Санкт-Петербургский государственный университет

Направление: Математика, механика

Специализация: Алгебра

Пейсаховский Пётр Олегович

Моделирование уклонения в задаче поиска на графе

Дипломная работа

Научный руководитель:
кандидат физ-мат. наук, старший преподаватель
Абрамовская Т.В.

Рецензент:
кандидат физ.-мат. наук, профессор
Бешко М. Ю.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Main Field of Study: Mathematics, mechanics

Area of Specialisation: Algebra

Pejsahosvky Petr Olegovich

Modelling deviation in a problem of graph search

Graduation Thesis

Scientific supervisor:
Senior Lecturer T. V. Abramovskaia

Reviewer:
Associate Professor M. Y. Beshko

Saint-Petersburg
2016

Содержание

1 Перечень условных обозначений

$\|C\|$	Количество полицейских
$\|V\|$	Количество вершин графа
$\mathbf{E}$	Математическое ожидание
$\mathcal{C}$	Игрок, играющий за полицейских
$\mathcal{R}$	Игрок, играющий за грабителя
σ_C	Функция, которая в рамках определенной стратегии сопоставляет новую позицию полицейских C' каждой конфигурации игры
σ_R	Функция, которая в рамках определенной стратегии сопоставляет новую позицию грабителя r' каждой конфигурации игры
A_C	Алгоритм для $\mathcal{C}$
$A_C(G)$	Множество стратегий для $\mathcal{C}$ на графе G, соответствующее алгоритму A_C
A_R	Алгоритм для $\mathcal{R}$
$A_R(G)$	Множество стратегий для $\mathcal{R}$ на графе G, соответствующее алгоритму A_R
$C = (c_1, ..., c_k)$	Список вершин, занимаемых фишками полицейских
$C\&R$	Игра «полицейские и грабитель»
$cn'(G)$	Эмпирическое полицейское число
$cn(G)$	Полицейское число графа

E	Множество ребер графа G
$G = G(V, E)$	Связный (если не указано обратное) неориентированный граф
$girth(G)$	Минимальная длина индуцированного цикла
I_C	Первый ход полицейских для некоторой стратегии на графе G
I_R	Первый ход грабителя для некоторой стратегии на графе G
k	Количество фишек у игрока, играющего за полицейских
$N(v)$	Множество вершин, смежных с v (соединенных с v ребром)
$P_C(v)$	Потенциал вершины $v \in V$, при расположении полицейских C
r	Вершина занимаемая грабителем
S	Текущая конфигурация игры
S_C	Стратегия для $\mathcal{C}$
S_R	Стратегия для $\mathcal{R}$
V	Множество вершин графа G
$v_1 \leftrightarrow v_2$	Обозначение, означающее, что вершины v_1 и v_2 — смежны (т.е. соединены ребром)
$W(C, R)$	Функция оценки позиции

2 Введение

В данной работе рассматриваются один из частных случаев задачи поиска на графах — игра «полицейские и грабитель». Здесь и далее вместо «полицейские и грабитель» будет использоваться англоязычное сокращение $C\&R$ (cops and robber).

Впервые определение игры $C\&R$ было дано в работе Winkler & Nowakowski [NW83], там рассматривался только случай с одним полицейским, и независимо Alain Quilliot.

В наиболее общем случае $C\&R$ можно описать как класс пошаговых игр на графах, в которых два игрока в каждый момент времени занимают некоторое множество вершин своими фишками, и один из двух игроков (играющий за *полицейских*) своими ходами (передвижениями фишек) «преследует» другого (грабителя) и пытается его «поймать». Количество фишек игроков, их начальная растановка, процесс преследования, правила захвата грабителя и прочее зависит от конкретных правил игры.

Наиболее распространённые правила (и в то же время наиболее простые) можно кратко описать так:

у грабителя 1 фишка, у полицейских — несколько. В начале игры игроки расставляют свои фишки в выбранные ими вершины и далее по очереди передвигают их вдоль рёбер графа, пока грабитель не будет пойман. Если игра не заканчивается за конечное количество ходов, то победа присуждается грабителю. Для избежания путаницы с другими возможными правилами, эта вариация игры называется в работе *простой $C\&R$-игрой*.

Кроме *простой $C\&R$-игры* существует довольно много её разновидностей, однако большинство из них лишь немного дополняют её правила. Вот некоторые примеры модификаций игры $C\&R$:

1. Игра с «быстрыми» грабителями и/или полицейскими (т.е. игрок может передвигать свои фишки вдоль нескольких рёбер за один ход)

2. Игра с неполной информацией (полицейские и грабитель имеют определённый «радиус видимости»)

3. Игра с «радиусом захвата» (полицейские захватывают грабителя на растоянии)

4. «Стреляющий» грабитель (и/или полицейские)

В статье [NS14] содержится более полный список модификаций игры, а также описания их правил и для некоторых из них доказываются содержательные теоремы (в частности там доказывается несколько утверждений про игру с быстрым грабителем).

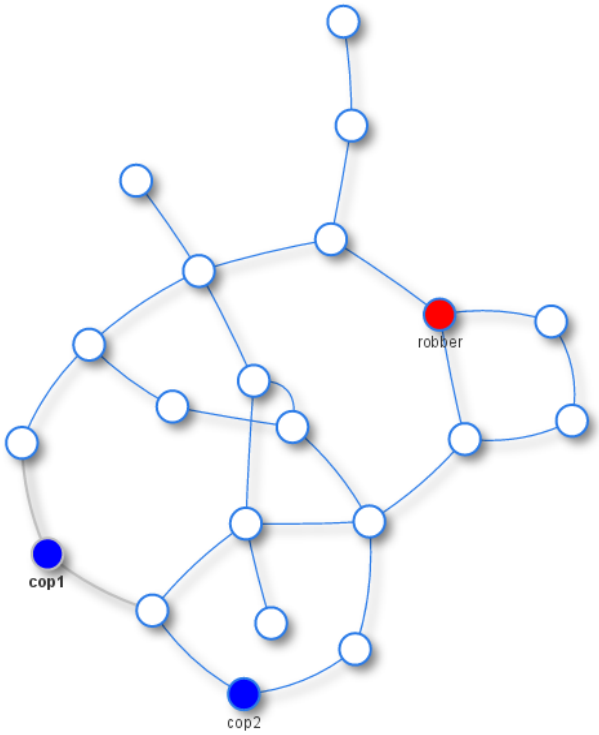

Рис. 1: Пример $C\&R$-игры с двумя полицейскими и одним грабителем на случайно сгенерированном связном неориентированном графе.

В данной работе будет рассматриваться наиболее простая вариация игры — простая $C\&R$-игра.

В этой работе для простой $C\&R$-игры строятся некоторые эмпирические алгоритмы («потенциальный алгоритм») для уклонения и поимки грабителя («сопряженный алгоритм»), а также для них даются некоторые числовые характеристики («эмпирическое полицейское число»).

Построенный в работе «потенциальный алгоритм» не является универсальным решением задачи, однако, он может оказаться пригодным в

некоторых частных случаях. В частности, в работе показывается работоспособность алгоритма в наиболее простых случаях: циклы, деревья, сетки, графы с некоторыми другими характеристиками вершин и минимальных циклов, и т.п. В работе также численными методами строятся некоторые статистики для случайно сгенерированных графов.

В рамках работы разработано приложение, моделирующее поведение полицейских и грабителей. Приложение состоит из трёх частей: основной модуль — `game.js` (программа, написанная на языке JavaScript), и два фронтенда: веб-страничка `index.html`, используемая для наглядной визуализации игры, и модуль для запуска программы в консольном режиме `ng.js`.

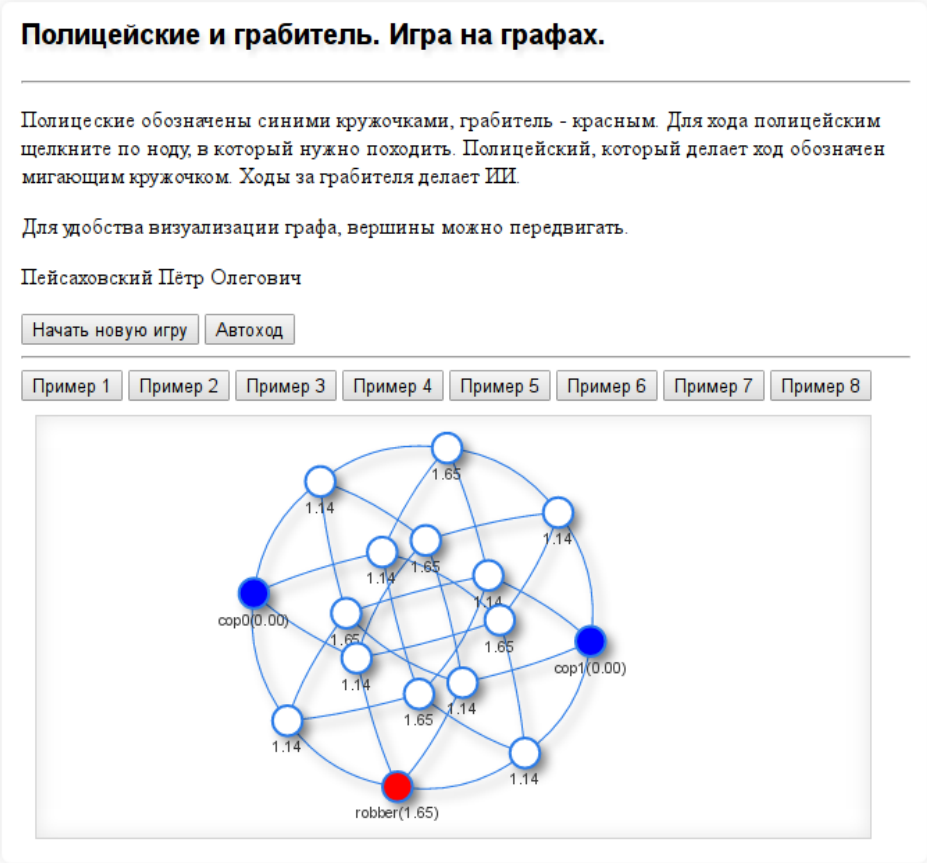

Рис. 2: Пример работы программы: визуализация $C\&R$ на тессеракте.

При запуске в визуальном режиме программа генерирует граф (либо случайный, либо один из выбранных частных случаев) и даёт возможность пользователю делать ходы за полицейского. Ходы за грабителя

делаются по упомянутому выше потенциальному алгоритму. У пользователя есть возможность тестировать «двойственный алгоритм» полицейских, нажимая кнопку «автоход».

В консольном режиме программа запускается посредством утилиты node.js. Консольный режим используется для построения статистик численным методом (т.е. для запуска большого количества игр на случайно сгенерированных графах и подведения статистик).

2.1 Мотивация

На данный момент существуют точные алгоритмы для поимки грабителя в $C\&R$-игре лишь в некоторых частных случаях. Так, например, в [NW83] рассматривается случай одного полицейского и строится необходимое и достаточное условие существования алгоритма для поимки грабителя (которое даёт ключ к построению алгоритма).

В работе [AF84] рассматривается способ поимки грабителя двумя полицейскими с помощью контроля каждым из полицейских одного наикратчайшего пути (на графах, на которых это возможно). Так, например, в случае сетки достаточно двух полицейских: один контролирует строку, другой — столбец. Поочерёдно сдвигая контролируемые столбцы и строки, грабитель прижимается в угол и там захватывается.

Однако, способ предлагаемый в [AF84] не может быть использован в общем случае. Столкнувшись с задачей $C\&R$-игр, мне в первую очередь стало интересно, как могли бы выглядеть алгоритмы уклонения или поимки грабителя, работающие пусть и не точно, но в какой-то степени эффективно на произвольных графах и с произвольным количеством полицейских.

В этой работе строятся эмпирические алгоритмы для построения стратегий поимки и уклонения грабителем.

2.2 Правила игры и постановка задачи

Как уже упоминалось в введении, существует огромное множество вариаций игры «полицейские и грабитель». Дадим детальное описание наиболее простой вариации:

Определение 1. *Простая C&R-игра.*

Пусть, дан связный неориентированный граф $G = G(V, E)$, где V — множество вершин графа, а E — множество его рёбер. В игре два участника. Первый из них играет за полицейских, второй — за грабителя. Для краткости в некоторых случаях для обозначения игроков будут использоваться сокращения $\mathcal{C}$ (cops — полицейские), $\mathcal{R}$ (robber — грабитель). У $\mathcal{C}$ имеется k фишек, а у $\mathcal{R}$ — одна (число фишек у каждого из игроков в течение игры не меняется). От начала и до конца игры оба игрока полностью видят текущее состояние игры, т.е. это игра с полной информацией. Игроки делают ходы по очереди.

В первый ход, основываясь только на графе G, $\mathcal{C}$ расставляет свои фишки $C = (c_1, ..., c_k)$, где $c_1, ..., c_k \in V$ в произвольные вершины графа.

Во второй ход, основываясь на графе G и расположении фишек соперника, $\mathcal{R}$ выбирает произвольную вершину $r \in V$ для своей фишки. Во все последующие ходы игроки двигают свои фишки вдоль рёбер графа. За свой ход $\mathcal{C}$ может подвинуть все свои фишки, а r только свою единственную. Допустимым также считается оставить одну или несколько фишек на месте. Две или более фишек $\mathcal{C}$ могут занимать одну и ту же вершину. Игра заканчивается в момент, когда хотя бы одна из c_i оказывается равной r, т.е. когда один из полицейских оказывается на той же вершине, что и грабитель. Если грабитель был пойман, то победа присуждается полицейским. Если игра не заканчивается за конечное количество ходов, то победа присуждается грабителю.

Определение 2. *Стратегия S_R для $\mathcal{R}$* есть пара $(\mathrm{I_R}, \sigma_R)$, где $\mathrm{I_R} \in V$ начальная позиция грабителя, и $\sigma_R : V^k \times V \to V$ — функция, которая сопоставляет новую позицию грабителя r' каждой конфигурации игры $S = (c_1, ..., c_k, r) \in V^k \times V$.

Определение 3. *Стратегия для* $\mathcal{C}$ — есть пара $S_C = (\mathrm{I_C}, \sigma_C)$, где $\mathrm{I_C} \in V^k$ начальные позиции k полицейских, и $\sigma_C : V^k \times V \to V^k$ функция, которая сопоставляет новые позиции полицейских $(c'_1, ..., c'_k)$ каждой конфигурации игры $S = (c_1, ..., c_k, r) \in V^k \times V$

Определение 4. *Полицейское число*(cop number) $cn(G)$ связного графа есть минимальное количество полицейских необходимое для достоверной поимки $\mathcal{R}$, в независимости от поведения $\mathcal{R}$. *Полицейским числом* несвязного графа будем называть сумму полицейских чисел его компонент связности.

Заметим, что полицейское число является характеристикой графа и не зависит от стратегий игроков.

В данной работе ставится задача построения алгоритмов уклонения для грабителя при $cn(G) > |C|$.

3 Некоторые факты о полицейском числе

Приведём некоторые известные факты о полицейском числе. В этом разделе все утверждения приводятся без доказательств, но при этом ко всем даны ссылки на источник, в котором эти доказательства при необходимости можно найти.

Определение 5. Род графа (см. [NS14])

Род графа — есть минимальный род ориентированного многообразия, на котором граф можно представить без самопересечений (т.е. без пересечений его рёбер). В свою очередь род ориентированного многообразия (или род поверхности) есть такое число n, что эта поверхность гомеоморфна сфере с n ручками.

Утверждение 1. (см. [NS14])
Если G — дерево, то $cn(G) = 1$.

Утверждение 2. (см. [NS14])
Если G — цикл и $|G| > 3$, то $cn(G) = 2$.

Утверждение 3. (см. [NS14])
Для сетки $cn(G) < 3$.

Утверждение 4. (см. [AF84])
Если G — планарный граф, то $cn(G) < 4$.

Утверждение 5. (см. [Sch01])
Для любого графа рода g, $cn(G) \leq \left\lfloor \frac{3}{2}g \right\rfloor + 3$.

На данный момент оценка полицейского числа через род графа является одной из самых эффективных (среди доказанных).

Гипотеза 1. (пока не доказана) (см. [NS14])
Для любого графа рода g, $cn(G) \leq g + 3$.

Утверждение 6. (см. [NS14])
Для любого N существует граф с $cn(G)$ большим N.

Определение 6. Обхват графа (см. [NS14])

Обхватом графа $girth(G)$ назовём минимальную длину его индуцированного цикла.

Утверждение 7. (см. [Fra87a])

Если G — граф с минимальной степенью вершин δ и $girth(G) \geq 5$, тогда $cn(G) \geq \delta$

Гипотеза 2. Гипотеза Мэйнела (см. [BB12])

Если G — граф с минимальной степенью вершин δ и $girth(G) \geq 5$, тогда $cn(G) \geq \delta$

Определение 7. Ловушка (pitfall) [AF84]

Назовём вершину p *ловушкой*, тогда и только тогда, когда $\exists d : N(p) \cup p \subset N(d)$.

Можно сказать, что ловушка — это вершина которая вместе со всеми своими соседями находится «под ударом» из некоторой вершины d.

Теорема 1. Теорема Айгнера-Фрома (о редуцировании графа) [AF84]

Полицейское число графа G равно единице тогда и только тогда, когда поочерёдно отнимая от графа ловушки (и свободные рёбра), граф можно редуцировать до точки.

4 Построение алгоритмов уклонения

В этой секции строятся алгоритмы уклонения и поимки. Под *алгоритмом уклонения* будем подразумевать функцию, сопоставляющую произвольному графу некоторое множество стратегий на этом графе:

$$A_R : G \mapsto \{S_R\}$$

аналогично для $\mathcal{C}$, алгоритм поимки есть отображение вида:

$$A_C : G \mapsto \{S_C\}$$

4.1 Простейшее уклонение методом потенциалов

Одна из наиболее интересных задач, связанных с играми $C\&R$ — это задача поиска *оптимальных* (т.е. приносящих победу игроку на тех графах, на которых это возможно) стратегий. Как показывает практика, задача поиска оптимальной стратегии на достаточно больших графах не поддаётся точному решению в общем случае (пока общего решения дающего результат за короткое время никто не нашел). Полным перебором искать стратегии можно только для очень небольших графов и при не большом количестве полицейских, так как сложность задачи обыстро растёт при увеличении параметров задачи. Например, для графа с заданной минимальной степенью вершины, как не трудно убедиться, количество стратегий оценивается следующим образом:

$$\text{Количество стратегий} \geq (\min_{v \in V} deg(v))^{(|V|^{|C|+1})}$$

Учитывая настолько быстро растущую оценку для количества стратегий, естественно возникает вопрос о поиске эмпирического алгоритма, который давал бы хорошие результаты в среднем или хотя бы в каких-то частных случаях.

Перед тем как строить алгоритм, моделирующий поведение грабителя (или генерирующий стратегию), сделаем несколько замечаний. Во-первых, если грабитель в свой ход решает подвинуть свою фишку из одной вершины в другую, значит позиция после его хода для него *более предпочтительна*. Иначе говоря, в множестве всех позиций игры

существует некоторый частичный порядок (возможно, нестрогий), причем каждой стратегии соответствует свой частичный порядок. Аналогично наоборот, если у нас есть частичный порядок, дающий возможность сравнивать соседние позиции (т.е. такие, что из одной можно перейти в другую ходом грабителя), то такому порядку, очевидно, соответствует некоторая стратегия (в частности, если множество стратегий вполне упорядочено).

Для простоты в работе рассматриваются стратегии, порождаемые полным порядком в множестве позиций.

Итак, пусть у нас есть вполне упорядоченное множество позиций игры. Так как игровых позиций конечное количество, то мы можем сопоставить каждой позиции некоторое число (очевидно, что можно целое, но далее мне будет удобнее использовать вещественные числа). Это число, соответствующее игровой позиции будем называть *оценкой позиции*.

Следующее замечание заключается в том, что на практике для моделирования $C\&R$ (т.е. при создании программы), даже предполагая детерминированность ходов игроков (т.е. то, что они ходят в соответствии с некоторой стратегией), нам вовсе не нужно знать их стратегию целиком, так как далеко не все игровые позиции могут оказать возможны (при фиксированной паре стратегий игроков). Иначе говоря, вместо того, чтобы делать программу, которая строит стратегию, которая как множество будет содержать довольно большое количество элементов, удобнее иметь программу, которая будет вычислять следующий ход, на основе текущей игровой ситуации по ходу игры, подобно тому, как шахматные компьютеры не просчитывают все возможные игровые ситуации, тратя вычислительные мощности лишь на те из них, которые реально произошли в игре.

Определение 8. *Функцией оценки позиции* назовем произвольное отображение из $W : V^k \times V \to \mathbb{R}$ Т.е. отображение, сопоставляющее вещественное число каждой игроков позиции.

Определение 9. *Потенциалом* P_C вершины $v \in V$ при фиксированном расположении полицейских $C = (c_1, ..., c_k)$ будем называть значение функции оценки позиции W на (C, v). Т.е. $P_C(v) = W(C, v)$, при фиксированных $c_1, ..., c_k$.

Определение 10. *Потенциальный алгоритм* (для грабителя)

Пусть рассматривается граф G и функция оценки позиции W. На каждом шаге будем выбирать ход следующим образом:

1. Если делаем первый ход:

   1.1. С помощью функции оценки позиции W считаем потенциалы всех вершин и ходим в вершину с максимальным потенциалом

2. Если делаем не первый ход:

   2.1. С помощью W считаем потенциалы соседних с грабителем вершин $v \in N(r)$ и вершины r

   2.2. Если потенциал r больше потенциалов соседних вершин, то пропускаем ход

   2.3. Иначе — выбираем соседнюю вершину с максимальным потенциалом и делаем ход в неё

Замечание 1. В потенциальном алгоритме, очевидно, присутствует некоторый произвол, связанный с тем, что потенциалы нескольких соседних вершин могут совпадать. Можно сказать, что каждому потенциальному алгоритму соответствует некоторое множество стратегий.

Определение 11. *Алгоритм 1* (для грабителя)

Назовем *Алгоритмом 1* потенциальный алгоритм для которого

$$P_C(v) = \min_{c_i \in C} dist(v, c_i),$$

где $dist(v_1, v_2), v_1, v_2 \in V$ есть расстояние от v_1 до v_2, т.е. число рёбер кратчайшего пути соединяющего эти две вершины и полностью лежащего в графе.

В этом алгоритме потенциал каждой вершины равен расстоянию от этой вершины до ближайшего полицейского.

Теорема 2. Грабитель, использующий «Алгоритм 1» на цикле длины больше 3-х при одном полицейском, не может быть пойман

► Так как длина цикла больше или равна четырём, то потенциал позиции $\mathcal{R}$ после первого хода грабителя будет как минимум 2 (напомним, что первый ход грабителя делается после первого хода полицейского). При этом после первого хода грабителя r будет находится на максимальном удалении от c_1 (по определению потенциального алгоритма). Очевидно, что после любого хода $\mathcal{C}$ вершина с максимальным потенциалом будет либо r либо соседней вершиной и по определению *алгоритма 1* $\mathcal{R}$ займёт её следующим ходом (вершин с максимальным потенциалом может быть две, но это совершенно неважно в данном случае). Таким образом, после хода $\mathcal{R}$ потенциал его текущей позиции всегда будет максимальным. Так как $\mathcal{C}$ за один ход не может уменьшить расстояние до $\mathcal{R}$ больше чем на единицу, то потенциал вершины занимаемой $\mathcal{R}$ всегда будет как минимум равен двум. □

Замечание 2. Перед тем, как доказывать следующую теорему заметим, что всякая сетка содержит простой цикл длины 4, более того, всякое ребро сетки содержится в некотором простом цикле длины 4, и что в сетке нет циклов длины 3.

Теорема 3. Грабитель, использующий «алгоритма 1» на сетке не может быть пойман одним полицейским

► Так как сетка содержит простой цикл длины 4, то, очевидно, что после первого хода грабителя его потенциал будет как минимум равен двум. Далее, пусть после некоторого хода полицейских потенциал r стал равен единице, т.е. r и c_1 соединены ребром. Но в этом случае C и r будут находится в одном цикле длины 4. Пойдем в соседнюю с r вершину v_0 в этом цикле (но не равную c_1). Так как в сетке нет циклов длины 3, то $\mathcal{C}$ не сможет захватить $\mathcal{R}$ следующим ходом. В противном случае получаем, что $c_1 \leftrightarrow R$, $c_1 \leftrightarrow v_0$, $R \leftrightarrow v_0$, т.е. имеем цикл длины три, чего не может быть (где под обозначением $v_1 \leftrightarrow v_2$ подразумеваем, что вершины v_1 и v_2 соединены ребром). Таким образом, потенциал r после каждого хода $\mathcal{R}$ будет не меньше двух, и, следовательно, $\mathcal{R}$ никогда не будет пойман. □

Теорема 4. Грабитель, использующий *алгоритм 1* не может быть пойман на графе с $\min\limits_{v \in V} deg(v) = k, \min\limits_{v \in V} girth(v) > 5$, при количестве полицейских равном k.

► Как показано в [NS14], на таком графе при любой игровой ситуации будет как минимум одна вершина $v \in V$, в которую $\mathcal{R}$ может походить, и $\mathcal{C}$ не сможет захватить его следующим ходом. Отсюда следует, что вершина v, что $\min\limits_{c_i \in C} dist(c_i, v) = P_C(v) > 1$, отсюда следует, что расстояние от грабителя до полицейских всегда будет больше единицы. □

Как показывает практика, простой потенциальный алгоритм далеко не всегда бывает эффективен.

Пример 1. Пример графов, на которых Алгоритм 1 не работает

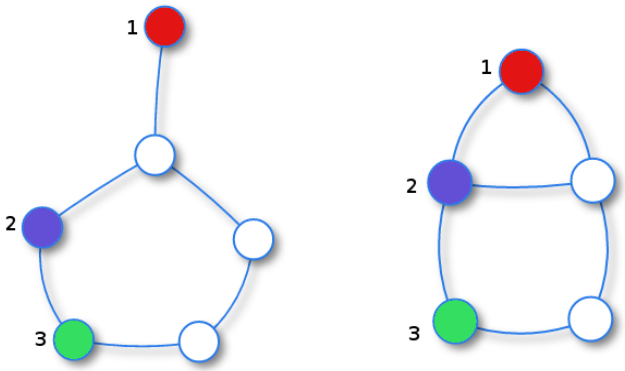

Рис. 3: Пример графов, на которых Алгоритм 1 не работает.

Видно, что если фишка $\mathcal{C}$ будет находиться на вершине 3 (зеленая), то наибольший потенциал будет у вершины 1 (красная) и $\mathcal{R}$, при возможности, её займёт. После того, как $\mathcal{C}$ подвинет свою фишку в вершину 2 (синяя), у $\mathcal{R}$ уже не будет возможности избежать поимки и он будет захвачен следующим ходом.

4.2 Улучшение алгоритма для некоторых частных случаев

Определение 12. Алгоритм 2

Введём обозначения:

$$\Delta_n(v) = \sum_{v_i \in N(v)} EP_n P(v_i),$$

где EN и последовательность EP_n есть некоторый набор вещественных эмпирических коэффициентов.

Алгоритмом 2 назовем потенциальный алгоритм с функцией $P_C(v)$, которая считается следующим образом:

1. На первом шаге присвоим каждой вершине v потенциал $p_1(v) = \min\limits_{c_i \in C} dist(v, c_i)$
2. Повторим следующие действия EN раз (обозначив номер повторения буквой n):

   Назначим новый потенциалы $P_n(v) = P_{n-1}(V) - \Delta(v) + \sum\limits_{v_i \in N(v)} \Delta(v_i)$
3. В качестве конечных потенциалов возьмем полученные значения $P_n(v)$, т.е. $P_C(v) := P_n(v)$

Алгоритм 2 — есть по сути модифицированный алгоритм 1 с «перетеканием потенциалов». Идея алгоритма 2 заключается в том, что если рядом с некоторой вершиной «много» вершин с «большим» потенциалом, то и у неё должен быть потенциал несколько больше, чем у вершины, у которой хоть и тоже расстояние до ближайшего полицейского, но не имеющей соседей с «большими» потенциалами.

Это, казалось бы, небольшое изменение даёт возможность $\mathcal{R}$ не заходить в тупики, в то время как «алгоритм 1» приводит к тому, что $\mathcal{R}$, убегая, непременно заходит в тупик, если есть такая возможность.

Определение 13. Тупик

Назовем вершину v *тупиком* в двух случаях:

1. Если $\deg(v) = 1$
2. Если для вершины v множество соседних не тупиковых вершин содержит не более одного элемента

Замечание 3. Очевидно, тупик является частным случаем ловушки

Обозначим через $G'(V', E')$ граф, который получается из $G(V, E)$ посредством редуцирования всех ловушек. Покажем, что при $k = 1$ и $cn(G) > 1$ для $\mathcal{R}$ существует эффективная стратегия, при которой он никогда не заходит в ловушки (можно сказать, что ходить в ловушки бесмысленно, и $\mathcal{R}$ может передвигаться только по редуцированному графу G').

Теорема 5. Если cn(G) > 1, то существует выигрышная стратегия для $\mathcal{R}$, при которой в каждый момент игры $R \in V'$

► Предположим, что такой стратегии нет. Но тогда $cn(G') = 1$, а значит в G' есть ловушки, что противоречит определению G'. □

Определение 14. *Алгоритм 3*

Алгоритм 3 определим как модификацию Алгоритма 2, в которой после подсчёта потенциалов вершин потенциал каждой вершины домножается на некоторый эмпирический параметр EP_p, в случае, если эта вершина является тупиком.

В случае, если $EP_p = 0$, Алгоритм 3, по сути редуцирует граф, отнимая от него тупики. Такой подход может оказаться эффективным в случае, если $cn(G) > 1$, однако, нужно учитывать, что, если $cn(G) = 1$, то может оказаться, что все вершины графа окажутся тупиками (например, если G — дерево). На практике, чтобы не иметь дела с пустыми графами, бывает удобнее положить EP_p некоторому небольшому числу.

4.3 Сопряженный алгоритм (для полицейских)

Вероятно, самым интересным достижением этой работы является «двойственный алгоритм» для полицейских, который можно построить как только есть какой-то алгоритм для грабителя с оценкой позиции. Идея двойственного алгоритма в том, что каждым своим ходом $\mathcal{C}$ минимизирует оценку позиции после следующего хода $\mathcal{R}$.

Определение 15. *Двойственный алгоритм*

Первый ход двойственного алгоритма определяется следущим образом: Обозначим максимальный потенциал при фиксированной расстановке полицейских через $M(C)$:

$$M(C) = \max_{v \in V} P_C(v).$$

Далее найдем минимум этой функции по всем возможным $C \in V^k$ и сделаем соответствующий ход. Последующие ходы будем делать по следующему принципу:

Пусть $C' = (c'_1, ..., c'_n)$ одно из расположений полицейских такое, что (их может быть несколько):

$$P_{C'}(\sigma_R(C', r)) = \min_{C'' \in N(c)} P_{C''}(\sigma_R(C'', r)),$$

где $C' \in N(C)$, а $N(C)$ — множество возможных конфигураций полицейских после одного их хода. Тогда из позиции C делаем ход в C'.

Замечание 4. Так же как и потенциальный алгоритм, двойственный алгоритм содержит некоторый произвол, связанный с тем, что разные игровые ситуации могут иметь одну и ту же оценку позиции. Можно сказать, что двойственному алгоритму соотвествует некоторое множество стратегий (а не одна).

Теорема 6. Пусть G — дерево, $k = 1$ и $\mathcal{C}$ использует сопряженный к «алгоритму 1» алгоритм, тогда $\mathcal{C}$ побеждает при любом поведении $\mathcal{R}$

► Пусть игроки сделали первые ходы (установили начальные позиции своих фишек на графе). В этом случае r и c_1 будут соединены

единственным путем (так как G — дерево). Очевидно, что следуя алгоритму, сопряженному алгоритму 1 $\mathcal{C}$ будет двигаться по этому пути. Учитывая конечность графа G, а также, что r не сможет попасть в те вершины, в которых побывал $\mathcal{C}$ (опять же в силу отсутствия в графе циклов), получаем, что $\mathcal{R}$ будет захвачен через конечное количество ходов. $\square$

5 Моделированние $C\&R$ на JavaScript

В этой части описывается построенная в ходе работы программа, моделирующая поведение грабителя и полицейских. Для моделирования грабителя используется Алгоритм 3, а для полицейских — двойственный к нему алгоритм.

5.1 Визуальное моделирование

Теоретические выкладки, таблицы и графики об играх на графах выглядят довольно безжизненно без возможности их наглядно представить. Именно поэтому, начиная заниматься этой дипломной работой, я первым делом сделал программу, наглядно представляющую процесс игры. Игра написана на JavaScript и выполненна в формате html веб-странички.

Вероятно, такой выбор инструментов разработки тут же вызовет некоторые вопросы, так как JavaScript довольно редко используется для математических расчётов. На самом деле, перед началом работы я рассматривал и несколько других вариантов, таких как: iPython (для которого есть специализированная среда для математических расчётов Spyder), C++ (на котором написаны утилиты из пакета Graphviz, предоставляющие удобный интерфейс для визуализации графов). Был также вариант с разработкой программы на php с генерацией визуализации графов посредством утилит Graphviz на стороне сервера.

Однако, после ознакомления с JavaScript-фрэймворком vis.js, все прочие варианты резко перестали быть актуальными. JavaScript с фреймворком vis.js как нельзя лучше подходит к задаче визуализации игр на графах, так как, кроме самой визуализации графов, предоставляет богатые возможности в динамике проводить с ними различные манипуляции (изменять структуру графа, цвет и размер вершин, добавлять обработчики различных событий).

Выбор JavaScript-html связки дал возможность сделать кроссплатформенное безопасное (с точки зрения распространения вирусов) приложение, которое (благодаря отсутствию серверной части) может работать на веб-сервере с самым скромным набором возможностей или даже работать локально. Также надо заметить, что по статистике JavaScript

заметно быстрее, чем python (по некоторым оценкам в 5 раз), хотя и уступает по быстродействию C++.

5.2 Консольное приложение

Для запуска консольного приложения понадобится установленная программа Node.js. Кроме того, приложению необходимо наличие библиотеки Math.js. Math.js — это популярная библиотека с открытым исходным кодом, предоставляющая программистам JavaScript широкий спектр математических возможностей. Официальный сайт библиотеки — http://mathjs.org/.

Основной объект игры имеет примерно следующий вид:

```
Game={
    CopsPos : [1,2,3],
    CopsAmount : 3,
    Robber : 5,
    CopToMove : 0,
    Matrix: [....],
    GameOver: false,
    GraphParams: {NodesAmount: 27, Dimensions: 3},

    AutoMoveCops: function(){ ....  },
    MoveCop: function (node){ .... },
    getRobberMove: function(){ ... }
    MoveRobber: function (){ ... },
    IsOnCop: function(r){ ... },
    init: function(N){ ... },
    ...
}
```

Здесь указаны только те члены объекта, которые необходимы при построении статистик. Полный листинг программы я тут не привожу, так как программа имеет размер более тысячи строк кода. Автор заинтересован в совершенствовании приложения, поэтому будет благодарен за любые комментарии отправленные по адресу mailto:peter.peysahovsky@gmail.com.

Поясним смысл полей и методов:

Таблица 1: Список основных полей и методов

Поле/функция	**Пояснение**
CopsPos	Массив номеров вершин, занимаемых $\mathcal{C}$
CopsAmount	Количество полицейских
Robber	Номер вершины, занимаемой грабителем
CopToMove	Номер полицейского, который делает ход (только для визуализации)
Matrix	Матрица инцидентности графа
GameOver	Флаг законченности игры
GraphParams	Параметры генерации графа
AutoMoveCops()	Делает ход за $\mathcal{C}$ двойственным алгоритмом
MoveCop()	Делает ход за $\mathcal{C}$ в указанную вершину
getRobberMove()	Вычисляет ход потенциальным алгоритмом для $\mathcal{R}$
MoveRobber()	Делает ход потенциальным алгоритмом за $\mathcal{R}$
IsOnCop()	Проверяет, оккупируется ли вершина полицейскими
init()	Инициализирует игру

Приведем пример минимальной программы, симулирующую игру на графе:

```
1  math = require("./mathjs/math.js");
2  require("./game.js");
3  Game.Generate = GraphGenerator.cube;
4  Game.GraphParams = {NodesAmount: 27, Dimensions: 3};
5  Game.init();
6  Game.CopsAmount = 2;
7  Game.CopsPos = [0,1];
8  Game.Robber = [5];
9  Game.MoveRobber();
10 Game.AutoMoveCops();
```

Данный пример в 10 строк кода является рабочей программой, которую можно запустить в Node.js. Пример генерирует кубический граф — сетку 3x3x3 и делает 1 ход грабителем и двумя полицейскими.

Если требуется запустить игру на графе другого типа, то в поле Game.Generate нужно записать ссылку на другую функцию, генерирующую граф (а точнее матрицу инцидентности), можно использовать одну

из готовых функций-примеров из объекта GraphGenereator или написать свою.

Проверять победу $\mathcal{C}$ можно с помощью функции `Game.IsOnCop()` (проверяя ей вершину грабителя — `Game.Robber`).

Победа в игре достаётся $\mathcal{R}$ по определению, если игра не заканчивается за конечное количество ходов. На практике это эквивалентно повторению какой-то игровой позиции 2 раза. С целью отслеживания такой ситуации каждой игровой позиции можно сопоставлять уникальный хэш-код, и проверять его повторение в течении игры. Такой подход реализован в консольном приложении для построения статистик `ng.js`.

6 Оценка эффективности пар алгоритмов с помощью эмпирического полицейского числа

Конечно же, после создания программы моделирующей поведение полицейских и грабителей, встает вопрос об оценке эффективности этих алгоритмов. Однако, моделирование поведения игроков не даёт возможности оценивать алгоритмы полицейских и грабителя по отдельности. Тем не менее, многие величины, которые можно оценить программой могут представлять интерес.

Определение 16. *Эмпирическое полицейское число* (для пары стратегий)

Предположим, у нас есть фиксированный граф G и пара фиксированных стратегий для обоих игроков. Обозначим S_C — стратегию $\mathcal{C}$, а S_R — стратегию $\mathcal{R}$. Назовём эмпирическим полицейским числом ($cn'(G)$) для тройки (G, S_C, S_R) минимальное количество полицейских необходимых для поимки грабителя на графе G при использовании стратегий S_C, S_R.

Определение эмпирического полицейского числа позволяет оценивать эффективность пары алгоритмов друг относительно друга при фиксированном графе.

В случае, если у нас есть не стратегии, а алгоритмы генерирующие некоторое множество стратегий, можно построить следующую характеристику пары алгоритмов:

Определение 17. *Среднее эмпирическое полицейское число* (для пары алгоритмов и некоторого множества графов)

Пусть есть некоторое конечное, множество неориентированных графов H. Каждому $G \in H$ алгоритмы (A_R и A_C) игроков сопоставляют некоторое множество стратегий, обозначим их как $A_C(G)$ и $A_R(G)$.

Далее составим множество элементарных событий:

$$\Omega = \{(G, S_C, S_R) \mid G \in H,\ S_C \in A_C(G),\ S_R \in A_R(G)\}$$

Будем считать $(\Omega, 2^{\Omega}, \mathbb{P})$ вероятностным пространством, при этом вероятностную меру $\mathbb{P}$ определим на элементарных событиях как константу равную $1/|\Omega|$. Очевидно, что это возможно для конечного множества элементарных событий.

Определим среднее эмпирическое полицейское число как мат. ожидание эмпирического полицейского числа по всем возможным графам из H. Иначе говоря:

$$cn'(H, A_C, A_R) = \mathbf{E}\, cn'(G, S_C, S_R)$$

В этой формуле происходит одновременно три усреднения, по множеству графов и по стратегиям полицейского и грабителя (напомним, что каждый алгоритм может сопоставлять графу некоторое множество стратегий).

Приведём три примера статистик, которые мне удалось посчитать за исторически короткий срок.

Первые две статистики считались для потенциального и модифицированного двойственного алгоритма. Отличие модифицированного алгоритма состоит в том, что первый ход делается не по алгоритму, а случайно. Такой подход даёт дополнительно возможность проследить зависимость результата игры (победа или поражение) от начального расположения полицейских.

(Исходники со всеми эмпирически подобранными параметрами приложены к работе, так что при желании все результаты можно легко перепроверить (или, изменив параметры, расчитать какие-то другие статистики)).

Итак, первая статистика, которую я посчитал — это проценты побед на n-мерном кубе. Под n-ным кубом я в данном случае подразумеваю граф-сетку 3x3...x3. Усреднение проводилось по стратегиям полицейских.

Таблица 2: Проценты побед на n-мерном кубе для разного количества полицейских (по результатам десяти испытаний)

Размерность	Вершины	$\|C\| = 1$	$\|C\| = 2$	$\|C\| = 3$	$\|C\| = 4$
1	4	100	100	100	100
2	16	0	100	100	100
3	64	0	60	100	100
4	256	0	0	80	100

Аналогичная статистика побед для n-мерного тора выглядит следующим образом.

Таблица 3: Проценты побед на n-мерного тора для разного количества полицейских (по результатам десяти испытаний)

Размерность	Вершины	$\|C\| = 1$	$\|C\| = 2$	$\|C\| = 3$	$\|C\| = 4$
1	4	100	100	100	100
2	16	0	100	100	100
3	64	0	0	100	100
4	256	0	0	0	100

Здесь видно, что начальное положение полицейских не имеет значения. Победа или поражение определяется только количеством полицейских.

Далее было посчитано среднее эмпирическое полицейское число для потенциального и двойственного к нему алгоритмов. Усреднение проводилось по случайно сгенерированным графам.

Способ генерации случайных графов следущий:

1. Запасаемся набором из N вершин (т.е. определяем множество V)
2. Для каждой пары неравных вершин (v_1, v_2) добавляем грань в E с вероятностью p.

Для представленного графика $p = 0.36$.

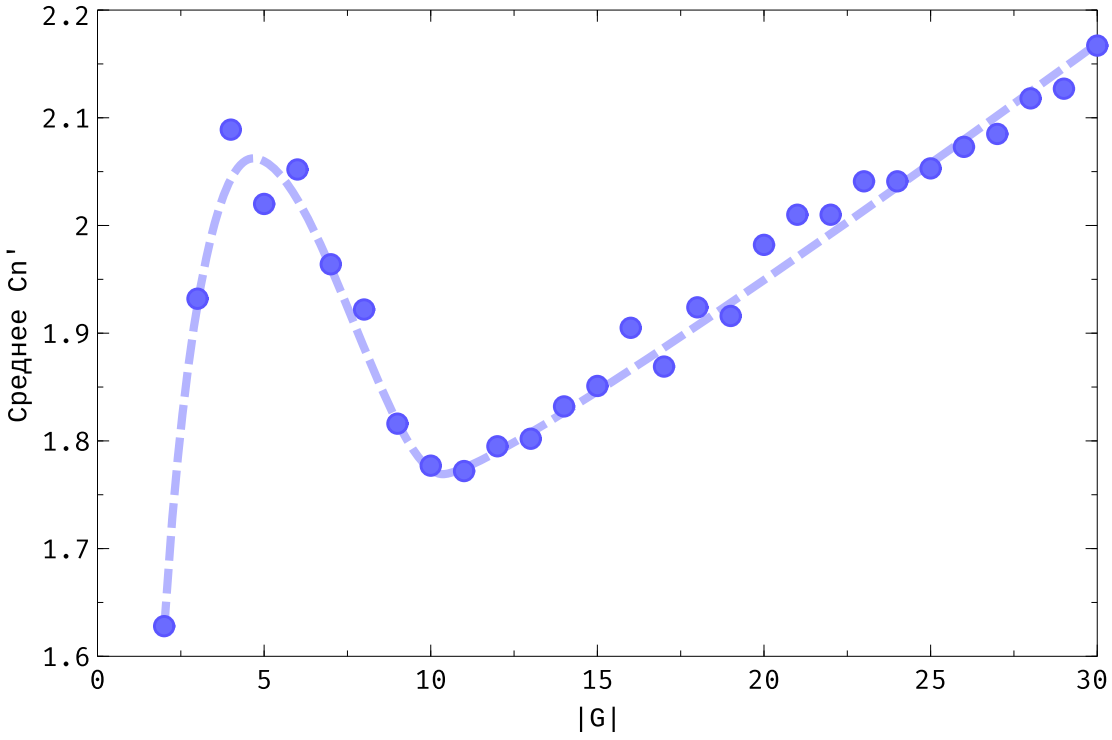

Рис. 4: Среднее эмпирическое полицейское число для случайных графов (по результатам игры на 29 000 графах).

Локальный максимум кривой связан с тем, что при небольшом количестве вершин очень велика вероятность несколько-компонентного графа (т.е. с двумя или более компонентами связности). В случае несколько-компонентного графа с количеством вершин меньше четырёх, полицейское число и эмпирическое полицейское всегда будет равно количеству компонент связности (т.к. $cn(G) = 1$ для любого связного графа с $|G| < 5$).

Из графика видно, что начиная с некоторого момента, среднее эмпирическое полицейское число монотонно растёт, причём зависимость очень похожа на линейную (хотя точно сказать по такому небольшому интервалу сложно). Стоит заметить, что если верить гипотезе Мэйнела [Fra87a], то полицейское число связного графа не может превышать константу умноженную на корень из количества вершин графа. В представленном на графике диапазоне гипотеза Мэйнэла выполняется и для среднего эмпирического полицейского числа (даже без ограничения на

количество компонент связности), однако, чтобы она выполнялась и далее в некоторый момент линейный характер cn'должен обязательно измениться, так как иначе линейная функция рано или поздно «обгонит» $d\sqrt{|G|}$ (где, d — некоторая константа). К сожалению, для существующего алгоритма построение статистики в таком широком диапазоне пока невозможно. Построение такого рода статистик может стать возможно после существенной оптимизации алгоритма или существенного увеличения доступных вычислительных мощностей.

7 Заключение

Результатом данной работы стали пара алгоритмов для грабителя и полицейских. Алгоритмы были протестированы для многомерных графов-сеток. Кроме того, был указан способ «бесплатного» построения алгоритма для полицейских, в случае наличия функции оценки позиции для грабителя (в частности, в случае наличия потенциального алгоритма). В работе вводится понятие «эмпирического полицейского числа», позволяющее давать оценку относительной эффективности пар алгоритмов (или стратегий), что продемонстрированно на примере пар потенциального и дуального алгоритмов.

Другим результатом работы стало приложение, предоставляющее удобный пользовательский и программный интерфейс для экспериментов, связанных с $C\&R$ (а также удобный способ демонстрации). Данное приложение публикуется вместе с этой работой.

Построенные в данной работе программные средства и алгоритмы нельзя воспринимать как конечную цель. Автор видит широкие возможности для дальнейшей работы. Так использованный в процессе построения статистик алгоритм содержит большое количество эмпирических параметров. Эти эмпирические параметры, возможно, имеет смысл находить генетическим алгоритмом или с помощью нейронных сетей. Кроме того, в работе построенны только самые простые статистики, в то время как потенциал программы позволяет строить множество «тонких» статистик.

Список литературы

[AF84] Martin Aigner and Michael Fromme. *A game of cops and robbers*, Discrete Appl. Math. 8 (1984), no. 1, 1–11.

MR 739593 (85f:90124)

[BB12] William Baird, Anthony Bonato *Meyniel's conjecture on the cop number: a survey*, Journal of Combinatorics Volume 0, Number 0 (2012), 1–8

[Fra87a] Peter Frankl *Cops and robbers in graphs with large girth and cayley graphs*, Discrete Applied Mathematics, 17 (1987), 301–305.

[NS14] Nicolas Nisse. *Algorithmic Complexity Between Structure and Knowledge.How Pursuit-evasion Games help*, HDR Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, Sophia Antipolis, France, (May 26th 2014), 1-43

[NW83] Richard J. Nowakowski and Peter Winkler. *Vertex-to-vertex pursuit in a graph. Discrete Mathematics*, 43 (1983), 235–239.

[Sch01] Bernd S. W. Schroder *The copnumber of a graph is bounded by* $\lfloor \frac{3}{2} genus(g) + 3 \rfloor$, Categorical perspectives, Trends in Mathematics (2001), 243–263