Ляхов Александр Федорович

ТРУДНО РЕШАЕМЫЕ ЗАДАЧИ И ИГРА «САПЁР»

Аннотация

В работе на основании теории сложности задач показано, что если алгоритм раскрытия некоторого поля игры «Сапёр» существует, то он имеет сложность NP. Результаты экспериментального исследования игры показали, что алгоритм раскрытия существует с некоторой вероятностью. Этот факт позволяет предложить новую классификацию сложности задач.

В приложении приводится описание методов оценки результатов в международных спортивных соревнованиях по игре в «Сапёр».

Ключевые слова: сложность алгоритмов, игра «Сапёр», классификация сложности задач.

Интерес к игре «Сапёр» (Minesweeper), поставляемой в пакете Windows, связан с несколькими причинами.

Во-первых, как показано в работах [1, 2], поиск алгоритма раскрытия игры связан с одной из фундаментальных проблем современной математики – доказательством неэквивалентности классов задач полиномиальной сложности и неполиномиальной сложности.

Во-вторых, процесс раскрытия игрового поля моделирует задачу распознавания графического изображения. Действительно, любое изображение образуется некоторыми связанными множествами пикселей, которые представлены в игре как мины. Исследование возникновения этих структур позволяет сделать ряд качественных заключений о структуре изображения.

В-третьих, поскольку в игре возможна случайная генерация сложных игровых полей равной сложности, то это позволяет внести в процесс игры элемент соревнования между игроками и превратить игру в новую спортивную игру [3] (см. Приложение).

ТРУДНО РЕШАЕМЫЕ ЗАДАЧИ

Проблема определения трудности решения математических задач возникла практически одновременно с возникновением математики. Человечество всегда волновал вопрос, почему одни задачи решаются легко, а другие с трудом, один человек мог решить задачу, а другой нет.

Другой важнейший вопрос, связанный с трудностью задач, может быть сформулирован так: если задачу не удаётся решить, то существует ли у неё решение[1].

Интерес к проблемам существования решения задач и создания алгоритмов обострился в конце XIX века. В это время появились различные механические устрой-

[1] Примеров таких задач, известных с древности, очень много, – это и квадратура круга, и великая теорема Ферма и т. п.

ства для автоматического проведения вычислений. Однако для реализации сложных алгоритмов требовались настоящие вычислительные машины. Это привело к тому, что были предложены различные теоретические модели идеальных вычислительных машин: машина Тьюринга, Поста и т. д.

С появлением реальных электронных вычислительных машин разработки по теории алгоритмов стали особенно актуальны и фактически обеспечили начало современной информационной революции.

На часть вопросов, которые исследовались ранее, ответы были получены, но возникли новые более глубокие проблемы, связанные с решением задач и вычислимостью функций.

Перед тем как проводить сравнительный анализ трудности задач, определим само понятие задачи и ряд характеристик задачи [1].

Под массовой задачей $П$ понимается некоторый общий вопрос, на который следует дать ответ. Обычно задача содержит несколько параметров, или свободных переменных, конкретные значения которых не определены.

Задача определяется следующей информацией:

1) общим списком всех параметров,

2) формулировкой тех свойств, которым должен удовлетворять ответ, то есть решение задачи.

Индивидуальная задача I получается из массовой задачи $П$, если всем параметрам массовой задачи присвоить конкретные значения.

Приведём пример наиболее широко изучаемых переборных массовых задач в теории алгоритмов.

Задача о коммивояжере

Путешественник должен объехать n городов, побывав в каждом городе один раз, и вернуться в исходный город, при этом затратить минимальную сумму на поездку.

Условие. Задано конечное множество $C = \{c_1, c_2, ..., c_m\}$ «городов», расстояние $d(c_i, c_j) \in Z^+$ для каждой пары городов $c_i, c_j \in C$ и граница $B \in Z^+$ (положительное целое число).

Вопрос. Существует ли «маршрут», проходящий через все города из C, длина которого не превосходит B?

Этот вопрос может быть записан в следующем виде: существует ли последовательность $\langle c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(m)} \rangle$, такая что

$$\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(1)}) \le B ?$$

Очевидно, что при малом количестве городов решение находится перебором возможных вариантов, но с ростом m количество различных вариантов маршрута растёт как $m!$.

Задача о рюкзаке

Дано n предметов, для каждого из которых известен объём a_i и стоимость b_i. Требуется этими предметами заполнить рюкзак ограниченного объема так, чтобы суммарная стоимость упакованных предметов была максимальной.

Условие. Задано конечное множество U, размер $S(u) \in Z^+$ и стоимость $V(u) \in Z^+$ каждого $u \in U$, положительные целые числа B и K.

Вопрос. Существует ли такое подмножество $U' \subseteq U$, что $\sum_{i \in U'} S(u) \le B$ и $\sum_{i \in U'} V(u) \ge K$?

Ещё одной широко известной массовой переборной задачей, о которой стоит упомянуть, является задача о разложении составного числа на множители.

Под алгоритмом задачи будем понимать общую выполняемую шаг за шагом процедуру, позволяющую её решить.

Будем говорить, что алгоритм решает массовую задачу $П$, если он применим к любой индивидуальной задаче I из $П$ и дает её решение[1].

[1] Заметим, что класс математических задач шире класса задач, решаемых алгоритмически [2].

Понятие эффективности алгоритма связано со всеми вычислительными ресурсами, необходимыми для его реализации. Обычно под «эффективным» алгоритмом решения задачи понимают самый быстрый алгоритм.

Время работы алгоритма удобно выражать в виде функции от одной переменной, характеризующей размер индивидуальной задачи, то есть объёмом входных данных, требуемых для описания задачи.

Описание задачи осуществляется следующим образом.

Введём Σ – множество символов, которые образуют некоторый алфавит. Цепочки из символов этого алфавита образуют слова, множество таких цепочек Σ^*. Любое подмножество $L \subseteq \Sigma^*$ будем называть языком в алфавите Σ.

Соответствие между задачами и языками устанавливаются с помощью схем кодирования.

Схема кодирования l задачи Π описывает каждую индивидуальную задачу из Π подходящими словами в некотором фиксированном алфавите Σ^*.

$$L[\Pi, l] = \left\{ x \in \Sigma^*,\ x - \text{код индивидуальной задачи} \right\}$$

Задача Π и схема кодирования l разбивают слова Σ^* на три класса: слова, не являющиеся кодами задачи Π, слова, являющиеся кодами индивидуальных задач Π с положительным ответом на вопрос, и слова с отрицательным ответом.

Если l и l' две схемы кодирования, то $L[\Pi, l]$, $L[\Pi, l']$ выполняются одновременно. Входная длина индивидуальной задачи I из Π определяется числом символов в цепочке, полученной применением к задаче I схемы кодирования для массовой задачи Π. Именно это число используется в качестве формальной характеристики размера индивидуальной задачи.

Приведём простейший пример кодирования массовой задачи.

1. Массовая задача может быть описана на естественном языке.

Дано число A и число B. Чему равно произведение чисел?

Индивидуальная задача. Дано число A, равное двум и число B, равное двум. Чему равно произведение чисел?

2. Математическая кодировка массовой задачи: A*B = ?

Индивидуальная задача 2*2=?

3. Массовая задача на языке С:

```
1. /*Произведение двух чисел
2. #include ⟨<studio.h>⟩
3. void main ()
   4. {
   5. int a,b,s;
   6. printf("\n Введите числа а и в");
   7. scanf("%d%d",&a,&b);
   8. s=a*b;
   9. printf("\n Произведение S= %d",s);
   10. }
```

Индивидуальная задача может быть получена двумя способами, либо через введение значений A и B, либо изменением восьмой строки s=2*2.

Очевидно, что в приведённом примере длина кода задачи зависит от длины записи перемножаемых чисел и результата[1].

В теории алгоритмов принято различать сложность алгоритма по изменению его длины в зависимости от входных параметров.

С каждой задачей Π связана не зависящая от схемы кодирования функция $Length : D_\Pi \to Z^+$ (D_Π – множество всех индивидуальных задач), которая «полиномиально эквивалентна» длине кода индивидуальной задачи, получаемой при любой разумной схеме кодирования ($Length[I]$ – функция длины задачи в символах выбранного алфавита).

Под полиномиальной эквивалентностью задач понимается следующее. Для любой разумной схемы кодирования l задачи Π существует два полинома p и p' такие, что $Length[I] \le p(|x|)$ и $|x| \le p'(Length[I])$, где $|x|$ – длина слова x.

Алгоритм решения задачи Π будем называть псевдополиномиальным по времени

[1] Для формализации понятия «алгоритм» необходимо зафиксировать определённую модель процесса вычислений.

алгоритмом, если его временная функция ограничена сверху полиномом от двух аргументов *Length* [*I*] и *Max* [*I*] (*Max* [*I*] – максимальное число в *I*).

Полиномиальным алгоритмом (или алгоритмом полиномиальной сложности) называется алгоритм, у которого временная сложность равна $O(p(n))$, где $p(n)$ – некоторая полиномиальная функция, а n – входная длина.

Приведём оценки полиномиальной сложности некоторых известных алгоритмов. Алгоритм быстрой сортировки элементов массива $O(n^2)$, алгоритм пирамидальной сортировки $O(n\log_2 n)$. Известен алгоритм умножения двух чисел с асимптотической сложностью $O(n\log n\log\log n)$, алгоритм умножения матриц $O(n^{2,2})$, алгоритм решения задачи линейного программирования – со сложностью $O(n^{3,5})$ [3], метод Гаусса решения систем линейных алгебраических уравнений имеет сложность $O(n^3)$, быстрое преобразование Фурье – $O(n\log n)$.

Заметим, что любой полиномиальный алгоритм является псевдополиномиальным по времени, но не наоборот.

Временная сложность алгоритма отражает требующиеся для его работы затраты времени. Это функция, которая каждой длине n (индивидуальной задачи) ставит в соответствие максимальное время, затрачиваемое алгоритмом на решение индивидуальных задач этой длины[1]. Функция временной сложности будет полностью определена, если зафиксирована схема кодирования, определяющая входную длину индивидуальной задачи, и выбрано вычислительное устройство (или его модель), определяющее время работы.

Заметим, что сложность алгоритма связана с используемой для решения задачи памятью. Например, пусть требуется определить число единиц в двоичной последовательности длины n. Алгоритм, который считает единицы, последовательно просматривая все элементы последовательности, имеет трудоёмкость $O(n)$. Можно предложить другой алгоритм, который требует 2^n ячеек памяти для хранения всех двоичных последовательностей длины n, причём номер ячейки – это двоичная последовательность длины n, а в самой ячейке хранится число единиц этой последовательности. Задача решается за одно обращение к памяти, то есть трудоёмкость алгоритма $O(1)$.

Полиномиальные алгоритмы обычно создаются в результате глубокого анализа задачи. Задачи, для которых найден полиномиальный алгоритм, относятся к множеству задач класса P (Polynomial).

Все неполиномиальные алгоритмы, как правило, считаются экспоненциальными. Большинство экспоненциальных алгоритмов являются вариантами полного перебора возможных частных решений задачи, и такие задачи относятся к задачам класса NP (Nondeterministically Polynomial).

Принято считать задачу труднорешаемой, если для неё не получен полиномиальный алгоритм.

Примером экспоненциальных задач являются выше описанные задачи о коммивояжере и рюкзаке. В работе [1] приводится 300 различных труднорешаемых задач.

Для исследования задач NP используют недетерминированный алгоритм, который состоит из двух этапов:

1) «угадывание» решения,

2) проверка найденного решения.

Заметим, что проверка осуществляется за полиномиальное время.

В 1971 году вышла работа С.А. Кука «Сложность процедур вывода теорем» [4], в которой было показано:

1. Если одна задача сводится к другой задаче за полиномиальное время, то любой полиномиальный алгоритм второй задачи может быть превращён в полиномиальный алгоритм решения первой задачи[2].

2. В качестве базовой модельной задачи класса NP была выбрана задача выполнимости.

---

[1] Практически во всех математических пакетах имеются встроенные функции, позволяющие определять время работы программы, то есть время индивидуальной задачи.

[2] Примером такой сводимости могут служить различные алгоритмы прямых методов решения систем линейных алгебраических уравнений $O(n^3)$ (метод Гаусса, LU-метод, метод прогонки и др.).

Задача выполнимости

Условие. Заданы множество булевых переменных U ($U = \{u_1, u_2, ...u_n\}$). Под набором значений истинности на множестве U будем понимать функцию $F: U \to \{T, F\}$ и набор C дизъюнкций над U.

Вопрос. Существует ли выполняющий набор значений истинности для C?

Например, заданы логические переменные X, Y, Z и набор логических операций $\wedge, \vee, \neg$. Существует ли функция $F = X \wedge Y \vee Z$, имеющая истинное значение при каких-то значениях X, Y, Z?

Ответ на этот вопрос может быть получен перебором подстановок значений переменных.

Теорема Кука: *Задача выполнимости есть NP-полная задача.*

Было показано, что любая задача из класса NP за полиномиальное время может быть сведена к задаче выполнимости.

Проверка решения каждой индивидуальной задачи из класса *NP* осуществляется за полиномиальное время, а выбор этого решения – случайным образом, то есть имеет место недетерминированный алгоритм решения. Число вариантов решений, которые надо проверить растёт экспоненциально[1] с ростом входных параметров задачи.

Вопрос о взаимоотношении классов *P* и *NP* является фундаментальным вопросом теории алгоритмов. Очевидно, что $P \subseteq NP$, и доказана следующая теорема. Если $\Pi \in NP$, то существует полином p такой, что Π может быть решена детерминированным алгоритмом с временной сложностью $O(2^{p(n)})$.

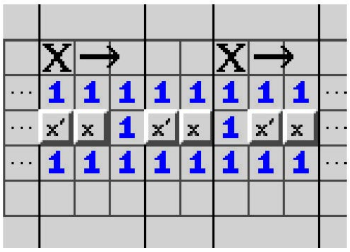


Рис. 1

Высказана гипотеза, что $P \neq NP$, но доказательства её нет. Если бы она была доказана, то для каждой конкретной задачи можно было бы определить, к какому классу она принадлежит, к классу *P* или к классу трудно решаемых задач *NP* [2].

Ответ на вопрос о неэквивалентности классов *P* и *NP* задач может быть получен, если будет найдена задача, принадлежащая *NP**P*, и для неё будет доказано отсутствие полиномиального алгоритма решения.

Поиском такой задачи и объясняется интерес к игре «Сапёр» (Minesweeper).

ПОИСК АЛГОРИТМА РАСКРЫТИЯ ИГРЫ «САПЁР»

Игра сапер входит в стандартный набор игр Windows. В ней имеется прямоугольное поле, состоящее из клеток. Одни клетки пустые, в других заложены мины. Открытие клеток ведется последовательно до первой ошибки или полного раскрытия поля. При открытии пустой клетки в ней появляется число, равное числу мин, находящихся в восьми клетках, которые окружают открытую клетку.

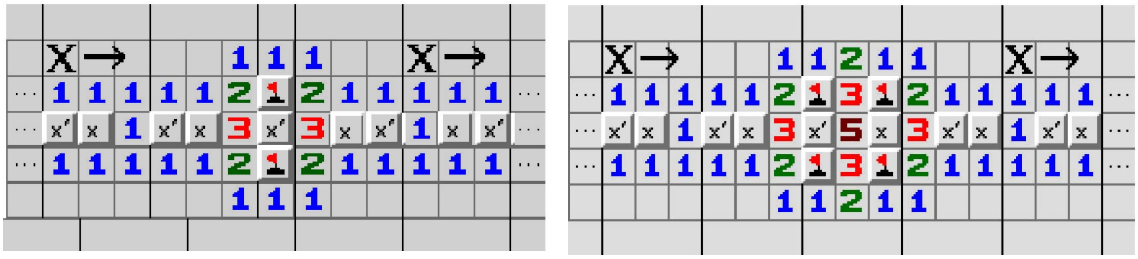
Из игровой практики известно, что при некоторых простых конфигурациях мин алгоритм открытия является последовательным линейным алгоритмом; в других, более сложных случаях, поиск алгоритма открытия мин осуществляется перебором возможных различных расстановок мин.

В работе [5] показано, что при переходе к бесконечному игровому полю в игре Сапёр могут возникать расстановки мин, которые можно представить как некоторые логические элементы.

Например, структура, приведённая на рис. 1, описывает перенос переменной *X*.

[1] К этим задачам относятся такие массовые задачи: составление расписаний (учебных, рабочих часов), планирование производства, минимизация максимальных затрат и т. д. [1].

[2] Интересно заметить, что математическим институтом Глена США предложена денежная премия в размере одного миллиона долларов за решение проблемы эквивалентности классов P и NP задач.



NOT NOT(NOT)

Рис. 2

Здесь x' – соответствует наличию мины, x – её отсутствию (можно рассматривать обратные значения x' – мины нет, x – мина есть).

На рис. 2 приведены структуры, соответствующие отрицанию (NOT) и двойному отрицанию (NOT(NOT)).

В этой же работе [5] показано, что в игре «Сапёр» на бесконечном поле могут возникать структуры, которые эквивалентны последовательности логических элементов, и, следовательно, некоторые индивидуальные задачи раскрытия игрового поля сводятся к задаче выполнимости Кука, то есть существует последовательность индивидуальных задач раскрытия игры «Сапёр», которая принадлежит к классу NP полных задач.

Однако игровая практика показывает, что этим не исчерпываются все возможные задачи игры. Действительно, при последовательном переборе возможных различных расстановок мин может возникнуть неоднозначность, то есть искомая мина с равной вероятностью может находиться в нескольких клетках. Например, пусть всё игровое поле раскрыто, а в области, ограниченной кластером из мин, должна находится одна мина (рис. 3). В этом случае мина с равной вероятностью находится в одной из двух клеток. Логический переход между внешним полем и областью, ограниченной кластером, невозможен.

Следовательно, в этом случае детерминированного алгоритма раскрытия игры не существует.

Возникает задача об определении вероятности появления замкнутых областей при различных концентрациях мин на поле [6].

Задача. По заданному размеру поля и случайно расставленным минам при различных концентрациях требуется определить критические значения концентрации, при которых происходит появление замкнутых кластеров и кластеров, разделяющих игровое поле на независимые части.

Анализ задачи о появлении кластеров из мин может быть проведён с помощью теории перколяции.

Слово перколяция и основные идеи теории перколяции были разработаны Симоном Бродбентом и Джоном Хаммерсли в 1957 г. при изучении ими явлений прохождения газов через угольный фильтр. Движение газа по лабиринтам из пор в угле они назвали перколяционным процессом (percolation – просачивание, протекание).

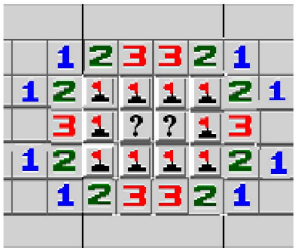


Рис. 3

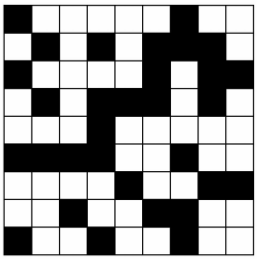


Рис. 4

Рассмотрим явление перколяции на простейшей квадратной сетке (рис. 4) [7]. Часть клеток сетки закрашивается в чёрный цвет. Доля закрашенных квадратов составляет $p = \frac{N_{черн.}}{N}$, N – общее число клеток, $N_{черн.}$ – чёрные клетки.

Квадраты для закрашивания выбираются двумя способами: либо случайно, либо по какому-то правилу. В первом случае говорят о случайной перколяции (перколяции Бернулли), во втором – о коррелированной.

В дальнейшем будем рассматривать случайные перколяции.

Цепочку связанных клеток одного типа называют кластером.

Кластер, соединяющий две противоположные стороны системы, называют перколяционным или бесконечным кластером[1].

Один из основных вопросов теории перколяции – при какой доле p закрашенных квадратов возникает цепочка черных квадратов, соединяющая верхнюю и нижнюю стороны сетки, то есть бесконечный кластер?

При случайной расстановке чёрных клеток такие цепочки могут возникать при разных концентрациях.

Однако если размер сетки L устремить к бесконечности, то критическая концентрация станет вполне определенной. Такую критическую концентрацию называют порогом перколяции.

По своей сути перколиционный порог или перколиционный переход является геометрическим фазовым переходом. Действительно критическая концентрация клеток разделяет все сетки на два вида (две фазы): в одном виде существуют конечные кластеры, в другом – существует один бесконечный кластер.

Многие важные характеристики кластера (длина корреляции, среднее число узлов) вблизи перехода описываются показательной функцией с различными критическими показателями.

ПОИСК КЛАСТЕРА ИЗ МИН НА ИГРОВОМ ПОЛЕ «САПЁР»

В работе [6] показано, что вероятность раскрытия поля «Сапёр» зависит от существования кластера из мин, отсекающего часть игрового поля.

1. Если на поле есть кластер из мин, делящий игровое поле на две независимые области, так что в каждой из этих областей есть мины, то алгоритм раскрытия такого поля не существует.

2. Если кластер из мин делит поле на область, содержащую все мины, и пустую область, то существование алгоритма раскрытия зависит от «толщины» кластера (то есть стороны наибольшего квадрата из заминированных клеток, целиком включенного в кластер). Если этот кластер содержит мины, которые не могут быть обнаружены при полном раскрытии первой области, то алгоритм раскрытия не существует.

Для исследования вопроса о возникновении бесконечного кластера из мин использовался численный эксперимент. Для построения зависимости вероятности появления кластера из мин от плотности из мин была модифицирована программа Cluster (рис. 5), описанная в работе [6].

Программа различает шесть видов кластеров: кластеры, соединяющие левую и правую стороны поля, верхнюю и нижнюю, верхнюю и левую, верхнюю и правую, нижнюю и левую, нижнюю и правую. Некоторые кластеры относятся к нескольким видам одновременно. Программа подсчитывает случаи возникновения кластеров каждого вида в отдельности, а затем находит их общее число за большое заданное число расстановок мин на поле. В программу была добавлена новая возможность – определять частоту появления и отсутствия кластеров, то есть оценивать вероятность существования алгоритма раскрытия поля.

Начальными данными для работы программы являются размеры поля (число клеток по горизонтали N_1 и по вертикали N_2),

[1] Для подобных кластеров в литературе используются названия «стягивающий кластер», «соединяющий кластер».

Рис. 5

число мин на поле M и количество опытов K, которое пользователь хотел бы провести. В каждом опыте программа случайным образом расставляет мины по клеткам поля.

В результате эксперимента была определена вероятность отсутствия кластера на квадратных полях разного размера и с различной плотностью мин. Вид зависимости этой частоты от плотности мин показан на рис. 6.

Можно видеть, что частота отсутствия кластеров зависит главным образом от плотности мин на игровом поле. При плотности 0,20 (игровое поле «Профессионал») с достаточно высокой вероятностью ~ 70% будет найден хотя бы один кластер, при плотности мин 0,15 (игровое поле «Любитель») около половины полей содержат кластер. На полях с меньшей плотностью кластеры появляются редко, и проблем при раскрытии практически не возникает.

Заметим, что вероятности раскрытия поля будут выше полученных вероятностей отсутствия кластеров из мин, так как часть найденных кластеров не отсекает поля или отсекает пустые области.

Полученные результаты позволяют сделать вывод о том, что вероятность существования алгоритма раскрытия игрового поля «Сапёр» тем меньше, чем выше плотность расположения мин, причем эта зависимость имеет нелинейный характер.

Рис. 6

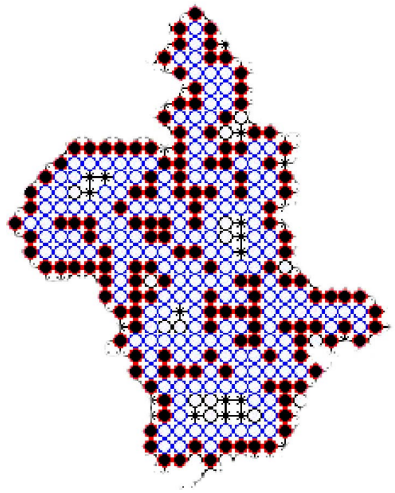


Рис. 7

Дальнейшие исследования показывают, что замкнутые кластеры при больших плотностях мин носят явно выраженный фрактальный характер (рис. 7) (фрактальная размерность при n = 1,317, ρ = 0,5).

Тот факт, что кластеры имеют фрактальную размерность, означает, что они обладают свойством масштабной инвариантности, поэтому исследование небольших по размерам решеток позволяет судить о бесконечных решетках. Данное свойство индивидуальных задач раскрытия игры «Сапер» позволяет говорить о новом виде их сводимости друг к другу.

ЗАКЛЮЧЕНИЕ

Проведённые исследования игры «Сапёр» позволяют сделать следующие выводы:

1. Алгоритм раскрытия игры «Сапёр» существует с некоторой вероятностью.

2. Если алгоритм игры раскрытия существует, то он может иметь неполиномиальную сложность NP.

3. Можно предложить новую классификацию сложности задач: задачи, имеющие полиномиальную сложность с некоторой вероятностью P (класс PP), задачи, имеющие неполиномиальный алгоритм с вероятностью P (класс NPP), и невычислимые задачи[1] с вероятностью P (класс NCP).

Литература

1. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.

2. *Картленд Н.* Вычислимость, введение в теорию реккурсивных функций. М.: Мир, 1983.

3. *Гордеев Э.Н.* Задача выбора и их решение // Сб. Компьютер и задача выбора. М.: Наука, 1989 г. С. 2–49.

4. *Cook S.A.* The complexity of theorem-proving procedures. Proc.3rd., Ann. ACM Symp. On Theory of Computing Machinery, New York. P. 151–158.

5. *Richard Kaye's* «Minesweeper is NP-complete», Mathematical Intelligencer. Vol 22. Number 2, P. 9–15, 2000.

6. *Виноградова Е.В., Ляхов А.Ф.* Вероятность существования алгоритма раскрытия игры «Сапёр» // Компьютерные инструменты в образовании, 2007, № 3. С. 72–79.

7. *А.Л. Эфрос.* Физика и геометрия беспорядка // Библиотечка «Квант». Вып. 19. М.: Изд. «Наука». Гл. редакция физ.-мат. литературы, 1982.

Приложение

СПОРТИВНЫЕ СОРЕВНОВАНИЯ ПО ИГРЕ «САПЁР»

Компания Microsoft рекламируя свой программный продукт, осуществляет поддержку информационной компании по проведению как национальных чемпионатов, так и чемпионата мира по игре «Сапер» (сайт www.minesweeper.ru). Как это принято во всех игровых видах, существуют национальные и мировые таблицы рейтингов игроков. Из последних новостей отметим, что 8

[1] Заметим ещё раз, что существуют алгоритмически невычислимые задачи [2].

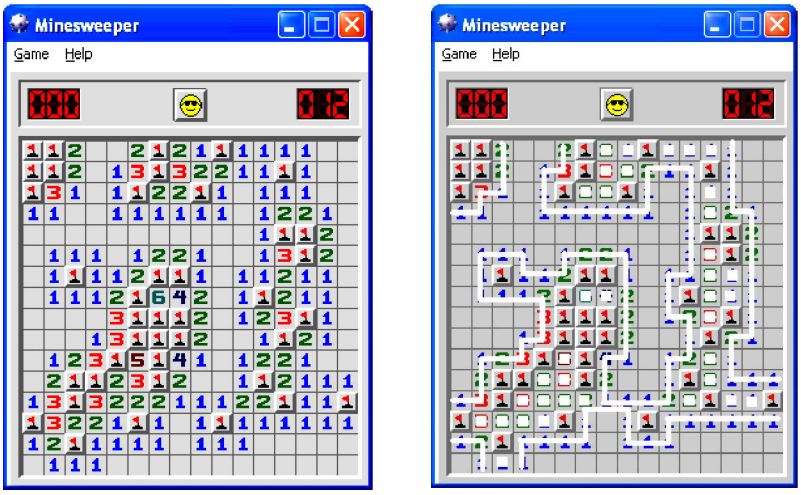


Рис. 1

апреля 2009 года завершился третий кубок России, а 20 июня 2009 года в шотландском городе Стирлинг проходил международный турнир по игре «Сапёр».

Соревнования в игре «Сапёр» проводятся на быстроту открытия игрового поля. Основное условие любого соревнования состоит в создании равных исходных позиций для соревнующихся игроков. Это условие в игре выполняется с помощью генерации игровых полей одинаковой сложности «Любитель» и «Профессионал».

Для оценки сложности игровой позиции используется минимальное число кликов, необходимых для его раскрытия, коэффициент 3*BV*.

Например, на рис. 1 приведено игровое поле, имеющее четыре области без мин (openings). Для их раскрытия требуется четыре клика и 36 клеток с цифрами, лежащие не на границе открытых областей. Итого для раскрытия игры требуется 40 кликов (3*BV* = 40).

Распределения 3*BV*

На графике (рис. 2) указаны вероятности получения различных значений 3*BV* на доске «Любитель» с плотностью мин равной $\rho = 0{,}16$.

Можно видеть, что распределение коэффициента 3*BV* близко к нормальному. Среднее значение $3BV \approx 64{,}92$. Заметим, что большая часть рекордов ставится на досках с 3*BV* менее 40, а вероятность получения такой доски составляет 0,008. Решением International Minesweeper Committee игровые поля с $3BV < 30$ не учитываются в рейтинге игроков.

Аналогичный вид имеет график распределения коэффициента 3*BV* на игровом поле «Профессионал», $\rho = 0{,}21$. В этом случае среднее значение $3BV \approx 173{,}58$. Простые поля с 3*BV* меньше 130 составляют всего 0,008 от общего числа досок. Поля с 3*BV* меньше 99 запрещены для использования в соревнованиях.

Заметим, что существует программа Minesweeper Clone 2007, проверяющая коэффициент 3*BV* и не позволяющая играть на полях с его малым значением.

Для оценки результативности игрока вводится скорость разминирования $V_{игр} = \frac{3BV}{T}$, где T – время игры. Однако этот коэффициент обладает рядом недостатков. Высокие скорости раскрытия достигаются на полях с большим коэффициентом 3*BV*. Это объясняется тем, что на таких игровых полях имеет место большая «плотность» клеток, дающих вклад в 3*BV*.

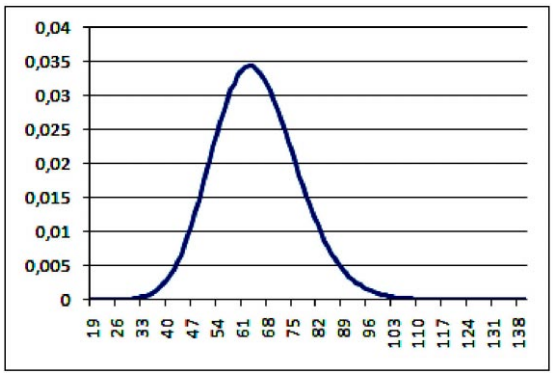


Рис. 2

В качестве альтернативы $V_{игр.}$ для устранения эффекта роста скорости при росте $3BV$ используется коэффициент IOS = Index Of Speed = $\log_T 3BV$. Этот коэффициент позволяет сравнивать по скорости как отдельных игроков, так и все без исключения игры одного игрока.

Вторым альтернативным коэффициентом является RQP = Rapport Qualite Prix = $\frac{T}{V_{игр.}}$. По этому коэффициенту имеет смысл сравнивать в основном игры с малыми временами.

Информация взята с сайта www.minesweeper_ru.

Abstract

In the paper it is shown based upon the theory of complexity of problems if a solving algorithm exists in a sample «Minesweeper» field, it has NP-complexity. Experimental treatments of the game show that the solving algorithm exists with certain probability. These results allow new classification of problems' complexity to be proposed.

Paper's Appendix contains description of methods for evaluating results of players at the international Minesweeper championship.

Ляхов Александр Федорович, кандидат физико-математических наук, доцент кафедры теоретической механики механико-математического факультета НГУ им. Н.И. Лобачевского, Lyakhov@mm.unn.ru