Discrete Applied Mathematics 320 (2022) 11–14

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

A faster algorithm for Cops and Robbers

Jan Petr, Julien Portier *, Leo Versteegen

Centre for Mathematical Sciences, Wilberforce Road, Cambridge CB3 0WA, United Kingdom

ARTICLE INFO

Article history:
Received 11 January 2022
Received in revised form 5 May 2022
Accepted 20 May 2022
Available online 9 June 2022

Keywords:
Cops and Robbers
Algorithm
Complexity

ABSTRACT

We present an algorithm of time complexity $O(kn^{k+2})$ deciding whether a graph G on n vertices is k-copwin. The fastest algorithm thus far had time complexity $O(n^{2k+2})$.

1. Introduction

Cops and Robbers, introduced by Nowakowski and Winkler [8] and Quillot [9], is a game played on a finite undirected graph G between two players, one of whom controls k cops while the other controls a single robber. It proceeds as follows: First, each of the cops chooses a starting vertex. Then, so does the robber. After this initialization round, the players take turns alternately, starting with the cops. During cops' turn, each cop moves to a vertex at distance at most 1. During robber's turn, the robber moves to a vertex at distance at most 1. At all stages of the game, multiple cops are allowed to stand on the same vertex. The cops win if at any point of the game there is a cop on the same vertex as the robber. Otherwise, that is, if the robber evades the cops forever, the robber wins.

We say that a graph G is *k-copwin* if there is a winning strategy for k cops. The *cop number of a graph G*, denoted $c(G)$, is the minimal k such that G is k-copwin. Note that the cop number is well defined, as having a cop on each vertex of the graph ensures cops' victory, hence $c(G) \leq n$.

Finding the cop number of a given graph is the central objective in the study of the game. As the cop number of a graph is the sum of cop numbers of its connected components, the attention can be restricted to connected graphs. Some connected graphs, such as Moore graphs and the incidence graph of a projective plane, have been shown to have a cop number on the order of $\sqrt{n}$, as follows from Theorem 3 in Aigner and Fromme [1]. Meyniel [5] conjectured that the maximum cop number among all connected graphs on n vertices $c(n)$ is $O(\sqrt{n})$. This conjecture remains unresolved to this day. It was proven independently by Scott and Sudakov [10] and by Lu and Peng [7] that $c(n) = O(n2^{-(1+o(1))\sqrt{\log_2(n)}})$.

The problem of determining the cop number of a given graph is EXPTIME-complete, as shown by Kinnersley [6]. Brandt, Pettie and Uitto showed in [2] that, conditional on the Strong Exponential Time Hypothesis, deciding whether a graph G on n vertices with $O(n \log^2 n)$ edges is k-copwin requires time $\Omega(n^{k-o(1)})$ in the worst-case. Clarke and MacGillivray [3] described an algorithm in time $O(n^{2k+2})$ that decides whether a graph G is k-copwin. In this paper, we present an algorithm in time $O(kn^{k+2})$ for deciding whether a graph is k-copwin.

* Corresponding author.
E-mail addresses: jp895@cam.ac.uk (J. Petr), jp899@cam.ac.uk (J. Portier), lvv23@cam.ac.uk (L. Versteegen).

https://doi.org/10.1016/j.dam.2022.05.019

2. The algorithm

In this section we present and analyse our new algorithm for determining whether k cops have a winning strategy on a graph G.

Let $G = (V(G), E(G))$ be a graph and k a natural number. We can assume $V(G) = [n] = \{1, \dots, n\}$. For ease of presentation, we order the cops arbitrarily and think of them as moving one after another in this order rather than simultaneously.

We define the state space

$$S = \{(p_0, p_1, \dots, p_k, t) \in [n]^{k+1} \times \mathbb{Z}_{k+1}\}.$$

The coordinates of these state space vectors are to be interpreted as follows. The last coordinate t tells us which piece has been moved last, meaning that the next piece to move is the robber if $t = k$ and the $(t+1)$-st cop if $t \neq k$. The other coordinates describe the positions of the pieces in the graph, where p_0 is the position of the robber and p_i for $i \in [k]$ is the position of the ith cop. We say that a state s is *cop-winning* if the cops can ensure to catch the robber from s.

We now define an oriented graph H with $V(H) = S$. A pair $(q, s) = ((p_0, \dots, p_k, t), (p'_0, \dots, p'_k, t'))$ of states belongs to the edge set $E(H)$ if and only if:

(1) $t' = (t+1) \mod (k+1)$, and
(2) $(p_t, p'_t) \in E(G)$ or $p_t = p'_t$, and
(3) $p_i = p'_i$ for all $i \neq t$.

An edge $(q, s) \in E(H)$ thus represents that it is possible to move from state q to state s.

For a state s, we denote by $I(s)$ its in-neighbours $\{q \in S : (q, s) \in E(H)\}$. We will also use the following data structures:

- COPSWIN: a boolean array indexed by S, initialized at all 0.
- COUNTER: an array of non-negative integers indexed by those states $q = (p_0, \dots, p_k, t)$ with $t = 0$. Initialized with out-degrees of the respective states, that is, with $1 + \deg_G(p_0)$.
- QUEUE: a queue of states, empty at the beginning.

COPSWIN is intended to mark states from which cops can win, COUNTER counts down the number of out-neighbours of a state that have not yet been marked as cop-winning.

Using these data structures we run an inductive algorithm that determines which states are cop-winning. The algorithm is listed in detail in Algorithm 1, but we shall briefly outline its functioning in natural language at this point.

Firstly, we mark all states for which a cop is placed on the same vertex as the robber as cop-winning. Here and throughout every state that is newly marked as cop-winning — and only these states — are enqueued in the queue. In particular, every state is enqueued at most once.

Secondly, we run through the enqueued states in FIFO-order and consider for each state $s = (p_0, \dots, p_k, t)$ its possible predecessors, i.e., its in-neighbours in H. If $t \neq 0$, then every s is reached by each of its predecessors through a cop move and therefore we may mark these states as cop-winning and enqueue them unless we have already done so before.

If $t = 0$, we cannot immediately mark the predecessors of s as cop-winning, since we must not expect the robber to run freely into their capture. We can however say for each predecessor of s, that the move leading to s is not a viable option for the robber. Thus we decrease the COUNTER at all predecessors s' of s by 1. Once COUNTER at s' arrives at zero, we know that in the state s' the robber has no moves that do not lead to capture, and hence we can mark and enqueue s'.

We begin by observing that each state is enqueued at most once, which ensures that the algorithm terminates.

Lemma 2.1. *Every $s \in S$ is enqueued at most once.*

Proof. Only states s with COPSWIN$(s) = 0$ get enqueued. Whenever a state s is enqueued, COPSWIN(s) becomes 1. □

We verify that cop-winning states are exactly those whose COPSWIN value is 1:

Lemma 2.2. *Let s be a state in S. Once Algorithm 1 finishes,* COPSWIN$(s) = 1$ *if and only if s is cop-winning.*

Proof. We first show that if COPSWIN$(s) = 1$ then s is cop-winning. To this end, assume for contradiction that $q = (p_0, \dots, p_k, t)$ is the first state that was assigned COPSWIN$(q) = 1$ during the run of Algorithm 1 without being cop-winning. All states s assigned COPSWIN$(s) = 1$ inside the first for-loop are trivially cop-winning, so q had to have been assigned COPSWIN$(q) = 1$ inside the while-loop. Let $s = (p'_0, \dots, p'_k, t')$ be the state in the while-cycle of which q got enqueued. Note that this means $q \in I(s)$, and therefore $t' = (t+1) \mod (k+1)$. Also, $t = k$, as else it would be possible to move the $(t+1)$-st cop so as to move from q to s, contradicting that q is not cop-winning. The state q got enqueued when COUNTER(q) reached zero, that is, after all its out-neighbours got enqueued. In other words, the states

Algorithm 1 Determining if a graph is k-copwin

Input: $G = ([n], E(G))$, $k \in \mathbb{N}$
Output: **true** if G is k-copwin, **false** if G is not k-copwin

```
if k ≥ n then
    return true
    halt
end if

Initiate COPSWIN, COUNTER, QUEUE

for s = (p_0, p_1, ..., p_k, t) ∈ S do
    for i ∈ [k] do
        if p_0 = p_i then
            enqueue s in QUEUE
            COPSWIN(s) := 1
            break
        end if
    end for
end for

while QUEUE not empty do
    dequeue s = (p_0, p_1, ..., p_k, t) from QUEUE
    if t ≠ 0 then
        for q ∈ I(s) do
            if COPSWIN(q) = 0 then
                enqueue q in QUEUE
                COPSWIN(q) := 1
            end if
        end for
    else
        for q ∈ I(s) do
            COUNTER(q) := COUNTER(q) − 1
            if COUNTER(q) = 0 then
                if COPSWIN(q) = 0 then
                    enqueue q in QUEUE
                    COPSWIN(q) := 1
                end if
            end if
        end for
    end if
end while

if there exists (p_1, ..., p_k) ∈ [n]^k s.t. for all p_0 ∈ [n] : COPSWIN((p_0, p_1, ..., p_k, 0)) = 1 then
    return true
else
    return false
end if
```

$(w_0, w_1, \ldots, w_k, 0)$ where $w_0 \in N(p_0) \cup \{p_0\}$ and $w_i = p_i$ for all $i \in [k]$ are all cop-winning. That means that wherever the robber moves from p_0, the resulting state is cop-winning, a contradiction.

It remains to prove that for every cop-winning state s, COPSWIN$(s) = 1$ once Algorithm 1 finishes. Assuming this were not the case, let q be a cop-winning state chosen from cop-winning states with COPSWIN$(q) = 0$ such that the number of moves m in which cops can catch the robber no matter how the robber moves is minimal. We have $m > 0$. If $t \neq k$, it is possible to move the $(t+1)$-st cop in such a way that cops can win in $m-1$ moves from the resulting state s. By the choice of q, s, COPSWIN$(s) = 1$. But then q would have been enqueued in the while-cycle of s, a contradiction.

Finally, assume $t = k$. Every robber's move from p_0 to $N(p_0) \cup \{p_0\}$ is losing in fewer than m moves. That means that all the states w in $\{(w_0, w_1, \ldots, w_k, 0) : w_0 \in N(p_0) \cup \{p_0\}\} = \{w \in S : (q, w) \in E(H)\}$ have been enqueued at some point, each of them decreasing COUNTER(q) by 1. After the last decrement, COPSWIN(q) becomes 1, a contradiction. □

Denoting the average degree of a graph G by $\bar{d}(G)$, we can express the time complexity of Algorithm 1 as follows.

Theorem 2.3. *Algorithm 1 determines whether a graph G is k-copwin and has time complexity $O(kn^{k+1}(\bar{d}(G)+k))$.*

Proof. The correctness of the algorithm has been verified by the previous lemma.

Now we move onto the time complexity. The initiation can be done in time $O(|S|) = O(kn^{k+1})$. The first for-cycle ends up doing $O(k|S|) = O(k^2n^{k+1})$ operations.

As by Lemma 2.1 each state $s = (p_0, p_1, \dots, p_k, t) \in S$ is enqueued at most once, the total number of operations inside the while-loop is $O(\sum_{s\in S} |I(s)|) = O(\sum_{s\in S}(\deg_G(p_0)+1)) = O(kn^k \cdot \sum_{v\in V(G)}(\deg_G(v)+1)) = O(kn^{k+1}(\bar{d}(G)+1))$.

Determining whether there exists $(p_1, \dots, p_k) \in [n]^k$ such that $\mathrm{COPSWIN}((p_0, p_1, \dots, p_k, 0)) = 1$ for all $p_0 \in [n]$ can be done in time $O(n^{k+1})$.

The time complexity of Algorithm 1 is therefore $O(kn^{k+1}) + O(k^2n^{k+1}) + O(kn^{k+1}(\bar{d}(G)+1)) + O(n^{k+1}) = O(kn^{k+1}(\bar{d}(G)+k))$. □

As $\bar{d}(G) < V(G)$, we have:

Corollary 2.4. *Algorithm 1 has time complexity $O(kn^{k+2})$.*

We conclude this article by mentioning two possible adaptations of the algorithm. Firstly, the algorithm could be easily changed to determine for each state not only if it is cop-winning, but also what is the minimum number of moves required to ensure catching the robber. Its time complexity would remain $O(kn^{k+1}(\bar{d}(G)+k))$.

Secondly, the algorithm can be adapted to variants on Cops and Robbers such as the deterministic version of Zombies and Survivors, introduced by Fitzpatrick, Howell, Messinger and Pike [4] — the zombies take the place of cops with the difference that in each move they move along a geodesic between them and the survivor (taking place of the robber). The only adjustment to the algorithm is a different definition of H, and, consequently, $I(s)$ and out-degrees of states. Computing all possible neighbours that lie on a geodesic between two vertices (here, the position of a cop and of the robber) can be done in time $O(n^3)$ using Floyd–Warshall algorithm for instance. The time complexity of the algorithm would thus be $O(kn^{k+1}(\bar{d}(G)+k) + n^3)$.

Acknowledgements

The authors would like to thank Professor Béla Bollobás for his valuable comments and the reviewers for their helpful suggestions.

The first two authors are supported by EPSRC (Engineering and Physical Sciences Research Council). The first author is also supported by the Department of Pure Mathematics and Mathematical Statistics of the University of Cambridge, and the second author is also supported by the Cambridge Commonwealth, European and International Trust. The third author is supported by the Trinity External Research Studentship.

References

[1] M. Aigner, M. Fromme, A game of cops and robbers, Discrete Appl. Math. (ISSN: 0166-218X) 8 (1) (1984) 1–12, http://dx.doi.org/10.1016/0166-218X(84)90073-8, URL https://www.sciencedirect.com/science/article/pii/0166218X84900738.

[2] S. Brandt, S. Pettie, J. Uitto, Fine-grained lower bounds on cops and robbers, in: Y. Azar, H. Bast, G. Herman (Eds.), 26th Annual European Symposium on Algorithms, ESA 2018, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 112, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, ISBN: 978-3-95977-081-1, 2018, pp. 9:1–9:12, http://dx.doi.org/10.4230/LIPIcs.ESA.2018.9, URL http://drops.dagstuhl.de/opus/volltexte/2018/9472.

[3] N.E. Clarke, G. MacGillivray, Characterizations of k-copwin graphs, Discrete Math. (ISSN: 0012-365X) 312 (8) (2012) 1421–1425, http://dx.doi.org/10.1016/j.disc.2012.01.002, URL https://www.sciencedirect.com/science/article/pii/S0012365X12000064.

[4] S. Fitzpatrick, J. Howell, M. Messinger, D. Pike, A deterministic version of the game of zombies and survivors on graphs, Discrete Appl. Math. (ISSN: 0166-218X) 213 (2016) 1–12, http://dx.doi.org/10.1016/j.dam.2016.06.019, URL https://www.sciencedirect.com/science/article/pii/S0166218X16302980.

[5] P. Frankl, Cops and robbers in graphs with large girth and Cayley graphs, Discrete Appl. Math. (ISSN: 0166-218X) 17 (3) (1987) 301–305, http://dx.doi.org/10.1016/0166-218X(87)90033-3, URL https://www.sciencedirect.com/science/article/pii/0166218X87900333.

[6] W.B. Kinnersley, Cops and robbers is EXPTIME-complete, J. Combin. Theory Ser. B (ISSN: 0095-8956) 111 (2015) 201–220, http://dx.doi.org/10.1016/j.jctb.2014.11.002, URL https://www.sciencedirect.com/science/article/pii/S0095895614001282.

[7] L. Lu, X. Peng, On Meyniel's conjecture of the cop number, J. Graph Theory 71 (2) (2012) 192–205, http://dx.doi.org/10.1002/jgt.20642, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/jgt.20642.

[8] R.J. Nowakowski, P. Winkler, Vertex-to-vertex pursuit in a graph, Discret. Math. 43 (2–3) (1983) 235–239, http://dx.doi.org/10.1016/0012-365X(83)90160-7.

[9] A. Quillot, Jeux et pointes fixes sur les graphes, (Ph.D. Dissertation), UniversitÉ de Paris VI, 1978.

[10] A. Scott, B. Sudakov, A bound for the cops and robbers problem, SIAM J. Discrete Math. 25 (2010) http://dx.doi.org/10.1137/100812963.