Discrete Applied Mathematics 172 (2014) 62–74

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

The complexity of pebbling reachability and solvability in planar and outerplanar graphs☆

Timothy Lewis, Charles A. Cusack*, Lisa Dion [1]

Department of Computer Science, Hope College, 27 Graves Place, Holland, MI 49422, United States

ARTICLE INFO

Article history:
Received 25 July 2013
Received in revised form 20 February 2014
Accepted 4 March 2014
Available online 22 March 2014

Keywords:
Graph pebbling
Planar
Outerplanar
NP-complete

ABSTRACT

Given a simple, connected graph, a *pebbling configuration* is a function from its vertex set to the nonnegative integers. A *pebbling move* between adjacent vertices removes two pebbles from one vertex and adds one pebble to the other. A vertex r is said to be *reachable* from a configuration if there exists a sequence of pebbling moves that places at least one pebble on r. A configuration is *solvable* if every vertex is reachable. We prove that determining reachability of a vertex and solvability of a configuration are NP-complete on planar graphs. We also prove that both reachability and solvability can be determined in $O(n^6)$ time on planar graphs with diameter two. Finally, for outerplanar graphs, we present a linear algorithm for determining reachability and a quadratic algorithm for determining solvability. To prove this result, we provide linear algorithms to determine all possible maximal configurations of pebbles that can be placed on the endpoints of a path and on two adjacent vertices in a cycle.

1. Introduction

A graph $G = (V, E)$ is a set V of *vertices* and a set E of pairs of vertices called *edges*. The maximum number of edges in the shortest path between any two vertices is called the *diameter* of G. G is *planar* if it can be embedded in a plane, i.e. if it can be drawn such that no two edges intersect. G is *outerplanar* if it can be embedded in a plane so that every edge is incident to the unbounded face. The *dual* of an embedding of a planar graph is the graph obtained by adding a vertex for each face and connecting vertices whose corresponding faces share an edge. The *weak dual* is the induced subgraph of the dual obtained by removing the vertex corresponding to the unbounded face. It is well known that the weak dual of an outerplanar graph is a forest. If a vertex v is adjacent to at least one vertex in a set $A \subseteq V$, then we will say that v is *adjacent to* A. A *dominating set* $S \subseteq V$ has the property that every vertex not in S is adjacent to S. The *domination number* of G is the cardinality of the smallest dominating set in G.

A *pebbling configuration* (or just *configuration*) is a function $C : V(G) \to \mathbb{N}$ (the nonnegative integers) where $C(v)$ represents the number of "pebbles" placed on vertex v. The size of C, denoted $|C|$, is the sum of the pebbles on the vertices of G. For $k \in \mathbb{N}$ we define $V_k = \{v \in V \mid C(v) \geq k\}$. That is, V_k is the set of vertices that have at least k pebbles. A *pebbling move* from a vertex u to an adjacent vertex v, denoted (u, v), removes two pebbles from u and adds one pebble to v, assuming u

☆ This work was supported by the National Science Foundation under grant 0851293.

* Corresponding author. Tel.: +1 616 395 7271; fax: +1 616 395 7123.
E-mail addresses: timothy.lewis@hope.edu (T. Lewis), cusack@hope.edu (C.A. Cusack), lisadion@umich.edu (L. Dion).

[1] Current address: Department of Electrical Engineering and Computer Science, University of Michigan, 1301 Beal Ave., Ann Arbor, MI 48109, United States.

http://dx.doi.org/10.1016/j.dam.2014.03.008
0166-218X/

has at least two pebbles. A vertex is *reachable* if it has at least one pebble assigned to it in the initial configuration C or it can receive a pebble through a sequence of pebbling moves. If every vertex in G is reachable, then C is said to be *solvable*. The *pebbling number* of G is the minimum integer k such that every pebble distribution of size k on G is solvable.

If ϕ is a sequence of pebbling moves, we define C_ϕ to be the configuration after the moves of ϕ have been performed on G with starting configuration C. A sequence of pebbling moves that accomplishes some goal (e.g. places one pebble on a specified vertex) is *minimal* if removing any moves renders the goal unattainable. A sequence is *minimum* if no other sequence accomplishes the same goal with less moves. If a pebbling sequence accomplishes some goal, some minimal sequence does as well. Thus, we assume throughout that all pebbling sequences are minimal.

Graph pebbling can be extended to weighted graphs by defining a weight function w that assigns a nonnegative integer to each edge [6,10,17]. A pebbling move from a vertex u to another vertex v removes $w(u, v)$ pebbles from u and adds one pebble to v, assuming $(u, v) \in E$ and u has at least $w(u, v)$ pebbles. Traditional pebbling on unweighted graphs can be thought of as a special case of pebbling on weighted graphs where the weight of each edge is 2. We further extend graph pebbling to directed weighted graphs using this definition of weight, with the only difference being that the direction of the edges matters. An alternative method of extending pebbling to weighted graphs is given in [12].

It has previously been shown that Reachable and Solvable are NP-complete [11,14,20], and that determining whether or not the pebbling number of a graph is at most k (Pebbling-Number) is Π_2^p-complete [14]. Techniques have been developed to compute the pebbling number for small graphs [17] and bounds on the pebbling number [9]. In addition, the pebbling number of graphs in a family known as split graphs can be computed in polynomial time [1].

For diameter two graphs, more is known. The pebbling number of a diameter two graph is either n or $n + 1$, and [4,3] imply that determining which it is can be accomplished in polynomial time. An explicit algorithm is given in [2], with a slight improvement given in [8]. Although Pebbling-Number is easier for diameter two graphs, Reachable remains NP-complete for diameter two graphs (D2-Reachable) [5]. However, D2-Reachable is decidable in polynomial time for diameter two graphs with pebbling number $n + 1$ or those with pebbling number n that have small connectivity [2].

Studying the complexity of Reachable restricted to planar and outerplanar graphs was suggested in [14]. We show that in the planar case, Reachable (P-Reachable) and Solvable (P-Solvable) remain NP-complete. However, Reachable and Solvable are both decidable in $O(|V|^6)$ time when restricted to planar graphs having diameter two (PD2-Reachable and PD2-Solvable). Finally, we present a linear algorithm for Reachable on outerplanar graphs (OP-Reachable), which implies that Solvable can be decided in $O(n^2)$ time on outerplanar graphs (OP-Solvable).

2. Planar graphs

First we will define Clause Cycle Planar 3SAT (CCP3SAT), a restricted form of 3SAT, and show that it is NP-complete. Then we will reduce from CCP3SAT to P-Reachable by constructing a planar graph using gadgets for each variable and clause so that under a certain pebbling configuration, a designated vertex r will be reachable if and only if the 3-CNF formula is satisfiable. We will construct variable gadgets with positive and negative edges corresponding to true and false assignments of the variables. To ensure our gadget enforces a valid truth assignment, each positive (resp. negative) edge can be pebbled along if and only if no pebbling moves have been made along any negative (resp. positive) edges. Then we will connect clause gadgets in a path so that each clause can only be pebbled from if it is pebbled to by at least one variable gadget and by the previous clause gadget. We will be able to pebble from the last clause to a specified target vertex if and only if each clause has a literal that evaluates to true.

2.1. Clause cycle planar 3SAT

Let F be a 3-CNF formula with n variables $\{v_1, \ldots, v_n\}$ and m clauses $\{c_1, \ldots, c_m\}$. Define $G(F) = (V, E)$, where $V = \{v_i \mid 1 \le i \le n\} \cup \{c_j \mid 1 \le j \le m\}$ and $E = \{(v_i, c_j) \mid v_i \in c_j \text{ or } \bar{v}_i \in c_j\} \cup \{(v_i, v_j) \mid j = (i + 1) \bmod n\}$. 3SAT restricted to instances of 3-CNF formula F where $G(F)$ is planar is called Planar 3SAT (P3SAT). P 3SAT was shown to be NP-Complete by Lichtenstein [13] by constructing an equivalent planar 3-CNF formula for a given general 3-CNF formula. The construction arranges clauses and variables in a grid (Fig. 1(a)) with the intersecting edges replaced with a crossover box (Fig. 2(a)). Then a cycle is drawn through the variables in the grid (Fig. 1(a)) that goes through each crossover box (Fig. 2(b)).

Given a 3-CNF formula F, define the graph $H(F) = (V, E)$, where $V = \{v_i \mid 1 \le i \le n\} \cup \{c_j \mid 1 \le j \le m\}$ and $E = \{(v_i, c_j) \mid v_i \in c_j \text{ or } \bar{v}_i \in c_j\} \cup \{(c_i, c_j) \mid j = (i + 1) \bmod m\}$. Define Clause Cycle Planar 3SAT (CCP3SAT) to be a subset of 3SAT where the graph $H(F)$ is planar. This is analogous to P3SAT except the clauses instead of the variables are connected by a cycle.

Theorem 1. *CCP3SAT is NP-complete.*

Proof. CCP3SAT is trivially in NP. We will show that CCP3SAT is NP-hard by showing that the 3-CNF formula reduced to in [13] is not only in P3SAT but also satisfies the requirements to be in CCP3SAT. Given a 3-CNF formula F, construct an equivalent 3-CNF formula F' using the reduction in [13]. Given the graph $G(F')$ from the reduction, remove the edges between variables in both the overall diagram (Fig. 1(a)) and the crossover boxes (Fig. 2(b)), and create a cycle between the clauses so that the graph remains planar as follows. First, draw a cycle going through all of the crossover boxes in the same order and by the same quadrants as [13] but connecting clauses instead of variables (Fig. 1(b)). Then create a path through the clauses

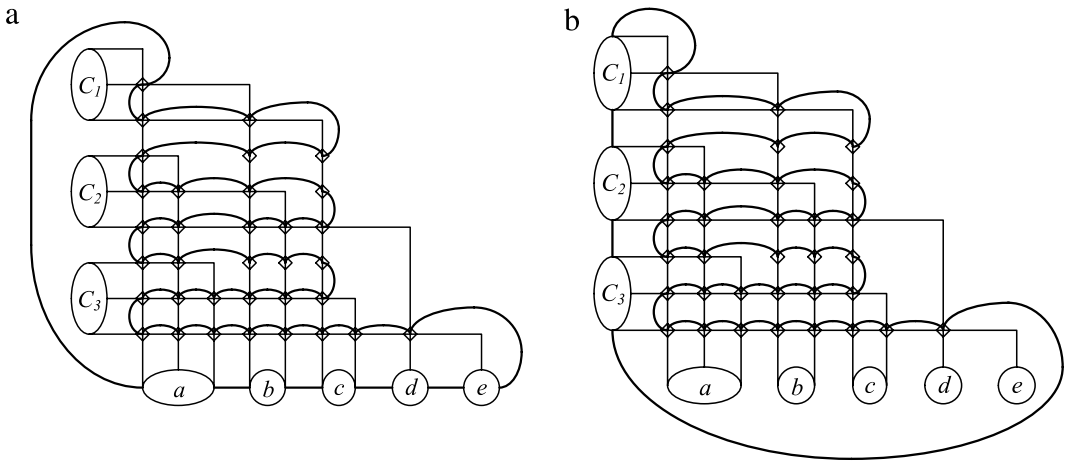

Fig. 1. (a) Lichtenstein's construction with variable cycle. (b) Lichtenstein's construction with the variable cycle replaced with a clause cycle.

in each crossover box depending upon which quadrant the path enters and exits the crossover box, either from top left to top right (Fig. 3(a)) or from bottom left to top right (Fig. 3(b)). This graph is planar and the clauses can be ordered so that this graph corresponds to $H(F')$, thus CCP3SAT is NP-hard.

2.2. Gadgets

We use several planar gadgets to create our reduction from CCP3SAT to Planar Reachable. When referring to instances of CCP3SAT, an arbitrary embedding in the plane will be assumed. As we describe each gadget we will also specify the pebbling configuration for the gadget. We will argue that under the conditions in which each gadget is used, no vertex can accumulate more than three pebbles and thus no edge can be pebbled along twice in a minimal set of pebbling moves. We define a *boundary edge* of a gadget to be one which connects a vertex in a gadget to a vertex outside the gadget.

2.2.1. Crossover XOR gadget

The crossover XOR gadget serves a dual purpose. It takes the place of an edge crossing and allows only one of the edges to be used in a minimal pebbling sequence, assuming at most one pebble enters from each boundary edge. The crossover XOR gadget is given in Fig. 4.

Assume that all boundary edges pebble into the gadget at most once. Then it is not difficult to see that x_i (resp. y_i) can pebble along its boundary edge only if x_{1-i} (resp. y_{1-i}) is pebbled to from its boundary edge. In other words, the boundary edges of the gadget correspond to an edge crossing. In addition, a move from x_i to x_{1-i} precludes a move from y_j to y_{1-j}, and vice-versa. Also notice that no vertex in the gadget can accumulate more than 3 pebbles. Finally notice that this subgraph is planar. In diagrams we represent a crossover XOR gadget as $\bigoplus$.

2.2.2. Variable gadget

The variable gadget has boundary edges corresponding to positive and negative instances of a given variable. The positive and negative boundary edges must be able to appear in any order when drawn on the plane so they can be connected to the clause gadgets in whatever order they appear. They should be designed such that each positive (resp. negative) boundary edge can be pebbled along once if and only if no negative (resp. positive) boundary edge has been pebbled along.

For a variable with m positive instances and n negative instances, create $m + n$ vertices and place two pebbles on each vertex. Also create mn crossover gadgets. Connect the vertices and gadgets as shown in Fig. 5(a).

Notice that even if each boundary edge pebbles into the gadget, no vertex can accumulate more than three pebbles. In order to pebble outward on a positive (resp. negative) boundary edge, the corresponding vertex with two pebbles must pebble through a path created by crossover XOR gadgets. If this is done then, by the nature of the crossover XOR gadgets, no negative (positive) boundary edge can be pebbled outward since its corresponding vertex with two pebbles must pebble through a path of crossover XOR gadgets which includes at least one already used by the positive (negative) path. Since the positive (negative) paths do not intersect with each other, every positive (negative) boundary edge can be pebbled outward simultaneously.

In order to attach each variable to the clauses that it appears in, it is necessary for the positive and negative boundary edges to interleave. However, none of these edges are allowed to intersect since our graph should be planar. It is easy to see that we can order the positive (resp. negative) edges so that they do not intersect with any other positive (negative) edges. However, if a positive and a negative boundary edge intersect, we can replace that intersection with a crossover gadget (Fig. 5(b)) since both edges cannot be simultaneously pebbled along anyway.

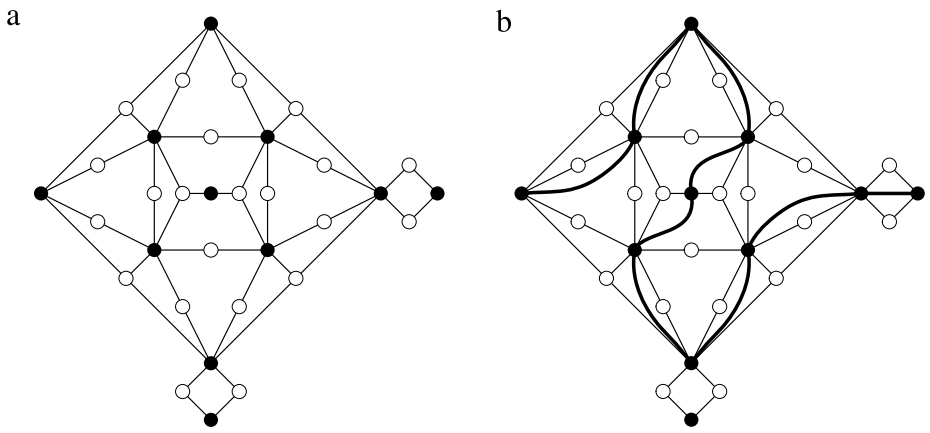

Fig. 2. (a) Lichtenstein's crossover box. (b) Variable path through crossover box.

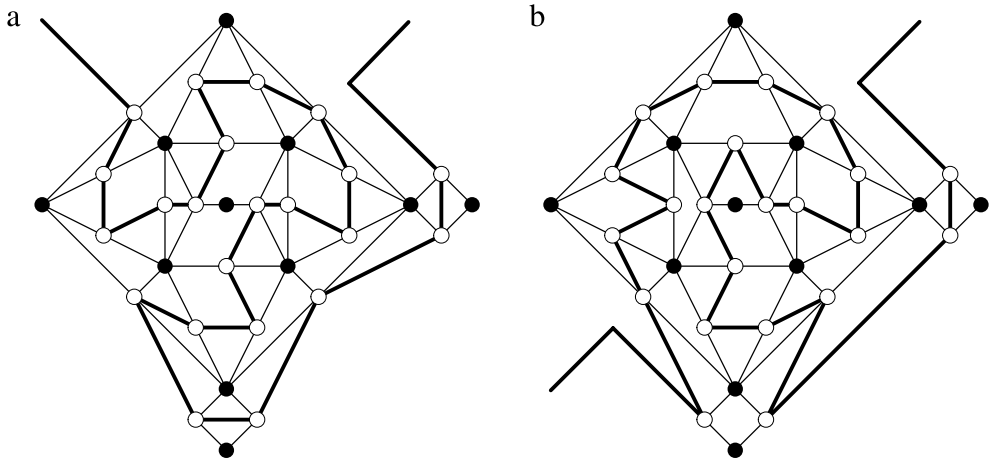

Fig. 3. Path through the clauses of a crossover box in 2 different manners: (a) top left to top right and (b) bottom left to top right.

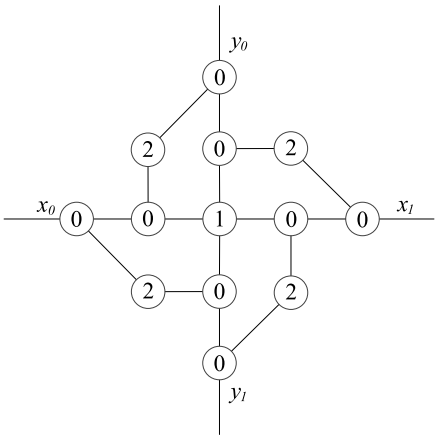

Fig. 4. Crossover XOR gadget with the number of pebbles indicated on each vertex.

2.2.3. Clause gadget

The clause gadget allows a pebbling move along the output edge if and only if a pebbling move is made from at least one edge connected to a variable gadget and from the input edge. The input and output edges are edges of the clause cycle, with the output edge from one clause gadget being the input edge of the next. We need to create two different clause gadgets depending upon whether all of the variables contained in a clause appear on the same side of the clause cycle. The gadget

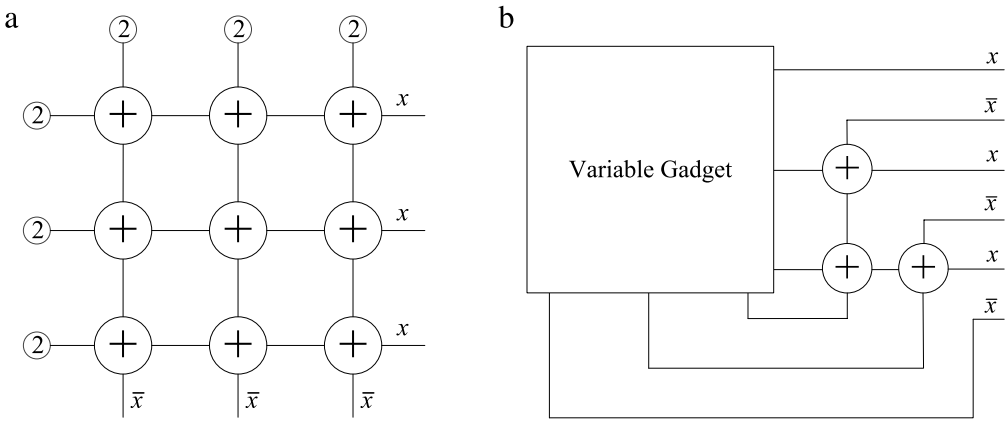

Fig. 5. (a) A variable gadget created with crossover XOR gadgets. (b) Variable gadget with edge intersections replaced by crossover XOR gadgets.

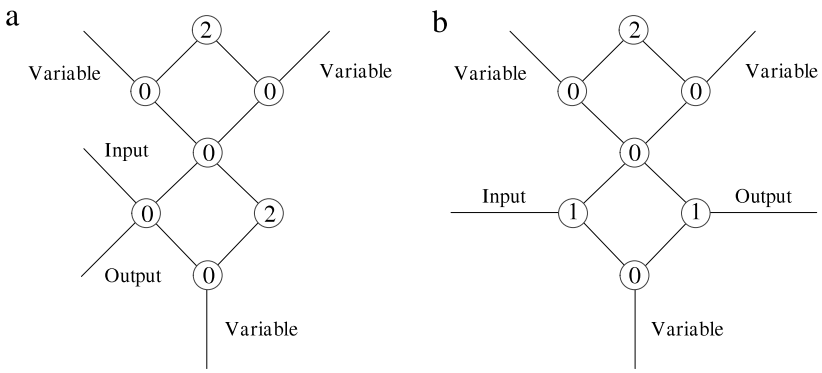

Fig. 6. Clause gadgets with (a) all variables on one side of the clause gadget, and (b) two variables on one side and one variable on the other side of the clause gadget.

in Fig. 6(a) will be used if they appear on the same side. If two variables appear on one side and the third variable appears on the other side, then the gadget in Fig. 6(b) is used.

Notice that a pebble can be moved along the output edge of each clause gadget if and only if a pebbling move is made along the input edge into the gadget and at least one variable edge pebbles into the gadget (assuming at most one pebble comes from each variable/input edge). Also notice that even if all five boundary edges are pebbled along once towards the gadget, no vertex in the gadget can accumulate more than three pebbles. Finally, notice that these subgraphs are planar.

2.3. Construction and proof

Let F be an instance of CCP3SAT with variables $\{v_1, \ldots, v_n\}$ and clauses $\{c_1, \ldots, c_m\}$. From $H(F)$, construct $H'(F)$ by replacing each variable with a corresponding variable gadget and each clause with a corresponding clause gadget, connecting the edges in the obvious way. Each variable gadget should have the correct number of positive and negative boundary edges and each clause gadget should be of the proper form depending upon the location of the variables it is connected to. Remove the edge between c_n and c_1. Create a vertex s with two pebbles and connect it to the input edge of c_1, and create a vertex r with zero pebbles and connect it to the output edge of c_n.

Lemma 2. *No vertex in $H'(F)$ can accumulate more than three pebbles after any sequence of pebbling moves.*

Proof. Assume otherwise and let ρ be a minimal sequence of pebbling moves that places four pebbles on some vertex. First note that ρ does not pebble across the same edge twice. Given that, previous comments imply that no vertex in a variable, clause, or crossover XOR gadget can accumulate four pebbles. Also, ρ does not place four pebbles on s or r because each has less than three pebbles and has degree one. Since this exhausts every vertex in $H'(F)$, no vertex can accumulate four pebbles.

This lemma implies that our gadgets will behave the way that we expect. In the crossover gadget, only one path can be used in a sequence of pebbling moves and a boundary edge can only be pebbled along from the gadget if the other corresponding boundary edge is pebbled along into the gadget. In the variable gadget, every positive (resp. negative) output edge can be pebbled from once if and only if no negative (positive) output edges have been pebbled from. In the clause gadget, the output edge can be pebbled along if and only if the input edge and at least one variable edge are pebbled along.

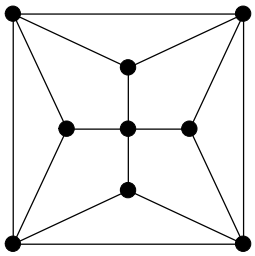

Fig. 7. The unique planar graph with diameter two with domination number three.

The following lemma is clear.

Lemma 3. *$H'(F)$ is planar.*

Theorem 4. P-Reachable *and* P-Solvable *are NP-complete, even if G has maximum degree four and each vertex starts with at most two pebbles.*

Proof. P-Reachable and P-Solvable are clearly NP. To show that P-Reachable is NP-hard, we will show that the graph $H'(F)$ can be constructed in polynomial time and that r is reachable in $H'(F)$ if and only if F is satisfiable.

Suppose that F is satisfiable and let $f : \{v_1, \ldots, v_n\} \to \{true, false\}$ be a satisfying truth assignment. For each variable v_i, if $f(v_i) = true$ (resp. $f(v_i) = false$), pebble along the positive (negative) edges of the variable gadget corresponding to v_i. Since each clause has a literal that evaluates to true, each clause gadget is pebbled to from a variable gadget. We can then pebble from s to r along the path through the clause gadgets. Thus if F is satisfiable then r is reachable.

Suppose that r is reachable. Then construct a satisfying truth assignment $f : \{v_1, \ldots, v_n\} \to \{true, false\}$ as follows. Since r is pebbled to, c_m must have received a pebble from some variable gadget v_i. If this move was along a positive (resp. negative) edge, assign $f(v_i) = true$ (resp. *false*). Notice that in order to pebble from a clause gadget c_i, a pebbling move must be made to c_i from c_{i-1}. Therefore each $c_{m-1}, \ldots, c_1$ must have been pebbled to by some variable gadget. Assign the corresponding variables in f in the same manner as before. If a variable is not assigned a truth value, arbitrarily assign it true or false. Since each variable gadget cannot simultaneously pebble along both positive and negative edges, f is well defined. Also, f satisfies F since each clause has a literal which evaluates to true. Thus if r is reachable then F is satisfiable by f.

It is clear from the construction that $H'(F)$ has maximum degree four and that each vertex starts with at most two pebbles. $H'(F)$ can clearly be computed in polynomial time since each clause gadget is of constant size and each variable gadget is of size $O(mn)$, where the variable appears m times in the positive instance and n times in the negative instance. Therefore, P-Reachable is NP-hard.

Notice that every vertex in a variable gadget is reachable using only pebbles from the gadget. Further, every vertex in a clause gadget is reachable if it receives at least one pebble from a variable gadget and one along the input edge. In other words, if r is reachable, then every vertex in all of the clause gadgets are reachable. Since s, the only other vertex in the graph, begins with pebbles, $H'(F)$ is solvable if and only if r is reachable. Therefore, P-Solvable is NP-hard as well.

3. Planar graphs with diameter two

In this section, we will establish that when restricted to planar graphs with diameter two, both solvable (PD2-Solvable) and reachable (PD2-Reachable) are in P. We will use results from [7] to establish an upper bound on $|V_2|$ in an unsolvable planar graph G with diameter two and apply this bound to a theorem from [2] to prove that the solvability of G can be determined in polynomial time. Throughout this section we will implicitly use the fact that a diameter two graph is solvable if any vertex can accumulate four pebbles.

The following is Theorem 2 from [7].

Theorem 5. *Every planar graph with diameter two has domination number at most two except for the unique graph with domination number three (Fig. 7).*

We now establish the following lemma:

Lemma 6. *Let G be a planar graph with diameter two with configuration C. If $|V_2| \geq 4$, then C is solvable.*

Proof. Let G be a planar graph with diameter two with $|V_2| = 4$.

If G has domination number one, let v be a dominating vertex. Since v is adjacent to every vertex it can accumulate at least two pebbles no matter which vertices begin with two pebbles. Then the remaining vertices can be pebbled to from v and C is solvable.

Let G have domination number two and assume C is an unsolvable configuration with $|V_2| = 4$. Let v_1 and v_2 be the two vertices in a minimum dominating set. If v_1 and v_2 are both in V_2, then every vertex is reachable and C is solvable. Similarly,

if exactly one is in V_2, then either v_1 and v_2 can both collect two pebbles or one of them can collect four pebbles, and C is solvable. So neither v_1 nor v_2 can be in V_2. Therefore there must be four vertices in V_2 that are adjacent to the dominating set. Clearly, neither v_1 nor v_2 can be adjacent to all four of these, and v_1 and v_2 cannot both be adjacent to two of them. So without loss of generality, let $t_1, t_2, t_3 \in V_2$ be adjacent to v_1 and $t_4 \in V_2$ be adjacent to v_2.

Since the graph is unsolvable, there must be some vertex r that is unreachable. Since r must be adjacent to the dominating set and it would be reachable if it was adjacent to v_1, it must be adjacent to v_2. By Lemma 14 from [5], there must be three vertices m_1, m_2, and m_3 adjacent to t_1, t_2, and t_3, respectively, each of which is also adjacent to r. It is not difficult to see that $\{v_1, v_2\} \cap \{m_1, m_2, m_3\} = \varnothing$ since r is not reachable. Now, m_1, m_2, and m_3 must be adjacent to the dominating set. If m_1 (similarly m_2, m_3) is adjacent to v_1, then t_2 and t_3 can pebble to v_1, t_1 and v_1 can pebble to m_1, and r is reachable. So m_1, m_2, and m_3 are all adjacent to v_2 (Fig. 8(a)). But this forms a subdivision of $K_{3,3}$ because if t_1, t_2, and t_3 are smoothed out, then m_1, m_2, and m_3 are each adjacent to v_1, v_2, and r (Fig. 8(b)). This contradicts G being planar. So a diameter two planar graph with domination number two is solvable if $|V_2| \geq 4$.

If G has domination number 3, then it is the unique graph from Fig. 7. By Corollary 17 from [5], if $|V_2| = 4$, then C is solvable.

Hence a planar graph G with diameter 2 is solvable if $|V_2| \geq 4$.

The following is Theorem 4.3 from [2].

Theorem 7. *Let G be a diameter two graph with a configuration C of pebbles such that the number of vertices with at least two pebbles is l. Then the solvability of G can be determined in $O(l! \cdot n^{2l-1}m)$ time, where n is the number of vertices in G and m is the number of edges.*

Our result follows:

Corollary 8. PD2-Solvable *is decidable in $O(n^6)$ time.*

Proof. By Lemma 6, if $|V_2| \geq 4$, then it is solvable. This can be determined in $O(n)$ time. Otherwise, the worst-case scenario is when $|V_2| = 3$. Since a planar graph has no more than $3n - 6$ edges, solvability can be determined in $O(3! \cdot n^{2(3)-1}(3n - 6)) = O(n^6)$ time by Theorem 7.

The algorithm from [2] to prove Theorem 7 can be easily modified to decide reachability in the same time, so PD2-Reachable can also be decided in $O(n^6)$ time. This result is interesting because both P-Reachable and D2-Reachable are NP-complete and likely not in P, but their intersection is in P.

4. Outerplanar graphs

In this section we present an algorithm to decide OP-Reachable in linear time. The algorithm makes use of a tree structure $T(G)$ that is associated with an embedding of an outerplanar graph G. Each node of $T(G)$ corresponds to either a chordless cycle or edge of G. The tree is related to the weak dual and is not necessarily unique for each embedding. We "prune" this tree by removing a leaf and making pebbling moves on the leaf's corresponding subgraph to the vertices that are also in the parent node or are the target vertex if the leaf is the root. This strategy is similar to the one used in [15] to find minimum dominating cycles in outerplanar graphs. For clarity, we will use the term *vertex* when speaking of the original graph and *node* when speaking of the associated tree. Similarly, we will use the term *target* when speaking of the original graph and *root* when speaking of the tree structure. When we speak of a node containing a vertex, we mean the node's corresponding subgraph in G contains that vertex.

4.1. An associated tree for outerplanar graphs

Given an outerplanar graph G and target vertex r, an associated tree $T(G)$ is constructed by first creating the weak dual of G, where each node of $T(G)$ corresponds to a chordless cycle of G. Then a node is added to $T(G)$ for each edge of G that is not on a cycle. Finally, edges are arbitrarily added between pairs of nodes from different trees whose corresponding subgraphs share a common vertex until the structure of $T(G)$ is a single tree. See Fig. 9 for an example. Notice that each node of $T(G)$ corresponds to either an edge or a chordless cycle of G. When we refer to the vertices of a node, we mean the vertices of G in the cycle/edge that corresponds to the node in $T(G)$. We will view $T(G)$ as a rooted tree by choosing one of the nodes containing r to be the root of $T(G)$.

Consider two adjacent nodes in $T(G)$. If either node corresponds to an edge in G, the nodes clearly share a single common vertex. If both nodes correspond to cycles in G, the nodes share either a single vertex or a pair of adjacent vertices (see Theorem 4 from [18]). For each non-root node in $T(G)$, we will refer to the vertices that it shares with its parent as *targets*. For the root node, the target vertex will be r.

It is well known that the weak dual of an outerplanar graph can be constructed in linear time [16,19]. By marking each edge that is contained in a cycle, we can add the nodes corresponding to the non-cycle edges in linear time. While computing the weak dual and these additional nodes, we can create a mapping from each node in $T(G)$ to the associated vertices in G and vice-versa. This allows us to connect all of the trees and determine the targets in linear time by using a breadth first search of G starting at r. Therefore, $T(G)$ can be constructed in linear time.

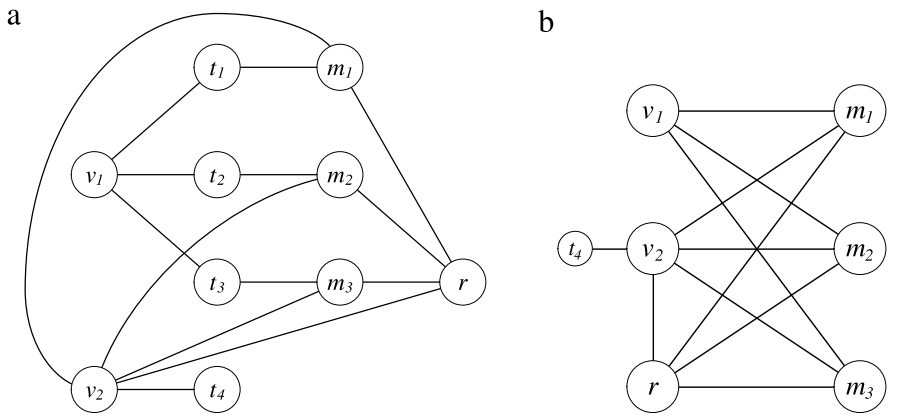

Fig. 8. (a) The subgraph of G used in the proof of Lemma 6. (b) The subgraph from (a) after smoothing out t_1, t_2, and t_3.

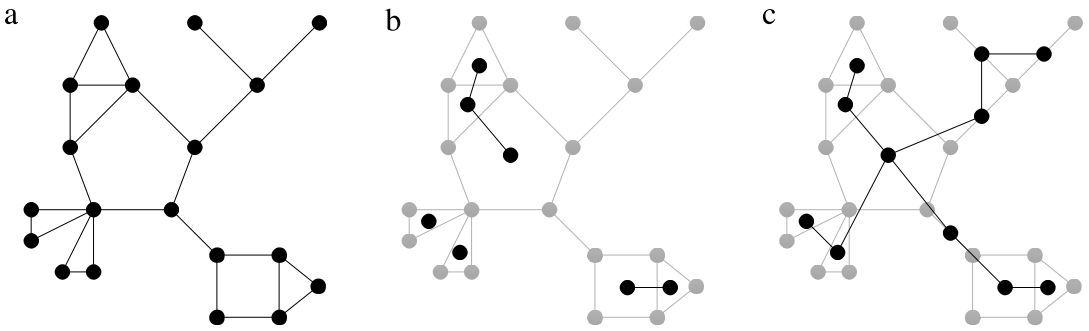

Fig. 9. (a) An outerplanar graph, (b) its weak dual, and (c) an associated tree.

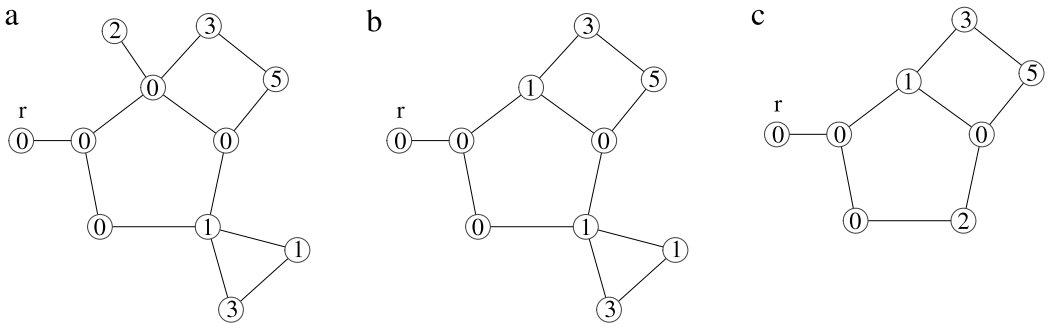

Fig. 10. (a) An outerplanar graph with target vertex r, (b) the graph after pruning a leaf corresponding to an edge, and (c) the graph after further pruning a leaf corresponding to a cycle with a single target vertex.

To prune the tree we select a leaf of $T(G)$, maximize the number of pebbles placed on the target vertices using only vertices from the leaf, and remove the leaf from $T(G)$. If the node has one target, we simply maximize the number of pebbles placed on the target. If the node has two targets, we find all of the maximal configurations that can be placed on the targets, taking all of these possibilities into account when pruning the parent node. Then the non-target vertices in the leaf are removed from G since they will not affect the reachability of r. Fig. 10 gives an example of pruning leaves with one target. An example of pruning leaves with two targets will be given after we characterize the maximal configurations.

Theorem 9. *Let G be an outerplanar graph with target vertex r and let $T(G)$ be an associated tree. If ϕ is a minimum pebbling sequence that places a pebble on r, then ϕ does not pebble from a target vertex of a node of $T(G)$ to a non-target vertex of that node.*

Proof. For nodes containing r the result is clear. Otherwise, suppose $(r_1, v) \in \phi$, where r_1 is a target and v a non-target vertex of some node. Let $\rho \subseteq \phi$ be the subsequence of moves that would not be legal if this move was removed. Since ϕ is minimal, ρ is a path from v to r. The target vertices of a node form a cut set that separates r and the non-target vertices of that node. Thus, ρ must pebble to one of the target vertices of the node. If $(w, r_1) \in \rho$ for some vertex w, ϕ contains a cycle and is not minimum. Thus, the node must contain a second target vertex r_2 that is adjacent to r_1 and $(w, r_2) \in \rho$ for some vertex w. In this case, the moves on the path from r_1 to r_2 can be replaced with the single move (r_1, r_2), and ϕ is not minimum.

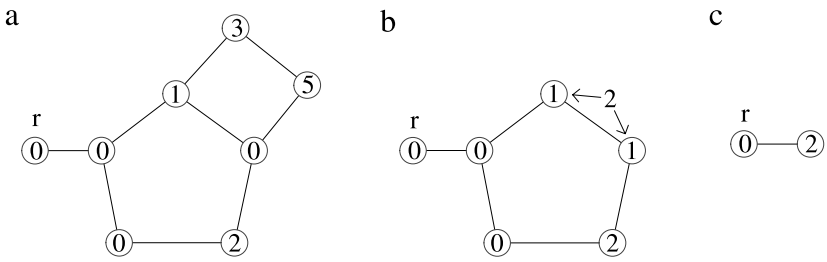

Fig. 11. (a) An outerplanar graph with designated target vertex r (from Fig. 10(c)), (b) the graph with a pool of size two resulting from pruning a leaf corresponding to a cycle with two targets, and (c) the graph after further pruning a leaf corresponding to a cycle that contains a pool.

Let ϕ be a minimum sequence that places a pebble on r, L be a leaf node and $\rho \subseteq \phi$ be the subset of moves from the non-target vertices of L. The preceding theorem implies that the moves from ρ can occur before any moves in $\phi - \rho$. Let G' be the subgraph of G with the non-target vertices of L removed. Then given the configuration C_ρ restricted to G', the sequence $\phi - \rho$ places a pebble on r. This leads to the following corollary.

Corollary 10. *Let G be an outerplanar graph with target vertex r and associated tree $T(G)$. Let L be a leaf of $T(G)$ and G' be the result of pruning L. Then r is reachable on G if and only if r is reachable on G'.*

This implies that r is reachable if and only if we can place a pebble on r by pruning every node of $T(G)$. The relationship between G and $T(G)$ implies that if each node can be pruned in linear time with respect to the number of vertices corresponding to that node, determining reachability of a vertex can be accomplished in linear time. If a node corresponds to an edge in G and the non-target vertex contains p pebbles, we simply move $\lfloor p/2 \rfloor$ pebbles onto the target. Pruning a node that is a cycle with one target is reducible from the two-target case by selecting one of the vertices adjacent to the target as a second target, pruning it as if it had two targets, and then moving all of the pebbles to the real target. In the next several sections we develop a linear algorithm to deal with the case of a cycle with two target vertices.

4.2. Pools on generalized cycles

As mentioned previously, for the two-target case we need to keep track of all of the possibilities of placing pebbles on both targets since we do not know beforehand how many pebbles we might need on each. In addition, it might be possible that (for instance) not placing three pebbles on one target allows us to place two additional pebbles on the other target, or any number of other possible trade-offs. As we prune the tree, the list of possibilities might grow too large to yield an efficient algorithm. Fortunately, we show that the optimal possibilities can be characterized by specifying three numbers—x pebbles that are placed on one target, y pebbles that are placed on the other target, and p pebbles that can be arbitrarily divided between the targets. We say these p pebbles form a *pool* between the targets. Fig. 11 gives an example of pruning a leaf with two targets. We will develop the details of how x, y, and p are determined in the next several sections.

When pruning a node, we need to take into account pools that were formed when the node's original children were pruned. We will show that the characterization just given remains even when these pools are present. In order to represent the pools, we will use a generalized version of a cycle on a weighted directed graph that has a vertex corresponding to each pool. But first we will look at a generalized path so we can prove results that ignore the possible moves between the target vertices.

4.3. Pooled paths

In this section we present several important results related to pebbling on paths. In particular, we are interested in maximizing the sum of the pebbles that can be placed on the two ends. Define PP_n (*Pooled Path*) to be the directed weighted path on vertices $v_1, \ldots, v_{2n-1}$ with edges (v_i, v_{i+1}) and (v_{i+1}, v_i) for $1 \le i \le 2n-2$, and $w(v_i, v_j) = 2$ if i is odd and 1 if i is even. Let $r^- = v_1$ and $r^+ = v_{2n-1}$ be the target vertices. For $1 < i < 2n-1$, define $W_i^- = \prod_{j=2}^{i} w(v_j, v_{j-1}) = 2^{\lfloor (i-1)/2 \rfloor}$ and $W_i^+ = \prod_{j=i}^{2n-2} w(v_j, v_{j+1}) = 2^{\lfloor (2n-1-i)/2 \rfloor}$. Notice that W_i^- (resp. W_i^+) pebbles on v_i allow 1 pebble to be placed on r^- (resp. r^+).

A configuration on the path graph P_n corresponds to the configuration on PP_n with each v_{2i-1} assigned the number of pebbles on the ith vertex in P_n and each v_{2i} assigned zero pebbles. On the other hand, notice that each pebble on a vertex of PP_n with even index can be moved to either adjacent vertex without costing any additional pebbles. Thus, PP_n is analogous to an extension of P_n that has the additional property that for each pair of adjacent vertices there may be some pebbles that can be placed on either vertex. These pebbles are said to form a *pool* between the two vertices.

Given a pebbling configuration C on PP_n, label the individual pebbles $1, 2, 3, \ldots, |C|$, beginning with every pebble on v_1, followed by every pebble on v_2, etc. This ordering is useful because a minimal sequence uses the closest pebbles to a target when pebbling to that target. Define i_k^- (resp. i_k^+) to be the smallest (resp. largest) index such that pebbles $1, \ldots, i_k^-$ (resp. $i_k^+, \ldots, |C|$) allow k pebbles to be placed on r^- (resp. r^+). For convenience we let $i_0^- = 0$ and $i_0^+ = |C| + 1$. Finally, define

$c_k^- = |\{i_j^+ : i_{k-1}^- < i_j^+ \leq i_k^-\}|$ (resp. $c_k^+ = |\{i_j^- : i_k^+ \leq i_j^- < i_{k-1}^+\}|$), which is said to be the *cost* of the kth pebble on r^- (resp. r^+). Notice that the cost of the kth pebble on r^- is the number of additional pebbles that could be placed on r^+ if only $k-1$ pebbles, as opposed to k, were placed on r^-. That is, if placing k pebbles on r^- allows for the placing of at most j pebbles on r^+ then placing $k-1$ pebbles on r^- allows for the placing of at most $j + c_k^-$ pebbles on r^+.

The following lemmas will be useful when we characterize the maximum number of pebbles that can be placed on r^- and r^+. For brevity we only include one proof of each respective result, but the other will hold because of the symmetry.

Lemma 11. *If $c_k^- = 0$ and l is the minimum value such that $i_l^+ \leq i_k^-$, then $c_l^+ \geq 2$. Similarly, if $c_k^+ = 0$ and l is the minimum value such that $i_l^- \geq i_k^+$, then $c_l^- \geq 2$.*

Proof. By choice of l, $i_l^+ \leq i_k^- < i_{l-1}^+$. Further, $c_k^- = 0$ implies that $i_{k-1}^- \geq i_l^+$. Thus $i_l^+ \leq i_{k-1}^- < i_k^- < i_{l-1}^+$, implying that $c_l^+ \geq 2$.

Lemma 12. *If $c_k^- \geq 2$ and l is the largest value such that $i_{k-1}^- < i_l^+$, then $c_l^+ = 0$. Similarly, if $c_k^+ \geq 2$ and l is the largest value such that $i_{k-1}^+ > i_l^-$, then $c_l^- = 0$.*

Proof. Since $c_k^- \geq 2$, $i_{k-1}^- < i_l^+ < i_{l-1}^+ \leq i_k^-$, so $c_l^+ = 0$.

Lemma 13. *If $c_k^- \geq 2$, then the i_k^-th pebble is on vertex v_i for some $n < i \leq 2n-1$. Similarly, if $c_k^+ \geq 2$, then the i_k^+th pebble is on vertex v_i for some $1 \leq i < n$.*

Proof. Since $c_k^- \geq 2$, there exists some b such that $i_{k-1}^- < i_{b+1}^+ < i_b^+ \leq i_k^-$. Let v_j be the vertex that has pebble $i_b^+ - 1$. Since pebbles $i_{b+1}^+, \ldots, i_b^+ - 1$ are sufficient to place 1 pebble on r^+, they must be able to place W_j^+ pebbles on vertex v_j. However, these same pebbles alone cannot be used to move 1 pebble to r^-. Thus $W_j^- > W_j^+$, implying that $j > n$. Since $i_b^+ \leq i_k^-$, the result follows.

Lemma 14. *If i_k^- is defined and pebble i_{k-1}^- is on vertex v_i for some $n < i \leq 2n+1$, then $c_k^- \geq 2$. Similarly, if i_k^+ is defined and pebble i_{k-1}^+ is on vertex v_i for some $1 \leq i < n$, then $c_k^+ \geq 2$.*

Proof. By definition of i_k^-, pebbles $i_{k-1}^- + 1, \ldots, i_k^-$ are sufficient to place 1 pebble on v_1. Since these pebbles are all on vertices with index at least $n+1$, they must place W_{n+1}^- pebbles on v_{n+1}. Since $W_{n+1}^- = 2W_{n+1}^+$, this implies that these pebbles can place at least 2 pebbles on v_n, so $c_k^- \geq 2$.

Lemmas 13 and 14 imply the following.

Corollary 15. *If $c_k^- \geq 2$, then $c_l^- \geq 2$ for all $l \geq k$ for which c_l^- is defined. Similarly, if $c_k^+ \geq 2$, then $c_l^+ \geq 2$ for all $l \geq k$ for which c_l^+ is defined.*

Define x to be the largest integer such that $c_x^- = 0$, or 0 if c_x^- is never 0; y to be the largest integer such that $c_y^+ = 0$, or 0 if c_x^+ is never 0; and p to be the largest integer such that $i_{x+p}^- < i_y^+$. Lemma 11 and Corollary 15 imply that $i_x^- < i_y^+$, so $p \geq 0$. It should be clear that it is possible to simultaneously place $x+p$ pebbles on r^- and up to y pebbles on r^+.

Lemma 16. *The following inequalities hold as written and when c_i^- is replaced with c_i^+ and x with y.*

1. *If $1 \leq i \leq x$, then $c_i^- \leq 1$.*
2. *If $x < i \leq x+p$, then $c_i^- = 1$.*
3. *If $i > x+p$ and c_i^- is defined, then $c_i^- \geq 2$.*

Proof. Lemma 12 implies that if $i \leq x+p$ then $c_i^- \leq 1$. Thus by choice of x, if $x < i \leq x+p$, then $c_i^- = 1$. By Lemma 11, $c_{x+p+1}^- \geq 2$ and by Corollary 15, if $i > x+p$ then $c_i^- \geq 2$.

By choice of p and since $c_i^- = 1$ for $x < i \leq x+p$, $y+p$ is the smallest value such that $i_{y+p}^+ > i_x^-$. Thus, a symmetric argument holds for c_i^-.

Corollary 17. *There exists a sequence of pebbling moves on PP_n that places $x+p-i$ pebbles on v_1 and $y+i$ pebbles on v_n for all $0 \leq i \leq p$. Further, no sequence of pebbling moves places more than $x+y+p$ pebbles total on r^- and r^+.*

Because the number of pebbles in a configuration can be exponential in the size of the problem, the techniques used in the proofs do not yield a linear algorithm to determine x, y, and p. Nonetheless, they can be computed in linear time, as we see next.

Theorem 18. *The values of x, y, and p can be determined in linear time with respect to the size of PP_n.*

Proof. For $1 \leq i \leq 2n-1$, define q_i^- to be the number of pebbles that can be placed on r^- using pebbles from vertices $v_1, \ldots, v_i$, and let $q_0^- = 0$. Define d_i^- to be the number of pebbles needed on v_i, in addition to the pebbles on $v_1, \ldots, v_{i-1}$, to place $q_{i-1}^- + 1$ pebbles on r^-. It is easy to see that $d_1^- = 1$. For $2 \leq i \leq 2n-2$, let $s_i^- = C(v_i) - w(v_i, v_{i-1}) \cdot d_{i-1}^-$.

If $s_i^- \geq 0$, then there are enough pebbles on v_i to place $1 + \lfloor s_i^-/W_i^- \rfloor$ additional pebbles on r^-. Therefore, $q_i^- = q_{i-1}^- + 1 + \left\lfloor s_i^-/W_i^- \right\rfloor$. After these pebbles are placed, there are $s_i^- \bmod W_i^-$ pebbles left on v_i, so $d_i^- = W_i^- - (s_i^- \bmod W_i^-)$.

If $s_i^- < 0$, there are not enough pebbles on v_i to place at least one more on r^-, so $q_i^- = q_{i-1}^-$ and $d_i^- = -s_i^-$.

We can use these equations to compute these values for all indices in linear time. We define and compute d_i^+ and q_i^+ symmetrically.

Define t_k^+ to be the maximum number of pebbles that could be placed on r^+ if k pebbles are placed on r^-. Let i be the smallest value such that $q_i^- \geq k$. Then $d_i^- + W_i^- \cdot (k - q_{i-1}^- - 1)$ pebbles are needed from v_i to add k pebbles to r^-. If we remove this many pebbles from v_i and recompute q_i^+, then $t_k^+ = q_i^+$. Therefore we can compute t_k^+ in linear time for a given k. Since $c_k^- = t_{k-1}^+ - t_k^+$, c_k^- can also be computed in linear time.

Let $a = q_n^- + 1$. That is, a is one more than the number of pebbles that can be placed on r^- using pebbles on $v_1, \ldots, v_n$. By Lemma 14, if $a+1$ pebbles can be placed on r^-, then $c_{a+1}^- \geq 2$. By Lemma 13 and choice of a, if $1 \leq b < a$ then $c_b^- < 2$. By Lemma 16, if $c_a^- \geq 2$ or a pebbles cannot be placed on r^-, then $x + p = a - 1$. Otherwise $x + p = a$. Thus, we can determine $x + p$ in linear time.

We can use a symmetric process to find $y + p$. We can also determine $y = t_{x+p}^+$ in linear time. From these we can find x, y, and p in constant time.

4.4. Pooled cycles

In this section, we utilize the results from the previous section to prove an important result for pebbling on cycles. Define PC_n (*Pooled Cycle*) to be the directed weighted graph obtained by adding a vertex to PP_n that represents the pool between v_1 and v_{2n-1}. That is, we add vertex v_0 and edges (v_0, v_1) and (v_0, v_{2n-1}) of weight 1, and edges (v_{2n-1}, v_0) and (v_1, v_0) of weight 2. Similar to the way that PP_n is analogous to P_n, PC_n is analogous to the cycle C_n with the additional property that for each pair of adjacent vertices in C_n there may be some pebbles that can be placed on either vertex. Again, these pebbles are said to form a *pool* between the vertices. Since our goal is to determine the maximal configurations of pebbles that can be added to r^- and r^+ from $v_2, \ldots, v_{2n-2}$, pebbles already on r^-, r^+, or v_0 are irrelevant in this context and will be ignored.

Theorem 19. *If ϕ is a pebbling sequence on PC_n that places a pebbles on r^- and b pebbles on r^+ then*

1. $b \leq y + p + \frac{x-a}{2}$,
2. $b \leq y - 2(a - (x + p))$, *and*
3. $b \leq x + y + p - a$.

Proof. Suppose ϕ is a minimal sequence that places a pebbles on r^- and b pebbles on r^+. Also suppose ϕ contains μ moves of the form (v_2, v_1) and ν moves of the form (v_{2n-2}, v_{2n-1}).

By Corollary 17,

$$\mu + \nu \leq x + y + p. \tag{1}$$

If $b \leq \nu$, then ϕ does not pebble from r^- to r^+. Then $a \geq \mu$ and $b = \nu + 2(\mu - a) \leq \nu + \frac{\mu-a}{2}$. On the other hand, if $b \geq \nu$, then ϕ does not pebble from r^+ to r^-. Then $a \leq \mu$ and $b = \nu + \frac{\mu-a}{2} \leq \nu + 2(\mu - a)$. In either case

$$b \leq \nu + 2(\mu - a) \tag{2}$$

and

$$b \leq \nu + \frac{\mu - a}{2}. \tag{3}$$

Now we can consider each case.

1. Notice that $\mu \leq x + p$ since if $\mu > x + p$, then $c_\mu^- \geq 2$, so there exists a minimal sequence that places a pebbles on r^- and b pebbles r^+ but contains the move (v_2, v_1) one less time. This and Eqs. (1) and (3) give $b \leq y + p + \frac{x-a}{2}$.
2. Similarly we can see that $\nu \leq y + p$. This and Eqs. (1) and (2) give $b \leq y - 2(a - (x + p))$.
3. Because pebbling between r^- and r^+ decreases the sum of the pebbles on each, $a + b \leq \mu + \nu$. This and Eq. (1) give $b \leq x + y + p - a$.

Theorem 19 characterizes the boundary on the number of pebbles that can be placed on r^- and r^+. Corollary 17 implies that all of the possibilities within the boundary are attainable when moves between the targets are allowed (see Fig. 12). Together, they imply that if x, y, and p can be determined, then any valid configuration of pebbles on r^- and r^+ can be determined. More specifically, if x pebbles are placed on r^-, y on r^+, and p on v_0, then any other valid configuration on r^- and r^+ is obtainable by making the appropriate moves between these three vertices.

4.5. Pebbling outerplanar graphs

We now have the tools necessary to prune the associated tree if the leaf is a cycle with two targets and we are ready for the main result of this chapter.

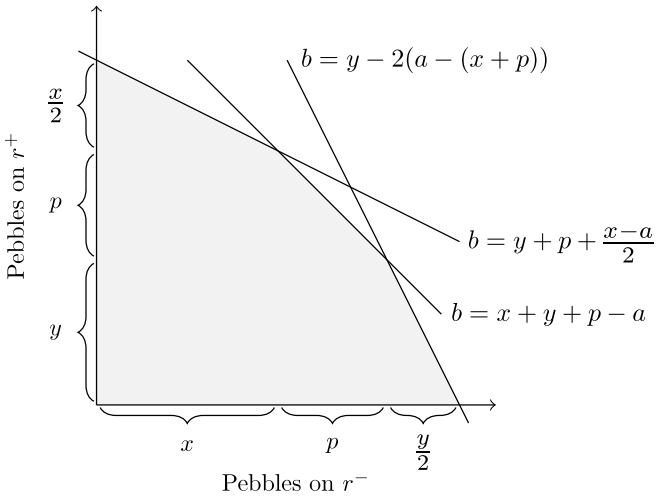

Fig. 12. The integer coordinates contained in the closed, shaded region represents all possible combinations of pebbles that can be placed on r^- and r^+ according to Corollary 17 and Theorem 19.

Theorem 20. OP-Reachable *is decidable in linear time.*

Proof. Let G be an outerplanar graph with target vertex r and pebbling configuration C. Construct the associated tree $T(G)$. For each cycle with one target (r^+), choose one of its neighbors on the cycle as the second target (r^-). Replace each cycle with a corresponding pooled cycle. Now prune $T(G)$ until it is empty. In particular, for leaves corresponding to an edge, move as many pebbles to the target as possible. For cycles, temporarily ignore vertex v_0 and the pebbles on r^+ and r^- and determine the values of x, y, and p according to Theorem 18. Add x pebbles to r^-, y pebbles to r^+, and p pebbles to v_0. For cycles that have one target, move as many pebbles as possible from r^- and v_0 to r^+. Then for any type of leaf, remove the non-target vertices of the leaf from G. Once all of the nodes from $T(G)$ have been pruned, we declare r reachable if and only if it now has at least one pebble. It is straightforward to see that the process is completed in linear time.

Since Solvable can always be decided by running an algorithm that decides Reachable at most $|V|$ times we also have the following.

Corollary 21. OP-Solvable *is decidable in* $O(|V|^2)$ *time.*

5. Further study

Since many problems, even those that are NP-complete, are decidable in linear time when restricted to outerplanar graphs, it would be interesting to determine if OP-Solvable is also decidable in linear time.

It should be clear that if Reachable is NP-complete for planar graphs of diameter k, then it is NP-complete for graphs with diameter greater than k. Similarly, if there is a polynomial time algorithm to decide Reachable on planar graphs of diameter k, then a polynomial time algorithm exists for planar graphs with diameter less than k. Given these facts, it would be interesting to determine which of several possible scenarios holds. Is it the case that Reachable is NP-complete for all $k > 2$? Or is there a polynomial time algorithm for any constant k? Or are there thresholds $k' < k''$ for which Reachable is NP-complete for $k \geq k''$ and decidable in polynomial time for $k \leq k'$?

Finally, how difficult is it to decide Pebbling-Number for planar and outerplanar graphs? Pebbling-Number was shown to be Π_2^p-complete in general [14]. Notice that for any class of graphs for which reachable can be determined in polynomial time, Pebbling-Number is no harder than NP-complete. Thus, we know that Pebbling-Number on outerplanar graphs is no harder than NP-complete. However, can it be decided in polynomial time? For planar graphs, we only know that it is no more difficult than Π_2^p-complete.

References

[1] L. Alcón, M. Gutierrez, G. Hurlbert, Pebbling in split graphs, 2012. ArXiv e-prints.
[2] A. Bekmetjev, C.A. Cusack, Pebbling algorithms in diameter two graphs, SIAM J. Discrete Math. 23 (2009) 634–646.
[3] A. Blasiak, J. Schmitt, Degree sum conditions in graph pebbling, Australas. J. Combin. 42 (2008) 83–90.
[4] T.A. Clarke, R.A. Hochberg, G.H. Hurlbert, Pebbling in diameter two graphs and products of graphs, J. Graph Theory 25 (1997) 119–128.
[5] C. Cusack, T. Lewis, D. Simpson, S. Taggart, The complexity of pebbling in diameter two graphs, SIAM J. Discrete Math. 26 (2012) 919–928.
[6] S. Elledge, G.H. Hurlbert, An application of graph pebbling to zero-sum sequences in abelian groups, Integers: Electron. J. Combin. Number Theory 5 (2005).
[7] W. Goddard, M.A. Henning, Domination in planar graphs with small diameter, J. Graph Theory 40 (2002) 1–25.

[8] D. Herscovici, B. Hester, G. Hurlbert, t-Pebbling and extensions, Graphs Combin. 29 (2013) 955–975.
[9] G. Hurlbert, A linear optimization technique for graph pebbling, 2011. ArXiv e-prints.
[10] G. Hurlbert, General graph pebbling, Discrete Appl. Math. 161 (2013) 1221–1231. Jubilee Conference on Discrete Mathematics.
[11] G. Hurlbert, H. Kierstead, On the complexity of graph pebbling, 2005. Unpublished.
[12] S. Jones, J.D. Laison, C. McLeman, K. Nyman, Weighted pebbling numbers on graphs, 2011. ArXiv e-prints.
[13] D. Lichtenstein, Planar formulae and their uses, SIAM J. Comput. 11 (1982) 329–343.
[14] K. Milans, B. Clark, The complexity of graph pebbling, SIAM J. Discrete Math. 20 (2006) 769–798.
[15] A. Proskurowski, M.M. Sysło, Minimum dominating cycles in outerplanar graphs, Int. J. Comput. Inf. Sci. 10 (1981) 127–139.
[16] A. Proskurowski, M. Sysło, Efficient vertex-and edge-coloring of outerplanar graphs, SIAM J. Algebra Discrete Methods 7 (1986) 131–136.
[17] N. Sieben, A graph pebbling algorithm on weighted graphs, J. Graph Algorithms Appl. 14 (2010) 221–244.
[18] M.M. Sysło, Characterizations of outerplanar graphs, Discrete Math. 26 (1979) 47–53.
[19] M. Sysło, An efficient cycle vector space algorithm for listing all cycles of a planar graph, SIAM J. Comput. 10 (1981) 797–808.
[20] N.G. Watson, The complexity of pebbling and cover pebbling, 2005. ArXiv Mathematics e-prints.