ROMAI J., v.21, no.2(2025), 89–98

A HEURISTIC ALGORITHM FOR 2-RESTRICTED OPTIMAL PEBBLING PROBLEM

Abdorreza Panahi[1], Gyula Y. Katona[2]

[1] *Department of Process Management, Budapest University of Economics and Business, Budapest, Hungary*

Department of Computer Science, Budapest University of Technology and Economics, Budapest, Hungary

abdorreza.panahi@uni-bge.hu, katona.gyula@vik.bme.hu

Abstract The graph pebbling problem involves the distribution and movement of indistinguishable pebbles on the vertices of a connected graph. A pebbling move consists of removing two pebbles from a vertex and placing one pebble on an adjacent vertex. A pebble distribution is considered *solvable* if it is possible to move at least one pebble to any vertex with a sequence of pebbling moves. A fundamental question in graph pebbling is to determine the minimum number of pebbles required to guarantee that any target vertex can be reached by a sequence of pebbling moves, regardless of the initial pebble distribution. This minimum number is known as the *pebbling number* of the graph, denoted by $\pi(G)$. The optimal pebbling number $\pi^*(G)$ of a graph G is the smallest number of pebbles that can be placed on its vertices such that the distribution is solvable.

Variations of the standard pebbling problem have been introduced and studied. One such variation is *k-restricted optimal pebbling*, where the limit of at most k pebbles per vertex applies only to the initial distribution, see [1]. This paper focuses on the specific case of $k = 2$, known as *2-restricted optimal pebbling* denoted by $\pi_2^*(G)$. This is an NP-hard problem, so we developed a greedy heuristic algorithm that attempts to determine $\pi_2^*(G)$. We show that for some graphs it is able to find the optimum, while in other cases it is far from it.

Keywords: Pebbling number, NP-complete, Heuristic algorithm.
2020 MSC: 68W25.

1. INTRODUCTION

Graph pebbling is a combinatorial game played on the vertices of a graph. The standard pebbling problem, first introduced by F.R.K. Chung, asks for the minimum number of pebbles needed to guarantee that a single pebble can be moved to any chosen vertex [2, 3]. A single pebbling takes 2 pebbles from a vertex and adds one pebble to one of its neighbor, the other pebble is thrown away. A pebble distribution is said to be *solvable* if at least one pebble can be moved to any vertex on the graph using a sequence of pebbling moves. The *pebbling number* $\pi(G)$ of a graph G is the minimum number of pebbles needed

so that, no matter how they are distributed on the vertices, the distribution is solvable. Another important parameter is the *optimal pebbling number* $\pi^*(G)$ of a graph G is the smallest number of pebbles that can be placed on its vertices such that the distribution is solvable. These problems have been a subject of significant research, with numerous variations and related problems explored to understand its properties and applications [4].

One such variation is the *k-restricted optimal pebbling* problem. This modification imposes a constraint on the initial distribution of pebbles, limiting the number of pebbles that can be placed on any single vertex. In this paper, we focus on the specific case where this restriction is $k = 2$, which we denote as 2-restricted optimal pebbling, with the corresponding pebbling number $\pi_2^*(G)$. Determining this value is a significantly more complex challenge than the standard version. The problem is NP-complete, which means finding the exact minimum number of pebbles is computationally intractable for large graphs [5]. Given this complexity, exact solutions seems impossible for real-world scenarios.

To address this computational challenge, this work proposes a novel greedy algorithm. The objective of this algorithm is to find a pebble distribution that attempts to find the optimal solution for $\pi_2^*(G)$. The remainder of this paper outlines our approach, the formal problem formulation, and the details of our proposed greedy algorithm. Finally, we show that for some graphs our algorithm is able to find the optimum, while in other cases it is far from it.

2. PROBLEM FORMULATION

Let the graph $G = (V, E)$ be a connected, simple graph with the set of vertices V and the set of edges E. A pebbling distribution is a function $f : V \to \mathbb{N}$ and in case of 2-restricted pebbling problem $f : V \to \{0, 1, 2\}$, where $f(v)$ is the number of pebbles on vertex v. Throughout the examples, the vertex v is shown by a red square if $f(v) = 2$ and it is distinguished with a red circle if $f(v) = 1$. A pebbling move at a vertex u with at least two pebbles allows us to remove two pebbles from u and place one pebble on an adjacent vertex v. All the other vertices without pebbles that are reachable by a sequence of pebble moves are shown by green circles. The k-restricted optimal pebbling problem is to find the minimum number of pebbles, m, such that there exists a distribution with m pebbles, where no vertex has more than k pebbles, and it is possible to move one pebble to any target vertex $t \in V$. The specific problem addressed here is the case where $k = 2$, where we seek to determine $\pi_2^*(G)$. It is proved in [5] that it is NP-hard to determine $\pi_2^*(G)$, we propose a heuristic greedy algorithm.

3. PROPOSED ALGORITHM

We define the set of vertices V_1, V_2 and the set of edges E_1 and E_2 to be able to determine the degree of vertices in the subgraphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$. Based on the current pebble distribution and the maximum degree over $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ separately, we will determine the vertex to assign one or two pebbles, respectively.

$V_1 = \{v_i \in V : v_i \text{ is reachable.}\}$
$V_2 = V - V_1$
$E_1 = \{\{v_i, v_j\} \in E : v_i \in V_1, v_j \in V_2\}$
$E_2 = \{\{v_i, v_j\} \in E : v_i, v_j \in V_2\}$

- **Initialization step:**
  Set $V_1 = \{\}$, then $V_2 = V - V_1 = V$ and $f(v) = 0$ for all $v \in V$. Therefore, after updating the set of edges in G_1 and G_2, we have $E_1 = \{\}$ and $E_2 = E$.

- **Main step:**
  If $V_2 = \{\}$, stop. The function f is a solvable pebble distribution. Otherwise, find the vertex $u \in V_1$ for which the degree d_u is maximal in the subgraph $G_1(V, E_1)$. Then find the vertex $v \in V_2$ for which the degree d_v is maximal in the subgraph $G_2(V, E_2)$. In case of ties we choose the vertex with the smaller index. By assigning one pebble to u we can add d_u vertices to V_1, or by assigning two pebbles to v we can add $d_v + 1$ vertices to V_1. If $2d_u < (d_v + 1)$ then we add v to V_1 and set $f(v) = 2$, otherwise we set $f(u) = 1$ and update V_2, E_1, and E_2, and repeat the main step.

We demonstrate the algorithm on the example graph shown on Figure 1.

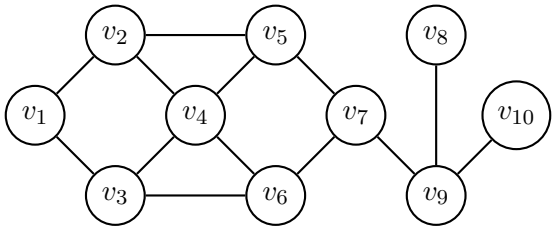


Fig. 1. An example graph.

At the initialization step $V_1 = \{\}$, then $V_2 = V - V_1 = V$ and $f(v) = 0$ for all $v \in V$. Since there is no vertex in V_1 then $E_1 = \{\}$ and $E_2 = E$. In the first iteration $d_u = 0$ since $E_1 = \{\}$. The degree of vertex v_4 is 4, which is maximal in $G_2(V, E_2)$. Now modify the value of $f(v)$ by changing $f(v_4) = 2$,

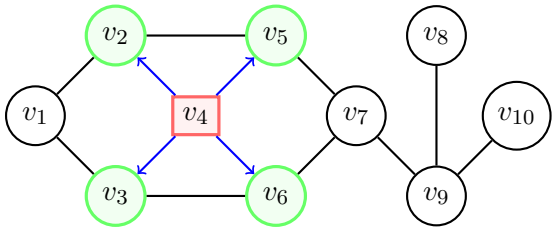


Fig. 2. G after the first iteration.

which assigns two pebbles to this vertex. The other elements in the algorithm are modified as follows:

$V_1 = \{v_4, v_2, v_3, v_5, v_6\}$
$V_2 = \{v_1, v_7, v_8, v_9, v_{10}\}$
$E_1 = \{\{v_2, v_1\}, \{v_3, v_1\}, \{v_5, v_7\}, \{v_6, v_7\}\}$
$E_2 = \{\{v_7, v_9\}, \{v_8, v_9\}, \{v_9, v_{10}\}\}$

In the second iteration, we find the maximum degree d_u in G_1 among the reachable vertices, which are shown by green circles: $d_u = 1$.

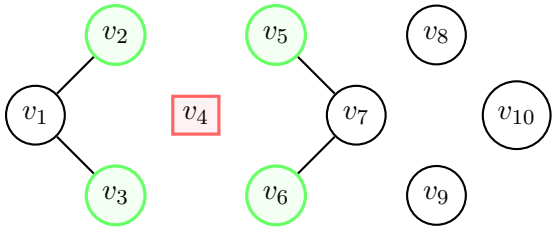


Fig. 3. G_1 after the first iteration.

Now, the degree of vertex v_9 is 3, which is maximal in G_2.

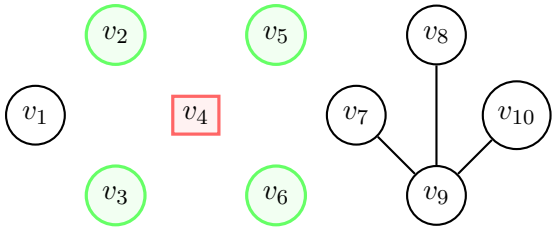


Fig. 4. G_2 after the first iteration.

Since $2d_u \geq (d_v + 1)$ then we add $v = v_9$ to V_1 and set $f(v_9) = 2$ and update V_2, E_1, and E_2 and repeat the main step. (See Fig. 5.)

$V_1 = \{v_4, v_2, v_3, v_5, v_6, v_7, v_8, v_9, v_{10}\}$
$V_2 = \{v_1\}$
$E_1 = \{\{v_2, v_1\}, \{v_3, v_1\}\}$
$E_2 = \{\}$

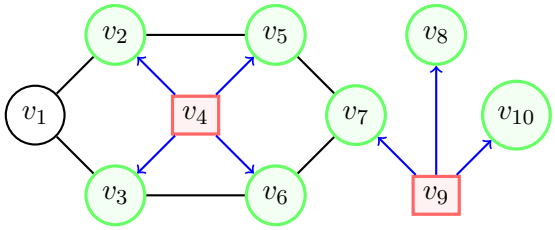


Fig. 5. G after the second iteration.

In the third iteration, the set of vertices in graph G_2 is empty and $d_u = 1$ where $u = v_2$. By assigning one pebble to the vertex $u = v_2$, we have all the vertices in V_1, so the distribution is solvable. (See Fig. 6.)

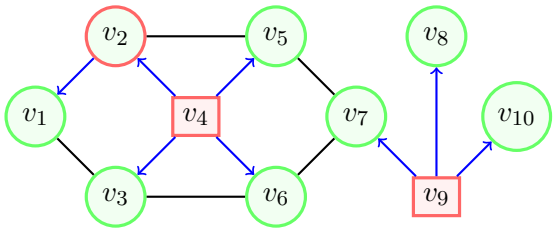


Fig. 6. G after the third iteration.

The number of pebbles the algorithm uses is $m = f(v_2) + f(v_4) + f(v_9) = 1 + 2 + 2 = 5$. This clearly shows that $\pi_2^*(G) \leq 5$. By case analysis it is not hard to see that in fact $\pi_2^*(G) = 5$, so in this case our algorithm works well.

For clarity, we provide a pseudo-code for our algorithm, see Algorithm 1.

Algorithm 1: Pebble Distribution Algorithm

Input: A graph $G = (V, E)$
Output: A solvable pebble distribution function $f : V \to \{0, 1, 2\}$
Initialization Step:
$V_1 \leftarrow \emptyset$
$V_2 \leftarrow V$
$E_1 \leftarrow \emptyset$
$E_2 \leftarrow E$
for *each* $v \in V$ **do**
  $f(v) \leftarrow 0$
end

Main Step:
while $V_2 \neq \emptyset$ **do**
  `// Construct the graphs` G_1 `and` G_2`.`
  $E_1 \leftarrow \{\{v_i, v_j\} \in E : v_i \in V_1, v_j \in V_2\}$
  $E_2 \leftarrow \{\{v_i, v_j\} \in E : v_i, v_j \in V_2\}$
  `// Find candidates for 1-pebble or 2-pebble assignments`
  $u \leftarrow$
   vertex with smallest index in V_1 maximizing degree d_u in $G_1(V, E_1)$
  $v \leftarrow$
   vertex with smallest index in V_2 maximizing degree d_v in $G_2(V, E_2)$
  `// Make a greedy choice.`
  if $2d_u < (d_v + 1)$ **then**
    $f(v) \leftarrow 2$
    Update V_1, V_2 by adding v and its neighbors to V_1
  else
    $f(u) \leftarrow 1$
    Update V_1, V_2 by adding neighbors of u to V_1
  end
end
return f

Running time of the algorithm

Theorem 3.1. *The algorithm terminates in polynomial time with respect to the number of vertices* $|V| = n$ *and edges* $|E| = m$.

Proof. To demonstrate that the algorithm is polynomial, we analyze the iteration count of the main loop and the computational cost of the operations within each iteration:

1 **Loop Convergence:** In each iteration of the `while` loop, at least one vertex is moved from the set of unreachable vertices V_2 to the set of reachable vertices V_1. Specifically:

- If a 2-pebble assignment is made, vertex v and all its neighbors in V_2 are moved to V_1 (at least 1 vertex).
- If a 1-pebble assignment is made, all neighbors of u in V_2 are moved to V_1 (at least 1 vertex, otherwise d_u would be 0).

Since $|V_2|$ starts at n and decreases by at least 1 in each step, the loop executes at most n times.

2 **Work Per Iteration:** Within each loop iteration, the algorithm performs the following:

- **Graph Partitioning:** Updating the edge sets E_1 and E_2 requires scanning the adjacency lists, which takes $O(n + m)$ time.
- **Degree Calculation:** Finding the vertices $u \in V_1$ and $v \in V_2$ with maximum degrees d_u and d_v involves a linear scan of the current edge sets, taking $O(n + m)$ time.
- **Set Operations:** Updating the vertex sets V_1 and V_2 takes at most $O(n)$ time.

3 **Total Complexity:** Combining the loop count and the work per iteration, the total time complexity is:

$$T(n, m) = O(n \cdot (n + m)) = O(n^2 + nm)$$

In the worst case of a dense graph where $m = O(n^2)$, the complexity is $O(n^3)$, so the algorithm is polynomial.

∎

4. PERFORMANCE ANALYSIS

Let $\pi_A(G)$ denote the size of the pebble distribution obtained by our algorithm. It is clear from the definitions that $\pi^*(G) \leq \pi_2^*(G) \leq \pi_A(G)$ hold for all graphs. $\pi_2^*(G)$ is not yet know for many graphs yet. However, if for some graph G we can show that $\pi^*(G) = \pi_A(G)$, then clearly we have $\pi^*(G) = \pi_A(G)$ as well.

Let as investigate the performance of the algorithm for some basic examples. First, it is easy to see what happens for complete graphs and stars.

Proposition 4.1. *If The maximum degree of an $n \leq 2$ vertex graph G is $n-1$, then the algorithm finds the optimum, so $\pi^*(G) = \pi_A(G) = 2$.*

Proof. The algorithm will place 2 pebbles at a vertex of maximum degree. Since this vertex is connected to all other vertices, this is already a solvable distribution, so the algorithm stops: $\pi_A(G) = 2$. On the other hand, it is obvious that 1 pebble cannot form a solvable distribution, thus $\pi^*(G) \geq 2$. ∎

Now we consider paths and cycles. In [6] the authors proved that $\pi^*(P_n) = \pi^*(C_n) = \lceil 2n/3 \rceil$ holds where P_n is a path, C_n is a cycle on $n \geq 3$ vertices. In [1] it was proved that $\pi_2^*(P_n) = \pi^*(P_n) = \lceil 2n/3 \rceil$.

Theorem 4.1. *Our algorithm finds the optimum for paths and cycles, i.e., if $n \geq 3$ then $\pi_A^*(P_n) = \pi_A^*(C_n) = \lceil 2n/3 \rceil$.*

Proof. Let $v_1, \ldots, v_n$ denote the vertices of P_n in the same order as they come on the path. It is clear that the algorithm will place 2 pebbles first to v_2. Now $d_u = 1$ and $d_v = 2$, thus in the following iterations the algorithm places 2 pebbles to $v_5, v_8, \ldots, v_{3\lfloor n/3 \rfloor - 1}$. If $n \equiv 0 \pmod 3$ then the algorithm terminates. If $n \equiv 1 \pmod 3$ then in the last iteration $d_u = 1$ and $d_v = 0$, so we place 1 pebble to v_n. If $n \equiv 2 \pmod 3$ then the algorithm will place one pebble to v_{n-2} and one to v_{n-1}. In each case $\pi_A^*(P_n) = \lceil 2n/3 \rceil$.

Now let $v_1, \ldots, v_n$ denote the vertices of C_n in the same order as they come on the cycle. It is clear that the algorithm will place 2 pebbles first to v_1. Now $d_u = 1$ and $d_v = 2$, thus in the following iterations the algorithm places 2 pebbles to $v_4, v_7, \ldots, v_{3\lfloor n/3 \rfloor - 2}$. If $n \equiv 0 \pmod 3$ then the algorithm terminates. If $n \equiv 1 \pmod 3$ then in the last iteration $d_u = 1$ and $d_v = 0$, so we place 1 pebble to v_{n-1}. If $n \equiv 2 \pmod 3$ then the algorithm will place one pebble to v_{n-3} and one to v_{n-2}. In each case $\pi_A^*(C_n) = \lceil 2n/3 \rceil$. ∎

5. PERFORMANCE ANALYSIS ON THE HUB-LEAF GRAPH

In this section, we show that our algorithm does not give a good approximation of $\pi_2^*(G)$ in general. We give an example of a class of graphs where $\pi_A^*(G) - \pi_2^*(G)$ and even $\pi_A^*(G)/\pi_2^*(G)$ can be arbitrarily large.

Let us define the Hub-Leaf graph $\mathrm{HL}(k, m)$. Take a complete bipartite graph on partite sets $\{x, y\}$ and $W = \{w_1, w_2, \ldots, w_k\}$ where $k \geq 2$. Now connect m new isolated vertex $z_{(i,1)}, \ldots, z_{(i,m)}$ to each vertex w_i. (See Fig. 7.)

Proposition 5.1. $\pi_2^*(\mathrm{HL}(k, m)) = 4$.

Proof. Placing 2-2 pebbles to x and y gives a solvable distribution. It it obvious that each w_i is reachable, but note that we can move one of the pebbles from

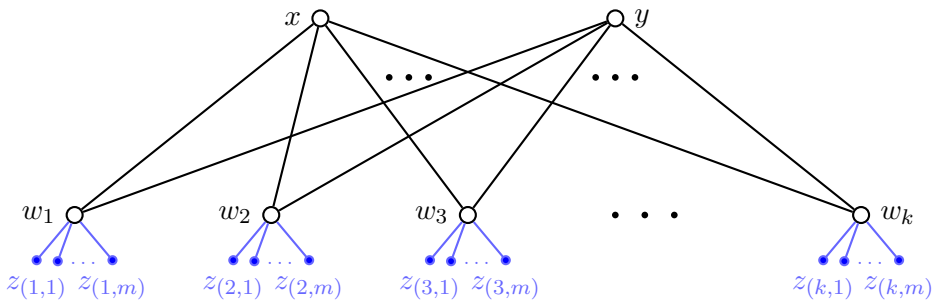


Fig. 7. The Hub-Leaf graph

both x and y to w_i. Thus, with an additional pebbling move, the degree one neighbors of w_i are also reachable. This proves that $\pi_2^*(\mathrm{HL}(k,m)) \leq 4$. On the other hand, it is easy to see by case analysis that 3 pebbles cannot give a solvable distribution. ∎

Proposition 5.2. *If* $m \geq 2k-2$ *then* $\pi_A^*(\mathrm{HL}(k,m)) = 2k$.

Proof. In the first iteration $V_1 = \emptyset, d_u = 0$ and $d_v = d(w_1) = m+2$, so 2 pebbles is placed on w_1. After this, V_1 and V_2 are updated: $V_1 = \{w_1, x, y, z_{(1,1)}, \ldots, z_{(1,m)}\}$ and $V_2 = V - V_1$.

In the second iteration we have $E_1 = \{xw_2, \ldots xw_k, yw_2, \ldots yw_k\}$, thus $d_u = k-1$. Since $d_v = d_{G_2}(w_2) = m > 2(k-1)-1$, now we place 2 pebbles to w_2. It is easy to see that in the following iterations we place 2 pebbles to each w_i. ∎

Since k can be chosen arbitrary large $\pi_A^*(\mathrm{HL}(k,m)) - \pi_2^*(\mathrm{HL}(k,m)) = 2k-4$ and

$$\frac{\pi_A^*(\mathrm{HL}(k,m))}{\pi_2^*(\mathrm{HL}(k,m))} = \frac{2k}{4} = \frac{k}{2}$$

can be arbitrary large.

Conclusion

The 2-restricted optimal pebbling problem, denoted by $\pi_2^*(G)$, presents a significant computational challenge due to its NP-hardness. In this paper, we have formulated the problem and proposed a greedy heuristic algorithm to estimate it. We found that although it is able to find the optimal value for some graphs, it can be very far from it in other cases. It would be interesting to characterize the graphs for which the algorithm finds the correct value. Also, we are planning to improve the heuristics to improve the performance of the algorithm.

References

[1] M. Chellali, T.W. Haynes, S.T. Hedetniemi, T. M. Lewis, *Restricted optimal pebbling and domination in graphs*, Discrete Applied Math. 221 (2017) pp. 46–53.

[2] F.R.K. Chung, *Pebbling in hypercubes*, SIAM J. Discrete Math. 2 (4) (1989) 467–472.

[3] A. Czygrinow, G. Hurlbert, GY. Y. Katona, L. F. Papp, *Optimal pebbling number of graphs with given minimum degree*, Discrete Applied Mathematics, 260 (2019) pp. 117–130.

[4] G. Hurlbert, *Graph pebbling*, Handbook of graph theory (2013): pp. 1428–1449.

[5] L.F. Papp, *Restricted optimal pebbling is NP-hard*, Proceedings of the 11th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications (2019)

[6] L. Pachtor, H.S. Snevily, B. Voxman, *On pebbling graphs*, Congr. Numer. 107 (1995) pp. 65–80.