〔論 文〕

論理的思考力育成授業のためのマインスイーパアプリの開発

千田 陽介 *1, 佐伯 透 *2

Development of a Minesweeper Application for Teaching Logical Thinking

Yosuke SENTA*1, Toru SAIKI*2

Abstract

With the aim of developing the logical thinking skills of collegians, the authors prototyped an application using the rules of the Minesweeper game. Minesweeper game is a logical puzzle, similar to Sudoku or Mastermind. The authors believe that jointly solving logic puzzles is a good way to train engineers for necessary logical thinking skills. However, the original Minesweeper is not completely a logic puzzle. Depending on the initial distribution of mines, some fields require guessing. Therefore, the authors developed an algorithm that logically solves Minesweeper. By using this algorithm, the application can serve a field that does not require guessing to solve. The application has several features for the development of advanced logical thinking skills. For example, opening a mining panel does not immediately end the game. Only the instructor's PC screen shows tips for solving the current situation. By conducting experimental lessons using the application, the authors concluded that although some ingenuity is required to progress in the classes, this application can be used to improve logical thinking ability. This paper describes the algorithms for solving the Minesweeper game; the application's features, which facilitate the smooth progression of logical thinking classes; and the results of experimental lessons using the application.

Key Words: Minesweeper, Logic puzzle, Logical thinking skill class

1　はじめに

筆者らが所属する久留米工業大学情報ネットワーク工学科では, 一年上期に「フレッシュマンセミナー」という授業がある. この授業は, 単位 (卒業) の仕組や図書館等の学内施設の使い方といったレクチャを数回行った後, 十名前後のグループとなって各教員に配属され, 各々の教員が新入生に役立つと考える独自のセミナを行う. その内容は数学や物理の基礎を復習するもの, Python によるプログラム, Maya を用いた基本的な 3D モデルの製作体験など多岐に渡る.

筆頭筆者 (千田) に所属されたグループでは皆で論理パズルを解くことを行っていた. 当情報ネットワーク工学科ではコンピュータプログラムの教育に力を入れている. 千田はプログラミングには論理的思考力が不可欠だと考えている. 例えばプログラムが思い通り動かない時, 今起きてる現象を矛盾なく説明できるモデルを構築し, その推理が正しいか確認することを何度も繰り返す. これはまさしく論理的思考と言ってよい. 論理パズルを解くことは論理的思考の訓練になると考えている. 実際「プログラマのための論理パズル」という本 (1) も存在する. さらに千田のセミナでは論理パズルを個人個人が解くのでなく全員で一つの問題を解くようにしている. 具体的には問題を黒板に板書し, 一人ずつ順番に前に出て「私はここが○○だと思います, なぜなら 〜 だからです」と発表してもらっている. これはエンジニアとしての必要技能: 自分の考えを言語化し論理的に他人に説明するプレゼンテーション能力を鍛えることも狙っているためである.

ここで問題なのは授業に用いる論理パズルの中身と質である. 一人ずつ順番に解くため少しずつ攻略できるものでないといけない. さらに勘や運といった要素はなるべく無い方がよい. またルールの説明にも多少なり時間がかかるので同じルールで幾つも問題が作れなくてはならない. そのような条件を満たすパズルとして, 数独 (ナンバーズプレース: ナンプレとも呼ばれる) とマスタマインドを利用していた. これがどのようなパズルなのかは次章で述べるが, 一つのパズルだと 2 時間 (2 週) が限度であり二つで 4 時間しか持たない. そこで以前よりこれらに続く第三のパズルを探していたが, マインスイーパはどうかと考えた. マインスイーパは 1960 年代に発明されて以来様々なプラットフォームに移植, 遊ばれている単純なゲームである (2). マインスイーパを教育に用いる取り組みとして, プログラミングの授業でマインスイーパを解くプログラムを作

*1 情報ネットワーク工学科　　*2 電子情報システム工学専攻
令和 3 年10月16日受理

	a	b	c	d	e	f	g	h	i
A					4		8		7
B			7		9	6		4	
C	3	5	4		1	7	6		
D			2						1
E					8				
F	7						3		
G			3	1	7		2	9	6
H		9		4	3		1		
I	1		8		6				

Fig. 1: A typical Sudoku puzzle

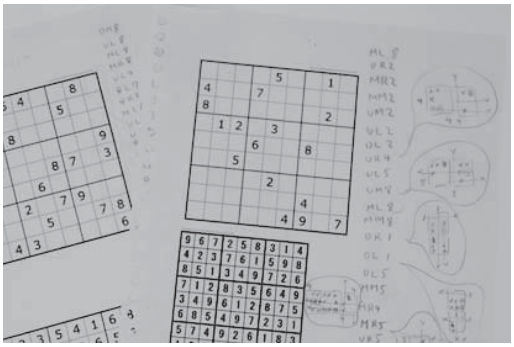

Fig. 2: Lecture notes for Sudoku class

るというものがある [3,4]. それに対し本稿にでは共同でマインスイーパを解き, 論理的思考力を鍛えることを目的としている. 既存のマインスイーパアプリケーションは, 失敗すると即ゲームオーバとなったりどうしても運が必要な場面が出たりと当セミナで考えているような運用に活用しづらい. そこで運を必要としない問題が提供でき, かつゲーム性よりも授業を遂行することに特化した UI を備えたアプリケーションを開発した. 本稿はこの論理的思考力を養う授業に特化したアプリケーションについて述べるものである.

2 論理パズル

2.1 数独・マスタマインド

論理パズルの世界を理解してもらうため, まず授業で活用していた数独とマスタマインドというパズルについて説明する. どちらも一つの問題を解くのに 30 分程かかり, 90 分授業で 3 問解くのを目標にしている.

数独は図 **1** のように 9×9 の桝に $1 \sim 9$ までの数を入れていくパズルである. ここで各行 (A ∼ I) 及び各列 (a ∼ i), さらに太線内で示した 3×3 のブロックの中には必ず $1 \sim 9$ までの数字が一つずつ入る. 授業では図 1 のような問題を黒板に大きく書く. そして順番に前に出て「私は Ah (実際は『ここ』とか『この』と言って図示させているが, 紙面の都合上記号で表記する) が 1 だと思います. なぜなら右上のブロックに注目すると Hg と Di に 1 があるため g 列と i 列, さらに Ce にも 1 があるため C 行に 1 を入れることができず残ったここにしか入れられないからです」などと発表してもらう. 次の者は同様に例えば, 今求めた Ah の 1 と既存の Ce, Ia から左上のブロックに着目して Bb を 1 だと言う. このように論理を重ね全ての桝を埋めていく. 逆に言えば一旦詰まると次に進めない. 桝を埋めていく過程で一ひねりしなければ答えが導き出せない時もある. 例えば図 1 の Gf がそれである. この Gf はこのまま (Ah, Bb の 1 という情報もなしに図 1 のまま) 8 だと導ける. 読者の楽しみとしてあえて解説はしないが筆者の知る限り二通りの理由付けが出来る. 学生の中には運悪くこのような桝しか残っていない時に順番が回ってくる時がある. そこで詰ると授業を進めることができない. 授業を円滑に進めるため, 教員は適切なヒントを与えたり代わりに解いてあげたりしなければならないが, アドリブだと教員もすぐ解法を思いつかないこともある. そのような事態を避けるため事前に問題を解き, どのような順番でどのように解くことができるか図 **2** のようにまとめた講義ノートを作っている. このため事前に準備していた以外の問題を出すことは出来ない. また実際に授業で解かれる工程は講義ノートの通りでは無いため, 今の進行具合ではどこのヒントが出せるか考えなくてはならない.

マスタマインドは隠された数列 (本来はピンの色だが色を数字に対応させれば同じことである) を限られた回数内で当てるパズル (ゲーム) である. 隠す数列の制約には様々な派生がある. 授業では 4 つの重複のない $0 \sim 9$ の数を使っている. 解答者は隠された数列だと思うものを答える. それに対し本当の答え (隠された数列) と位置も数字も合っている個数 h と, 位置は合ってないものの数字はあっている個数 b をヒントとして返す (本稿では $[h, b]$ と表現する). 例えば, 0942 が隠された数列として 2187 と答えた場合, どちらにも 2 があるのでヒントは $[0, 1]$ となる. また 2345 と答えたら 2 や 4 からヒントは $[1, 1]$ となる. このパズルはこれまで出てきたヒントと矛盾ない数列を思いつくことが重要である. そのためヒントを間違えてはいけない. 授業では 25 行程度の BASIC プログラムを走らせたポケットコンピュータを使うことで間違いを防いでいた. 学生は一人ずつ順番に正しいと思う数列を答えてもらうが, その際なぜその数列を選んだのか論理的理由を添えることも必須とした. しかし短時間のうちにここまで出て来た全てのヒントと矛盾ない数列を探し出すことは難しい. ある程度はゲス (guess) を許容せざるを得ないがその匙加減が難しい. さらに最初の数回はゲスで数列を出さざるを得ないが, まぐれでかなり良いヒント ($[0, 0]$ とか $[1, 3]$ とか) が出ることもあり, 一問にかかる時間にバラツキがある.

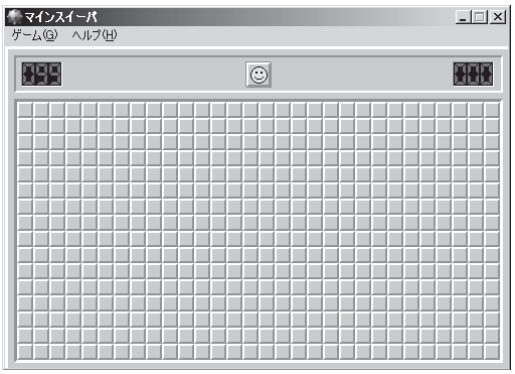

Fig. 3: Capture of Minesweeper game (Windows Xp version)

2.2　マインスイーパ

マインスイーパは歴史の古いビデオゲームであり様々なプラットフォームに移植されている. 図 **3** に Windows Xp (クラシックスタイルテーマ) のマインスイーパを示す. 図のようにパネルが並んでいる. パネルにはランダムで機雷 (「地雷」「爆弾」と呼ばれていることもある) が隠されている. マウスクリック等でパネルを選択すると開くことができるが, もしそこが機雷ならゲームオーバとなる. 機雷でない場合パネルの 8 近傍に存在する機雷の数が開示される. フィールド (本稿では面と表現する) 内の機雷ではないパネルをすべて開けばゲームクリアとなる. 図 **4** (a) を例にマインスイーパの解き方を説明する. 数値の書かれたパネルは既に開いたものとする. 逆に何も書かれてない空白のパネルはまだ開いてないパネルであり, 機雷が潜んでいるかどうか分からない. まず Da の '1' に着目する. これはこのパネルの 8 近傍に機雷が 1 つ潜んでいることを示している. 枠外は機雷なしとみなされるため Da の 8 近傍で未開のパネルは Cb 1 個のみであり, ここに機雷が隠されていると分かる. 次に Ca に着目するとここも '1' である. このパネルの 8 近傍の機雷の数は 1 であるが, Cb が機雷だと確定したため残る Ba, Bb には機雷が隠されていないことが分かる. 本稿ではこの状態を「安全」と称する. 同様に Db に着目すると Cc, Dc も安全だと分かる. 安全なパネル Ba, Bb, Cc, Dc を開くことでその数値 (8 近傍の機雷の数) を見ることができる. 仮に同図 (b) のようになったとする. 機雷があると確定した Cb には 'M' という文字を入れているが, これは機雷が隠れていると分かったので目印 (機雷マーク) を付けたものである. 図 (b) で Ba, Cc に着目すると Aa, Ab, 及び Bc, Bd, Cd, Dd が安全だと分かる. これらを選択し同 (c) のような数値が得られたとすると, Bb より Ac は機雷, Bc もしくは Bd より Ad は安全だと分かり, (d) のように全パネルを開くことができる.

このようにマインスイーパは論理的に機雷を探し当てるゲームである. そのためプロジェクタで図 3 のような既存のアプリケーション画面を投影し, 順番に「自分はここが機雷だと思う. なぜなら〜」「〜といった理由から, ここは安全だと思う」などと発表しつつクリアを目指せば授業出来るように思える. しかし既存のアプリケーションを授業に活用するにあたって問題が二つある.

一つ目の問題は最後まで論理的に解けるかどうかは機雷の配置次第だということである. 例えば図 **5** の状態は Bb と Ac に機雷があり Ab, Bc は安全なパターンとその逆のパターンどちらでも矛盾が無い. 一般的なマインスイーパアプリケーションでは最後の方でこのような状態になることがある. その場合運に頼らざるを得ずゲスがはずれた場合ゲームオーバとなる. 運の要素は初手にもある. マインスイーパの最初は図 3 のように全パネルが閉じた状態であり, ゲームを始めるにあたってどれかを開かないといけない. 選んだパネルがたまたま機雷だった場合もゲームオーバとなる. ただしこれは実装依存であり, 例えば X Window System 用マインスイーパ (xmine 1.0.3) では初手を打った後にそのパネル及び 8 近傍を避けて機雷を配

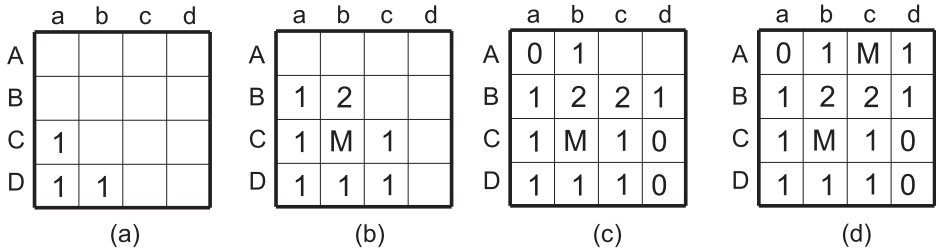

Fig. 4: Sample of Minesweeper solving process

	a	b	c	d
A	1			1
B	2			2
C	M	3	3	M
D	M	2	2	M

Fig. 5: Unsolvable pattern without guess

置する作りになっている (`xmine.c` 921 行目, `layout board()` 関数内参照). 運の要素を排除し, 必ず論理的に解ける機雷配置の面を提供するアプリケーションとして UnambiSweeper[5] がある. 作者のブログ [6] によるとこのプログラムは, 初手が打たれた後にランダムに機雷を配置し, それが論理的に最後まで解けるか試す (解けなかったら再配置する) しくみとのことであるが具体的な実装方法は明らかにされていない. また UnambiSweeper はスマートフォン用のアプリケーションである. これを授業に使うにはスマートフォンの画面をプロジェクタに投影しなければならない. 不可能ではないが簡単ではない.

もう一つの問題は機雷が潜んでいるパネルを開くと即ゲームオーバになることである. これはゲームの方向性としてまったく正しい. しかし授業に用いるには, ある学生が誤るとここまで積み上げて来たものが台無しになる. 全体責任とは言え他の受講者にとって理不尽であろう. また三面クリアしたら終了などと言う運用では授業時間の見積もりが難しくなる. 誤って機雷を開いたとしてもミスだということのみ示してプレイが続行できる方がよい.

このような観点から新しい授業用のマインスイーパアプリケーションを作成した. このアプリケーションの設計思想は以下の通りである.

- 必ず論理的に解ける機雷配置の面を提供する
- 四隅は初手用として必ず安全とする
- 間違えて機雷を選んでもゲームオーバとならない
- プロジェクタに投影する画像と教員手元の PC 画像との内容を変える (教員画像にはヒントを提示する)

3 問題生成アルゴリズム

3.1 アルゴリズムの概要

必ず論理的に解ける機雷配置 (面) を提供する方法は「1. 乱数で適当に機雷を配置」し, 「2. その配置で最後まで論理的に解けるか確認」, 「3. もし解けなければ 1 に戻る」しくみを作ることである. このしくみの実現には 2 の任意の面に対し論理的に解いてみるアルゴリズムが不可欠である. マインスイーパは歴史あるゲームである. そのためマインスイーパを解くアルゴリズムの研究は数多くある [7,8]. これらの研究のほとんどはゲームとしてのマインスイーパを高確率で解くことを目指している. 本来のマインスイーパゲームは図 5 のような運の要素もある. そのため各パネルにおける機雷の存在確率を計算し, より安全なパネルを選ぶ戦略を取っている. 確率が高いパネルを選んで間違っても運が悪かったで済む. 言い換えれば False Positive 的な判断を許容している. 一方本用途では運の要素を排除した面を作ることを目的としているため逆に, 突き詰めれば論理的に解ける配置を解けないと判断する False Negative 的な間違いを許容する. そのため確率的な扱いは行わない.

今回開発した解法アルゴリズムの手順を以下に示す. ただしこれは既に作られた面を解くアルゴリズムであり, 別途四隅を除くパネルに乱数で機雷を設置して面を作る処理が必要である.

1. 四隅のパネルを開く
2. 機雷マークを除去したバッファを作る
3. 未開パネルで単純に安全だと分かるパネルがあれば開き, その後 2 に戻る
4. 未開パネルで単純に機雷だと分かるパネルがあれば機雷マークを記し 2 に戻る
5. テンプレートマッチングで安全もしくは機雷だと分かるパネルがあれば, それぞれ開いたり機雷マークを記したりしてから 2 に戻る
6. 未開パネルが無ければ論理的に解ける面, あれば解けない面と判定する

このうち手順 1 に関しては四隅には機雷を置かないというルール上の話である. 「開く」処理を行うとその 8 近傍の機雷数が分かる. 続く手順 2 は以後の手順 3 〜 5 を円滑に行うための処理である. パネルの数値はその 8 近傍の機雷の値を示して

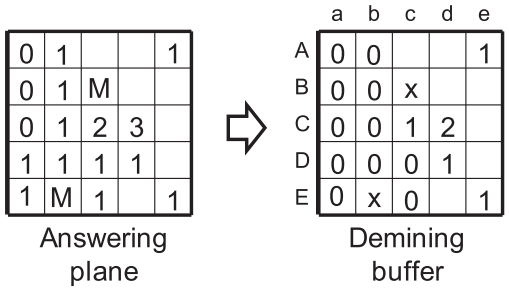

Fig. 6: Virtual demining

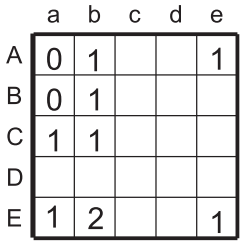

Fig. 7: Solvable pattern using multiple panels

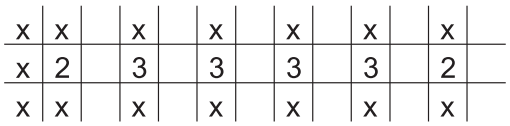

Fig. 8: Extreme example of logically solvable pattern

いる. 逆に考えると機雷はその 8 近傍の値を 1 上げていることになる. 仮に機雷が除去されれば 8 近傍のパネルの値は 1 下がる. 手順 2 はこの作業を行うものである. 図 **6** にその例を示す. 解答中の面 (左) に対し, 右のような機雷を除去して数値を修正したバッファを作る. ここで記号 'x' は「空白ではない」という意味である.

図 6 (右) の Bb の '0' はその 8 近傍に未発見の機雷が無いという意味である. そのため 8 近傍の空白 (図では Ac) は安全なので開いてもよい (上記手順 3). 一方でパネル Cc の '1' は 8 近傍に未発見の機雷が 1 つ存在するという意味である. Cc の 8 近傍の空白は Bd のみであり未発見の機雷数と等しい. このためここには機雷が隠されていることが分かる (上記手順 4).

3.2　テンプレートマッチング

前節で述べたアルゴリズム (手順 3, 4) は隣接するパネルの情報だけを用いて解いていた. マインスイーパでは複数のパネルの情報から機雷の有無を推定することもある. 例えば 図 **7** では Ca もしくは Ea の '1' より Da か Db どちらかに機雷が存在しているはずである. 一方で Cb の '1' はこの Da か Db のどちらかにある機雷を示していることから Bc, Cc, Dc は安全だと分かる.

動的に複数のパネルをグループ化し再帰的処理を行えば, このような複数の情報を組み合わせて論理的に解くアルゴリズムは作れるかもしれない. そのようなアルゴリズムがあれば例えば図 **8** のような配置も解けるであろう (一番右の列 3 個の空白は論理的に安全である). しかし先述したようにここでのアルゴリズムは False Negative な動きを目指す. 図 8 のような配置が起こり, かつこれを解かなければ打開できないようなシチュエーションはまずあり得ない. このような配置は「論理的に解けない」と判断しても実用上問題ない. そのような考えから予めパターン (テンプレート) を用意し, それに当てはまるかどうかで解決するテンプレートマッチングを使うことにした. 図 **9** にテンプレートの例を示す. 数値が書かれたパネルは既に開いたもので数値はそれ自身が持つ値を示す. 'X' は既に開いたパネルもしくは壁, 除去した機雷 (図 6 右での 'x') など未知の機雷が潜んでいる可能性が無いパネルである. '*' はさらに開いているかどうかも問わない. 一方 'Y' は '*' と同じく開いているかどうかすら問わないパネル, 'Z' は未開パネルである. このパターンにマッチした場合, 'Y' は安全, 'Z' は機雷だと確定する. そのため 'Y' に対応するパネルがもし開かれてないのなら開くことができるし, 'Z' に対応するパネルには機雷

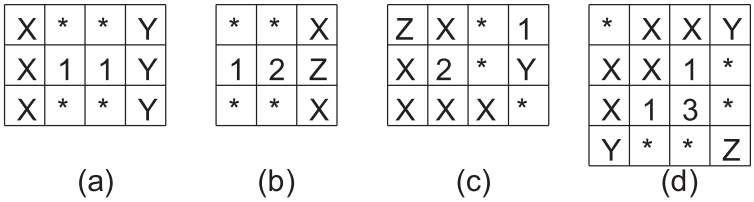

Fig. 9: Template sample for logically solving Minesweeper

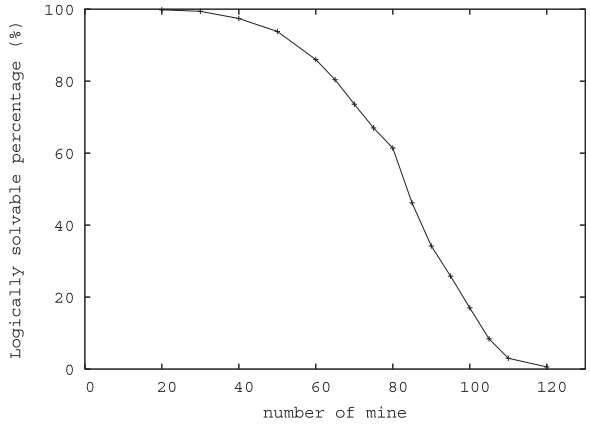

Fig. 10: Relation between the number of mine and solvable percentage without guess (30×16 panel)

マークが置ける. 図 7 の Ca, Cb 辺りは図 9 (a) とマッチする. Y に相当する Bc, Cc, Dc は安全で開くことができる. これらを開いた後の配置は同図 (b) とマッチするため Ec が機雷だと分かる.

テンプレートは実際に前節の手順 1〜6 を行うプログラムを実行し, 最後まで解けなかった面を表示, 論理的に解ける箇所を目視で探しそれに合うものを作成して増やしていった. また作成したテンプレートに対し回転対称や鏡面対称が存在できるならそれも追加した. 図 9 のパターンなら (a) で 4 つ, (b) で 4 つ, (c) で 8 つ (d) で 4 つの計 20 個のテンプレートが作れる. ここで (c) の右上の角の '1' はテンプレート外に近接する 5 つのパネルも安全すなわち 'Y' である. 処理の都合上テンプレートマッチングは $M \times N$ の長方形で行っており, この '1' に近接する 5 つのパネルを 'Y' と記述しようとすると 5×4 の大きさにしなければならない. テンプレートが大きいとマッチング処理に時間がかかる. この 'Y' を記述しなくても別の機会で別のテンプレートにマッチすることから, 小さなテンプレートを多く作る戦略に基づき記述してない. 本稿執筆時点でテンプレートは 248 個である. その中には図 8 のような稀有なパターンは入れてないが特に問題は生じていない.

なおマインスイーパ終盤の戦略として残りの機雷数をヒントに使うこともある. しかし個人的にこれは機雷掃討をメタファとしたゲームの戦略としてふさわしくないと考えている. 本アプリではその感性に基づき, 設置する機雷の数にある程度の振れ幅を持たせかつ公開しないことでこの戦略は使えないようにしている.

4 論理的に解ける場の確率

読者の中で過去 (ランダムに機雷を配置する仕様の) マインスイーパで遊んだ際, 「今のこの面が最後まで運に頼らなくても済む確率はどれくらいだろう」と思ったことはないだろうか. 前章のアルゴリズムでマインスイーパを論理的に解くプログラムを手に入れることができた. せっかくなので面の広さや機雷の数と, 論理的に最後まで解ける面が出現する確率の関係を調べてみた. ここで確率は乱数で作った 500 面のうち最後まで解けた面数から求めた. 図 3 に示した Windows 版マインスイーパの上級コースは 30×16 の面に 99 個の機雷を置いている. これが最後まで論理的に解ける確率を求めてみた所 17.4% であった. ただし Windows 版マインスイーパは「四隅は安全」というルールが無いため実際はこれより多少低いであろう. 一方で図 8 のようなパターンを含み解けないと判定されたものの実は解ける面の存在は否定できない. その存在は逆に確率を上げる方向に作用するが数値として確認できる程にはならないと考えている.

図 **10** に設置した機雷と論理的に解ける面となる確率の関係を示す. 図のように機雷が増えれば増える程, 論理的に解ける面になる確率は減少し, その傾向はロジスティック曲線のような形をしている. 30×16 の面には 480 個のパネルがあり, 機雷 99 個はそのうちの約 20.6% を占めることになる. これを機雷密度と呼ぶこととし, 同じ機雷密度でも面の広さが異なれば論理的に解ける確率が変わるのか調べてみた. 図 **11** に結果を示す. 図から同じ機雷密度でも面が広くなれば論理的に解ける面が出にくくなる傾向があることが分かる. これは面が広くなればなる程, 端のパネルが占める割合が減るためではないかと考えている. 端のパネルは 8 近傍のうち 3 個が壁なため既に開かれている状態である. その分難易度が下がり解き易いのではないだろうか. この件は非常に興味深いものの当研究の趣旨とは逸れてしまうので本稿ではこれ以上追及しない.

5 アプリケーション

3 章で述べたアルゴリズムを使うことで論理的に最後まで解くことの出来る面を提供することはできた. アプリケーションとして完成させるにはユーザインタフェイスを用意しなくてはいけない. ユーザインタフェイスを作ること自体は技術的に

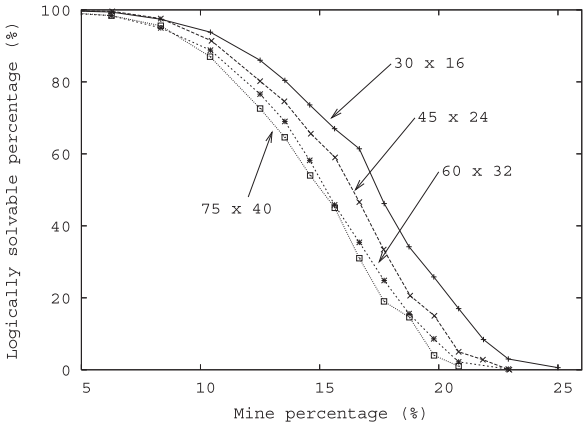

Fig. 11: Relationship between logically solvable percentage and field size

難しいことはない. 機雷配置から求めた各パネルの 8 近傍の数値を配列に記憶し, ユーザがマウスで該当するパネルをクリックしたら記憶した数値を提示するだけである.

ユーザインタフェイスを実装する上で気を配ったことはゲーム性より論理的思考力を鍛える授業の教材としての役割に重きを置いた所である. 授業を円滑に進めるため大きく二つの工夫を入れている. 一つ目の工夫は, 間違えて機雷が隠れているパネルを開いても即ゲームオーバとならないことである. 間違えた旨を提示するだけで元に戻してゲームを続行することができる. 二つ目の工夫は教員用画面と提示用画面を分けていることである. これは Microsoft Power Point アプリケーションのスライドショー機能をイメージすればよいかと思う. Power Point のスライドショーでは手元の PC 画面と液晶プロジェクタ等に投影されている画面の表示は異なっている. それと同じよう教員用画面にのみ授業を進める上で役立つ情報を提示している. これは図 2 に示した数独用講義ノートのような位置付けである. 提示情報の中には機雷のないパネルに機雷マークを置いたらそれが間違いだと示すものがある. 本来のマインスイーパゲームでは機雷マークを間違えて置いても何も起こらない. しかしそこに機雷があるという前提でパズルを解き進めるとそのうち破綻が起こる. この仕様は常に緊張感を維持してゲームに挑ませる効果があるが, 本用途のように授業で順番に解いていく場合あまり望ましくない. 機雷の無い所に機雷マークを置いたら手元の PC だけ分かるようにしている. 教員は適切なタイミングでここはおかしいのではと言える. また今の状態で論理的に解けるパネルも提示出来るようにしている. どのパネルが論理的に解けるかは今の状態に対し 3.1 節の手順 2〜5 を行ってみればよい. ただし手順に記した「2 に戻る」ことまでしてしまうと最後まで解くことになる. 「2 に戻る」所まで来た段階で処理を中断し処理前後の差分から解けるパネルを抽出している. このしくみの都合上, 論理的に解けるパネルを全て表示している訳ではなく, 少なくとも提示したパネルは論理的に解けるという意味付けとなる. 図 **12** に Microsoft Visual C++ 2008 Express Edition を用いて作成したアプリケーションの画面を示す.

6　実験授業

はじめに述べたように, このアプリケーションは 1 年生上期の授業「フレッシュマンセミナー」で活用するために作られた. しかし 2021 年度はコロナ禍の影響を受け該セミナはほとんど遠隔授業となり, 少人数グループに分けての対面授業は無く実際に活用されることは無かった. そこで代わりに久留米工業大学千田研究室に配属された 3 年生で 1 回, 4 年と大学院生の混成で 2 回の計 3 回実験授業を行ってみた. あくまで私見となるが, 今までセミナで行ってきた数独やマスタマインドに近い盛り上がりがあり数独, マスタマインドに続く第三の論理的思考力を訓練する教材になる手ごたえを感じた. 数独, マスタマインドは順番に一人一手ずつ進めていたが, 当アプリケーションでは一手 (一つだけパネルを開くか機雷マークを付ける) だと効率が悪い. 学生の提案もあり 3 〜 5 手ずつ進めた. 機雷密度 20%, 広さ 32×16 を 1 面解くのにおおよそ 30 分くらい要し, 数独やマスタマインドと同じくらいであった.

想定外のこととしてマインスイーパに慣れた学生の存在が少なからずいたことである. マインスイーパは Windows 7 まで OS 付属の標準ゲームであったため実際に深く遊んでいたようである. はじめのうちは慣れた学生と初見の学生との能力差が大きく, 自分の番で取り決めていた数の手を進める時間に顕著な差があった. 回を重ねる毎にその差は縮まっていくが, 授業を進める上で特に初期段階では工夫が必要だと感じた. 現在 (Windows 10) では, マインスイーパはプレインストールされておらず Microsoft Store からインストールする形となっている. 無料とは言えわざわざインストールするとは思えず今後はマインスイーパを経験している学生が減り逆に授業に用いやすくなるのではと考えている.

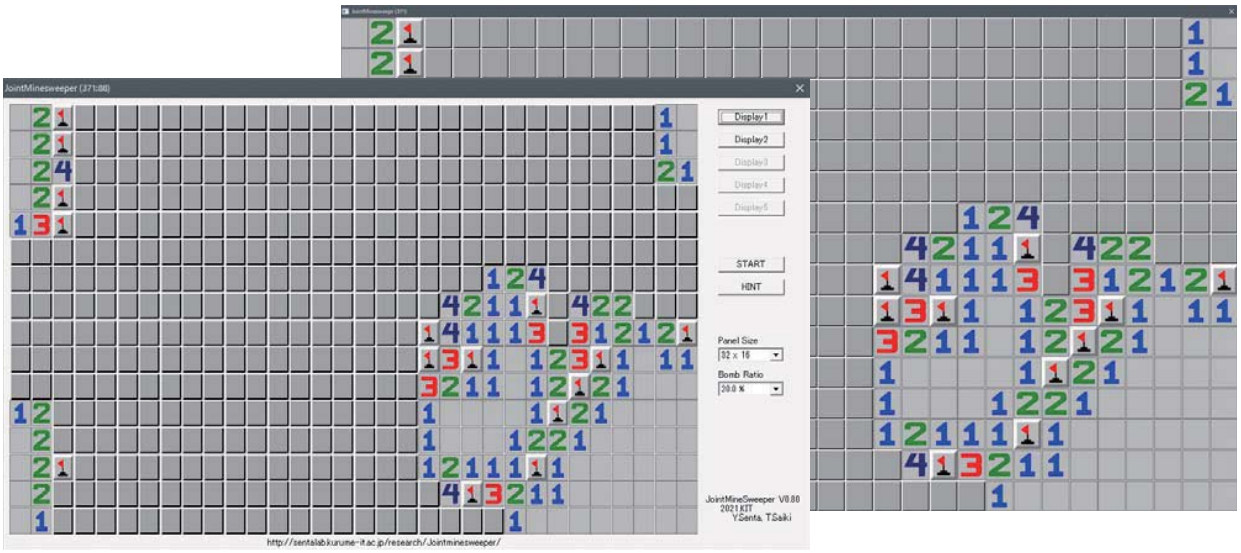

Fig. 12: Capture image of the prototype application in this research (left for facilitator, right for student)

7 おわりに

工学, 特に情報系の技能には論理的思考力が不可欠である. 論理的思考力を育てる授業に使うことを目的にマインスイーパを題材にした教材を試作した. 授業中, 多人数で一つの面を交代で共同で解くためにはマインスイーパが本来持つ運要素を除去する必要がある. そこで乱数で作成したフィールドが論理的に解くことができるか確認し, 確認できたものを提供する作りにした. この作りにはマインスイーパを実際に解くアルゴリズムが必要である. まず最初に簡単な問題に落とし込み, 単純に分かる部分を解き, 最後はテンプレートマッチングで解く, ということを繰り返すようなアルゴリズムを作成した. テンプレートマッチングを用いているため, 実際は論理的に解けるものも解けないと判定する可能性はある. しかし論理的に解ける面を提供するという用途では問題ない.

このアルゴリズムを搭載した教材用アプリケーションを作成した. このアプリケーションはマルチディスプレイに対応し, 液晶プロジェクタ等に投影し教室で共有する画面と, 手元のノート PC のディスプレイに表示する教員用画面とで異なる表示を行っている. 教員用画面は状況に応じた様々なヒントを表示することで円滑に授業を進めることが出来る.

コロナ禍で対面授業が大幅に縮小されたため, 構想時にターゲットとしていた 1 年生に対して授業を行うことが出来なかった. 代わりに 3, 4 年生及び大学院生で実験授業を行った. 有名なゲームであることから経験者が少なからず存在し, 未経験者との能力の差が大きい. これは論理的思考力と言うより慣れの問題であり, 回を重ねる毎に差が縮まっていった. 特に初期段階ではうまく教室をハンドリングする必要がある. 最後に今回試作したプログラム (Windows 用) は `http://sentalab.kurume-it.ac.jp/research/Jointminesweeper/` にて公開している. 授業やセミナ等で活用いただければ幸いである.

文 献

(1) D. E. Shasha (吉平訳), プログラマのための論理パズル, オーム社 (2009)
(2) Wikipedia: Minesweeper (video game), `https://en.wikipedia.org/wiki/Minesweeper_(video_game)`
(3) 朝廣, マインスイーパを題材とした演習, 九州産業大学
(4) 中村, ゲームプログラミングを題材としたプログラミング教育の試み, 情報教育シンポジウム (2001)
(5) 運ゲー排除マインスイーパー – UnambiSweeper, `https://dnek.net/ja/unambi/`
(6) DENK'a blog (2018.12.06), `https://blog.dnek.net/entry/2018/12/06/230114`
(7) 大森, 井上, 可能パターン完全検索による minesweeper の solver 生成, 情報処理学会研究報告 (2012)
(8) 鈴木, 孫, 湊, BDD/ZDD を用いたマインスイーパーの爆弾配列パタンの列挙, 人工知能学会全国大会 (2016)