LETTER Special Section on Foundations of Computer Science

Generalized Chat Noir is PSPACE-Complete*

Chuzo IWAMOTO[†a)], ***Member***, **Yuta MUKAI**[††], **Yuichi SUMIDA**[†††], ***Nonmembers***, ***and*** **Kenichi MORITA**[†], ***Member***

SUMMARY We study the computational complexity of the following two-player game. The instance is a graph $G = (V, E)$, an initial vertex $s \in V$, and a target set $T \subseteq V$. A "cat" is initially placed on s. Player 1 chooses a vertex in the graph and removes it and its incident edges from the graph. Player 2 moves the cat from the current vertex to one of the adjacent vertices. Players 1 and 2 alternate removing a vertex and moving the cat, respectively. The game continues until either the cat reaches a vertex of T or the cat cannot be moved. Player 1 wins if and only if the cat cannot be moved before it reaches a vertex of T. It is shown that deciding whether player 1 has a forced win on the game on G is PSPACE-complete.

key words: *PSPACE-complete, computational complexity, two-player game, Chat Noir*

1. Introduction

Chat Noir is a cat capture game on a particular graph having a regular board-like structure (see Fig. 1). The graph is composed of $n \times n$ vertices, and all vertices except boundary vertices have degree six. There are $4(n-1)$ boundary vertices in an $(n \times n)$-graph (You can play a (11×11)-Chat Noir at the web site [1]. The French word "Chat Noir" means "Black Cat").

Initially, the cat is on a non-boundary vertex, and several vertices are colored grey. Grey vertices are regarded as removed vertices, and the cat cannot be moved to them. Alternately, (i) player 1 chooses a vertex in the graph and removes it and its incident edges from the graph, and (ii) player 2 moves the cat from the current vertex to one of the adjacent vertices. Here, player 1 must not remove the vertex on which the cat is currently placed. The game continues until either the cat reaches a boundary vertex or the cat cannot be moved. Player 1 wins if and only if the cat cannot be moved before it reaches a boundary vertex.

In this paper, we study the computational complexity of the generalized version of Chat Noir.

Manuscript received March 15, 2012.
Manuscript revised June 8, 2012.

[†]The authors are with the Graduate School of Engineering, Hiroshima University, Higashihiroshima-shi, 739–8527 Japan.
[††]The author is with Fujitsu Corporation, Kawasaki-shi, 211–8588 Japan.
[†††]The author is with West Japan Railway Company, Osaka-shi, 530–8341 Japan.

*This research was supported in part by Scientific Research Grant, Ministry of Japan.

a) E-mail: chuzo@hiroshima-u.ac.jp

DOI: 10.1587/transinf.E96.D.502

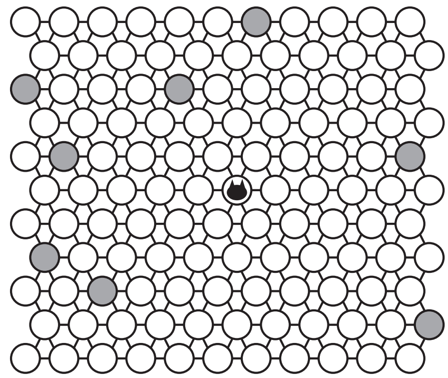

Fig. 1 An initial configuration on an (11×11)-vertex graph.

GENERALIZED CHAT NOIR
INSTANCE: An undirected graph $G = (V, E)$, an initial vertex $s \in V$, and a target set $T \subseteq V$.
QUESTION: Does player 1 have a forced win in the following game played on G? A cat is initially placed on s. Player 1 chooses a vertex and removes it and its incident edges from the graph. Here, the vertex on which the cat is currently placed must not be chosen or removed. Player 2 moves the cat from the current vertex to one of the adjacent vertices. Players 1 and 2 alternate removing a vertex and moving the cat, respectively. The game continues until either the cat reaches a vertex of T or the cat cannot be moved. Player 1 wins if and only if the cat cannot be moved before it reaches a vertex of T.

From the definition, one can see that the player 1 wins if he leaves the cat on a connected component which does not contain any vertex of T. The problem is in PSPACE because a game can last at most $|V| - 1$ removals of vertices. We will prove that the problem is PSPACE-complete.

A lot of two-player games have been shown to be PSPACE-complete. In Garey and Johnson's survey book [6], the following PSPACE-complete games are listed: Generalized HEX [3]; Generalized Geography and Kayles, Variable Partition Truth Assignment, Sift, Alternating Hitting Set [8]; and Sequential Truth Assignment [9].

As for games on the $(n \times n)$-extension of a square grid, Othello [7], Rush Hour [4], and Amazons [5] are known to be PSPACE-hard.

HEX is a game on an $(n \times n)$-hexagonal grid (see [2]). Even and Tarjan proved the PSPACE-completeness of the generalized version of HEX [3]. Their generalized HEX is played on an arbitrary graph (and not a regular board-like hexagonal grid). The instance is a graph $G = (V, E)$ and two specified vertices $s, t \in V$. Players 1 and 2 alternate choosing a vertex from $V - \{s, t\}$, with those chosen by player 1 being colored blue and those chosen by player 2 being colored red. The game continues until all such vertices have been colored, and player 1 wins if and only if there is a path from s to t in G that passes through only blue vertices. (This description of the HEX rule is from [6]). The question is to decide whether player 1 has a forced win on the game on G.

In this paper, we also define the generalized Chat Noir as a game on an arbitrary graph. The differences between HEX and Chat Noir are as follows: (i) Player 2 in HEX can choose a vertex among *arbitrary* non-colored vertices. On the other hand, player 2 in Chat Noir can only move the cat from the current vertex to one of the *adjacent* non-colored vertices (here, removed vertices in Chat Noir are called colored vertices in this sentence). (ii) Vertices chosen by player 2 in HEX are colored by red, while vertices chosen by player 2 in Chat Noir are not colored.

2. Reduction from Quantified 3SAT to Chat Noir

2.1 Transformation from a Quantified Boolean Formula to a Graph

The following definition of QUANTIFIED 3SAT is mostly from [LO11] in [6]. This is a well-known PSPACE-complete problem.

QUANTIFIED 3SAT
INSTANCE: Set $U = \{x_1, x_2, \ldots, x_n\}$ of variables, quantified Boolean formula $F = (Q_1x_1)(Q_2x_2)\cdots(Q_ix_i)\cdots(Q_nx_n)E$, where $E = c_1 \wedge c_2 \wedge \cdots \wedge c_j \wedge \cdots \wedge c_m$ is a Boolean expression in conjunctive normal form with three literals per clause c_j, and each Q_i is either $\forall$ or $\exists$.
QUESTION: Is F true?

Without loss of generality, we can assume that Q_1 is $\forall$ and the quantifiers are alternately $\forall$ and $\exists$. For example, let

$$c_1 = (\overline{x_1} \vee x_2 \vee x_3), \quad c_2 = (x_1 \vee x_2 \vee x_4),$$
$$c_3 = (\overline{x_1} \vee x_3 \vee x_4), \quad c_4 = (\overline{x_2} \vee x_3 \vee \overline{x_4}).$$

It is easy to verify that $F_1 = \forall x_1 \exists x_2 \forall x_3 \exists x_4 (c_1 \wedge c_2 \wedge c_3)$ is true. However, $F_2 = \forall x_1 \exists x_2 \forall x_3 \exists x_4 (c_1 \wedge c_2 \wedge c_3 \wedge c_4)$ is false, since there are no assignment values for x_2 and x_4 that simultaneously satisfy c_1, c_2, c_3, and c_4 when $x_1 = 1$ and $x_3 = 0$.

We present a transformation from an arbitrary quantified Boolean formula F in conjunctive normal form with three literals per clause to a graph $G = (V, E)$, an initial vertex $s \in V$, and a target set $T \subseteq V$, such that F is true if and only if player 1 has a forced win on the game on G.

Let n and m be the numbers of variables and clauses of F, respectively. Without loss of generality, we assume that n is an even number. The graph G has $9n/2 + 3m + 3$ vertices given as follows (see Fig. 2):

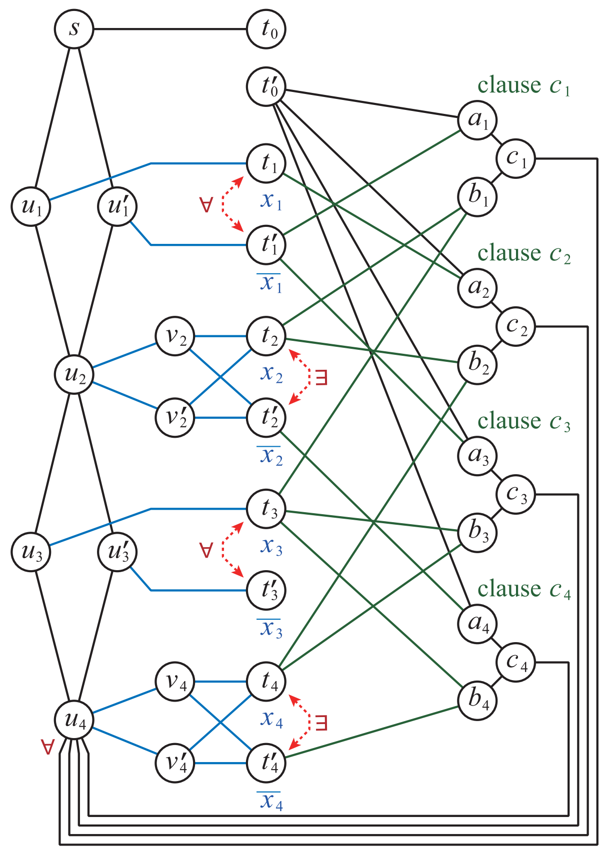

Fig. 2 The graph G_2 transformed from $F_2 = \forall x_1 \exists x_2 \forall x_3 \exists x_4 (c_1 \wedge c_2 \wedge c_3 \wedge c_4)$, where $c_1 = (\overline{x_1} \vee x_2 \vee x_3)$, $c_2 = (x_1 \vee x_2 \vee x_4)$, $c_3 = (\overline{x_1} \vee x_3 \vee x_4)$, and $c_4 = (\overline{x_2} \vee x_3 \vee \overline{x_4})$.

$$\begin{aligned} V = \{\ & s, t_0, t'_0, \\ & t_1, t'_1,\ t_2, t'_2,\ \ldots,\ t_n, t'_n, \\ & u_1, u'_1,\ u_2,\ u_3, u'_3,\ u_4,\ \ldots,\ u_{n-1}, u'_{n-1},\ u_n, \\ & v_2, v'_2,\ v_4, v'_4,\ \ldots,\ v_n, v'_n, \\ & a_1, b_1, c_1,\ a_2, b_2, c_2,\ \ldots,\ a_m, b_m, c_m.\ \} \end{aligned}$$

Here, s is the initial vertex, and $T = \{t_0, t'_0, t_1, t'_1, \ldots, t_n, t'_n\}$ is the target set of G.

For each $i \in \{1, 2, \ldots, n\}$, vertices t_i and t'_i are labeled with x_i and $\overline{x_i}$, respectively (see Fig. 2). Later, one can see that vertex t_i (resp. t'_i) is removed by player 1 if $x_i = 1$ (resp. $\overline{x_i} = 1$).

The connections among $s, u_1, u'_1, u_2, u_3, u'_3, u_4, \ldots, u_n$ are as follows. For every $l \in \{1, 2, \ldots, n/2\}$, vertex u_{2l-2} is connected to u_{2l-1} and u'_{2l-1} by two edges, and vertices u_{2l-1} and u'_{2l-1} are connected to u_{2l} by two edges, where the initial vertex s is regarded as u_0.

For every $l \in \{1, 2, \ldots, n/2\}$, vertices t_{2l-1} and t'_{2l-1} are connected to u_{2l-1} and u'_{2l-1}, respectively. Also, vertices t_{2l} and t'_{2l} are connected to v_{2l} and v'_{2l} by 2×2 edges. Furthermore, v_{2l} and v'_{2l} are connected to u_{2l} by two edges. Vertex s is connected to t_0.

For every $j \in \{1, 2, \ldots, m\}$, vertex c_j is connected to a_j and b_j. Vertices a_j and b_j are connected to four vertices in T such that two of the four are connected to a_j, and the remain-

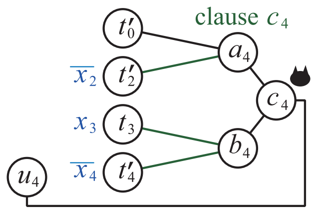

Fig. 3 Clause $c_4 = (\overline{x_2} \vee x_3 \vee \overline{x_4})$ is false when $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$.

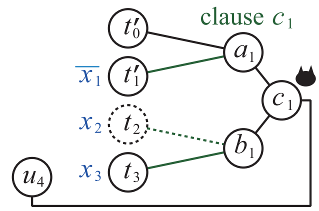

Fig. 4 Clause $c_1 = (\overline{x_1} \vee x_2 \vee x_3)$ is true when $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$. Vertex t_2 has been removed, which implies $x_2 = 1$.

ing two are connected to b_j. (Connection between T and $\{a_j, b_j\}$ is given in the next paragraph.) Vertices c_j, a_j, b_j, and those four vertices in T compose a seven-vertex binary tree, which corresponds to a clause (see Fig. 3).

Connection between T and $\{a_1, b_1, a_2, b_2, \ldots, a_m, b_m\}$ is constructed as follows. For example, suppose that $c_1 = (\overline{x_1} \vee x_2 \vee x_3)$ (see Figs. 2 and 4). Then, a_1 is connected to t'_1 (labeled with $\overline{x_1}$), and b_1 is connected to t_2 and t_3 (labeled with x_2 and x_3, respectively). Furthermore, a_1 is also connected to vertex t'_0 in some technical reason. In the same manner, for every clause c_j, we add two edges between vertex c_j and $\{a_j, b_j\}$, and add four edges between $\{a_j, b_j\}$ and T.

Finally, vertex u_n (see u_4 of Fig. 2) is connected to $c_1, c_2, \ldots, c_m$ by m edges. This completes the construction of the graph $G = (V, E)$, vertex $s \in V$, and set $T \subseteq V$.

2.2 Char Noir on the Constructed Graph

In the following, we will show that F is true if and only if player 1 has a forced win on the game on G.

Initially, the cat is placed on the initial vertex s, and the first move is player 1. (Recall that players 1 and 2 are a vertex remover and a cat mover, respectively.)

The first move of player 1 is to remove the vertex $t_0 \in T$. (If player 1 does not remove t_0, then player 2 moves the cat to $t_0 \in T$, and player 2 wins immediately.) The first move of player 1 is forced.

The first move of player 2 is to move the cat to one of the two vertices u_1 and u'_1. If player 2 moves the cat to u_1 (resp. u'_1), then the player 1's next move is to remove vertex $t_1 \in T$ (resp. $t'_1 \in T$) by the same reason as the previous paragraph. The second move of player 1 is also forced. (Removing t_i (resp. t'_i) corresponds to the assignment $x_i = 1$ (resp. $\overline{x_i} = 1$). See x_2 of Fig. 4 later.)

For the second move of player 2, there seem to be two choices: (i) He moves the cat back to s, or (ii) he moves the cat to u_2. If player 2 moves the cat to s from u_1 (resp. from u'_1), then player 1 wins by removing u'_1 and u_2 (resp. u_1 and u_2) in that order. Hence, player 2 is forced to choose (ii) as his second move.

When the cat is on the vertex u_2, player 1 is forced to remove one of vertices t_2 and t'_2. This is because if player 1 remove a vertex, say, v_2 (which is not t_2 or t'_2), then player 2 moves the cat to v'_2, and player 2 wins in his next move (since v'_2 is connected to $t_2, t'_2 \in T$).

For the third move of player 2, there seem to be six choices: $u_1, u'_1, v_2, v'_2, u_3$, or u'_3. For a reason similar to the second move of player 2, the cat is forced to be moved to one of u_3 and u'_3.

By continuing this observation, one can see that player 2 will move the cat to u_n (see u_4 in Fig. 2), and player 1 removes one of t_n and t'_n. Again, in the same reason as the previous paragraph, player 2 is forced to move the cat to one of the m vertices $c_1, c_2, \ldots, c_m$.

In the following, for simplicity of exposition, we suppose that player 2 has moved the cat on the path s, u_1, u_2, u'_3, u_4, and player 1 has removed vertices t_1, t_2, t'_3, t_4 (where t_1 and t'_3 were forcedly removed), which correspond to assignment $(x_1, x_2, x_3, x_4) = (1, 1, 0, 1)$.

Suppose that player 2 moves the cat from u_4 to c_4 (see Fig. 3). In this case, player 2 reaches one of the four vertices t'_0, t'_2, t_3, and t'_4 in four steps, and he wins. (The trivial verification is left to the reader.)

Consider a different case from the previous paragraph. Suppose that player 2 moves the cat from u_4 to c_1 (see Fig. 4). In this case, one of the four vertices t'_0, t'_1, t_2, and t_3 has been removed (see t_2 in the figure). In general, if at least one of the four leaves is removed in the seven-vertex binary tree, then player 2 cannot move the cat from the root to any of the remaining leaves (i.e., t'_0, t'_1, and t_3). Namely, (i) player 1 removes a_1, (ii) player 2 moves the cat to b_1 or u_4, and then (iii) player 1 removes t_3.

Assume that player 1 has a winning strategy. Recall that, in the first $2n + 1$ moves, player 1 removed $n + 1$ vertices from T, and player 2 moved the cat from s to u_n. Note that, when the cat was on vertex u_{2l}, player 1 had two choices (removal of t_{2l} or t'_{2l}) for each $l \in \{1, 2, \ldots, n/2\}$. This corresponds that variable x_{2l} is quantified by $\exists$ for each $l \in \{1, 2, \ldots, n/2\}$.

At the $(2n+2)$nd move, player 2 moves the cat from u_n to one of the m vertices $c_1, c_2, \ldots, c_m$. The assumption that player 1 has a winning strategy implies that at least one of the four leaves of the seven-vertex binary tree rooted at c_j has been removed for all $j \in \{1, 2, \ldots, m\}$ (see Fig. 4). If t_i (resp. t'_i) is a removed vertex of a seven-vertex binary tree rooted at c_j, then clause c_j is satisfied by literal x_i (resp. $\overline{x_i}$). Hence, if player 1 has a winning strategy, then F is true.

Assume that player 1 has no winning strategy. This implies that, at the $(2n + 2)$nd move, there is a seven-vertex binary tree such that none of its four leaves have been removed (see Fig. 3). Let c_j be the clause which corresponds to such a binary tree. The clause c_j is satisfied by none of its three literals (see $\overline{x_2}, x_3, \overline{x_4}$ of Fig. 3). Hence, if player 1 has no winning strategy, then F is false.

3. Conclusion

In this paper, we proved that the generalized Chat Noir played on an arbitrary graph is PSPACE-complete. The complexity of Chat Noir played on the $(n \times n)$-extension of a regular hexagonal grid is an interesting open problem.

References

[1] http://www.gamedesign.jp/flash/chatnoir/chatnoir.html
[2] http://en.wikipedia.org/wiki/Hex_(board_game)
[3] S. Even and R.E. Tarjan, "A combinatorial problem which is complete in polynomial space," J. Assoc. Comput. Mach., vol.24, no.4, pp.710–719, 1976.
[4] G.W. Flake and E.B. Baum, "Rush hour is PSPACE-complete, or why you should generously tip parking lot attendants," Theor. Comput. Sci., vol.270, no.1/2, pp.895–911, 2002.
[5] T. Furtak, M. Kiyomi, T. Uno, and M. Buro, "Generalized Amazons is PSPACE-complete," Proc. 19th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, pp.132–137, 2005.
[6] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, NY, USA, 1979.
[7] S. Iwata and T. Kasai, "The Othello game on an $n \times n$ board is PSPACE-complete," Theor. Comput. Sci., vol.123, no.2, pp.329–340, 1994.
[8] T.J. Schaefer, "On the complexity of some two-person perfect-information games," J. Comput. Syst. Sci., vol.16, pp.185–225, 1978.
[9] L.J. Stockmeyer and A.T. Meyer, "Word problems requiring exponential time," Proc. 5th Ann. ACM Symp. on Theory of Computing, pp.1–10, 1973.