_k(v)$ (this subgraph is biconnected because it includes a cycle). Therefore, $\overline{G}^\pi_{k+1}(v)$ is biconnected if and only if $\overline{G}^\pi_k(v)$ is biconnected by Lemma 14.

By the induction, $\overline{G}^\pi_{\alpha(v)}(v)$ is biconnected if and only if $\overline{G}^\pi_0(v)$ is biconnected, and therefore $\overline{G}^\pi(v)$ is biconnected if and only if $G^\pi(v)$ is biconnected. □

Lemma 15 *Algorithm 2 runs in $O(|R(v)|)$ time.*

Proof of Lemma 15 Line 1 identifies the vertices of the pocket graph and can be executed in $O(|R(v)|)$ time by iterating through the list containing $R(v)$. Lines 2–10 construct the neighbor components and store the partitioned edge sets and set of adjacent pockets for each neighbor component, as defined in Definition 3. This process can be executed in $O(|R(v)|)$ time as described in Sect. 4.3. Similarly, the construction and assessment of the neighbor pockets in Lines 11–14 can be executed in $O(|R(v)|)$ time as described in Sect. 4.3. The edge set $\overline{E}^\pi(v)$ is constructed in Lines 15–24; the set of edges added is reduced by Corollary 2, and runs in $O(|R(v)|)$ time. Finally, Line 26 assesses the biconnectivity of $G^\pi(v)$ through the function $IsBiConnected(\overline{G}^\pi(v))$, which runs in $O(|V^\pi(v)| + |\overline{E}^\pi(v)|) \leq O(|R(v)|)$ time using a depth-first search algorithm [20,30]. Therefore, the entire algorithm runs in $O(|R(v)|)$ time. □

5 Conclusions

Graph partitioning problems applied to geographic zoning typically seek to generate contiguous zones that optimize a specified objective (e.g., maximum shape compactness, maximum election competitiveness) while satisfying other constraints (e.g., population balance). If heuristics such as local search are applied to find good (but suboptimal) sets of zones, one must be able to quickly verify whether a particular transition leads to an infeasible set of zones. When each zone must comprise a contiguous area of land, local search requires an efficient algorithm to determine whether a desired transition causes contiguity to be violated. By integrating contiguity information at both the zone and unit levels, the geo-graph is able to efficiently evaluate contiguity constraints in local search approaches to graph partitioning. When a unit is removed from a zone, knowing the composition of that zone's pockets allows post-removal contiguity to be assessed by only examining the units whose boundaries share at least a single point with the removed unit, as shown in Theorem 1. Simple search algorithms can assess the pockets of all zones in $O(m(G)^3)$ time and search the subgraphs of $R(v)$ in $O(m(G)|R(v)|)$ time [22]. However, each of these two algorithms ignores the underlying structure of the geo-graph. By exploiting this structure, the algorithms presented in this paper can assess the pockets of all zones in $O(m(G)^2)$ time and assess contiguity in $O(|R(v)|)$ time. When applied in geographic zoning problems, these efficient algorithms can help practitioners explore the set of zoning options more quickly, particularly when the number of units is large, such as when creating United States Congressional Districts.

When single-vertex transfer is implemented, geo-graph partitioning shares many similarities to dynamic graph contiguity. Dynamic graphs are characterized by a single graph under iterative update and query operations. Update operations make small changes to the structure of the graphs, typically by inserting or removing an edge or vertex, while query operations request information about graph properties such as connectivity of the current graph. At least one of these operations may run in polylogarithmic time in the size of the graph by creating and updating a hierarchical decomposition of the elements of the graph, such as the balanced piece decomposition used by Frigioni and Italiano [16]. In contrast, the algorithmic improvements attained in this paper are derived from the structure of the graph; the data structures used by these algorithms are based on standard graph search procedures, and hence, may be attractive to practitioners in their simplicity. By using only graph structure to reduce computation, it is possible that more insightful data structures may be applied in the future to achieve further reductions in computation. For example, the current algorithm for determining zone pockets is currently executed anew each time the structure of $H(G)$ changes; an algorithm that takes advantage of this dynamic structure instead of computing from scratch may be able to further reduce computation.

The geo-graph is tailored for geographic zoning problems where a large number of units are grouped into a relatively small number of zones. The run times of the proposed algorithms support this tailoring; in the worst case, determining contiguity depends on the number of zones and the number of units that share at least a single point with the boundary of the transferred unit. Furthermore, if this algorithm determines that the unit cannot be transferred without violating contiguity constraints, then local search will

need to identify a new transition and potentially execute many contiguity queries. In such cases, pockets do not need to be reassessed since no unit has yet been transferred, and only the $O(|R(v)|)$ portion of the algorithm needs to be repeated. A fundamental effect of this tailoring is that, while local search can consider a vast set of possible zoning options, these options must be explored individually during the search process. Practitioners must strike a balance between the potential quality of their solutions and the time required to generate those solutions. Once a practitioner chooses the desired granularity of their basic units, the geo-graph algorithms proposed in this paper will allow them to explore their zoning options more efficiently than simple search methods currently suggested in the literature [27].

Appendix A: Additional geo-graph lemmas

The following lemmas from [22] are applied in the proofs contained in this paper:

Lemma 16 *Let $G = (V, E, B, z)$ be a geo-graph, $C^S \subseteq V$ be a tangle-free closed strand on which $x, v, y \in V$ appear consecutively, and $W_1, W_2 \subseteq R(v)$ be the (x, y)-perimeters on v.*

A. For some $j \in \{1, 2\}$, $W_j \cap N(v) \subseteq C^S \cup Int(C^S)$ and $W_{3-j} \cap N(v) \subseteq C^S \cup Ext(C^S)$
B. If $C = C^S$ is a cycle, then for some $j \in \{1, 2\}$, $W_j \subseteq C \cup Int(C)$ and $W_{3-j} \subseteq C \cup Ext(C)$. Furthermore, W_j is unbroken and therefore is an x, y-walk on $R(v) \cap (C \cup Int(C))$.

Lemma 17 *Let $G = (V, E, B, z)$ be a geo-graph, with $C^S \subset V$ being any tangle-free closed strand, and $x \in V - C^S$. The following properties hold:*

A. For every $y \in N(x) - C^S$, $y \in Int(C^S)$ if and only if $x \in Int(C^S)$.
B. If $C = C^S$ is a cycle, then for every $y \in R(x) - C$, $y \in Int(C)$ if and only if $x \in Int(C)$.
C. If $C = C^S$ is a cycle and $B(x) \cap B(v_0) \neq \emptyset$, then $x \notin Int(C)$.
D. If $y \in V - C^S$ such that $x \in Int(C^S)$ and $y \in Ext(C^S)$, then each x, y-path, P, has $P \cap C^S \neq \emptyset$.
E. If $C = C^S$ is a cycle and $y \in V - C$ such that $x \in Int(C)$ and $y \in Ext(C)$, then each x, y-strand, S, has $S \cap C \neq \emptyset$.
F. If G is zone-connected with $z(x) = i$ for some $i \in M(G)$, $C = C^S$ is a cycle, and $C \subseteq V(j)$ for some $j \in M(G) - i$, then zone i is surrounded by C if and only if $x \in Int(C)$.

References

1. Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning: a survey. INTEGRATION. VLSI J. **19**, 1–81 (1995)
2. Barth, D., Pellegrini, F., Raspaud, A., Roman, J.: On bandwidth, cutwidth, and quotient graphs. Theor. Inform. Appl. **29**(6), 487–508 (1995)
3. Becker, R.I., Lari, I., Lucertini, M., Simeone, B.: Max-min partitioning of grid graphs into connected components. Networks **32**(2), 115–125 (1998)

4. Bender, M.A., Farach-Colton, M.: The level ancestor problem simplified. Theor. Comput. Sci. **321**(1), 5–12 (2004)
5. Bozkaya, B., Erkut, E., Laporte, G.: A tabu search heuristic and adaptive memory procedure for political districting. Eur. J. Oper. Res. **144**(1), 12–26 (2003)
6. Buchsbaum, A.L., Georgiadis, L., Kaplan, H., Rogers, A., Tarjan, R.E., Westbrook, J.R.: Linear-time algorithms for dominators and other path-evaluation problems. SIAM J. Comput. **38**(4), 1533–1573 (2008)
7. Butler, D., Cain, B.E.: Congressional Redistricting: Comparative and Theoretical Perspectives. MacMillan Publishing Company, New York (1992)
8. Chlebíková, J.: Approximating the maximally balanced connected partition problem in graphs. Inf. Process. Lett. **60**, 225–230 (1996)
9. di Cortona, P.G., Manzi, C., Pennisi, A., Ricca, F., Simeone, B.: Evaluation and Optimization of Electoral Systems. Society for Industrial and Applied Mathematics, Philadelphia (1999)
10. D'Amico, S., Wang, S., Batta, R., Rump, C.: A simulated annealing approach to police district design. Comput. Oper. Res. **29**, 667–684 (2002)
11. Demetrescu, C., Eppstein, D., Galil, Z., Italiano, G.F.: Dynamic graph algorithms. In: Atallah, M.J., Blanton, M. (eds.) Algorithms and Theory of Computation Handbook, 2nd edn. pp. 9.1–9.28. Chapman & Hall/CRC, London (2010)
12. Eppstein, D., Galil, Z., Italiano, G.F., Spencer, T.H.: Separator based sparsification for dynamic planar graph algorithms. In: Proceedings of the 25th Annual ACM Symposium on Theory of Computing, pp. 208–217. ACM, New York, NY (1993)
13. Eppstein, D., Italiano, G.F., Tamassia, R., Tarjan, R.E., Westbrook, J., Yung, M.: Maintenance of a minimum spanning forest in a dynamic plane graph. J. Algorithms **13**(1), 33–54 (1992)
14. Fiduccia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. In: DAC '82: Proceedings of the 19th Design Automation Conference, pp. 175–181. IEEE Press, Piscataway, NJ (1982)
15. Firmani, D., Italiano, G.F., Laura, L., Orlandi, A., Santaroni, F.: Computing strong articulation points and strong bridges in large scale graphs. In: Klasing, R. (ed.) Experimental Algorithms: Proceedings of the 11th International Symposium, SEA 2012, Bordeaux, France, June 7–9, 2012, pp. 195–207. Springer, Berlin (2012)
16. Frigioni, D., Italiano, G.F.: Dynamically switching vertices in planar graphs. Algorithmica **28**(1), 76–103 (2000)
17. Henzinger, M.R., King, V.: Randomized fully dynamic graph algorithms with polylogarithmic time per operation. J. ACM **46**(4), 502–516 (1999)
18. Henzinger, M.R., King, V.: Maintaining minimum spanning forests in dynamic graphs. SIAM J. Comput. **31**(2), 364–374 (2001)
19. Holm, J., de Lichtenberg, K., Thorup, M.: Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. J. ACM **48**(4), 723–760 (2001)
20. Horowitz, E., Sahni, S.: Fundamentals of Computer Algorithms. Computer Science Press, Rockville (1978)
21. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Syst. Tech. J. **49**(1), 291–307 (1970)
22. King, D.M., Jacobson, S.H., Sewell, E.C., Cho, W.K.T.: Geo-graphs: an efficient model for enforcing contiguity and hole constraints in planar graph partitioning. Oper. Res. **60**(5), 1213–1228 (2012)
23. Lengauer, T., Tarjan, R.E.: A fast algorithm for finding dominators in a flowgraph. ACM Trans. Program. Lang. Syst. **1**(1), 121–141 (1979)
24. Reif, J.H.: A topological approach to dynamic graph connectivity. Inf. Process. Lett. **25**(1), 65–70 (1987)
25. Ricca, F.: A multicriteria districting heuristic for the aggregation of zones and its use in computing origin-destination matrices. INFOR **42**(1), 61–77 (2004)
26. Ricca, F., Scozzari, A., Simeone, B.: Political districting: from classical models to recent approaches. 4OR **9**(3), 223–254 (2011)
27. Ricca, F., Simeone, B.: Local search algorithms for political districting. Eur. J. Oper. Res. **189**, 1409–1426 (2008)
28. Salgado, L.R.B., Wakabayashi, Y.: Approximation results on balanced connected partitions of graphs. Electron. Notes Discret. Math. **18**, 207–212 (2004)

29. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
30. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972)
31. Tavares-Pereira, F., Figueira, J.R., Mousseau, V., Roy, B.: Multiple criteria districting problems: the public transportation network pricing system of the Paris region. Ann. Oper. Res. **154**, 69–92 (2007)
32. Thorup, M.: Near-optimal fully-dynamic graph connectivity. In: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, pp. 343–350. ACM, New York, NY (2000)
33. United States Census Bureau: Apportionment Population and Number of Representatives, by State: Census 2000. http://www.census.gov/population/www/cen2000/maps/files/tab01.pdf (2000). Accessibility Verified on 27 May 2011
34. United States Census Bureau: Tallies of Census Blocks By State or State Equivalent. http://www.census.gov/geo/www/2010census/census_block_tally.html (2011). Accessibility Verified on 27 May 2011
35. Wang, J.P.: Stochastic relaxation on partitions with connected components and its application to image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **20**(6), 619–636 (1998)
36. West, D.B.: Introduction to Graph Theory, 2nd edn. Prentice Hall, Upper Saddle River (2001)
37. Whitney, H.: Non-separable and planar graphs. Trans. Am. Math. Soc. **34**(2), 339–362 (1932)