Optimum Solution Search for FreeCell by Applying the Strongly Connected Component Decomposition Algorithm for Directed Graphs

Shuji Jimbo
Guest Researcher, Graduate School of Environmental, Life, Natural Science and Technology, Okayama University

E-mail address: jimbo-s@okayama-u.ac.jp

Abstract

The author is developing a computer-aided proof to demonstrate that there exists no Eulerian circuit of the 15-vertex complete graph that satisfies specified conditions. Depth-first search serves as the underlying principle for this proof. However, due to the limitations of the author's computational resources, a brute-force approach is deemed impractical. The primary goal of this research is to explore the effectiveness of depth-first search (DFS) algorithms in the context of the single-player card game FreeCell. The ultimate objective is to develop a program that can determine any shortest solution sequence to any given FreeCell problem. The research will focus on developing fundamental implementation techniques for DFS, along with advanced search strategies informed by domain knowledge of FreeCell mechanics and heuristics. In particular, application of the number of strongly connected components in a directed graph constructed from the card arrangement in the FreeCell tableau presents a promising approach to reducing the search space. Currently in the initial phases of development, this report primarily delves into the program's development strategy.

Keywords. computer-based proof, FreeCell, depth-first search, strongly connected component.

1 Introduction

The author is developing a computer-based proof that no Eulerian circuit of a complete graph with 15 vertices can be constructed that satisfies the following condition[3][4].

> Condition: There are always at least 11 occurrences of vertices between two occurrences of the same vertex on the Eulerian circuit.

The fundamental principle underlying the proof is depth-first search, which is augmented by the collision avoidance technique with hash tables. Nevertheless, a brute-force search is unlikely to be an effective solution in the computing environment available to the author.

It is desirable to reduce the search space as much as possible by utilizing knowledge in the domain related to Eulerian circuits of complete graphs, which is the object of the search.

The objective of this research is to enhance depth-first search techniques by developing a program that finds any shortest solution to a single-player card game called FreeCell. Although the target problems are relatively small-scale and can be handled by today's common PCs, there exist instances of the FreeCell game that would take a simple program a very long time to solve. This research will endeavor to develop fundamental implementation techniques for depth-first search, as well as search methods based on relevant domain knowledge about the target of the search. In particular, the application of the number of strongly connected components of a directed graph derived from the ordering of cards in the tableau of FreeCell is expected to be an effective means of reducing the search space.

The author is developing a FreeCell solver. The solver is intended to be used to find an optimum solution or to prove that a given problem is unsolvable. The current stage of development is preliminary stage, and this report primarily outlines the program's development policy.

2 Preliminaries

2.1 The Rule of FreeCell

FreeCell is a solitaire card game that has been played on Windows OS for a long time. It is a one-person strategy game with no element of luck. The rules of FreeCell are described below. Initially, 52 cards are dealt into eight piles (the initial position). Each of the first (left) four piles has seven cards, and each of the last (right) four piles has six cards. The set of eight piles is called the tableau. There are four free cells. There is a place called the foundation. The names of parts of the FreeCell board are described in Fig. 1.

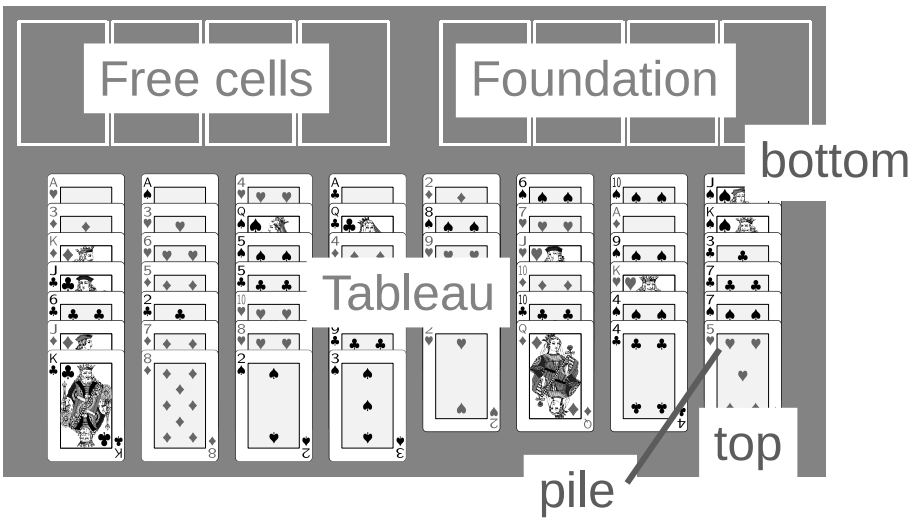

Figure 1: The names of parts of the FreeCell board

Only one card can be placed in each free cell. The four frames in the upper right corner of Fig. 1 represent the foundation. If a card x is placed in the four frames, x is the highest ranked card of the same suit as x in the foundation, and all cards of the same or

lower rank as x are in the foundation. In the initial position of FreeCell, the foundation is always empty.

The goal in the game is to move all cards into the foundation. Only one card in a free cell or on top of a tableau pile can be moved per turn. There are three kinds of legal move destinations for a card x as follows.

(a) The foundation, when the rank of x is A or there is the card y in the foundation such that the suit of y is the same as that of x and the rank of y is one less than that of x.

(b) An empty free cell or an empty tableau pile.

(c) On a top card y in a tableau pile, when the color of y is different from that of x and the rank of y is one greater than that of x.

An illustration of how the cards are moved is shown in Fig. 2. When there are no more legal moves in a turn, the game is lost.

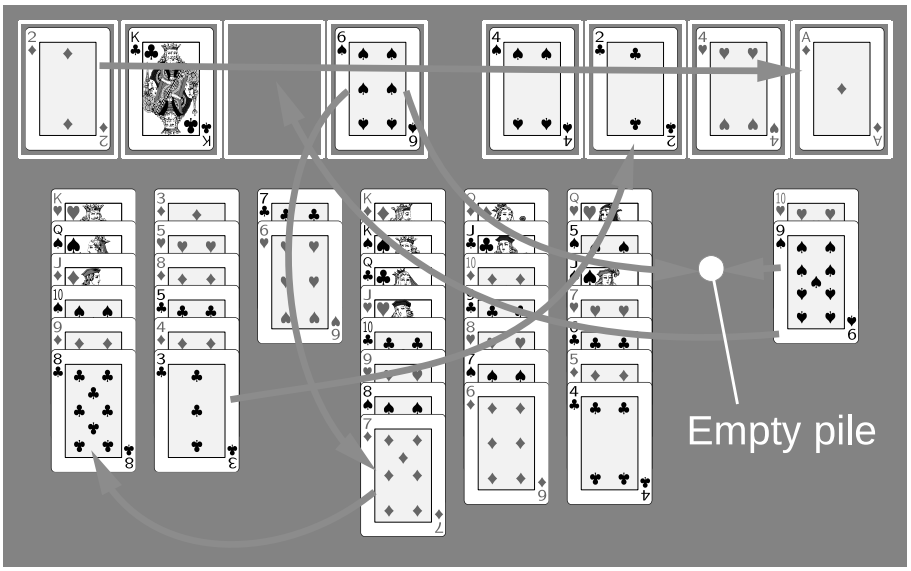

Figure 2: How the cards are moved

2.2 Examples of FreeCell Game

If a FreeCell problem (initial position) is picked according to a uniform distribution, it has a solution with very high probability. However, there is a possibility of an unsolvable problem in several tens of thousands of randomly selected ones. For example, Microsoft FreeCell No. 11982 is a well-known FreeCell problem with no solution (see Fig. 3). Hereafter, problem Microsoft FreeCell No. n will be referred to as MS n for any positive integer n less than 10,000,000. A test solver can prove the unsolvability of MS 11982. It showed that there is no legal move sequence longer than 1,416 moves when the limit of the length of legal move sequence is 2,000.

Problem MS 1941 is notorious for being extremely difficult for humans. However, a test solver solves it in a few seconds. The length of shortest solutions to MS 1941 is 88.

On the other hand, problem MS 5 is notorious for being extremely difficult for computers that try to find an optimum solution. Since a test solver shows that MS 5 has no solution of length 89 or less and it finds a solution of length 90, the length of shortest solutions to MS 5 is 90. However, it took about 50 minutes of user time for the search above. Notice that if we let the test solver solve MS 5 without requiring a high-quality solution, it finds a solution in several seconds. FreeCell problem MS 5 is shown in Fig. 4.

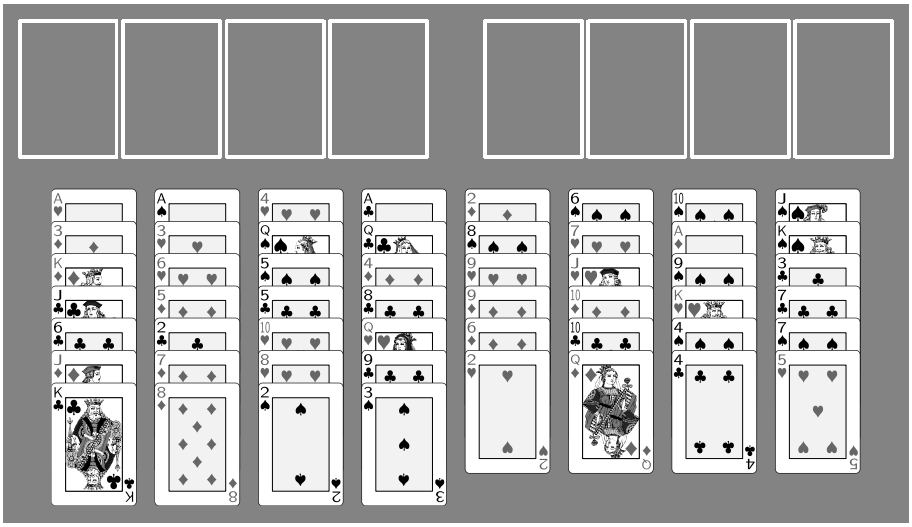

Figure 3: FreeCell problem MS 11982

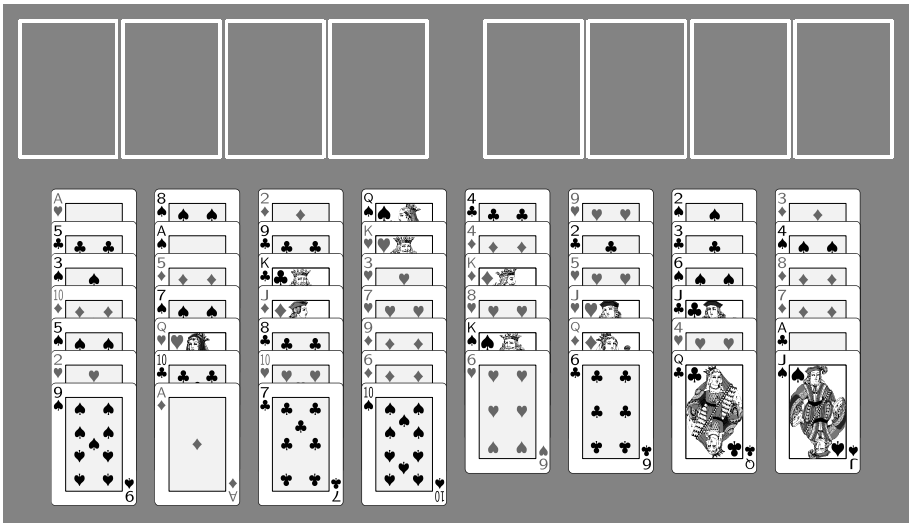

Figure 4: FreeCell problem MS 5

3 Development of a FreeCell Solver

3.1 Design Strategy to the FreeCell Solver

Generally, almost all FreeCell problems can be quickly solved by a simple solver if solution quality is not a concern. Proving a given FreeCell problem unsolvable may be very time-

consuming, depending on the initial position, with a simple solver. Additionally, finding any shortest solution of a given FreeCell problem can be very time-consuming. MS 5 described in Fig. 4 is an example of such a FreeCell problem. Based on the above, the goal of this research is to develop a FreeCell solver that can quickly find any shortest solution to any given problem and, given an unsolvable problem, prove that the problem is unsolvable in a short time.

The design strategy for the FreeCell solver is listed below.

- One-pass DFS (depth-first search) is employed instead of iterative deepening DFS.
- An optimization routine called Mudatori is used during the DFS.
- There are FreeCell positions where the best move can be detected simply by inspecting basic conditions, without the need for searching. The FreeCell solver has the ability to detect such FreeCell positions.
- An algorithm in graph theory that decomposes a given directed graph into strongly connected components is applied to the design of the admissible heuristic function.

3.2 One-pass v.s. Iterative Deepening

An iterative deepening method is often applied in the search for optimum solutions by depth-first search. However, for some FreeCell problems such as MS 5, the one-pass method is more advantageous. In the one-pass method, when a solution is found during the DFS, the search is not stopped and the upper bound of the solution cost is set to one less than that of the found solution, and the search continues.

Both figures in Fig. 5 show the operation of depth-first search. The one-pass method is shown on the left and the iterative deepening method is shown on the right. For the search space, the following inequality is expected to hold for many difficult problems.

$$S < X + Y + Z.$$

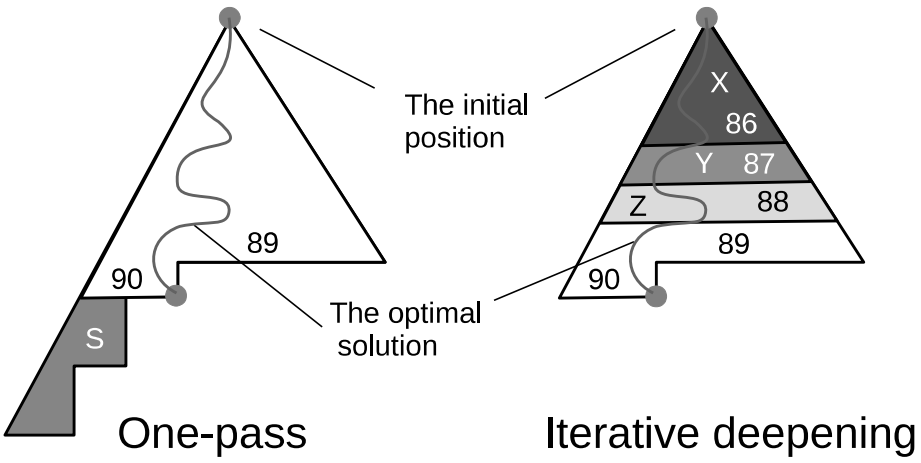

Figure 5: One-pass v.s. iterative deepening

3.3 Optimization Routine Mudatori

Once the DFS finds a solution, an optimization routine called Mudatori is applied to the solution. The principle of Mudatori is described below.

Let the card at place p be moved to place q, and denote this move by $p \to q$. Assume that q is not the foundation. Then, a move sequence $p \to q, \cdots, q \to r$ appears. The possible purposes of move $p \to q$ are the following only.

(a1) To put another card at p after this move. Thus, a move of the form $x \to p$ appears after this move as $p \to q, \cdots, x \to p, \cdots, q \to r$.

(a2) In the case where p is the top of a tableau pile, to move the card that appeared at p after this move elsewhere. Thus, a move of the form $p \to x$ appears after this move as $p \to q, \cdots, p \to x, \cdots, q \to r$.

(b) In the case where q is the top of the tableau pile, to put another card at q after this move. Thus, a move of the form $x \to q$ appears after this move as $p \to q, \cdots, x \to q, \cdots, q \to r$.

If no move of the form $x \to p$, $p \to x$, or $x \to q$ appears before the move $q \to r$ appears, then we may delete move $p \to q$ and change move $q \to r$ to $p \to r$. The optimization routine Mudatori is the procedure of repeatedly applying the above principle to a FreeCell solution until it can no longer be applied.

An example of the Mudatori is shown below. The following is a solution of MS 5 of length 99, where each single-digit hexadecimal number represents a location on the FreeCell board. Numbers from 0 to 3 represent free cells, numbers from 4 to b represent tableau piles, and numbers from c to f represent the maximum rank of each suit in the foundation.

$4 \to 0, 4 \to 1, 4 \to 2, 0 \to 4, 2 \to 8, 4 \to 0, 4 \to 2, 1 \to 4, 2 \to b, 0 \to b,$
$4 \to 0, 4 \to 1, 4 \to 2, 4 \to e, 0 \to e, 5 \to d, 7 \to 0, 7 \to 6, 2 \to 6, 7 \to 5,$
$\cdots\cdots\cdots$
$5 \to d, 4 \to d, 7 \to d, 9 \to f, a \to f, 1 \to f, 5 \to f, 4 \to f, 4 \to d, 7 \to f,$
$4 \to f, 7 \to d, 4 \to d, 0 \to d, 7 \to c, 7 \to e, 7 \to f, 2 \to f, 8 \to c.$

By applying Mudatori to the solution, we have the following solution of length 94.

$4 \to 0, 4 \to 1, 4 \to 8, 4 \to b, 0 \to b, 1 \to 0, 4 \to 1, 4 \to 2, 4 \to e, 0 \to e,$
$5 \to d, 7 \to 0, 7 \to 6, 2 \to 6, 7 \to 5, 7 \to 2, 7 \to e, 9 \to 3, 9 \to 4, a \to 7,$
$\cdots\cdots\cdots$
$1 \to f, 5 \to f, 4 \to f, 4 \to d, 7 \to f, 4 \to f, 7 \to d, 4 \to d, 0 \to d, 7 \to c,$
$7 \to e, 7 \to f, 2 \to f, 8 \to c.$

3.4 Detection of the Best Move without Searching

If we can move a card c with rank A, then it is the best move to move c to the foundation. If there is a card c with rank A in the foundation and we can move the card d with rank 2 and the same suit as c, then it is the best move to move d to the foundation. This idea can be generalized as follows. Let card c be able to be moved to the foundation in the current position. If c satisfies the following condition, then it is the best move in the current position to move c to the foundation.

Condition: Either the rank of c is A or 2, or the two cards of the opposite color to c and two ranks lower than c are both present in the foundation.

Once the best move is found in a position, the search for all other moves in the position may be cut.

For example, if we move the ten of clubs on the Jack of hearts in the position depicted on the upper side in Fig. 6, 26 cards can be moved to the foundation without searching. Then, we get the position depicted on the underside. Furthermore, if we move the ten and the Jack of diamonds to the foundation in the position depicted on the underside, all remaining 16 cards can be moved to the foundation without searching.

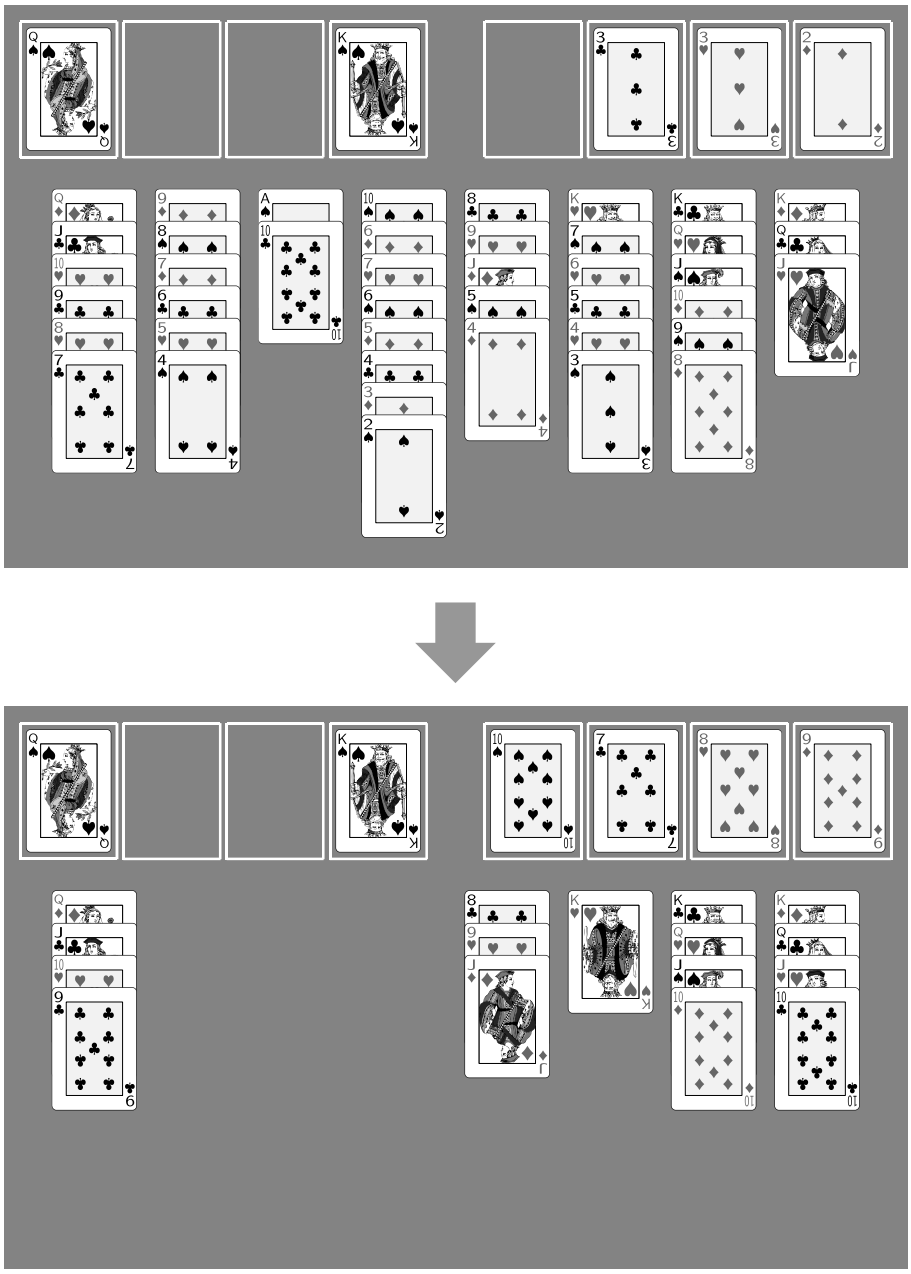

Figure 6: Example of best moves obtained without searching

3.5 Admissible Heuristic Functions

For a FreeCell position P, let $L(P)$ denote the number of moves from the initial position to P. We say that a heuristic function $f(P)$ for FreeCell position P is *admissible* if, for any FreeCell position P, $L(P) + f(P)$ is a lower bound on the length of the solution obtained by extending P.

Let $f(P)$ be an admissible heuristic function, and n a positive integer constant. Suppose that DFS is restricted so that when it reaches position P with the inequality $n < L(P) + f(P)$, it always backtracks. Then, DFS will always find a solution of length n or less if such a solution exists. The larger the admissible heuristic function $f(P)$, the more effective it is in narrowing the search space of the restricted DFS.

For FreeCell position P, let $f_0(P)$ denote the number of cards present outside the foundation. Function $f_0(P)$ is the most basic admissible heuristic function. Let c be a card in a tableau pile S. Card c is said to be a *reversal card* if a card with rank less than that of c exists below c in S. For FreeCell position P, let $f_1(P)$ denote the number of reversal cards in P. Function $f_0(P) + f_1(P)$ is an admissible heuristic function, which is very easy to implement and quite effective.

For a FreeCell position P, let S be a card sequence obtained by arranging all the cards in the tableau of P in a line. Sequence S is said to be *obtained by merging the tableau piles in* P if for any two cards x and y in the same tableau pile, y is behind x in S if y is below x in the pile. A card c in a card sequence S is said to be a *reversal card* if a card with rank lower than that of c exists behind c in S.

For FreeCell position P, let $g(P)$ denote the minimum number of reversal cards in S when S runs in the set of all the card sequences obtained by merging tableau piles in P. Then, function $f_0(P) + g(P)$ is an admissible heuristic function, and $f_1(P) \leqq g(P)$ holds for any P.

Card sequence S with $g(P)$ reversal cards represents the optimum strategy to solve P when infinitely many free cells are available. Repeat the following steps until all the cards have been moved to the foundation.

(a) If there is a card c in a free cell that can be moved to the foundation, then move c to the foundation.

(b1) Otherwise, let c be the head card of S. If c is a reversal card, then remove c from S and move c to a free cell.

(b2) Otherwise, remove c from S and move c to the foundation

Note that if neither the condition in (a) nor in (b1) holds, then the operation in (b2) can be carried out. The value $f_0(P) + g(P)$ is the number of moves that appear while P is solved by using the strategy above in the case where infinitely many free cells are available.

The value of function $g(P)$ for any given position P can be easily calculated using a SAT solver or IP (integer programming) solver. However, the calculation methods using those solvers are too computationally intensive to be incorporated into DFS. An admissible heuristic function that can be regarded as an approximation of $g(P)$ is adopted in the article[5]. The calculation method also seems to be too computationally intensive.

For a FreeCell position P, construct a directed graph $G(P) = (V, E)$ called a *tableau dependency graph* as follows. The vertex set V is the set of all the cards in the tableau

except the reversal cards. An ordered pair $(x, y) \in V \times V$ is in the directed edge set E if x is immediately below y in a tableau pile (tableau edge) or x and y are with the same suit and the rank of x is greater than that of y (rank edge). For a FreeCell position P, let function $h(P)$ denote the minimum size of set S of vertices in $G(P)$ such that $G(P) - S$ has no cycles. Then, $f_1(P) + h(P) \leqq g(P)$ holds for any FreeCell position P.

3.6 Approximation of Function $h(P)$ by Kosaraju's Algorithm

For a FreeCell position P, let $k(P)$ denote the number of strongly connected components of $G(P)$ consisting of multiple vertices. Then, $f_1(P) + k(P) \leqq f_1(P) + h(P) \leqq g(P)$ holds for any FreeCell position P. Function $k(P)$ can be easily calculated by applying Kosaraju's algorithm[1][2] to $G(P)$. Kosaraju's algorithm decomposes a given directed graph into strongly connected components in linear time. Function $f_0(P) + f_1(P) + k(P)$ is expected to be a promising admissible heuristic function for a FreeCell solver. The functions $f_0(P) + f_1(P)$ and $f_0(P) + f_1(P) + k(P)$ are proposed as an admissible heuristic function to be used in the design of the FreeCell solver under development.

4 Concluding Remarks

Development of a FreeCell solver to find any shortest solution to a given problem or to prove the unsolvability of a given problem has been described. The FreeCell solver under development is based on the following strategies. The program is a one-pass DFS. The optimization routine Mudatori is used during the DFS. Best moves are detected without searching in the FreeCell positions that satisfy certain conditions. The algorithm to find the strongly connected components of a directed graph called Kosaraju's algorithm is applied to the design of the admissible heuristic functions.

Future tasks to be addressed include to complete the FreeCell solver program according to the proposed strategies, and to conduct experiments to investigate the differences in the size of search spaces and the performance improvement among heuristic functions $f_0(P) + f_1(P)$, $f_0(P) + f_1(P) + k(P)$, and $f_0(P) + g(P)$.

Acknowledgment

This work was supported by the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms.* Addison-Wesley series in computer science and information processing. Addison-Wesley, 1983.

[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, fourth edition.* MIT Press, 2022.

[3] Shuji Jimbo. Partition of an eulerian circuit search problem for the complete graph of order 15. *RIMS Research Institute for Mathematical Sciences Publications*, 2265:73–77, 2023.

[4] Shuji Jimbo and Akira Maruoka. The upper bound on the eulerian recurrent lengths of complete graphs obtained by an ip solver. In *International Workshop on Algorithms and Computation*, pages 199–208. Springer, 2019.

[5] Gerald Paul and Malte Helmert. Optimal solitaire game solutions using a* search and deadlock analysis. In *Proceedings of the International Symposium on Combinatorial Search*, pages 135–136, 2016.