

ISSN 2949-5598

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Том 33 № 2 2026

Новосибирск
Издательство Института математики

МЕТОД ВЕТВЕЙ И ОТСЕЧЕНИЙ ДЛЯ ЗАДАЧИ ВЗВЕШЕННОЙ 3-РАСКРАСКИ ГРАФА

Г. А. Жуков^{1, a}, А. И. Ерзин^{1, 2, b}, Р. В. Плотников^{3, c}

¹ Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия

² Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия

³ ООО «Ледас»,
ул. Николаева, 11/5, 630090 Новосибирск, Россия

E-mail: ^ag.zhukov@ng.su.ru,
^badilerzin@math.nsc.ru, ^crvplotnikov@gmail.com

Аннотация. Для рёберно-взвешенного неориентированного графа рассматривается NP-трудная задача раскраски вершин в три цвета при условии минимизации суммарного веса рёбер, соединяющих вершины одного цвета. Данная задача математически эквивалентна известной задаче о максимальном разрезе на k частей (о максимальном k -разрезе, maximum k -cut), где требуется максимизировать вес рёбер между вершинами из различных подмножеств. В настоящей работе представлен точный алгоритм для случая $k = 3$, реализованный в рамках схемы ветвей и отсечений (branch-and-cut). В то время как большинство существующих исследований сфокусированы на классах допустимых неравенств, ключевым вкладом данной работы является метод уменьшения глубины дерева ветвления на основе декомпозиции вершин, интегрированный в схему branch-and-cut. Вычислительные эксперименты демонстрируют высокую эффективность разработанного алгоритма. Табл. 4, ил. 1, библиогр. 28.

Ключевые слова: 3-раскраска, метод ветвей и отсечений, задача о максимальном k -разрезе.

1. Обзор литературы

В настоящей работе исследуется следующая NP-трудная задача. Дан взвешенный неориентированный граф $G = (V, E)$, каждому ребру $e \in E$ приписан положительный вес w_e . Требуется раскрасить вершины графа в три цвета так, чтобы сумма весов рёбер, соединяющих вершины

одного цвета (одноцветных рёбер), была минимальной. Будем называть такую постановку задачей о минимальном 3-разбиении, поскольку в англоязычной литературе она известна как *minimum 3-partition problem*. В эквивалентной формулировке максимизации суммарного веса рёбер разреза, т. е. рёбер, соединяющих вершины разных кластеров (цветов), имеем задачу о максимальном 3-разрезе — *maximum 3-cut problem*.

1.1. Задачи раскраски графов. Исследуемая задача тесно связана с классическими проблемами раскраски графов: 1) определить хроматическое число $\chi(G)$ графа G , т. е. минимальное число цветов для правильной раскраски, в которой смежные вершины имеют разные цвета; 2) установить, допускает ли граф правильную раскраску в три цвета.

Задача определения $\chi(G)$ входит в список из 21 NP-полной задачи Карпа [1]. Много работ посвящено точным и приближённым методам её решения; фундаментальные обзоры представлены в [2–4]. Классической конструктивной эвристикой является алгоритм DSATUR [5], который динамически упорядочивает вершины, выбирая на каждом шаге ту, что имеет наибольшее число разноцветных соседей (степень насыщения).

В классе точных методов ключевая роль за работой [6], где предложен подход *branch-and-price*, моделирующий задачу как разбиение на независимые множества и решающий её с помощью процедуры генерации столбцов. Для больших графов более целесообразно применение метаэвристических подходов. В частности, гибридный эволюционный алгоритм из [7] показывает хорошие результаты за счёт применения специализированных операторов скрещивания и интеграции метаэвристики поиска с запретами, а в [8] авторы используют декомпозицию вершин для итеративного упрощения графа и сужения пространства поиска.

Значительная часть теоретических исследований обращена к гипотезе Бородина — Косточки, согласно которой хроматическое число удовлетворяет неравенству $\chi(G) \leq \max\{\omega, \Delta - 1\}$ при максимальной степени графа $\Delta \leq 9$, где ω — размер максимальной клики. Так, в [9] доказано, что некоторые гипотезы, на первый взгляд кажущиеся более слабыми, чем гипотеза Бородина — Косточки, на самом деле ей эквивалентны. В [10] получен аналогичный результат для списочной раскраски: неравенство $\chi < \Delta$ выполняется для всех графов с достаточно большим Δ при условии $\omega < \Delta$. В [11] представлен детерминированный алгоритм, строящий $O(k\Delta)$ -раскраску вершин за $O(\Delta/k) + \log n$ раундов, где $k \in [1, O(\Delta)]$.

Задача проверки существования правильной раскраски графа в три цвета NP-полна даже для планарных 4-регулярных графов [12]. Лучший известный точный алгоритм для общего случая имеет сложность $O(1,3217^n)$ [13], а для планарных графов без треугольников существует линейный алгоритм 3-раскраски [14].

1.2. Аппроксимационные алгоритмы для задачи о максимальном k -разрезе. Задача о максимальном k -разрезе является обобщением классической задачи о максимальном разрезе (maximum 2-cut). В [15] предложен приближённый алгоритм на основе полуопределённого программирования (semidefinite programming, SDP), гарантирующий коэффициент аппроксимации 0,87856 для $k = 2$. В [16] авторы обобщили этот результат для задачи о максимальном k -разрезе, предложив SDP-релаксацию с гарантированной точностью, зависящей от k (для $k = 3$ коэффициент составлял 0,800207). Впоследствии данные оценки были уточнены: в частности, для $k = 3$ соответствующий коэффициент был улучшен до 0,836008 [17].

1.3. Полиэдральная комбинаторика и метод ветвей и отсечений. Один из наиболее эффективных подходов для точного решения задачи о максимальном k -разрезе — это метод ветвей и отсечений (branch-and-cut), использующий свойства полиэдра задач.

Существует две основные формулировки задачи: вершинная и рёберная. Вершинные формулировки используют булевы переменные x_{ik} , указывающие, принадлежит ли вершина i кластеру k (покрашена ли вершина i в цвет k). Такие модели обладают недостатком — симметрией (перестановка цветов меняет решение, но не изменяет значения целевой функции). Для борьбы с симметрией применяются специальные неравенства и методы изоморфного отсечения [18].

Рёберные формулировки более перспективны для метода ветвей и отсечений. Они используют переменные y_{ij} , которые ассоциированы с рёбрами и равны 1, если вершины i и j лежат в разных кластерах (имеют разные цвета). Однако такие формулировки содержат большое число неравенств (порядка $\binom{n}{k+1}$). Для получения верхних (для задачи на максимум) оценок также используют формулировку полуопределённого программирования.

Статья [19] — одна из ключевых работ в области полиэдрального анализа задачи о k -разбиении. Её авторы детально изучили полиэдр задачи для произвольного k и описали несколько важнейших классов отсечений: общие кликовые неравенства (general clique inequalities), неравенства типа «колесо» (wheel inequalities), неравенства типа «велосипедное колесо» (bicycle wheel inequalities). Эти классы неравенств широко применяются в современных алгоритмах ветвей и отсечений.

Автор [20] через анализ гиперметрических неравенств объясняет высокое качество нижних оценок, получаемых с помощью релаксации модели полуопределённого программирования для задачи о k -разбиении. При этом доказано, что полуопределённая релаксация неявно содержит ослабленные версии этих ограничений, и установлены точные границы

максимального нарушения фасетоопределяющих неравенств (треугольных и кликовых) её допустимыми решениями.

Во многих работах метод ветвей и отсечений применяется для решения задач о минимальном k -разбиении и максимальном k -разрезе. Фундаментальный обзор математических формулировок, допустимых неравенств и ключевых элементов метода ветвей и отсечений для задачи о максимальном k -разрезе представлен в работе [21]. В [22] для задачи о минимальном k -разбиении предложен алгоритм ветвей и отсечений, использующий для вычисления нижних оценок формулировку полуопределённого программирования, а для поиска допустимых решений — новую эвристику итеративной кластеризации. Вычислительный эксперимент показал, что данная эвристика находит лучшие допустимые решения, чем классические техники округления с помощью гиперплоскостей.

Авторы [23] провели сравнительный анализ четырёх постановок задачи о минимальном k -разбиении: двух вершинных и двух рёберных. Они исследовали структуру политопов для рёберных моделей и описали новый класс фасетоопределяющих неравенств. Эксперименты показали, что вершинные формулировки обладают слабой линейной релаксацией, тогда как рёберные обеспечивают значительно более качественные оценки. На основе рёберной формулировки была предложена эффективная реализация метода ветвей и отсечений.

В работе [24] изучается обобщение задачи о максимальном k -разрезе, в котором дополнительно требуется, чтобы каждый из получаемых k кластеров индуцировал связный подграф. Авторы предложили новую модель целочисленного линейного программирования, а также метод ветвей и отсечений. Для повышения вычислительной эффективности применён анализ изоморфизма графов, позволяющий учитывать структуру рассматриваемых экземпляров.

2. Постановка задачи

Сформулируем задачу взвешенной 3-раскраски графа. Рассмотрим простой неориентированный граф $G = (V, E)$, где V — множество вершин, а E — множество рёбер. Каждому ребру $e \in E$ сопоставлен положительный вес $w(e) > 0$. Задача заключается в раскраске вершин графа в три цвета так, чтобы минимизировать суммарный вес рёбер, соединяющих вершины одного цвета.

Эта задача NP-трудна. Действительно, известно, что задача 3-раскрашиваемости графа NP-полна даже для планарных 4-регулярных графов [12]. Покажем сводимость этой классической задачи к рассматриваемой. Положим веса всех рёбер равными единице, т. е. $w(e) = 1$ для любого $e \in E$. В рассматриваемой постановке оптимальное значение целевой функции равно нулю тогда и только тогда, когда граф допускает

правильную 3-раскраску. Следовательно, существование полиномиального алгоритма для задачи взвешенной 3-раскраски влекло бы за собой полиномиальную разрешимость NP-полной задачи 3-раскрашиваемости графа, что невозможно в предположении $P \neq NP$.

Существует несколько постановок этой задачи. Рассмотрим следующую линейную модель, которую будем использовать в методе ветвей и отсечений. Введём булевы переменные:

- $x_{ik} = 1$, если вершина i имеет цвет k , и 0 иначе ($i \in V, k \in \{1, 2, 3\}$);
- $y_{ij} = 1$, если вершины i и j имеют одинаковый цвет, и 0 иначе ($(i, j) \in E$).

Задачу взвешенной 3-раскраски можно сформулировать в виде следующей модели целочисленного линейного программирования:

$$\begin{aligned} \min \quad & \sum_{(i,j) \in E} w_{ij} y_{ij}, \\ & \sum_{k=1}^3 x_{ik} = 1, \quad i \in V, \\ & y_{ij} \geq x_{ik} + x_{jk} - 1, \quad (i, j) \in E, k \in \{1, 2, 3\}, \\ & x_{ik} \in \{0, 1\}, \quad i \in V, k \in \{1, 2, 3\}, \\ & 0 \leq y_{ij} \leq 1, \quad (i, j) \in E. \end{aligned}$$

Правая часть ограничения $y_{ij} \geq x_{ik} + x_{jk} - 1$ всегда принимает целочисленные значения, поскольку переменные x_{ik} булевы. Так как целевая функция линейно зависит от y_{ij} (с положительными весами) и минимизируется, в оптимальном решении переменные y_{ij} будут принимать минимально допустимые значения, удовлетворяющие системе ограничений. Следовательно, при фиксированных x_{ik} переменные y_{ij} примут значения 0 или 1. Это позволяет опустить требование целочисленности y_{ij} и рассматривать их как непрерывные переменные на отрезке $[0, 1]$.

3. Метод ветвей и отсечений

В современной практике решения задач оптимизации широко применяются коммерческие решатели: CPLEX, Gurobi и др., которые предоставляют обширные возможности для настройки алгоритма ветвей и отсечений. В данном исследовании использовался решатель SCIP — свободно распространяемое программное обеспечение с открытым исходным кодом. Выбор SCIP обусловлен его высокой гибкостью в настройке метода ветвей и отсечений: пакет предоставляет широкий спектр точек настройки, детально описанных в официальной документации. В настоящей работе требуются следующие ключевые параметры настройки алгоритма.

1. Управление процессом ветвления — стратегия выбора переменной для фиксации её значения.
2. Стратегия обхода дерева поиска — порядок обработки узлов (в глубину, в ширину и другие альтернативные подходы).
3. Генерация отсечений — добавление ограничений для сужения пространства поиска.

Далее опишем метод декомпозиции вершин графа. Алгоритм декомпозиции основывается на построении разбиения множества вершин графа V на следующие подмножества:

- множество граничных вершин V_b (сепаратор);
- набор кластеров $\{V_1, V_2, \dots, V_k\}$.

При этом должны выполняться условия

- 1) $V = V_b \cup V_1 \cup V_2 \cup \dots \cup V_k$, причём $V_b, V_1, \dots, V_k$ попарно не пересекаются;
- 2) для любых $i \neq j$ между вершинами кластеров V_i и V_j отсутствуют рёбра.

Пример декомпозиции вершин графа представлен на рис. 1.

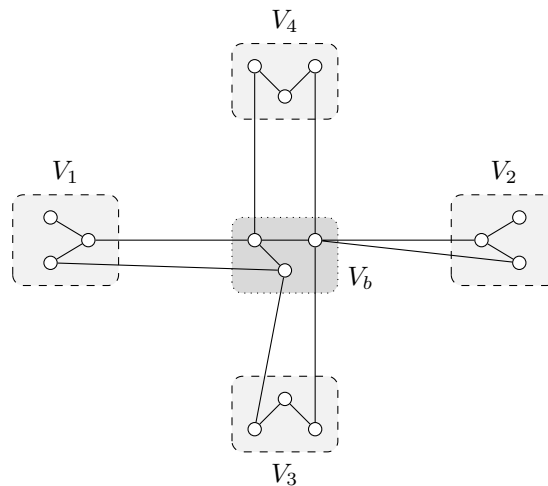


Рис. 1. Схематичное представление декомпозиции вершин графа

Если значения переменных x_{ik} зафиксированы для каждой граничной вершины (вершины, принадлежащей множеству граничных вершин V_b), то исходная задача распадается на набор независимых подзадач. В силу отсутствия рёбер между вершинами различных кластеров оптимальное решение для каждого кластера определяется исключительно цветами граничных вершин и может быть найдено независимо. Это позволяет

- 1) отсечь текущую ветвь в узле дерева поиска;

- 2) решить совокупность подзадач независимо;
- 3) передать полученное решение в основной алгоритм.

Однако на практике применение стандартных техник отсечения приводит к тому, что даже в случае разреженных графов мощность множества граничных вершин превышает максимальную глубину ветвления, определяемую как число вершин с зафиксированным цветом. Вследствие этого предложенный подход в своей исходной формулировке не обеспечивает снижения вычислительной трудоёмкости. Таким образом, необходимо модифицировать предложенную схему. Реализуем схему ветвления по вершинам с приоритетным выбором граничных вершин. Для текущей вершины i создаются подзадачи, соответствующие фиксации цвета ($x_{ik} = 1, k \in \{1, 2, 3\}$). В этом случае глубина узла в дереве поиска равна числу вершин, для которых цвет уже задан. Обозначим через d максимальную глубину дерева ветвления — ключевой параметр алгоритма. На глубине d выполняется следующая последовательность действий.

1. На основе вершин с зафиксированными цветами строится соответствующая частичная раскраска графа.
2. В множество подзадач добавляется новая задача, определяемая частичной раскраской, найденной на предыдущем шаге.
3. Текущая ветвь дерева ветвления исключается из дальнейшего рассмотрения.

Выполнение алгоритма предполагает обработку набора подзадач. Для этого используется выделенный набор вычислительных потоков (thread pool), в котором задачи решаются параллельно по мере поступления. Решение задач осуществляется по следующей схеме.

Шаг 1. Вычисление нижней оценки.

Шаг 1.1. Решается релаксация задачи целочисленного линейного программирования (ЦЛП), в которой переменные x_{ik} , соответствующие граничным вершинам, рассматриваются как непрерывные, при этом фиксируется часть переменных — те, для которых цвет задан в текущем частичном решении.

Шаг 1.2. Если полученная нижняя оценка позволяет отсечь текущую ветвь дерева поиска, то процесс завершается. Иначе осуществляется переход к следующему шагу.

Шаг 2. Поиск оптимального решения для текущей ветви.

Шаг 2.1. Решается задача ЦЛП, в которой цвета вершин, определённые в текущем частичном решении, фиксируются.

Шаг 2.2. Полученное решение сравнивается с наилучшим известным на данный момент решением.

ШАГ 2.3. Если новое решение превосходит текущее лучшее, то оно передаётся в основной алгоритм для обновления рекорда.

В связи с использованием многопоточной обработки возникает потребность в синхронизации потоков. Однако детальное обсуждение методов синхронизации выходит за рамки настоящего исследования. Следует выделить существенный фактор, влияющий на производительность алгоритма, — частоту обновления рекорда. Поскольку данная операция выступает точкой синхронизации, её частое выполнение приводит к снижению степени параллелизма. Во избежание этого целесообразно инициализировать алгоритм решением, близким к оптимальному. Для нахождения начального решения рационально использовать метаэвристические методы. При реализации предложенного подхода был использован алгоритм имитации отжига. Ключевые преимущества алгоритма имитации отжига — высокая скорость работы и качество получаемых решений.

Эффективность предложенного метода существенно зависит от мощности множества граничных вершин: сокращение мощности этого множества позволяет уменьшить глубину ветвления d , тем самым обеспечивая выигрыш в производительности. Если не ограничивать число вершин в кластерах, то решение задачи декомпозиции вершин графа вырождается в разбиение, включающее множество граничных вершин и один кластер. Во избежание тривиального решения введём дополнительное ограничение на максимальный размер кластера. Для декомпозиции множества вершин предлагается комбинированный подход, сочетающий

- рандомизированный алгоритм;
- алгоритм имитации отжига.

Схема рандомизированного алгоритма выглядит следующим образом.

ШАГ 1. Вершины перемешиваются в случайном порядке.

ШАГ 2. Для каждой вершины v выполняется следующая проверка.

ШАГ 2.1. Если v не смежна ни с одним из существующих кластеров, то

- если существует кластер, для которого не нарушается ограничение на размер, то вершина v добавляется в него;
- иначе создаётся новый кластер, состоящий из вершины v .

ШАГ 2.2. Если v смежна ровно с одним кластером, то

- вершина v добавляется в кластер при соблюдении ограничения на его размер;
- иначе вершина v добавляется в множество граничных вершин.

ШАГ 2.3. Если v смежна более чем с одним кластером, то вершина v добавляется в множество граничных вершин.

Алгоритм имитации отжига, применяемый для декомпозиции вершин графа, использует экспоненциальную функцию снижения температуры. Функция энергии определяется как мощность множества граничных вершин. На каждой итерации случайно выбранная вершина кластера добавляется в множество граничных вершин. После этого для соседей данной вершины, принадлежащих множеству граничных, проверяется возможность добавления в один из кластеров; при выполнении условий вершина добавляется в соответствующий кластер.

Осталось рассмотреть три ключевых компонента: стратегию обхода дерева поиска, правило выбора вершины ветвления и генерацию отсечений. В качестве стратегии обхода выбран поиск в глубину (depth-first search). Такой выбор обусловлен низким потреблением памяти, а также возможностью эффективного пересчёта релаксаций при переходе к дочерним узлам (за счёт использования «горячего» старта и других механизмов современных решателей). Эффективный метод выбора переменной ветвления даёт стратегия, основанная на псевдостоимостях (pseudo-costs [25]). Решатель SCIP вычисляет эти значения для каждой переменной и предоставляет программный интерфейс для доступа к ним. В качестве вершины ветвления выбирается граничная вершина v , цвет которой ещё не зафиксирован, обладающая максимальной суммарной псевдостоимостью. Суммарная псевдостоимость вершины v определяется как величина $\sum_{k=1}^3 c(x_{vk})$, где $c(x_{vk})$ — псевдостоимость переменной x_{vk} .

Для генерации отсечений используются неравенства типа «велосипедное колесо» [19], построенные на цикле длины 3, показавшие наибольшую эффективность в предварительном численном эксперименте.

4. Вычислительный эксперимент

Вычислительный эксперимент проведён на персональном компьютере с процессором Intel Core i9-13900HK и 32 ГБ оперативной памяти. Алгоритм реализован на языке C++ с использованием оптимизационного пакета SCIP версии 9.2.3. Оценка эффективности предложенного подхода проведена в сравнении с базовой конфигурацией решателя SCIP. Для корректности в обеих версиях использованы идентичные настройки: стратегия обхода в глубину и генерация отсечений типа «велосипедное колесо». В дальнейшем разработанный метод ветвей и отсечений будем называть SCIP-mod, а базовую конфигурацию — SCIP-std.

Для эксперимента сформированы четыре группы тестовых графов:

- выборка примеров из работы [24];
- набор компонент связности сетей мобильной связи;
- графы, полученные с помощью генератора [26], предоставленного организаторами конкурса MOTOR 2025 Challenge 5 [27];

Таблица 1

Графы из статьи [24]

n	m	SCIP-mod		SCIP-std		$\delta, \%$
		t, c	N	t, c	N	
50	306	64	20 829	132	12 153	51,5
		68	23 114	119	8 780	42,9
		74	20 114	95	7 313	22,1
		118	29 058	141	13 994	16,3
		183	31 873	234	38 438	21,8
70	232	24	15 309	59	7 682	59,3
	235	8	7 069	13	1 640	38,5
	245	20	10 309	57	4 752	64,9
	250	50	19 474	147	15 551	66
	254	91	20 884	145	23 051	37,2
80	346	2	1 166	3	351	33,3
	350	232	26 991	321	37 931	27,7
		508	66 392	538	95 395	5,6
		439	42 431	364	51 425	-20,6
		280	38 163	269	29 807	-4,1
93	593	56	9 020	32	2 301	-75
114	546	9	2,184	8	1 012	-12,5
Общее время работы		2 226 с		2 677 с		16,8

- unit-disk-графы.

Примеры графов, сгенерированных с помощью генератора, и unit-disk-графов доступны для скачивания по ссылке [28]. Unit-disk-графы генерировались следующим образом: в единичном квадрате $[0, 1] \times [0, 1]$ случайным образом размещались n точек, соответствующих вершинам графа. Две вершины соединялись ребром тогда и только тогда, когда расстояние между соответствующими точками не превышало порогового значения $2r$ (что эквивалентно пересечению кругов радиуса r с центрами в этих точках).

Результаты численного эксперимента для каждой из четырёх рассматриваемых групп тестовых примеров представлены в виде табл. 1–4, в которых приняты следующие обозначения:

- n — число вершин графа;
- m — число рёбер;
- t — время работы алгоритма в секундах;
- N — число узлов дерева поиска, просмотренных в ходе работы алгоритма;

Таблица 2

Компоненты связности сетей мобильной связи

n	m	SCIP-mod		SCIP-std		δ , %
		t , с	N	t , с	N	
53	293	53	14 780	87	15 610	39,1
56	406	33	4 680	36	4 753	8,3
70	600	336	31 457	441	26 784	23,8
		705	46 076	1 355	204 065	47,9
114	891	1 450	338 470	1 673	366 494	13,3
Общее время работы		2 577 с		3 592 с		28,3

• δ — относительное сокращение времени вычисления в процентах, рассчитываемое по формуле

$$\delta = \frac{t_S - t_M}{t_S} \cdot 100\%, \quad (1)$$

где t_S и t_M — время работы SCIP-std и SCIP-mod соответственно.

В последней строке каждой таблицы приведено суммарное время работы стандартной и модифицированной версий решателя для соответствующей группы примеров, а также итоговый относительный выигрыш по времени, вычисленный по формуле (1).

Анализ полученных результатов показывает, что выигрыш по времени составил 16,8% для первой группы, 28,3% для второй, 33,1% для третьей и 41,2% для четвёртой. Стоит отметить, что в первой группе встречались отдельные задачи, на которых модифицированная версия уступала стандартной. Во всех остальных случаях предложенный метод показал меньшее время работы.

Что касается размера дерева поиска, то сокращение числа просмотренных узлов было зафиксировано примерно для 30% примеров первой группы, 80% — второй, 28% — третьей и 60% — четвёртой групп.

Отдельного пояснения требует тот факт, что размер дерева поиска сократился не во всех тестовых примерах. Причина заключается в различных схемах ветвления. В стандартной настройке решатель использует классическое ветвление по дробной переменной x_{vc} . В предлагаемой модификации применяется ветвление по вершинам: для выбранной вершины v создаются три дочерние ветви, фиксирующие её цвет ($x_{v1} = 1$, $x_{v2} = 1$, $x_{v3} = 1$ соответственно). Данная схема ветвления не была интегрирована в базовую версию SCIP по следующим причинам.

1. Реализация пользовательского правила ветвления вносит дополнительные вычислительные накладные расходы.

Таблица 3

Графы, полученные генератором MOTOR 2025 Challenge 5

n	m	SCIP-mod		SCIP-std		$\delta, \%$
		t, c	N	t, c	N	
90	355	42	11 676	76	4 113	44,7
	361	163	24 990	251	21 390	35,1
	372	42	11 230	116	5 132	63,8
	374	111	25 073	226	15 456	50,9
	385	55	15 576	116	6 844	52,6
95	350	48	10 325	65	6 109	26,2
	354	48	9 652	95	6 528	49,5
	360	83	12 727	137	9 717	39,4
	365	87	16 545	306	38 650	71,6
	391	212	30 481	234	25 712	9,4
100	410	433	49 392	460	48 541	5,9
	411	218	31 424	358	33 417	39,1
	413	300	35 442	366	27 556	18
	432	199	28 003	295	19 632	32,6
	467	236	27 887	310	17 049	23,9
105	357	228	39 061	250	24 574	8,8
	374	120	21 851	120	12 771	0
	376	155	12 725	214	19 137	27,6
	378	440	41 482	556	51 183	20,9
	386	163	25 063	241	20 831	32,4
110	323	324	34 535	384	61 219	15,6
	354	150	22 749	189	19 435	20,6
	362	57	11 899	83	5 779	31,3
	417	244	27 790	440	41 916	44,6
	471	538	40 114	1 515	200 173	64,5
Общее время работы		4 696 с		7 019 с		33,1

2. Предварительный эксперимент показал, что в рамках стандартной настройки переход к ветвлению по вершинам на значительной части тестов приводит к снижению производительности по сравнению со встроенным механизмом.

Результаты экспериментов подтверждают, что предложенная техника уменьшения глубины ветвления обеспечивает существенное увеличение производительности.

Заключение

В работе предложен новый метод построения оптимального решения для задачи о минимальном 3-разбиении. Основу этого метода составляет

Таблица 4

Unit-disk-графы

n	m	SCIP-mod		SCIP-std		$\delta, \%$
		t, c	N	t, c	N	
90	296	145	54 362	254	91 569	42,9
	304	160	44 775	201	44 140	20,4
		270	103 814	410	82 672	34,1
	308	297	54 997	905	263 058	67,2
	312	68	29 836	132	28 477	48,5
95	324	779	144 914	1 025	419 258	24
	339	318	78 337	354	39 948	10,2
	345	177	52 559	251	51 534	29,5
	346	2 332	222 172	3 075	958 663	24,2
	366	346	44 692	578	63 006	40,1
100	358	2 699	477 727	4 422	1 431 610	39
	367	137	34 526	419	132 001	67,3
	374	539	82 168	1 274	387 737	57,7
	376	206	37 958	376	36 415	45,2
	383	2 516	263 350	5 000	2 250 000	49,7
Общее время работы		10 990 c		18 676 c		41,2

сокращение глубины дерева ветвления посредством разбиения вершин графа на множество граничных вершин и кластеры, между которыми отсутствуют рёбра, с последующим приоритетным ветвлением по граничным вершинам. Результаты проведённого численного эксперимента подтвердили высокую эффективность разработанного метода.

Финансирование работы

Исследование выполнено в рамках государственного задания Институт математики им. С. Л. Соболева (проект № FWNF-2026-0021). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. Karp R. M. Reducibility among combinatorial problems // Complexity of computer computations. Proc. Symp. (New York, USA, March 20–22, 1972). New York: Plenum Press, 1972. P. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

2. **Borodin O. V.** Colorings of plane graphs: A survey // *Discrete Math.* 2013. V. 313, No. 4. P. 517–539. DOI: 10.1016/j.disc.2012.11.011.
3. **Husfeldt T.** Graph colouring algorithms // *Topics in chromatic graph theory.* Cambridge: Cambridge Univ. Press, 2015. P. 277–303. (Encycl. Math. Appl.). DOI: 10.1017/CB09781139519793.016.
4. **Kostochka A. V., Yancey M.** On coloring of sparse graphs // *Computer science—Theory and applications. Proc. 8th Int. Comput. Sci. Symp. in Russia (Yekaterinburg, Russia, June 25–29, 2013).* Heidelberg: Springer, 2013. P. 224–234. (Lect. Notes Comput. Sci.; V. 7913). DOI: 10.1007/978-3-642-38536-0_20.
5. **Brélaz D.** New methods to color the vertices of a graph // *Commun. ACM.* 1979. V. 22, No. 4. P. 251–256. DOI: 10.1145/359094.359101.
6. **Mehrotra A., Trick M. A.** A column generation approach for graph coloring // *INFORMS J. Comput.* 1996. V. 8, No. 4. P. 344–354. DOI: 10.1287/ijoc.8.4.344.
7. **Galinier P., Hao J.-K.** Hybrid evolutionary algorithms for graph coloring // *J. Comb. Optim.* 1999. V. 3. P. 379–397. DOI: 10.1023/A:1009823419804.
8. **Moalic L., Gondran A.** Variations on memetic algorithms for graph coloring problems // *J. Heuristics.* 2018. V. 24, No. 4. P. 1–24. DOI: 10.1007/s10732-017-9354-9.
9. **Cranston D. W., Rabern L. W.** Coloring a graph with $\Delta - 1$ colors: Conjectures equivalent to the Borodin–Kostochka conjecture that appear weaker // *Eur. J. Comb.* 2015. V. 44. P. 23–42. DOI: 10.1016/j.ejc.2014.09.006.
10. **Choi I., Kierstead H. A., Rabern L. W.** The list version of the Borodin–Kostochka conjecture for graphs with large maximum degree // *Discrete Math.* 2023. V. 346, No. 11. Article ID 113300. 12 p. DOI: 10.1016/j.disc.2022.113300.
11. **Maus Y.** Distributed graph coloring made easy. Ithaca, NY, 2021. 22 p. (e-Print Archive / Cornell Univ.; arXiv:2105.05575). DOI: 10.48550/arXiv.2105.05575.
12. **Dailey D. P.** Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete // *Discrete Math.* 1980. V. 30, No. 3. P. 289–293. DOI: 10.1016/0012-365X(80)90236-8.
13. **Meijer L.** 3-Coloring in time $O(1.3217^n)$. Ithaca, NY, 2023. 17 p. (e-Print Archive / Cornell Univ.; arXiv:2302.13644). DOI: 10.48550/arXiv.2302.13644.
14. **Dvořák Z., Kawarabayashi K.-I., Thomas R.** Three-coloring triangle-free planar graphs in linear time // *ACM Trans. Algorithms.* 2011. V. 7, No. 4. Article ID 41. 14 p. DOI: 10.1145/2000807.2000809.
15. **Goemans M., Williamson D. P.** Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming // *J. ACM.* 1995. V. 42, No. 6. P. 1115–1145. DOI: 10.1145/227683.227684.
16. **Frieze A., Jerrum M.** Improved approximation algorithms for MAX- k -CUT and MAX BISECTION // *Algorithmica.* 1997. V. 18. P. 67–81. DOI: 10.1007/BF02523688.

17. **De Klerk E., Pasechnik D. V., Warners J. P.** On approximate graph colouring and MAX- k -CUT algorithms based on the ϑ -function // J. Comb. Optim. 2004. V. 8, No. 3. P. 267–294. DOI: 10.1023/B:JOCO.0000038911.67280.3f.
18. **Margot F.** Pruning by isomorphism in Branch-and-cut // Math. Program. Ser. A. 2002. V. 94. P. 71–90. DOI: 10.1007/s10107-002-0358-2.
19. **Chopra S., Rao M.** The partition problem // Math. Program. 1993. V. 59, No. 1. P. 87–115. DOI: 10.1007/BF01581239.
20. **Eisenblätter A.** The semidefinite relaxation of the k -partition polytope is strong // Integer programming and combinatorial optimization. Proc. 9th Int. IPCO Conf. (Cambridge, USA, May 27–29, 2002). Heidelberg: Springer, 2002. P. 273–290. (Lect. Notes Comput. Sci.; V. 2337). DOI: 10.1007/3-540-47867-1_20.
21. **De Sousa R.** Global optimization of the maximum K -cut problem: PhD Thesis. Montreal, 2018. 95 p.
22. **Ghaddar B., Anjos M. F., Liers F.** A branch-and-cut algorithm based on semidefinite programming for the minimum k -partition problem // Ann. Oper. Res. 2011. V. 188, No. 1. P. 155–174. DOI: 10.1007/s10479-008-0481-4.
23. **Ales Z., Knippel A.** The K -partitioning problem: Formulations and branch-and-cut // Networks. 2020. V. 76, No. 3. P. 323–349. DOI: 10.1002/net.21944.
24. **Healy P., Jozefowicz N., Laroche P., Marchetti F., Martin S., Róka Z.** A branch-and-cut algorithm for the connected max- k -cut problem // Eur. J. Oper. Res. 2024. V. 312, No. 1. P. 117–124. DOI: 10.1016/j.ejor.2023.06.015.
25. **Benichou M., Gauthier J. M., Girodet P., Hentges G., Ribiere G., Vincent O.** Experiments in mixed-integer linear programming // Math. Program. 1971. V. 1. P. 76–94. DOI: 10.1007/BF01584074.
26. **Kovrizhnykh N. A.** 3-Coloring: Graph generator for MOTOR 2025 Challenge 5. 2025. URL: github.com/sedefe/3-coloring (accessed: 6.08.2026).
27. **MOTOR 2025 challenges.** Novosibirsk: IM SO RAN, 2025. URL: old.math.nsc.ru/conference/motor/2025/challenges.html (accessed: 6.08.2026).
28. **Dataset of generated and unit-disk graphs.** 2025. URL: <https://disk.yandex.ru/d/cbMBI1Rp0J535Q> (accessed: 6.08.2026).

Жуков Георгий Алексеевич
Ерзин Адиль Ильясович
Плотников Роман Викторович

Статья поступила
17 декабря 2025 г.
После доработки —
15 февраля 2026 г.
Принята к публикации
23 марта 2026 г.

BRANCH-AND-CUT METHOD FOR THE WEIGHTED 3-COLORING PROBLEM

G. A. Zhukov^{1, a}, *A. I. Erzin*^{1, 2, b}, and *R. V. Plotnikov*^{3, c}

¹Novosibirsk State University,

2 Pirogov Street, 630090 Novosibirsk, Russia

²Sobolev Institute of Mathematics,

4 Acad. Koptug Avenue, 630090 Novosibirsk, Russia

³Ledas Ltd.,

11/5 Nikolaev Street, 630090 Novosibirsk, Russia

E-mail: ^ag.zhukov@ngsu.ru,

^badilerzin@math.nsc.ru, ^crvplotnikov@gmail.com

Abstract. This study addresses the NP-hard problem of coloring the vertices of an edge-weighted undirected graph with 3 colors, aiming to minimize the total weight of edges connecting vertices of the same color. This problem is mathematically equivalent to the well-known Maximum k -Cut problem, where the objective is to maximize the weight of edges between vertices from different subsets. In this work, we present an exact algorithm for the case $k = 3$, implemented within the branch-and-cut framework. While most existing research focuses on classes of valid inequalities, the key contribution of this paper is a method for reducing the depth of the branching tree based on vertex decomposition integrated into the branch-and-cut scheme. Computational experiments demonstrate the high efficiency of the proposed algorithm. Tab. 4, illustr. 1, bibliogr. 28.

Keywords: 3-coloring, branch-and-cut, maximum k -cut.

References

1. **R. M. Karp**, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, Proc. Symp. (New York, USA, March 20–22, 1972) (Plenum Press, New York, 1972), pp. 85–103, DOI: 10.1007/978-1-4684-2001-2_9.

2. **O. V. Borodin**, Colorings of plane graphs: A survey, *Discrete Math.* **313** (4), 517–539 (2013), DOI: 10.1016/j.disc.2012.11.011.
3. **T. Husfeldt**, Graph colouring algorithms, in *Topics in Chromatic Graph Theory* (Cambridge Univ. Press, Cambridge, 2015), pp. 277–303 (Encycl. Math. Appl.), DOI: 10.1017/CB09781139519793.016.
4. **A. V. Kostochka** and **M. Yancey**, On coloring of sparse graphs, in *Computer Science — Theory and Applications*, Proc. 8th Int. Comput. Sci. Symp. in Russia (Yekaterinburg, Russia, June 25–29, 2013) (Springer, Heidelberg, 2013), pp. 224–234 (Lect. Notes Comput. Sci., Vol. 7913), DOI: 10.1007/978-3-642-38536-0_20.
5. **D. Brélaz**, New methods to color the vertices of a graph, *Commun. ACM* **22** (4), 251–256 (1979), DOI: 10.1145/359094.359101.
6. **A. Mehrotra** and **M. A. Trick**, A column generation approach for graph coloring, *INFORMS J. Comput.* **8** (4), 344–354 (1996), DOI: 10.1287/ijoc.8.4.344.
7. **P. Galinier** and **J.-K. Hao**, Hybrid evolutionary algorithms for graph coloring, *J. Comb. Optim.* **3**, 379–397 (1999), DOI: 10.1023/A:1009823419804.
8. **L. Moalic** and **A. Gondran**, Variations on memetic algorithms for graph coloring problems, *J. Heuristics* **24** (4), 1–24 (2018), DOI: 10.1007/s10732-017-9354-9.
9. **D. W. Cranston** and **L. W. Rabern**, Coloring a graph with $\Delta - 1$ colors: Conjectures equivalent to the Borodin–Kostochka conjecture that appear weaker, *Eur. J. Comb.* **44**, 23–42 (2015), DOI: 10.1016/j.ejc.2014.09.006.
10. **I. Choi**, **H. A. Kierstead**, and **L. W. Rabern**, The list version of the Borodin–Kostochka conjecture for graphs with large maximum degree, *Discrete Math.* **346** (11), ID 113300 (2023), DOI: 10.1016/j.disc.2022.113300.
11. **Y. Maus**, Distributed graph coloring made easy (Ithaca, NY, 2021) (e-Print Archive / Cornell Univ., arXiv:2105.05575), DOI: 10.48550/arXiv.2105.05575.
12. **D. P. Dailey**, Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete, *Discrete Math.* **30** (3), 289–293 (1980), DOI: 10.1016/0012-365X(80)90236-8.
13. **L. Meijer**, 3-Coloring in time $O(1.3217^n)$ (Ithaca, NY, 2023) (e-Print Archive / Cornell Univ., arXiv:2302.13644), DOI: 10.48550/arXiv.2302.13644.
14. **Z. Dvořák**, **K.-I. Kawarabayashi**, and **R. Thomas**, Three-coloring triangle-free planar graphs in linear time, *ACM Trans. Algorithms* **7** (4), ID 41 (2011), DOI: 10.1145/2000807.2000809.
15. **M. Goemans** and **D. P. Williamson**, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, *J. ACM* **42** (6), 1115–1145 (1995), DOI: 10.1145/227683.227684.
16. **A. Frieze** and **M. Jerrum**, Improved approximation algorithms for MAX- k -CUT and MAX BISECTION, *Algorithmica* **18**, 67–81 (1997), DOI: 10.1007/BF02523688.

17. **E. De Klerk, D. V. Pasechnik, and J. P. Warners**, On approximate graph colouring and MAX- k -CUT algorithms based on the ϑ -function, *J. Comb. Optim.* **8** (3), 267–294 (2004), DOI: 10.1023/B:JOCO.0000038911.67280.3f.
18. **F. Margot**, Pruning by isomorphism in Branch-and-Cut, *Math. Program., Ser. A*, **94**, 71–90 (2002), DOI: 10.1007/s10107-002-0358-2.
19. **S. Chopra and M. Rao**, The partition problem, *Math. Program.* **59** (1), 87–115 (1993), DOI: 10.1007/BF01581239.
20. **A. Eisenblätter**, The semidefinite relaxation of the k -partition polytope is strong, in *Integer Programming and Combinatorial Optimization*, Proc. 9th Int. IPCO Conf. (Cambridge, USA, May 27–29, 2002) (Springer, Heidelberg, 2002), pp. 273–290 (Lect. Notes Comput. Sci., Vol. 2337), DOI: 10.1007/3-540-47867-1_20.
21. **R. de Sousa**, Global optimization of the maximum K -cut problem, *PhD Thesis* (Montreal, 2018).
22. **B. Ghaddar, M. F. Anjos, and F. Liers**, A branch-and-cut algorithm based on semidefinite programming for the minimum k -partition problem, *Ann. Oper. Res.* **188** (1), 155–174 (2011), DOI: 10.1007/s10479-008-0481-4.
23. **Z. Ales and A. Knippel**, The K -partitioning problem: Formulations and branch-and-cut, *Networks* **76** (3), 323–349 (2020), DOI: 10.1002/net.21944.
24. **P. Healy, N. Jozefowicz, P. Laroche, F. Marchetti, S. Martin, and Z. Róka**, A branch-and-cut algorithm for the connected max- k -cut problem, *Eur. J. Oper. Res.* **312** (1), 117–124 (2024), DOI: 10.1016/j.ejor.2023.06.015.
25. **M. Benichou, J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent**, Experiments in mixed-integer linear programming, *Math. Program.* **1**, 76–94 (1971), DOI: 10.1007/BF01584074.
26. **N. A. Kovrizhnykh**, 3-Coloring: Graph generator for MOTOR 2025 Challenge 5 (2025), URL: github.com/sedefe/3-coloring (accessed: 6.08.2026).
27. MOTOR 2025 Challenges (IM SO RAN, Novosibirsk, 2025), URL: old.math.nsc.ru/conference/motor/2025/challenges.html (accessed: 6.08.2026).
28. Dataset of generated and unit-disk graphs (2025), URL: <https://disk.yandex.ru/d/cbMBI1Rp0J535Q> (accessed: 6.08.2026).

Georgy A. Zhukov
 Adil I. Erzin
 Roman V. Plotnikov

Received December 17, 2025
 Revised February 15, 2026
 Accepted March 23, 2026