УДК 519.174.7

В. Б. Гольдштейн

Кафедра анализа данных факультета инноваций и высоких технологий МФТИ;
ООО «Яндекс»

О проблеме Борсука для $(0,1)$- и $(-1,0,1)$-многогранников в пространствах малой размерности

Изучается классическая гипотеза Борсука о разбиении множеств на части меньшего диаметра. Гипотеза доказывается для $(0,1)$-векторов при $n \leqslant 9$ и для $(-1,0,1)$-векторов при $n \leqslant 6$. Здесь n — это размерность.

Ключевые слова: проблема Борсука, диаметр, алгоритмы раскраски.

1. Введение

В настоящей работе речь пойдет о классической гипотезе Борсука в комбинаторной геометрии. Сперва мы расскажем об истории возникновения и опровержения гипотезы (параграфы 1.1 и 1.2). Затем мы сформулируем основную задачу нашей работы и полученные нами результаты (параграф 1.3). В последнем параграфе текущего раздела мы опишем дальнейшую структуру статьи.

1.1. История возникновения гипотезы Борсука

Рассмотрим произвольное ограниченное неодноточечное множество $V \subset \mathbb{R}^n$. *Диаметром* множества V называется величина

$$\operatorname{diam} V = \sup_{\mathbf{a},\mathbf{b} \in V} \rho(\mathbf{a},\mathbf{b}),$$

где $\rho(\mathbf{a},\mathbf{b})$ — евклидово расстояние между векторами.

Представим V в виде

$$V = V_1 \cup V_2 \cup \ldots \cup V_f,$$

где каждое множество V_i имеет диаметр, строго меньший диаметра V. Это всегда возможно, поскольку любое множество может быть заключено в n-мерный куб со стороной $\operatorname{diam} V$, который в свою очередь можно разделить на конечное число кубиков произвольно малого наперед заданного диаметра. Таким образом, корректно определена величина $f(V)$, равная минимальному числу f частей меньшего диаметра, на которые можно разбить множество V.

Положим

$$f(n) = \max_{V \subset \mathbb{R}^n} f(V).$$

Описанная выше конструкция с покрытием множества кубом и разбиением этого куба на более мелкие кубики позволяет получить оценку $f(n) \leqslant \left(\lceil \sqrt{n} \rceil + 1\right)^n$. В то же время, беря в качестве V множество вершин правильного n-мерного симплекса, мы видим, что $f(V) = n + 1$, и, значит, $f(n) \geqslant n + 1$. Более того, К. Борсук доказал в 1933 году (см. [1]), что $f(B^n) = n + 1$, где B^n — n-мерный шар. Наконец, очевидно равенство $f(1) = 2$, и не очень трудно показать, что $f(2) = 3$. Последний факт также установил Борсук с помощью одного более раннего результата Й. Пала (см. [1] и [2]). В итоге Борсук задал вопрос: *верно ли, что всегда* $f(n) = n + 1$?

Впоследствии большинство специалистов в области верило в то, что ответ на вопрос Борсука должен быть положительным. И потому довольно быстро появился термин «гипотеза Борсука». Естественно, гипотеза гласила, что $f(n) = n + 1$, хотя сам Борсук столь определенных предположений не делал.

Проблема Борсука — это одна из основополагающих задач комбинаторной геометрии. В разное время ей занимались многие известные специалисты. О ней есть книги (см. [3—8]) и обзоры (см. [9—13]), в которых можно найти подробную историю возникновения и развития проблематики. В следующем параграфе мы остановимся на наиболее важном для нас аспекте: речь пойдет о *контрпримерах* к гипотезе Борсука.

1.2. Опровержение гипотезы

В течение шестидесяти лет гипотеза Борсука оставалась ни доказанной, ни опровергнутой. Был получен ряд промежуточных результатов. Например, в 1945 году Г. Хадвигер показал, что любое множество с гладкой границей допускает разбиение на нужное число частей (см. [14]). А на рубеже 1980—90-х гг. О. Шрамм (см. [15]) и Ж. Бурген–Й. Линденштраусс (см. [16]) установили самую лучшую из известных верхних оценок для $f(n)$: $f(n) \leqslant (1.224\ldots + o(1))^n$. С одной стороны, результат Хадвигера кажется чем-то очень близким к доказательству гипотезы: аппроксимируем произвольное множество гладким, и все получится. С другой стороны, экспоненциальная верхняя оценка выглядит слишком далекой от истины, если верить в гипотезу.

И все-таки гипотеза оказалась неверна. В 1993 году Дж. Кан и Г. Калаи построили первый контрпример к ней. Они доказали, что $f(n) \geqslant (1.203\ldots + o(1))^{\sqrt{n}}$ и что эта оценка превосходит $n + 1$, начиная с размерности 2015 (см. [17]). В дальнейшем был получен ряд усилений этого результата. Эти усиления мы укажем в таблице 1.

Таблица 1

Авторы публикации	Год	Ссылка	Размерность контрпримера к гипотезе $n \geqslant$
Дж. Кан, Г. Калаи	1993	[17]	2015
А. Нилли	1994	[18]	946
Б. Вайссбах, Й. Грей	1997	[19]	903
А. М. Райгородский	1997	[20]	561
Б. Вайссбах	2000	[21]	560
А. Хинрихс	2001	[22]	324
О. Пихурко	2002	[23]	323
А. Хинрихс, Х. Рихтер	2003	[24]	298

Вместе с тем известно, что гипотеза верна при $n \leqslant 3$ (см. [1, 25—29]). Наконец, асимптотическая нижняя оценка Кана–Калаи была уточнена А. М. Райгородским (см. [30]) в 1999 году. И теперь мы знаем, что

$$(1.2255\ldots{+}o(1))^{\sqrt{n}} \leqslant f(n) \leqslant (1.224\ldots{+}o(1))^n, \quad f(n) = n{+}1 \quad n \leqslant 3, \quad f(n) > n{+}1 \quad n \geqslant 298.$$

Отметим, что все нижние оценки для $f(n)$, а также практически все контрпримеры к гипотезе Борсука получены с помощью построения множеств $V \subset \{0, 1\}^n$ и $V \subset \{-1, 0, 1\}^n$.

Выпуклые оболочки таких множеств принято называть $(0,1)$-*многогранниками* и $(-1,0,1)$-*многогранниками*. Последние иногда называют еще *кросс-политопами*. Именно им и будет посвящена дальнейшая работа.

1.3. Постановка задачи и формулировки результатов

В 1999 году Г. М. Циглер начал изучение задачи Борсука для (0,1)-многогранников в малых размерностях. Совместно с учениками он показал, что такие многогранники допускают гипотетическое разбиение на $n+1$ часть меньшего диаметра при всех $n \leqslant 9$ (см. [31—35]). Иными словами, если и существуют контрпримеры к гипотезе Борсука в размерностях $n \leqslant 9$, то они не могут быть получены с помощью упомянутых конструкций.

В случае растущей размерности аналогичными задачами занимался А. М. Райгородский (см. [36—40]).

Поскольку разбиение многогранника на части равносильно разбиению множества его вершин (диаметры достигаются только на парах вершин), то мы будем работать только с конечными множествами $V \subset \{0,1\}^n$ и $V \subset \{-1,0,1\}^n$. Дадим несколько определений.

Графом расстояний множества $V \subset \mathbb{R}^n$ *с расстоянием* d называется граф $G_d(V)$, вершинами которого являются элементы множества V, а ребрами соединены точки на расстоянии d. Более формально, граф $G_d(V)$ есть пара, состоящая из множества вершин V и множества ребер

$$E = \{\{\mathbf{a},\mathbf{b}\}|\ \mathbf{a} \in V,\ \mathbf{b} \in V,\ \rho(\mathbf{a},\mathbf{b}) = d\}.$$

Далее, назовем *графом расстояний множества* V *с расстоянием в отрезке* $[l,r]$ граф $G_{[l,r]}(V)$, вершинами которого являются элементы множества V, а ребрами соединены точки на расстоянии из отрезка $[l,r]$. Более формально, граф $G_{[l,r]}(V)$ есть пара, состоящая из множества вершин V и множества ребер

$$E = \{\{\mathbf{a},\mathbf{b}\}|\ \mathbf{a} \in V,\ \mathbf{b} \in V,\ \rho(\mathbf{a},\mathbf{b}) \in [l,r]\}.$$

Вместе с тем под *графом диаметров множества* V мы понимаем граф расстояний с расстоянием, равным величине $\operatorname{diam} V$. Будем обозначать граф диаметров через $G_{\operatorname{diam} V}(V)$.

Наконец, *хроматическое число* $\chi(G)$ *графа* G — это минимальное число цветов, в которые можно так покрасить вершины графа G, чтобы концы любого ребра имели разные цвета.

Очевидно, что для конечного множества $V \subset \mathbb{R}^n$ выполнено $f(V) = \chi\left(G_{\operatorname{diam} V}(V)\right)$. Таким образом, в новых терминах гипотеза Борсука говорит о том, что $\chi\left(G_{\operatorname{diam} V}(V)\right) \leqslant n+1$. А результаты группы Циглера сводятся к тому, что $\chi\left(G_{\operatorname{diam} V}(V)\right) \leqslant n+1$ для всех $V \subset \{0,1\}^n$ и для всех $n \leqslant 9$.

Про случай $(-1,0,1)$-многогранников ничего известно не было.

Сформулируем основные результаты настоящей работы.

Теорема 1. *Для любого* $n \leqslant 9$ *и для всех* $V \subset \{0,1\}^n$ *выполнено* $\chi\left(G_{\operatorname{diam} V}(V)\right) \leqslant n+1$.

Теорема 2. *Для любого* $n \leqslant 6$ *и для всех* $V \subset \{-1,0,1\}^n$ *выполнено* $\chi\left(G_{\operatorname{diam} V}(V)\right) \leqslant n+1$.

Теорема 2 принципиально новая, а теорема 1 опирается на принципиально новый подход, хотя ее результат и совпадает с ранее доказанным. Более того, именно подход из теоремы 1 позволяет обосновать теорему 2.

Упомянутый выше новый подход основан на построении алгоритма, который позволяет осуществить значительное сокращение перебора на множестве всех графов диаметров,

которые нас интересуют. Сложность в том, что полный перебор требует порядка 10^{150} — 10^{250} операций, что необозримо. Нам удается этого избежать. В последующих разделах мы докажем теоремы 1 и 2, описав соответствующий алгоритм.

1.4. Дальнейшая структура статьи

Оставшаяся часть работы состоит из следующих разделов: «Описание алгоритма», «Доказательства лемм» и «Работа программы».

В разделе «Описание алгоритма» будет в подробностях изложен алгоритм, с помощью которого мы доказываем гипотезу Борсука в малых размерностях. Также будут приведены псевдокоды основных функций, использованных в алгоритме. В этом же разделе по ходу описания будут сформулированы все леммы, на которых основаны производимые в алгоритме действия.

Доказательства всех фактов вынесены в раздел «Доказательства лемм», чтобы не отвлекать от самого алгоритма.

В разделе «Работа программы» будут приведены результаты запуска программы. Также будет приведена информация о времени работы программы и его изменении при применении так называемых отсечений, описанных в алгоритме.

2. Описание алгоритма

Этот раздел мы разобьем на несколько параграфов. В параграфе 2.1 мы скажем общие слова об устройстве нашего алгоритма. В параграфе 2.2 мы поговорим об отсечениях перебора и их роли в работе алгоритма. В последующих параграфах мы опишем все использованные нами отсечения.

2.1. Устройство алгоритма

Если говорить совсем грубо, нам нужно перебрать все подмножества V' множества $\{0,1\}^n$ или все подмножества V' множества $\{-1,0,1\}^n$ и для каждого из таких подмножеств проверить, что $\chi\left(G_{\operatorname{diam} V'}(V')\right) \leqslant n+1$. Разумеется, полный перебор здесь невозможен. Впоследствии, рассматривая так называемые отсечения (которые и помогут нам значительно сократить перебор), мы, в частности, убедимся в том, что не всегда следует пробегать $\mathbf{V} = \{0,1\}^n$ или $\mathbf{V} = \{-1,0,1\}^n$ целиком: зачастую достаточно взять некоторое собственное подмножество $V \subset \mathbf{V}$ и только из него извлекать в свою очередь подмножества V'. Поэтому ниже мы будем использовать обозначение V вместо $\mathbf{V}$.

Далее извлечение V' из V мы организуем как наращивание множества V' путем последовательного добавления в строящееся множество одной вершины за другой. И иногда мы будем прерывать процесс за счет различных соображений, которые мы будем называть *отсечениями перебора*. Так или иначе это будут соображения, которые показывают, что при дальнейшем увеличении множества V' проверяемое нами свойство $\chi\left(G_{\operatorname{diam} V'}(V')\right) \leqslant n+1$ заведомо выполняется.

Напомним, что *подграфом* графа $G = (V, E)$ называют такой граф $G' = (V', E')$, что $V' \subseteq V$ и

$$E' \subseteq \{\{u, v\} \in E | u \in V',\ v \in V'\}.$$

А *индуцированным подграфом* графа $G = (V, E)$ (индуцированным на множество $V' \subseteq V$) называют подграф, содержащий все ребра из E, оба конца которых принадлежат V', т.е. $G[V'] = (V', E')$, где

$$E' = \{\{u, v\} \in E | u \in V', v \in V'\}.$$

Также граф $G = (V, E)$ называют *пустым*, если $E = \varnothing$.

Введем некоторые обозначения.

- $C = n + 1$ — максимальное количество цветов, которое можно использовать при раскраске графа диаметров.
- d — диаметр подмножеств $V' \subseteq V$.
- $F_d = G_{[d+1,\operatorname{diam} V]}(V)$ — *граф запретов*. В этом графе ребрами соединены вершины, которые не могут одновременно входить в множество $V' \subseteq V$. При упомянутом выше процессе наращивания множества V' мы будем следить за тем, чтобы очередная вершина не образовала ребро в графе запретов с какой-либо из уже выбранных вершин.
- $H_d = G_d(V)$ — *проверяемый граф*. Это граф, который содержит все графы диаметров множеств $V' \subseteq V$. Так, для фиксированного V' с $\operatorname{diam} V' = d$ граф диаметров $G_{\operatorname{diam} V'}(V')$ равен индуцированному подграфу $H_d[V']$.

В новых обозначениях утверждение $\operatorname{diam} V' = d$ эквивалентно следующим двум условиям:

1. $F_d[V']$ — пустой граф;
2. $H_d[V']$ — не пустой граф.

Когда значение d будет предполагаться фиксированным, мы будем обозначать проверяемый граф и граф запретов просто как H и F соответственно.

Отныне описание алгоритма мы будем вести, излагая ключевые функции алгоритма в виде псевдокода. Все технические подробности мы будем опускать.

Для каждой функции будут описаны следующие составляющие:

- название функции;
- входные параметры, с которыми работает функция;
- возвращаемое значение — результат работы функции;
- описание действий, выполняемых функцией.

Для примера, запишем самые общие функции 2.1.1 и 2.1.2 (и равносильную ей 2.1.3), призванные доказать теоремы 1 и 2.

2.2. Отсечения перебора

Мы уже знаем, что основная функция алгоритма занимается перебором всех подмножеств $V' \subseteq V$, где в свою очередь $V \subseteq \mathbf{V}$. Как было сказано в начале параграфа 2.1, перебор производится путем последовательного добавления в строящееся множество одной вершины за другой. И при добавлении очередной вершины всякий раз нужно, естественно, рассматривать один из двух вариантов:

1. добавить вершину в множество V';
2. не добавлять вершину в множество V'.

Функция 2.1.1 Проверка гипотезы Борсука

procedure CHECKHYPOTHESIS(n, V)

Входные параметры: размерность пространства n и множество V.

Выходные значения: булево значение 0 или 1.

end procedure

Описание действий. Данная функция просто разделит задачу на несколько подзадач. Для всех $d \in [1, \operatorname{diam} V]$ с помощью функции «Проверка гипотезы для заданного диаметра» она установит, верно ли, что для любого $V' \subseteq V$ с $\operatorname{diam} V' = d$ выполнено $\chi(G_{\operatorname{diam} V'}(V')) \leqslant C$.

Если хотя бы для одного d неравенство не выполнено, то гипотеза неверна и мы возвращаем 0; иначе возвращаем 1.

Далее будем использовать более короткий псевдокод.

procedure CHECKHYPOTHESIS(n, V)
for $d = 1$ to $\operatorname{diam} V$ **do**
if не верно **«Проверка гипотезы для заданного диаметра»**(n, V, d) **then**
return 0
end if
end for
return 1
end procedure

Функция 2.1.2 Проверка гипотезы для заданного диаметра

procedure CHECKHYPOTHESISFORDIAM(n, V, d)

Входные параметры: размерность пространства n, множество V и d.

Выходные значения: булево значение 0 или 1. Возвращаем 1, если для любого подмножества $V' \subseteq V$ с $\operatorname{diam} V' = d$ выполнено $\chi(G_d(V')) \leqslant C$.

end procedure

Функция 2.1.3 Проверка возможности раскраски

procedure CHECKCOLORING(H, F, C)

Входные параметры: H — проверяемый граф, F — граф запретов, C — максимальное количество цветов. Множества вершин у графов H и F совпадают и равны V.

Выходные значения: булево значение 0 или 1. Возвращаем 1, если для любого подмножества $V' \subseteq V$ выполнены следующие условия:

1. $F[V'] = (V', \varnothing)$, а $H[V']$ не пуст.
2. $\chi(H[V']) \leqslant C$.

end procedure

Отсечения, о которых мы подробнее скажем в следующих параграфах, помогают осуществлять выбор из указанных вариантов. Благодаря им для большинства вершин имеет место только один вариант. Более того, благодаря им же для некоторых вершин и вовсе вариантов нет: они даже не попадают в V (и именно за счет этого мы сразу заменяем **V** на V).

Каждое отсечение будет выглядеть как некоторый отдельный модуль. Обязанностями этого модуля будут:

1. обработка сигналов о совершаемых действиях, предпринимаемых перебором;

2. сообщение перебору о необходимости продолжения;

3. сообщение перебору о выполнении некоторых действий.

Иными словами, каждый модуль будет следить за некоторыми аспектами, по которым сможет сделать вывод о том, что продолжение перебора не требуется.

Кроме этого, модуль может запросить у перебора выполнение следующих действий:

1. включить вершину v в множество V';

2. не включать вершину v в множество V'.

Каждый модуль может за один раз запросить множество таких действий.

Функция 2.2.1 будет выполнять перебор и исполнять роль менеджера между модулями отсечений.

2.3. Отсечение пустоты подграфа F_d

Отсечение следит за тем, чтобы в генерируемое множество не попали две смежные в графе F_d вершины.

При получении информации о добавлении в множество новой вершины v отсечение проверяет, что в In нет вершин, смежных с v в F_d. Также отсечение сообщает, что все смежные с v вершины должны немедленно отправиться в Out.

Это очень простое отсечение существенно сокращает количество вариантов в одной из двух ветвей перебора. Когда перебор рассматривает случай добавления вершины в множество In, от трех до двадцати вершин в зависимости от размерности автоматически попадают в множество Out.

2.4. Отсечение максимальности по включению

Это отсечение следит за тем, чтобы генерируемое множество было максимальным по включению, т.е. чтобы в него нельзя было добавить ни одной вершины v, не нарушив свойства пустоты подграфа F_d. Дело в том, что для подтверждения гипотезы Борсука достаточно рассматривать только максимальные по включению графы, так как выполнена следующая очевидная лемма 1.

Лемма 1. *Пусть граф G является подграфом графа H. Тогда $\chi(G) \leqslant \chi(H)$.*

Вершину будем называть *потенциально* из In, если она не принадлежит Out. Положим $Potential = \{v : \ v \in V \setminus Out\}$.

При поступлении информации о том, что вершина v добавляется в Out, отсечение проверяет, что v смежна в F_d с вершиной потенциально из In. Если это условие не выполнено, то по окончании перебора в множество In будет возможно добавить вершину v, а следовательно, In не будет максимальным по включению. Поэтому продолжение перебора не требуется. Информация о том, что вершина добавляется в In, никак не используется.

Это отсечение (равно как и предыдущее) очень просто в реализации и почти не требует вычислительных ресурсов. Однако количество отсеченных ветвей вполне существенно.

2.5. Отсечения, связанные с симметрией

Этот тип отсечений основан на простой мысли, что симметричные подмножества порождают изоморфные графы. Поэтому достаточно раскрашивать только одно из таких подмножеств. Реализация этих отсечений требует некоторого компромисса между количеством отсеченных веток перебора и алгоритмической сложностью проверки симметрии.

Функция 2.2.1 Основной перебор

procedure MainBruteForce($H, F, C, Cuts, In, Out$)

Входные параметры: H — проверяемый граф, F — граф запретов, C — максимальное количество цветов, $Cuts$ — множество отсечений перебора.

Выходные значения аналогичны значениям функции 2.1.3.

Определим переменные, которые будут модифицироваться в процессе принятия решений о взятии или не взятии вершин.

$In \leftarrow \varnothing$ — множество вершин, которые должны входить в V'.

$Out \leftarrow \varnothing$ — множество вершин, которые не должны входить в V'.

$Unknown \leftarrow V$ — множество вершин, для которых еще не определено, в какое из множеств In или Out они попадут.

Данная функция рекурсивная. Будем подразумевать, что In и Out передаются как параметры. Выше же описаны начальные их значения. Мы не будем отдельно поддерживать множество $Unknown$, так как оно всегда может быть вычислено как $V \setminus In \setminus Out$.

if $Unknown = \varnothing$ **then return** $\chi(H[In]) \leqslant C \ \ \& \ \ F[In] = (In, \varnothing)$

end if

$v \in Unknown$ — произвольный элемент множества.

Для вершины v необходимо рассмотреть два варианта: поместить ее в множество In или в множество Out.

Добавление в множество In.

$In \leftarrow In \cup \{v\}$

Сообщить всем отсечениям из $Cuts$, что вершина v добавлена в множество In.

if хотя бы одно отсечение сообщило о прекращении перебора **then**

    перейти к рассмотрению следующего варианта.

end if

Поместить все действия, которые сообщили модули отсечений, в очередь $Events$.

while $Events \neq \varnothing$ **do**

    Выполнить действие, указанное в начале очереди.

    Удалить выполненное действие из очереди.

    Сообщить всем отсечениям из $Cuts$ о совершенном действии.

    if хотя бы одно отсечение сообщило о прекращении перебора **then**

        перейти к рассмотрению следующего варианта.

    end if

    Добавить все вновь сообщенные действия от модулей в очередь $Events$.

end while

if не $MainBruteForce(H, F, C, Cuts, In, Out)$ **then return** 0

end if

Добавление в множество Out выполняется аналогично.

Оба варианта либо были отсечены, либо при дальнейшем рассмотрении вернули 1, поэтому **return** 1

end procedure

2.5.1. Изометрическое преобразование

Определение 1. *Пусть, как и прежде, $\rho(\mathbf{a},\mathbf{b})$ — расстояние между точками $\mathbf{a}$ и $\mathbf{b}$. Рассмотрим два множества точек $V_1' = \{\mathbf{g}_1,\ldots,\mathbf{g}_m\}$ и $V_2' = \{\mathbf{h}_1,\ldots,\mathbf{h}_m\}$. Если существует такая биекция $f:\ V_1' \longleftrightarrow V_2'$, что всегда $\rho(\mathbf{a},\mathbf{b}) = \rho(f(\mathbf{a}),f(\mathbf{b}))$, то будем говорить, что V_1' изометрично V_2'.*

Справедлива следующая очевидная лемма.

Лемма 2. *Если V_1' и V_2' изометричны, то для любых l,r, удовлетворяющих неравенству $l \leqslant r$, граф $G_{[l,r]}(V_1')$ изоморфен графу $G_{[l,r]}(V_2')$.*

Благодаря этому утверждению достаточно рассматривать только одно множество V' из каждого класса изометричных множеств. Ниже мы скажем, какое именно.

Определение 1. *Введем лексикографический порядок на множестве всех подмножеств V' данного множества V. При этом элементы множества V мы считаем занумерованными, т.е. само множество V заранее упорядочено — например, лексикографически. Пусть, стало быть, $V_1' = \{\mathbf{a}_1,\mathbf{a}_2,\ldots,\mathbf{a}_l\}$ и $V_2' = \{\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_m\}$ — подмножества множества V, причем $\mathbf{a}_1 < \mathbf{a}_2 < \ldots < \mathbf{a}_l$ и $\mathbf{b}_1 < \mathbf{b}_2 < \ldots < \mathbf{b}_m$. Будем говорить, что $V_1' < V_2'$, если существует такой номер i, что $\mathbf{a}_1 = \mathbf{b}_1,\ldots,\mathbf{a}_i = \mathbf{b}_i, \mathbf{a}_{i+1} < \mathbf{b}_{i+1}$.*

В процессе перебора мы для каждого класса изометричных множеств будем рассматривать только минимальное в лексикографическом порядке множество в этом классе.

Далее, имеет место следующее простое утверждение.

Лемма 3. *Пусть множества V_1', V_2' изометричны и $V_1' < V_2'$. Пусть $V_2' = \{\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_m\}$. Рассмотрим произвольное множество $V_2'' = \{\mathbf{b}_1,\mathbf{b}_2,\ldots,\mathbf{b}_m,\mathbf{b}_{m+1},\ldots,\mathbf{b}_{m+t}\}$, полученное в результате пополнения множества V_2'. Тогда существует множество V_1'', которое изометрично множеству V_2'' и лексикографически меньше него.*

Лемма 3 позволяет прекратить перебор, как только обнаружится, что текущее множество In лексикографически больше некоторого изометричного ему множества In'.

В следующих двух пунктах мы рассмотрим два вида изометрии, которые мы будем использовать для наших отсечений.

2.5.2. Отсечение изометрии отражения

Один из примеров биекции f, которая сохраняет расстояние, — это отражение. В следующих леммах, поясняющих, что мы имеем в виду под отражением, мы отдельно рассмотрим случаи $\mathbf{V} = \{0,1\}^n$ и $\mathbf{V} = \{-1,0,1\}^n$. При этом в последнем случае мы для удобства осуществим трансляцию и станем говорить о $\mathbf{V} = \{0,1,2\}^n$.

Лемма 4. *Пусть $\mathbf{a} = (a_1,\ldots,a_n)$, $a_i \in \{0,1\}$. Тогда отображение $f_{\mathbf{a}}$, переводящее каждый $\mathbf{x} = (x_1,\ldots,x_n)$, $x_i \in \{0,1\}$, в $((a_1+x_1) \mod 2,\ldots,(a_n+x_n) \mod 2)$, является изометрией для любого $V' \subseteq \{0,1\}^n$.*

Лемма 5. *Пусть $\mathbf{a} = (a_1,\ldots,a_n)$, $a_i \in \{0,1\}$. Далее, пусть $a \in \{0,1\}$, $x \in \{0,1,2\}$. Положим*

$$\begin{cases} d(0,x) = x, \\ d(a,1) = 1, \\ d(1,x) = 2-x,\ x \in \{0,2\}. \end{cases}$$

Тогда отображение $f_{\mathbf{a}}$, переводящее каждый $\mathbf{x} = (x_1,\ldots,x_n)$, $x_i \in \{0,1,2\}$, в $(d(a_1,x_1),\ldots,d(a_n,x_n))$, является изометрией для любого $V' \subseteq \{0,1,2\}^n$.

Функция 2.5.1 Проверка существования отражения в $\mathbf{V} = \{0,1\}^n$

```
1: procedure CheckReflection01(A)
2:     A — проверяемое множество.
3:     Выходные значения: 1 — если существует лексикографически меньшее B, полученное отражением; 0 — иначе.
4:     if 0 ∉ A then
5:         Симметрия f_a при любом a ∈ A переведет A в множество B, содержащее 0. Следовательно, B < A.
6:         return 1
7:     else
8:         Если множество B симметрично множеству A и лексикографически меньше него, то оно должно содержать 0. Но только симметрия f_a с каким-либо a ∈ A переводит A в множество с 0.
9:         Таким образом, мы свели задачу к рассмотрению стольких симметрий, сколько есть элементов в множестве A: f_a, где a ∈ A.
10:        for a ∈ A do
11:            if f_a(A) < A then return 1
12:            end if
13:        end for
       return 0
14:    end if
15: end procedure
```

Приведем алгоритм, который для данного множества $A = In$ определяет, существует ли множество $B = In'$, лексикографически меньшее A и полученное из A отражением. Реализации для $\mathbf{V} = \{0,1\}^n$ и $\mathbf{V} = \{0,1,2\}^n$ приведены в функциях 2.5.1 и 2.5.2 соответственно.

2.5.3. Отсечение изометрии перестановки координат

Имеет место следующая очевидная лемма.

Лемма 6. *Пусть $p = \{p_1, \ldots, p_n\}$ — перестановка чисел от 1 до n и $\mathbf{x} = (x_1, \ldots, x_n)$, $x_i \in \{0,1\}$. Тогда отображение $f_p(\mathbf{x}) = (x_{p_1}, \ldots, x_{p_n})$ является изометрией для любого множества точек V'.*

Приведем приближенный алгоритм, который для данного множества A определяет, существует ли множество B, лексикографически меньшее A и полученное из A перестановкой координат. Реализация записана в функции 2.5.3.

Для иллюстрации приведем пример, в котором множество не является лексикографически минимальным, однако не детектируется ни одним из алгоритмов:

000000000
000001111
000010000

В этом примере во всех отрезках биты упорядочены. Симметрия отражения также не забракует этот пример. Чтобы обнаружить этот случай, а равно и многие другие случаи, введем дополнительную проверку — функцию 2.5.4.

Функция 2.5.2 Проверка существования отражения в $\mathbf{V} = \{0,1,2\}^n$

procedure CHECKREFLECTION012(A)
    A — проверяемое множество.
    Выходные значения: 1 — если существует лексикографически меньшее B, полученное отражением; 0 — иначе.
    $minElement \leftarrow min(A)$
    if $2 \in minElement$ **then**
        Если в минимальном элементе множества A присутствует координата 2, то симметрия $f_{(0,\ldots,1,\ldots,0)}$ дает лексикографически меньшее множество B. **return** 1
    else
        Тогда минимальный элемент множества B, без ограничения общности, не имеет координаты 2. Для достижения приемлемой скорости работы мы не будем рассматривать все возможные изометрии, которые приводят к отсутствию координаты 2, а ограничимся только отражениями $f_{P(\mathbf{a})}$, где $\mathbf{a} \in A$, а $P(\mathbf{a}) = \mathbf{v}$ — точка, координаты которой определяется следующим образом:

$$v_i = \begin{cases} 0, & a_i \in \{0,1\}, \\ 1, & a_i = 2. \end{cases}$$

        Пойдя на такой шаг, мы уже не гарантируем, что для любого множества мы корректно определим, минимально ли оно лексикографически. Однако такая проверка покрывает большое количество симметрий. Данная проверка используется в отсечениях, поэтому если функция не находит симметричного множества, то перебор продолжится. Такого рода ошибки могут только замедлить алгоритм, однако он по-прежнему будет работать верно.
        Таким образом, мы свели задачу к рассмотрению $|A|$ симметрий.
        for $\mathbf{a} \in A$ **do**
            if $f_{P(\mathbf{a})}(A) < A$ **then return** 1
            end if
        end for
        return 0
    end if
end procedure

Функция 2.5.3 Проверка существования перестановки в $\mathbf{V} = \{0,1,2\}^n$

procedure CHECKPERMUTATION012(A)

A — проверяемое множество.

Выходные данные: 1 — если существует лексикографически меньшее B, полученное перестановкой; 0 — иначе.

$minElement \leftarrow min(A)$

if биты в $minElement$ не упорядочены **then**

$minElement$ имеет вид, отличный от $(0,0,\ldots,0,1,\ldots,1,2,\ldots,2)$. Но тогда изометрия, упорядочивающая биты этого минимального элемента, дает лексикографически меньшее множество B.

return 1

else

Минимальный элемент множества B, изометричного множеству A, должен быть также с упорядоченными битами; иначе существует $B' < B$. Это не определяет симметрию однозначно, так как одинаковые биты могут переставляться между собой произвольным образом.

Пусть $\mathbf{e}_1$ — минимальный элемент множества A, т.е. $\mathbf{e}_1 = (0,0,\ldots,0,1,\ldots,1,2,\ldots,2)$. При этом нулей l_0, единиц l_1, а двоек l_2. Тогда, не изменяя $\mathbf{e}_1$, можно произвольным образом переставлять элементы в трех отрезках:

$ORDER\_SEGMENTS = [0, l_0),\ [l_0, l_0 + l_1),\ [l_0 + l_1, l_0 + l_1 + l_2)$.

for $i \in [2, |A|]$ **do**

if в $\mathbf{e}_i$ хотя бы в одном из отрезков $ORDER\_SEGMENTS$ элементы не упорядочены **then**

существует симметрия, преобразующая $\mathbf{e}_i$ в лексикографически меньший вектор и не изменяющая при этом все предыдущие $\mathbf{e}_j$, где $j < i$.

return 1

end if

Предыдущее условие не выполнилось, поэтому каждый отрезок упорядочен, т.е. сам имеет вид $0,0,\ldots,0,1,\ldots,1,2,\ldots,2$. Разделим каждый отрезок из $ORDER\_SEGMENTS$ на три соответствующих отрезка постоянства.

end for

return 0

end if

Стоит заметить, что алгоритм проверяет лишь необходимое условие существования изометрии. Это некоторый компромисс между скоростью и точностью алгоритма, так как проверять все перестановки крайне не эффективно. При использовании битовых операций для определения упорядоченности и нахождения отрезков время работы алгоритма — $O(k)$, где k — количество битов в каждом числе.

end procedure

Функция 2.5.4 Проверка «изометрии количества» в $\mathbf{V} = \{0,1,2\}^n$

procedure CHECKIZOMETRIANUMBERS012(A)
A — проверяемое множество.
Выходные данные: 1 — если существует лексикографически меньшее B, полученное перестановкой; 0 — иначе.
Пусть $A = [\mathbf{e}_1 < \mathbf{e}_2 < \ldots < \mathbf{e}_k]$.
Введем обозначение $num(\mathbf{e})$ ($\mathbf{e} \in A$) для тройки (k_0, k_1, k_2), в которой k_0 — количество нулей, k_1 — количество единиц, k_2 — количество двоек в векторе $\mathbf{e}$.
Далее будем использовать лексикографическое сравнение для этих троек.
if $num(\mathbf{e}_1)$ не минимально **then**
return 1
Перестановка, упорядочивающая биты элемента с минимальной тройкой num, преобразует A в меньшее множество.
else
Далее будем действовать так же, как и в функции 2.5.3, но на каждом отрезке будем проверять $num(\mathbf{e}_i)$.
Пусть $\mathbf{e}_1$ имеет вид $(0,0,\ldots,0,1,\ldots,1,2,\ldots,2)$. Первые l_0 элементов равны 0, следующие l_1 — 1, последние l_2 — 2.
Тогда, не изменяя вектор $\mathbf{e}_1$, можно произвольным образом переставлять элементы в трех отрезках:
$ORDER\_SEGMENTS = [0, l_0)$, $[l_0, l_0 + l_1)$, $[l_0 + l_1, l_0 + l_1 + l_2)$.
for $i \in [2, |A|]$ **do**
Пусть $num_{[l,r]}(\mathbf{e}_i)$ — тройка из количеств нулей, единиц и двоек у вектора на отрезке $[l, r]$.
if Для некоторого $j > i$ и $[l, r] \in ORDER\_SEGMENTS$ выполнено $num_{[l,r]}(\mathbf{e}_i) > num_{[l,r]}(\mathbf{e}_j)$ **then**
существует изометрия, преобразующая $\mathbf{e}_j$ в вектор, лексикографически меньший $\mathbf{e}_i$, и при этом не изменяющая все предыдущие $\mathbf{e}_t$, где $t < i$.
return 1
end if
Как и в 2.5.3, разделим каждый отрезок из $ORDER\_SEGMENTS$ на три части.
end for
return 0
end if
end procedure

2.6. Отсечение раскраски графа

Целью перебора является проверка всех подмножеств множества V на возможность раскрасить графы, индуцированные на них, в $n+1$ цвет. При некоторых принятых решениях в переборе сгенерированный граф гарантированно можно будет раскрасить в требуемое число цветов. Так, если граф $G[V \setminus Out]$ возможно раскрасить в нужное количество цветов, то продолжение перебора, очевидно, не требуется.

Нахождение точного значения хроматического числа — слишком трудоемкая задача. Однако нам достаточно получить хорошую верхнюю оценку, что проще. Как и при всех других отсечениях, мы будем исходить из баланса времени работы проверки и количества успешно отсеченных веток перебора.

2.6.1. Отсечение жадной раскраски

Это отсечение реализует очень простой способ раскраски, когда каждой вершине присваивается минимальный цвет, в который не покрашен ни один ее сосед. Более подробная реализация записана в функции 2.6.1.

В принципе существует огромное количество способов усовершенствования такого алгоритма. В основном все усовершенствования опираются на изменение порядка раскраски вершин. Однако они имеют существенный для нашей задачи недостаток: требуется рассмотреть все ребра графа, т.е. время работы пропорционально размеру графа.

На самом же деле, в процессе перебора нам необходимо раскрашивать огромное количество графов, которые отличаются друг от друга удалением и добавлением нескольких вершин. Заранее, стало быть, зафиксируем порядок, в котором вершины будут получать свои цвета. Такое ограничение алгоритма может привести к значительному огрублению верхних оценок хроматического числа. Однако оно же позволит быстро пересчитывать

Функция 2.6.1 Жадная раскраска графа G

```
1: procedure SimpleGreedyColoring(G)
2:     G = (V, E) — раскрашиваемый граф.
3:     Выходные данные: для каждой вершины цвет при жадной раскраске.
4:     UNCOLORED = V
5:     while UNCOLORED ≠ ∅ do
6:         v = наиболее подходящая вершина из UNCOLORED. (Выбирается некоторая
       вершина из еще непокрашенных. Существует огромное количество способов выбирать
       такую вершину: по наименьшей степени, наименьшей сумме степеней соседей и т.д.)
7:         INC_COLORS = ∅
8:         for u, что (u, v) ∈ E  &  u ∉ UNCOLORED do
9:             INC_COLORS = INC_COLORS ∪ {color[u]}
10:        end for
11:        color[v] = mex(INC_COLORS)
           (Здесь mex(A) = min(ℕ \ A), т.е. наименьшее натуральное число, не входящее в
       A (предполагается 0 ∈ ℕ).)
12:        UNCOLORED = UNCOLORED − {v} (удалить вершину v из множества не
       покрашенных)
13:    end while
       return color
14: end procedure
```

оценки при добавлении вершины в граф и удалении вершины из графа.

Изначально с помощью функции 2.6.1 мы для каждой вершины получим ее цвет при соответствующем методе раскраски. После этого для каждой вершины найдем мультимножество цветов ее соседей, которые были раскрашены алгоритмом раньше. В функциях 2.6.2 и 2.6.3 описаны производимые операции при обработке событий в отсечении. Такой алгоритм проверки раскраски работает чрезвычайно быстро, однако дает очень грубые оценки. В итоге скорость работы позволяет применять это отсечение на каждом шаге перебора, что дает существенное количество отсеченных веток перебора.

2.6.2. Отсечение раскраски табу

Генетические алгоритмы и алгоритмы имитации отжига используются при решении многих задач, раскраска графа — одна из них. Основным недостатком таких алгоритмов является время их работы. В принципе, это время никак не ограничено: просто чем дольше работает алгоритм, тем точнее результат он может получить. На первый взгляд, такие алгоритмы совершенно не подходят для нашего случая. Однако оказалось, что даже при небольшом количестве итераций результаты раскраски получаются очень даже не плохими.

В данной работе был применен алгоритм имитации отжига, основанный на табу-поиске (см. [41]). Алгоритм запускался не более чем на 300 итераций и использовался не с самого начала перебора, но лишь тогда, когда множество $G[V \setminus Out]$ состояло из небольшого

Функция 2.6.2 Отсечение жадной раскраски графа. Вариант помещения вершины в Out

procedure OutGreedyColoringCut($colors$, v)

Для реализации этого отсечения в процессе перебора будем поддерживать раскраску графа $G[V \setminus Out]$ жадным алгоритмом и другую дополнительную информацию, связанную с раскраской.

$colors[v]$ — цвет вершины v в раскраске.

$inc\_colors[v][c]$ — количество соседей вершины v, имеющих цвет c.

Пусть (VP, EP) — множества вершин и ребер графа $G[V \setminus Out]$.

Выходные данные: 0 — если не нужно продолжать перебор; 1 – если нужно.

(Помещение вершины в множество Out по сути обозначает удаление вершины из рассматриваемого нами графа.)

$c = colors[v]$

for u, что $(u, v) \in EP$ и $u > v$, т.е. u красится жадным алгоритмом позже v **do**

$inc\_colors[u][c] -= 1$ (вершина удалена, и мы обновляем число соседей)

if $inc\_colors[u][c] = 0$ & $c < colors[u]$ **then**

(Цвет вершины u изменился, так как появился меньший незадействованный цвет.)

$colors[u] = c$

Рекурсивно аналогичным образом обновим значения для всех соседей вершины u.

end if

end for

$\chi = max(color)$

(Максимальный цвет также возможно находить быстро, если воспользоваться структурой данных куча или же хранить в массиве количество вершин каждого цвета.)

return $\chi > n + 1$ (Продолжать нужно, если не удалось раскрасить в $n + 1$ цвет.)

end procedure

Функция 2.6.3 Отсечение жадной раскраски графа. Отмена помещения вершины в Out

procedure UNDOOUTGREEDYCOLORINGCUT($colors$, v)

Будем использовать те же обозначения, что и в процедуре 2.6.2. Данная функция позволяет быстро пересчитать раскраску графа $G[V \setminus Out]$ при обратном ходе перебора.

Пусть $G[V \setminus Out] = (VP, EP)$.

(Перемещение вершины из множества Out обратно в множество $Unknown$ по сути обозначает добавление вершины в рассматриваемый нами граф.)

$colors[v] = mex(\{c | inc\_colors[v][c] > 0\})$ (Выбираем mex среди цветов соседей.)

$c = colors[v]$

for u, что $(u, v) \in EP$ и $u > v$ **do**

$inc\_colors[u][c] += 1$

if $inc\_colors[u][c] = 1$ & $c = colors[u]$ **then**

Цвет вершины u изменился, так как текущий цвет стал задействован одной из соседних вершин.

while $inc\_colors[u][color[u]] > 0$ **do**

$color[u] += 1$ (Увеличиваем цвет, пока не найдем не задействованный.)

end while

Рекурсивно обновляем значения для всех соседей вершины u.

end if

end for

Процедура ничего не возвращает, так как вызывается при обратном ходе перебора и не может его останавливать.

end procedure

количества вершин. В начале перебора очень хорошо действуют гораздо более простые и быстрые способы отсечения.

Использование большого количества итераций алгоритма для получения более точной раскраски оказалось не эффективным. Быстрее можно уменьшить количество вершин, сделав еще несколько шагов перебора, а затем еще раз запустить алгоритм раскраски.

Хотя при добавлении и удалении вершины изменение и пересчет цвета может коснуться всех вершин, однако в большинстве случаев при удалении одной вершины цвет меняется лишь у нескольких вершин. В связи с этим на перерасчет всей раскраски требуется в среднем лишь 3–5 действий.

2.7. Дополнительные отсечения

Для ускорения программы были применены некоторые отсечения, основанные на очень простых утверждениях из работы [32].

Лемма 7. *Пусть* $\mathbf{V} = \{0, 1\}^n$ *и* d *— некоторое число, квадрат которого нечетен. Тогда граф* $G_d(\mathbf{V})$ *двудольный.*

Лемма 7 позволяет нам не рассматривать графы с нечетным квадратом диаметра множества вершин.

Лемма 8. *Пусть* $\mathbf{V} = \{0, 1\}^n$ *и* d *— некоторое число, квадрат которого четен. Тогда граф* $G_d(\mathbf{V})$ *состоит из двух изоморфных компонент связности.*

Лемма 8 позволяет нам перебирать подграфы только одной из двух компонент связности. Такое отсечение очень существенно, ведь его использование фактически «извлекает корень» из общего числа операций.

3. Доказательства лемм

Все леммы, сформулированные в алгоритме, очень просты. Нам представляется сколь-нибудь разумным привести здесь лишь доказательства лемм 4 и 5.

Лемма 4. *Пусть* $\mathbf{a} = (a_1, \ldots, a_n)$, $a_i \in \{0, 1\}$. *Тогда отображение* $f_{\mathbf{a}}$, *переводящее каждый* $\mathbf{x} = (x_1, \ldots, x_n)$, $x_i \in \{0, 1\}$, *в* $((a_1{+}x_1) \mod 2, \ldots, (a_n{+}x_n) \mod 2)$, *является изометрией для любого* $V' \subseteq \{0, 1\}^n$.

Доказательство леммы 4. В пространстве $\{0, 1\}^n$ квадрат евклидова расстояния можно вычислить как $\rho^2(\mathbf{x}, \mathbf{y}) = \sum\limits_{i=1}^{n} (x_i + y_i) \mod 2$. Подставим функцию $f_{\mathbf{a}}$ в выражение $\rho^2(f_{\mathbf{a}}(\mathbf{x}), f_{\mathbf{a}}(\mathbf{y}))$:

$$\rho^2(f_{\mathbf{a}}(\mathbf{x}), f_{\mathbf{a}}(\mathbf{y})) = \sum_{i=1}^{n} (((x_i + a_i) \mod 2) + ((y_i + a_i) \mod 2))) \mod 2 =$$

$$= \sum_{i=1}^{n} ((x_i + y_i) \mod 2) + \sum_{i=1}^{n} (2a_i \mod 2) = \sum_{i=1}^{n} (x_i + y_i) \mod 2 = \rho^2(\mathbf{x}, \mathbf{y}).$$

Лемма доказана.

Лемма 5. *Пусть* $\mathbf{a} = (a_1, \ldots, a_n)$, $a_i \in \{0, 1\}$. *Далее, пусть* $a \in \{0, 1\}$, $x \in \{0, 1, 2\}$. *Положим*

$$\begin{cases} d(0, x) = x, \\ d(a, 1) = 1, \\ d(1, x) = 2 - x, \ x \in \{0, 2\}. \end{cases}$$

Тогда отображение $f_{\mathbf{a}}$, *переводящее каждый* $\mathbf{x} = (x_1, \ldots, x_n)$, $x_i \in \{0, 1, 2\}$, *в* $(d(a_1, x_1), \ldots, d(a_n, x_n))$, *является изометрией для любого* $V' \subseteq \{0, 1, 2\}^n$.

Доказательство леммы 5. Квадрат евдклидова расстояния в $\{0, 1, 2\}^n$ можно вычислить как $\rho^2(\mathbf{x}, \mathbf{y}) = \sum\limits_{i=1}^{n} (x_i - y_i)^2$. Покажем, что $(x - y)^2 = (d(a, x) - d(a, y))^2$ для любого a.

Рассмотрим два случая:

1. если $a = 0$, то $d(a, t) = t$, поэтому равенство очевидно;
2. если $a = 1$, то следует рассмотреть три подслучая и убедиться в корректности равенства прямым вычислением:
   - если $x = y$, то обе части равенства равны 0;
   - если $x = 0, y = 1$, то обе части равны 1;
   - если $x = 0, y = 2$, то обе части равны 2.

Так как все слагаемые суммы равны, то и сами суммы равны. Лемма доказана.

4. Работа программы

Для проверки гипотезы был осуществлен перебор с описанными методами отсечений. Программа была реализованна на языке программирования C++ для минимизации второстепенных накладных расходов.

Для уменьшения вероятности ошибки в программе был применен метод модульного тестирования. Ключевые части программы перед каждым запуском проходят проверку на заранее подготовленных входных данных.

Вообще говоря, программа корректно работает для любых размерностей, однако количество операций растет сверхэкспоненциально с ростом размерности. Поэтому при текущем наборе отсечений удалось добиться разумно быстрой работы только для случаев, описанных в теоремах 1 и 2.

Запуск программы производился на компьютере с частотой процессора 3.2 Ггц. Программа работала на одном процессоре и не использовала параллельных вычислений. Благодаря большому количеству различных отсечений все варианты стало возможно рассмотреть за несколько минут. Время работы программы при использовании всех отсечений указано в таблице 2.

Таблица 2

Множество $\mathbf{V}$	Время работы
$\{0,1\}^n$, $n \leqslant 6$	меньше секунды
$\{0,1\}^7$	5 секунд
$\{0,1\}^8$	10 секунд
$\{0,1\}^9$	20 минут
$\{-1,0,1\}^n$, $n < 6$	меньше секунды
$\{-1,0,1\}^n$, $n = 6$	18 минут

При отказе от отсечений, связанных с раскраской и симметрией, время работы указано в сравнительной таблице 3. В первом столбце таблицы написано множество, на котором работала программа. В последующих столбцах написано время работы программы с различными отсечениями в минутах.

Таблица 3

Множество $\mathbf{V}$	Все отсечения	Без раскраски	Без симметрии	Без раскраски и симетрии
$\{0,1\}^n$, $n \leqslant 6$	0	0	0	0
$\{0,1\}^7$	0	0	0	2
$\{0,1\}^8$	0	11	2	67
$\{0,1\}^9$	20	$> 10^4$	$> 10^4$	$> 10^4$

Как видно из сравнительной таблицы, наибольший вклад дает отсечение раскраски, однако использование отсечения симметрии как дополнительного существенно ускоряет работу программы.

Литература

1. *Borsuk K.* Drei Sätze über die n-dimensionale euklidische Sphäre // Fundamenta Math.— 1933.— V. 20.— P. 177—190.
2. *Pál J.* Über ein elementares Variationsproblem // Danske Videnskab. Selskab. Math.-Fys. Meddel. — 1920. — V. 3, N 2.

3. *Болтянский В. Г., Гохберг И. Ц.* Теоремы и задачи комбинаторной геометрии. — М: Наука, 1965.

4. *Boltyanski V. G., Martini H., Soltan P. S.* Excursions into combinatorial geometry. — Berlin: Springer, 1997.

5. *Brass P., Moser W., Pach J.* Research problems in discrete geometry. — Berlin: Springer, 2005.

6. *Райгородский А. М.* Проблема Борсука.— М.: МЦНМО, 2006.

7. *Райгородский А. М.* Линейно-алгебраический метод в комбинаторике.— М.: МЦНМО, 2007.

8. *Райгородский А. М.* Системы общих представителей в комбинаторике и их приложения в геометрии.— М.: МЦНМО, 2010.

9. *Raigorodskii A. M.* Coloring distance graphs and graphs of diameters // сдано в печать.

10. *Raigorodskii A. M.* Three lectures on the Borsuk partition problem // London Mathematical Society Lecture Note Series. — 2007. — V. 347. — P. 202–248.

11. *Raigorodskii A. M.* The Borsuk partition problem: the seventieth anniversary // Math. Intelligencer. — 2004. — V. 26, N 3. — P. 4–12.

12. *Райгородский А. М.* Проблема Борсука и хроматические числа метрических пространств // УМН. — 2001.— Т. 56, вып. 1. — С. 107–146.

13. *Райгородский А. М.* Вокруг гипотезы Борсука // Итоги науки и техники.— Сер. «Современная математика». — 2007. — Т. 23. — С. 147–164.

14. *Hadwiger H.* Überdeckung einer Menge durch Mengen kleineren Durchmessers // Comm. Math. Helv. — 1945/46. — V. 18. — P. 73–75; Mitteilung betreffend meine Note: Überdeckung einer Menge durch Mengen kleineren Durchmessers // Comm. Math. Helv.— 1946/47.— V. 19.— P. 72—73.

15. *Schramm O.* Illuminating sets of constant width // Mathematika.— 1988.— V. 35.— P. 180—189.

16. *Bourgain J., Lindenstrauss J.* On covering a set in $\mathbb{R}^d$ by balls of the same diameter // Geometric Aspects of Functional Analysis (J. Lindenstrauss and V. Milman, eds.), Lecture Notes in Math., V. 1469.— Berlin: Springer-Verlag, 1991.— P. 138—144.

17. *Kahn J., Kalai G.* A counterexample to Borsuk's conjecture // Bulletin (new series) of the AMS. — 1933. — V. 29, N 1. — P. 60–62.

18. *Nilli A.* On Borsuk's problem // Contemporary Mathematics.— 1994.— V. 178.— P. 209—210.

19. *Grey J., Weissbach B.* Ein weiteres Gegenbeispiel zur Borsukschen Vermutung // Univ. Magdeburg, Fakultät für Mathematik.— 1997.— Preprint 25.

20. *Райгородский А. М.* О размерности в проблеме Борсука // УМН. — 1997.— Т. 52, вып. 6.— С. 181—182.

21. *Weissbach B.* Sets with large Borsuk number // Beiträge zur Algebra und Geometrie.— 2000.— V. 41.— P. 417—423.

22. *Hinrichs A.* Spherical codes and Borsuk's conjecture // Discr. Math.— 2002.— V. 243.— P. 253—256.

23. *Pikhurko O.* Borsuk's conjecture fails in dimensions 321 and 322.— arXiv:math.CO/0202112.

24. *Hinrichs A.*, *Richter C.* New sets with large Borsuk numbers.
http://www.minet.unijena.de/hinrichs/paper/18/borsuk.pdf

25. *Eggleston H. G.* Covering a three-dimensional set with sets of smaller diameter // J. London Math. Soc. — 1955. — V. 30. — P. 11—24.

26. *Heppes A.* Térbeli ponthalmazok felosztása kisebb átméröjü részhalmazok összegére // A magyar tudományos akadémia.— 1957.— V. 7.— P. 413—416.

27. *Grünbaum B.* A simple proof of Borsuk's conjecture in three dimensions // Proc. Cambridge Philos. Soc. — 1957. — V. 53. — P. 776—778.

28. *Макеев В. В.* Об аффинных образах ромбододекаэдра, описанных вокруг трехмерного выпуклого тела // Зап. научн. семин. ПОМИ. — 1997.— Т. 246.— С. 191—195.

29. *Макеев В. В.* Аффинно-вписанные и аффинно-описанные многоугольники и многогранники // Зап. научн. семин. ПОМИ. — 1996.— Т. 231.— С. 286—298.

30. *Райгородский А. М.* Об одной оценке в проблеме Борсука // УМН. — 1999. — Т. 54, вып. 2. — С. 185–186.

31. *Ziegler G. M.* Lectures on 0/1-polytopes // Polytopes — Combinatorics and Computation (G. Kalai and G.M. Ziegler, eds.). // DMV-seminar.— Birkhäuser-Verlag Basel, 2000.— V. 29.— P. 1—44.

32. *Ziegler G. M.* Coloring Hamming graphs, optimal binary codes, and the 0/1-Borsuk problem in low dimensions // Lect. Notes Comput. Sci. — 2001. — V. 2122. — P. 159—171.

33. *Payan C.* On the chromatic number of cube-like graphs // Discrete Math. — 1992.— V. 103.— P. 271—277.

34. *Schiller F.* Zur Berechnung und Abschätzung von Färbungszahlen und der ϑ-Funktion von Graphen // Diplomarbeit.— Berlin: TU, 1999,

35. *Petersen J.* Färbung von Borsuk-Graphen in niedriger Dimension // Diplomarbeit.— Berlin: TU, 1998.

36. *Райгородский А. М.* Проблема Борсука для $(0, 1)$-многогранников и кросс-политопов // Доклады РАН. — 2000.— Т. 371.— С. 600—603.

37. *Райгородский А. М.* Проблема Борсука для $(0, 1)$-многогранников и кросс-политопов // Доклады РАН. — 2002.— Т. 384.— С. 593—597.

38. *Райгородский А. М.* Проблема Борсука для целочисленных многогранников // Матем. сборник.— 2002.— Т. 193, № 10.— С. 139—160.

39. *Райгородский А. М.* Проблемы Борсука, Грюнбаума и Хадвигера для некоторых классов многогранников и графов // Доклады РАН. — 2003. — Т. 388, № 6. — С. 738—742.

40. *Райгородский А. М.* Проблемы Борсука и Грюнбаума для решетчатых многогранников // Известия РАН. — 2005. — Т. 69, № 3. — С. 81–108.

41. http://www.springerlink.com/content/y766j23246l1674/referrers/

Поступила в редакцию 15.10.2011