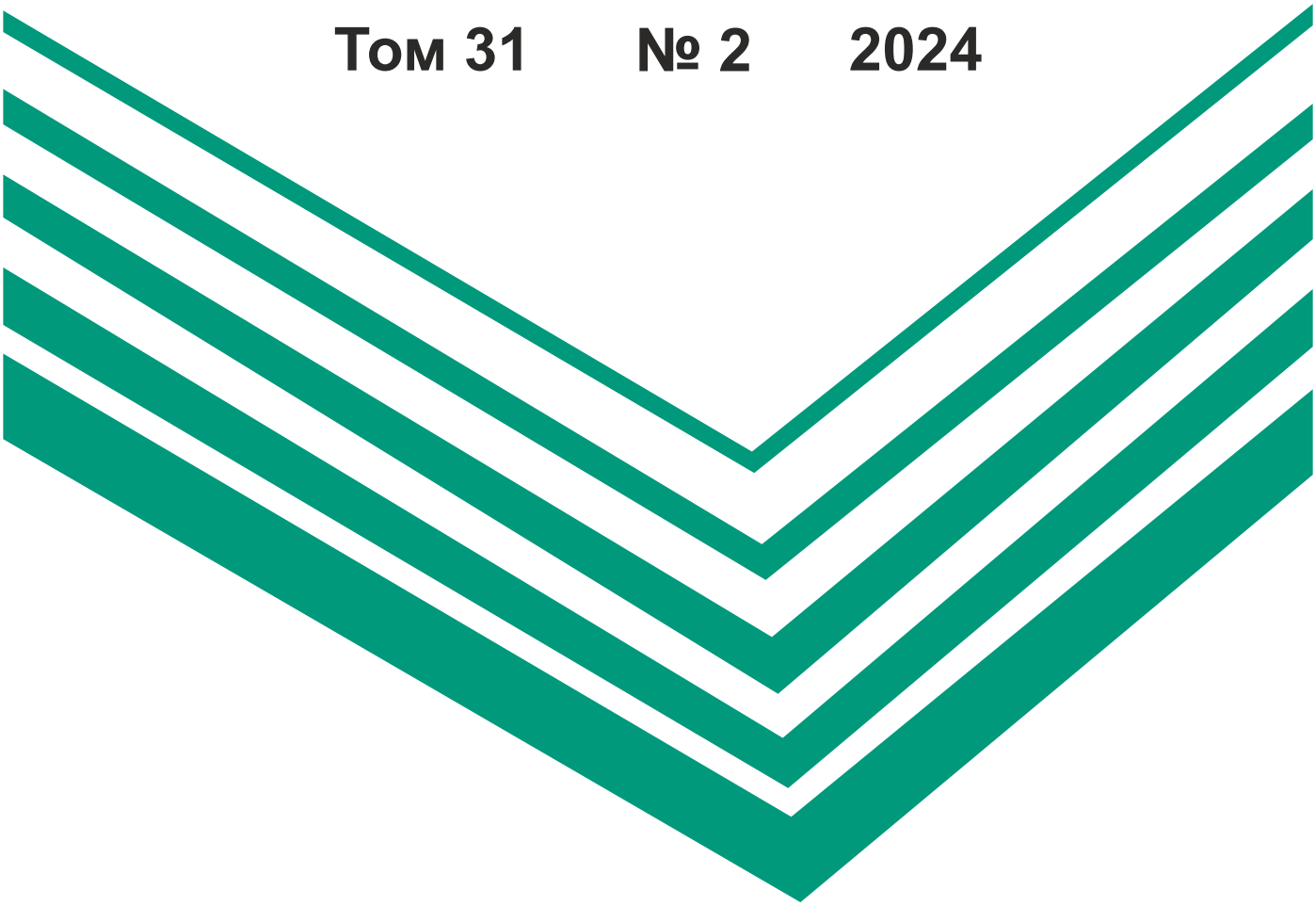
ISSN 2949-5598

# ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Том 31 № 2 2024



Новосибирск
Издательство Института математики

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2024. Т. 31, № 2. С. 28–45

УДК 519.8 DOI: 10.33048/daio.2024.31.797

# УСТОЙЧИВОСТЬ ВЕРШИННЫХ ПОКРЫТИЙ В ИГРЕ С КОНЕЧНЫМ ЧИСЛОМ ШАГОВ

*В. Л. Береснев*[1, a], *А. А. Мельников*[1, b], *С. Ю. Утюпин*[2, c]

[1] Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
[2] Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия
E-mail: [a] beresnev@math.nsc.ru,
[b] melnikov@math.nsc.ru, [c] stepan.utyupin@gmail.com

**Аннотация.** Задача о вечном вершинном покрытии является вариантом задачи о вершинном покрытии графа и может рассматриваться как динамическая игра двух сторон (Атакующего и Защитника) с бесконечным числом шагов. На каждом шаге имеется размещение охранников по вершинам графа, образующее вершинное покрытие. Атакующий выбирает для атаки одно из рёбер графа, а Защитник должен переместить охранника вдоль атакованного ребра из одной вершины в другую. Кроме того, Защитник может переместить любое подмножество остальных охранников из вершин, в которых они находились до атаки, в одну из незанятых соседних вершин, чтобы получить новое вершинное покрытие.

В статье описана процедура, позволяющая ответить на вопрос, существует ли выигрышная стратегия для Защитника, позволяющая защищать вершинное покрытие в течении заданного числа шагов. Для построения стратегии Защитника задача представляется в виде динамической игры Штакельберга, на каждом шаге которой взаимодействие противоборствующих сторон формализуется как двухуровневая задача математического программирования. Ил. 6, библиогр. 11.

**Ключевые слова:** динамическая игра Штакельберга, вечное вершинное покрытие, защита рёбер графа, алгоритм проверки устойчивости.

## Введение

Модели вечной защиты рёбер графа появились в 1990-х гг. в качестве формализации для стратегии императора Константина по защите

Римской империи [1]. В этих моделях рёбра графа защищаются от бесконечной последовательности атак. Защита осуществляется с помощью охранников, размещаемых в вершинах графа и способных перемещаться по его рёбрам после атаки.

В данной статье рассматривается задача о вечном вершинном покрытии, сформулированная в [2]. Эту задачу можно интерпретировать как бесконечную игру двух сторон — Атакующего и Защитника. Первоначально Защитник размещает в вершинах заданного графа охранников так, что вершины, в которых размещены охранники, образуют вершинное покрытие. Далее на каждом шаге Атакующий выбирает для атаки любое ребро, в одной из вершин которого нет охранника. В ответ Защитник перемещает охранника из вершины атакованного ребра в другую вершину этого ребра. Кроме того, Защитник имеет право из любой вершины переместить охранника вдоль любого ребра в соседнюю вершину, если в ней не было охранника, либо охранник был перемещён из этой вершины. В одну вершину не могут быть перемещены несколько охранников, при этом один и тот же охранник не может быть перемещён более одного раза на одном шаге. Если реконструированное размещение охранников в вершинах образует вершинное покрытие, то начинается следующий шаг с той же последовательностью действий. Если же с помощью указанной процедуры не удаётся построить вершинное покрытие, т. е. хотя бы одно ребро окажется незащищённым, то Атакующий побеждает и игра завершается. Победителем становится Защитник, если он может защититься от бесконечной последовательности атак.

Задача нахождения минимального числа охранников (eternal vertex cover number, EVCN), образующих вершинное покрытие, для которого в рассматриваемой игре может быть построена выигрышная стратегия Защитника, есть задача о вечном вершинном покрытии. В [3] установлено, что эта задача NP-трудна, а в [4] показано, что задача остаётся NP-трудной для двудольных графов. В статье [5] авторы предлагают новую нижнюю оценку для EVCN произвольного графа через мощность вершинного покрытия минимального размера, содержащего все точки сочленения данного графа. Также предлагается квадратичный по времени алгоритм для вычисления EVCN для хордальных графов. В [6] для некоторых классов графов построены полиномиальные алгоритмы вычисления EVCN. В статье [7] предложен полиномиальный алгоритм для специализированных древовидных графов. В [8] получен полиномиальный алгоритм для поиска EVCN в цепных графах и кографах, а также предложен алгоритм с линейным временем для поиска EVCN для разделённых графов, подкласса хордальных графов. В [3] авторы показывают, что для задачи о вечном вершинном покрытии существует 2-приближённый полиномиальный по времени алгоритм.

Можно заметить, что перечисленные и многие родственные работы либо предлагают алгоритмы для конкретных семейств графов, либо приводят оценки для общих или частных случаев. В данной статье мы рассматриваем ситуацию, когда задан граф, на структуру которого не наложено никаких ограничений, а также дано некоторое вершинное покрытие без каких-либо предположений о нём. Рассматривается вопрос, какой игрок выигрывает, если игра «Атакующий — Защитник» начнётся с заданного вершинного покрытия.

Ответ на этот вопрос требует анализа всех бесконечных последовательностей атак. На практике достаточно рассматривать последовательности некоторой достаточно большой длины из-за конечности множества возможных состояний игры, однако стоит ожидать, что эта длина будет неизвестной. В этой работе мы предлагаем процедуру, позволяющую ответить на вопрос, существует ли выигрышная стратегия для Защитника, позволяющая защищать заданное вершинное покрытие на протяжении наперёд заданного конечного числа шагов. Данная процедура является обобщением процедуры, предложенной в [9].

Для построения стратегии Защитника рассматриваемая игра представляется в виде динамической игры Штакельберга. На каждом шаге взаимодействие противоборствующих сторон формализуется в виде задачи двухуровневого математического программирования. Решение задачи верхнего уровня (задачи Атакующего) определяет ребро, атака которого приводит к максимальному числу незащищённых рёбер после реконструкции Защитником размещения охранников. Решение задачи нижнего уровня (задачи Защитника) при заданном решении Атакующего изменяет размещение охранников с целью минимизировать число незащищённых рёбер.

Приводятся результаты вычислительных экспериментов с использованием предложенной процедуры.

## 1. Формулировка задачи, рассматриваемой на шагах исследуемой игры

Пусть $G = (I, J)$ есть ориентированный граф без петель с множеством вершин $I$ и множеством дуг $J$. Для всякого $j \in J$ обозначим через $i(j)$ и $k(j)$ начальную и конечную вершины дуги $j$ соответственно и будем называть эти вершины соседними. Отметим, что рассмотрение ориентированного графа не уменьшает общности двухуровневой задачи, рассматриваемой на каждом шаге исследуемой игры. В самом деле, в неориентированном графе каждое ребро можно заменить двумя противоположно ориентированными параллельными дугами. При этом будем считать, что перемещение охранника по дуге происходит от начальной вершины к конечной вершине дуги.

Вершинным покрытием в графе $G$ считаем всякое подмножество множества $I$, содержащее начальную или конечную вершину любой дуги из множества $J$.

Для формальной записи двухуровневой задачи введём следующие обозначения:

• $x_{i0} \in \{0,1\}$ — заданные величины, определяющие начальное для рассматриваемого шага вершинное покрытие: $x_{i0} = 1$, если в вершине $i \in I$ размещён охранник, и $x_{i0} = 0$ в противном случае;

• $x_i \in \{0,1\}$ — переменные, задающие размещение охранников по завершении шага: $x_i = 1$, если в вершине $i \in I$ размещён охранник, и $x_i = 0$ в противном случае;

• $y_j \in \{0,1\}$ — переменные, обозначающие атакованную на шаге дугу: $y_j = 1$, если дуга $j \in J$ атакована, и $y_j = 0$ в противном случае;

• $u_j \in \{0,1\}$ — переменные, обозначающие дуги, которые по завершении шага оказались незащищёнными: $u_j = 1$, если в начальной и конечной вершинах дуги $j \in J$ не размещён охранник, и $u_j = 0$ в противном случае;

• $v_j \in \{0,1\}$ — переменные, обозначающие перемещение охранников по дугам на шаге: $v_j = 1$, если охранник перемещается по дуге $j \in J$ из вершины $i(j)$ в вершину $k(j)$, и $v_j = 0$ в противном случае.

С использованием введённых обозначений модель, формализующая взаимодействие сторон на шаге рассматриваемой игры, записывается как следующая задача двухуровневого математического программирования:

$$\max_{(y_j)} \sum_{j \in J} u_j^*, \tag{1}$$

$$\sum_{j \in J} y_j \leqslant 1, \tag{2}$$

$$x_{i(j)0} \geqslant y_j, \quad j \in J, \tag{3}$$

$$x_{i(j)0} + x_{k(j)0} + y_j \leqslant 2, \quad j \in J, \tag{4}$$

$$y_j \in \{0,1\}, \quad j \in J, \tag{5}$$

$(x_j^*), (v_j^*), (u_j^*)$ — оптимальное решение задачи:

$$\min_{(x_i),(v_j),(u_j)} \sum_{j \in J} u_j, \tag{6}$$

$$v_j \geqslant y_j, \quad j \in J, \tag{7}$$

$$x_{i0} \geqslant \sum_{j\colon i(j)=i} v_j, \quad i \in I, \tag{8}$$

$$1 - x_{i0} \geqslant \sum_{j\colon k(j)=i} v_j - \sum_{j\colon i(j)=i} v_j, \quad i \in I, \tag{9}$$

$$x_i = x_{i0} + \sum_{j\colon k(j)=i} v_j - \sum_{j\colon i(j)=i} v_j, \quad i \in I, \tag{10}$$

$$x_{i(j)} + x_{k(j)} + u_j \geqslant 1, \quad j \in J, \tag{11}$$

$$x_i, u_j, v_j \in \{0, 1\}, \quad i \in I,\ j \in J. \tag{12}$$

Данная модель, как и всякая двухуровневая задача, включает задачу верхнего уровня (1)–(5) (задачу Атакующего) и задачу нижнего уровня (6)–(12) (задачу Защитника).

Целевые функции (1) и (6) обеих задач показывают, сколько дуг осталось без защиты после реконструкции вершинного покрытия на рассматриваемом шаге. Неравенство (2) означает, что на рассматриваемом шаге атакованной может быть только одна дуга, а из неравенств (3)–(4) следует, что атакованной может быть только дуга, у которой охранник размещён в начальной вершине и нет охранника в конечной вершине. Ограничения (7)–(9) задачи нижнего уровня формализуют правила возможных перемещений охранников вдоль соответствующих дуг. Условие (7) означает, что если дуга атакована, то охранник перемещается в конечную вершину дуги. Из неравенства (8) следует, что если в рассматриваемом на этом шаге вершинном покрытии в вершине $i$ размещён охранник, то его перемещение возможно не более чем по одной дуге, исходящей из этой вершины, а если в вершине $i$ охранника нет, то перемещение охранников по дугам, исходящим из вершины $i$, невозможно. Аналогично из неравенства (9) с учётом условия (8) получаем, что если в рассматриваемом на этом шаге вершинном покрытии в вершине $i$ нет охранника, то $\sum\limits_{j\colon k(j)=i} v_j \leqslant 1$. Это означает, что если охранник не размещён в вершине $i$, то не более чем по одной дуге, входящей в вершину $i$, может быть перемещён охранник. Если же в вершине $i$ размещён охранник, то из неравенств (8) и (9) получаем

$$1 \geqslant \sum_{j\colon i(j)=i} v_j \geqslant \sum_{j\colon k(j)=i} v_j.$$

Это означает, что по одной из дуг, входящих в вершину $i$, может быть перемещён охранник, но только в случае, когда охранник, размещённый в вершине $i$, также перемещается. Равенство (10) определяет новое размещение охранников в результате их перемещения на рассматриваемом шаге. Неравенство (11) заставляет переменную $u_j$ принимать ненулевое значение, когда в начальной и конечной вершинах дуги $j$ нет охранников.

Допустимым решением двухуровневой задачи на рассматриваемом шаге игры назовём пару $((y_j), ((x_i), (v_j), (u_j)))$, где $(y_j)$ — допустимое решение задачи верхнего уровня, а $((x_i), (v_j), (u_j))$ — оптимальное решение задачи нижнего уровня. Допустимое решение будет оптимальным, если значение целевой функции на этом решении не меньше, чем на других

допустимых решениях. Вычисление оптимального решения позволяет проверить текущее вершинное покрытие на устойчивость к атаке. Если оптимальное значение целевой функции равно нулю, то ответ положительный. Если же это значение больше нуля, то существует допустимое решение задачи верхнего уровня (атакуемое ребро) для которого не существует реконструкции вершинного покрытия, приводящей к новому вершинному покрытию.

## 2. Устойчивость вершинных покрытий

Как следует из сказанного выше, возможность восстановления (реконструкции) вершинного покрытия после атаки определяется значениями целевых функций задач верхнего и нижнего уровней в двухуровневой модели (1)–(12), рассматриваемой на каждом шаге исследуемой игры.

Вершинное покрытие $C_0 = (x_{i0})$ будем называть *устойчивым к атаке дуги* $j_0 \in J$, если оптимальное значение целевой функции задачи Защитника равно нулю при допустимом решении $(y_j)$, $y_{j_0} = 1$, задачи Атакующего.

Вершинное покрытие $C_0 = (x_{i0})$ назовём 1-*устойчивым,* если оптимальное значение целевой функции задачи Атакующего равно нулю, иначе покрытие $C_0$ будем называть *неустойчивым.*

Отсюда следует, что процедура проверки 1-устойчивости заданного вершинного покрытия $C_0 = (x_{i0})$ сводится к проверке устойчивости этого покрытия к атаке всякой допустимой для атаки дуги $j_0 \in J$. Эта проверка может быть ускорена, если воспользоваться следующей леммой.

**Лемма 1.** *Пусть* $(y_j)$ *— допустимое решение задачи Атакующего и существует решение* $((x_i), (v_j), (u_j))$ *задачи Защитника при решении* $(y_j)$ *со значением целевой функции, равным нулю. Если* $j' \in \{j \in J \mid v_j = 1\}$ *и допустимое решение* $(y'_j)$ *задачи Атакующего таково, что* $y'_{j'} = 1$*, то оптимальное значение целевой функции задачи Защитника равно нулю при решении задачи Атакующего* $(y'_j)$.

Доказательство. Рассмотрим оптимальное решение $((x_i), (v_j), (u_j))$ задачи Защитника, соответствующее решению $(y_j)$ задачи Атакующего, и заметим, что это решение есть допустимое решение задачи Защитника при решении $(y'_j)$. Действительно, ограничение (7) выполняется, поскольку $v_{j'} = 1$, а ограничения (8)–(12) справедливы, поскольку не зависят от решения $(y'_j)$. Таким образом, получаем, что $((x_i), (v_j), (u_j))$ — оптимальное решение задачи Защитника при решении $(y'_j)$ со значением целевой функции, равным нулю.

**Следствие 1.** *Если для заданного вершинного покрытия при атаке дуги* $j \in J$ *существует реконструкция, дающая* 1-*устойчивое покрытие,*

*определяемое решением* $((x_i),(v_j),(u_j))$ *задачи Защитника, то такая реконструкция возможна при атаке любой дуги* $j' \in \{j \in J \mid v_j = 1\}$.

Резюмируя сказанное, получим следующую процедуру проверки 1-устойчивости заданного вершинного покрытия $C_0 = (x_{i0})$. Процедура состоит из начального шага и некоторого числа однотипных основных шагов.

На начальном шаге определяем множество

$$J_0 = \{j \in J \mid x_{i(j)0} = 1,\ x_{k(j)0} = 0\},$$

задающее допустимые решения задачи Атакующего.

На основном шаге рассматриваем множество $J_0$. Если $J_0 = \varnothing$, то процедура проверки 1-устойчивости завершится с положительным результатом. Если $J_0 \neq \varnothing$, то выбираем $j_0 \in J_0$ и вычисляем оптимальное решение $((x_i),(v_j),(u_j))$ задачи Защитника при допустимом решении $(y_j)$, $y_{j_0} = 1$, задачи Атакующего. Если оптимальное значение целевой функции задачи не равно нулю, то процедура проверки завершается с отрицательным результатом. В противном случае полагаем $J_0 := J_0 \setminus \{j \in J \mid v_j = 1\}$ и переходим к следующему основному шагу.

## 3. Устойчивость вершинного покрытия в игре с конечным числом шагов

Рассмотрим игру, состоящую из $L > 1$ шагов, с начальным вершинным покрытием $C_0 = (x_{i0})$. Пусть на шагах $1, 2, \ldots, L-1$ оптимальное значение целевой функции задачи Атакующего равно нулю, а на шаге $L$ оно больше нуля. Это означает, что в результате $L-1$ шагов получено вершинное покрытие такое, что после атаки некоторой дуги успешная реконструкция этого вершинного покрытия невозможна. Заметим, что отсюда не следует, что при начальном покрытии $C_0$ у Защитника нет выигрышной стратегии. Действительно, полученное к шагу $L$ вершинное покрытие есть результат последовательных изменений вершинных покрытий, построенных на предыдущих шагах. Каждое такое изменение определяется оптимальным решением задачи Защитника, которое может быть не единственным. Следовательно, при соответствующем выборе оптимальных решений на шагах, предшествующих шагу $L$, может быть получено вершинное покрытие, которое будет устойчивым к атаке любого ребра на шаге $L$.

Вершинное покрытие $C_0 = (x_{i0})$ назовём *$L$-устойчивым,* если оно 1-устойчиво и для любого допустимого решения задачи Атакующего существует оптимальное решение задачи Защитника, дающее $(L-1)$-устойчивое вершинное покрытие. Если покрытие $L$-устойчиво, то число $L$ будем называть *степенью устойчивости.*

Следовательно, при проверке $L$-устойчивости заданного вершинного покрытия для каждого допустимого решения задачи Атакующего может потребоваться исследование, вообще говоря, всех оптимальных решений задачи Защитника. Таким образом, алгоритм проверки $L$-устойчивости исходного вершинного покрытия $C_0 = (x_{i0})$ включает два этапа. На первом этапе проверяем 1-устойчивость исходного вершинного покрытия, а на втором этапе для допустимых решений задачи Атакующего проверяем существование оптимального решения задачи Защитника, дающего $(L-1)$-устойчивое вершинное покрытие.

При этом в ходе работы алгоритма формируем и используем список $D$ проверенных вершинных покрытий, содержащий информацию о степенях устойчивости этих покрытий. Для вершинного покрытия $C \in D$ определяем величины $v(C)$ и $u(C)$, при этом $v(C) = 0$, если $C$ не 1-устойчиво, и $v(C) = l$, если $C$ — $l$-устойчивое вершинное покрытие; $u(C) = l$, если $C$ не является $l$-устойчивым покрытием.

В начале первого этапа алгоритма проверяем наличие вершинного покрытия $C_0$ в списке $D$. Если $C_0 \in D$ и $v(C_0) = 0$, то вершинное покрытие не 1-устойчиво, и алгоритм проверки $L$-устойчивости завершает работу. Если $C_0 \in D$ и $u(C_0) \leqslant L$, то вершинное покрытие не $L$-устойчиво и алгоритм также завершает работу. Если $C_0 \in D$ и $v(C_0) > 0$, то начинается второй этап алгоритма. Если $C_0 \notin D$, то выполняем процедуру проверки 1-устойчивости $C_0$, и в зависимости от результата завершаем проверку $L$-устойчивости $C_0$ с отрицательным исходом или полагаем $v(C_0) = 1$, включаем $C_0$ в список $D$ и начинаем второй этап алгоритма.

Второй этап алгоритма аналогичен процедуре проверки 1-устойчивости вершинного покрытия. На начальном шаге формируем множество $J_0 \subseteq J$, задающее допустимые решения задачи Атакующего. Далее идёт последовательность основных шагов, на каждом из которых для некоторого $j \in J_0$ проверяем существование $(L-1)$-устойчивого вершинного покрытия, полученного после атаки дуги $j$. Несложно увидеть, что число таких шагов можно уменьшить аналогично тому, как это сделано для случая проверки 1-устойчивости в разд. 2.

На основном шаге рассматриваем множество $J_0$, задающее допустимые и не просмотренные решения задачи Атакующего. Если $J_0 = \varnothing$, то исходное вершинное покрытие $C_0$ $L$-устойчиво, и алгоритм завершается с положительным результатом. Если $J_0 \neq \varnothing$, то выбираем $j_0 \in J_0$ и выполняем процедуру поиска оптимального решения задачи Защитника при атаке дуги $j_0$, дающего $(L-1)$-устойчивое вершинное покрытие. В ходе этой процедуры, включающей некоторое число однотипных итераций, строим множество *запрещённых* вершинных покрытий $C(j_0)$, включающее рассмотренные и отброшенные покрытия $C'$, не являющиеся $(L-1)$-устойчивыми. На первой итерации $C(j_0) = \varnothing$.

На каждой итерации рассматриваем задачу Защитника с дополнительными ограничениями, отсекающими решения, приводящие к вершинным покрытиям из множества $C(j_0)$. Если $C' = (x'_i)$ и $C' \in C(j_0)$, то соответствующее дополнительное ограничение имеет вид

$$\sum_{i \in I} x'_i x_i \leqslant \sum_{i \in I} x'_i - 1.$$

Пусть на очередной итерации оптимальное значение целевой функции задачи Защитника с дополнительными ограничениями, отсекающими запрещённые вершинные покрытия из множества $C(J_0)$, равно нулю. Если $((x_i), (v_j), (u_j))$ — оптимальное решение, то $C = (x_i)$ — новое вершинное покрытие. Если $C \in D$ и $v(C) = 0$ или $u(C) \leqslant L - 1$, то включаем $C$ в множество $C(j_0)$ и начинаем следующую итерацию на текущем шаге. Если $C \in D$ и $v(C) \geqslant L-1$, то полагаем $J_0 = J_0 \setminus \{j \mid v_j = 1\}$ и переходим к следующему шагу. Если $C \in D$ и $0 \leqslant v(C) < L - 1$, либо $C \notin D$, то обращаемся к алгоритму проверки $(L-1)$-устойчивости покрытия $C$. Если алгоритм проверки завершается с положительным результатом, то полагаем $J_0 = J_0 \setminus \{j \mid v_j = 1\}$ и включаем $C$ в список $D$, полагая $v(C) = L-1$. После этого переходим к следующему основному шагу. Если алгоритм проверки $(L-1)$-устойчивости покрытия $C$ завершается с отрицательным результатом, то включаем покрытие в множество запрещённых покрытий $C(j_0)$ и начинаем следующую итерацию на текущем шаге.

Пусть на очередной итерации текущего шага значение целевой функции задачи Защитника с дополнительными ограничениями больше нуля. Это означает, что вершинных покрытий, кроме рассмотренных и включённых в множество запрещённых, не существует, поэтому исходное вершинное покрытие не $L$-устойчиво, и алгоритм проверки $L$-устойчивости завершается с отрицательным результатом.

Описанный алгоритм рекурсивный, и в ходе его работы при проверке $l$-устойчивости покрытия $C$ проводится проверка $(l-1)$-устойчивости некоторого множества вершинных покрытий $C'$, порождённых покрытием $C$. Однако число вызовов функции проверки устойчивости и в целом трудоёмкость алгоритма значительно снижаются за счёт использования списка $D$ проверенных вершинных покрытий. В результате каждое вершинное покрытие, рассмотренное в ходе работы алгоритма, проверяется на $l$-устойчивость, $l = 1, 2, \ldots, L-1$, не более одного раза.

## 4. Показательный пример

На рис. 1 приведён пример графа и его вершинного покрытия, которое 1-устойчиво, но не 2-устойчиво.

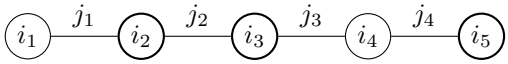



*Рис. 1.* Исходное покрытие

На рис. 1 множество выделенных вершин $\{i_2, i_3, i_5\}$, в которых размещены охранники, образует вершинное покрытие. На первом шаге у Атакующего есть возможность атаковать одно из рёбер $j_1, j_3, j_4$. Легко видеть, что при любой атаке перемещение охранников по рёбрам $j_1, j_3, j_4$ соответственно приведёт к новому вершинному покрытию. Вместе с тем у Атакующего есть выигрышная стратегия в игре из двух шагов. Если на первом шаге атаковать ребро $j_3$, то Защитник переместит охранника из вершины $i_3$ в вершину $i_4$, что приведёт к размещению охранников, показанному на рис. 2.

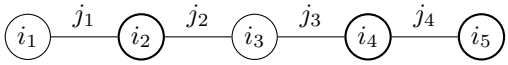



*Рис. 2.* Результат первой атаки

У Защитника есть также возможность переместить охранника из вершины $i_2$ в вершину $i_1$ или $i_3$. Однако любое из этих перемещений нарушит вершинное покрытие. Перемещение охранника из вершины $i_5$ недопустимо, так как в единственную соседнюю вершину $i_4$ уже был перемещён охранник. Таким образом, единственное возможно перемещение охранников на первом шаге приводит к вершинному покрытию $\{i_2, i_4, i_5\}$.

На втором шаге Атакующий атакует ребро $j_2$. Защитник обязан переместить охранника из вершины $i_2$ в вершину $i_3$. Тогда получаем размещение охранников, показанное на рис. 3.

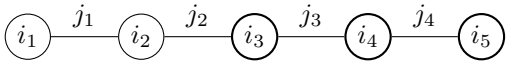



*Рис. 3.* Результат второй атаки

Других возможностей для перемещения охранников у Защитника нет, так как у охранников, находящихся в вершинах $i_4$ и $i_5$, соседние вершины заняты. При этом ребро $j_1$ оказалось не защищённым. Таким образом, исходное покрытие $\{i_2, i_3, i_5\}$ не 2-устойчиво.

## 5. Вычислительные эксперименты

В ходе проведённых экспериментов исследованы вычислительные возможности предложенного алгоритма проверки устойчивости вершинных

покрытий, а также влияние на его производительность некоторых используемых процедур. Расчёты проводились на компьютере с четырёхъядерным процессором 3,3 ГГц и 16 ГБ оперативной памяти. Алгоритмы написаны на языке программирования Julia [10], а для поиска оптимальных решений возникающих задач целочисленного линейного программирования использовался решатель Cbc COIN-OR [11].

**5.1. Генерация графов.** В экспериментах используем случайно генерируемые связные графы разной плотности. Плотность определяется как отношение числа рёбер в графе к числу рёбер в клике с тем же числом вершин.

Построение графа с заданным числом вершин $n$ начинается с построения остовного дерева. Такое дерево может иметь специальный вид (путь, звезда и т. д.), а может быть случайно сгенерированным. Случайно сгенерированное дерево получается в результате последовательного просмотра вершин и соединения рассматриваемой вершины со случайно выбранной ранее просмотренной вершиной. Затем случайным образом выбирается некоторое число дополнительных рёбер, каждое из которых соединяет некоторую пару случайно выбранных вершин.

**5.2. Генерация покрытий.** Поскольку нахождение минимального покрытия есть труднорешаемая задача, в экспериментах используются начальные покрытия, у которых ни одно из собственных подмножеств не остаётся покрытием. Такие покрытия назовём *неприводимыми*. Граф может иметь несколько неприводимых покрытий разных размеров, и генерация различных неприводимых покрытий в наших экспериментах организована как рандомизированный жадный алгоритм.

Алгоритм строит неприводимое покрытие в два этапа. В начале первого этапа все вершины графа считаются непомеченными. Далее последовательно случайным образом выбираются тройки непомеченных вершин и помечается та вершина, которая инцидентна наибольшему числу рёбер с непомеченными концевыми вершинами. Этот процесс повторяется до тех пор, пока не останется рёбер, у которых не помечены обе концевые вершины. На втором этапе в некотором случайно выбранном порядке просматриваются помеченные вершины. Если у всех рёбер, инцидентных рассматриваемой вершине, обе концевые вершины помечены, то рассматриваемая вершина становится непомеченной. Вершины, оставшиеся помеченными после второго этапа, образуют неприводимое вершинное покрытие.

**5.3. Влияние списка проверенных покрытий.** Здесь приводятся результаты вычислительного эксперимента, позволяющего оценить влияние использования списка проверенных вершинных покрытий на время

работы алгоритма проверки устойчивости покрытий. Рассмотрено 300 примеров вершинных покрытий различных графов с числом вершин 15, которые проверялись на 3-устойчивость. Для рассматриваемых примеров покрытий, которые оказались 3-устойчивыми, сравнивается время работы алгоритма с использованием списка проверенных покрытий и без использования списка. Время исчисляется числом обращений в ходе работы алгоритма к процедуре проверки устойчивости покрытия к атаке ребра, т. е. числом вычислений оптимального решения задачи Защитника. Для каждого примера вычисляется величина $s = \left(1 - \frac{t_1}{t_2}\right) \cdot 100$, где $t_1$ — время работы алгоритма с использованием списка, $t_2$ — время работы алгоритма без использования списка. Эта величина характеризует то, насколько велико уменьшение времени работы алгоритма при использовании списка по сравнению со временем работы алгоритма без использования списка.

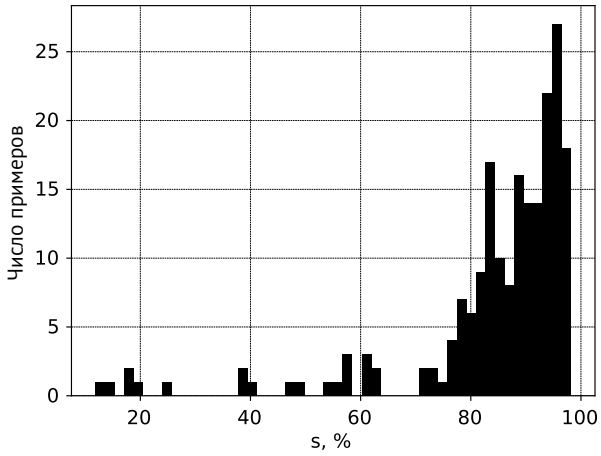



*Рис. 4.* Ускорение алгоритма при использовании списка

На рис. 4 представлена гистограмма распределения полученных значений $s$. По горизонтали указаны возможные значения $s$, а по вертикали число примеров, соответствующих данному $s$. Как можно видеть, в большинстве случаев использование списка сокращает время работы алгоритма более чем на 80%. Это, в частности, свидетельствует о большой повторяемости вершинных покрытий, рассматриваемых в ходе работы алгоритма.

**5.4. Время работы алгоритма.** Проведённые вычислительные эксперименты позволяют сделать некоторые выводы о времени работы алгоритма при проверке $L$-устойчивости вершинных покрытий в зависимости

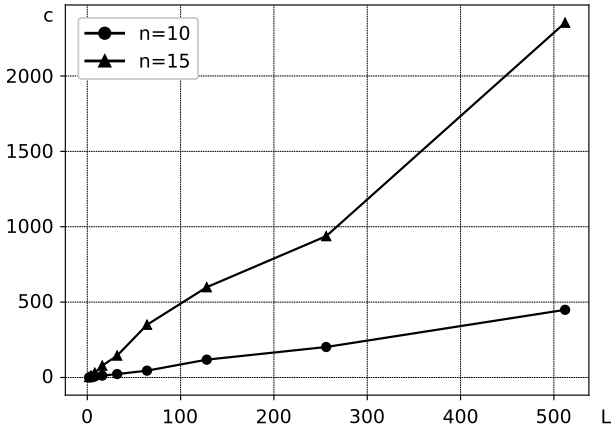



*Рис. 5.* Зависимость времени работы алгоритма от степени устойчивости $L$

от числа вершин графа и проверяемой степени устойчивости вершинного покрытия.

На рис. 5 показана зависимость времени работы алгоритма (в секундах) от степени устойчивости $L$ для числа вершин $n \in \{10, 15\}$. В ходе эксперимента для каждого значения $L$ и каждого числа вершин рассмотрено 75 примеров вершинных покрытий различных графов. Из этих покрытий были удалены неустойчивые, а для остальных вычислено среднее время работы алгоритма. Как можно видеть, временная сложность алгоритма возрастает линейно при увеличении $L$.

На рис. 6 показана зависимость времени работы алгоритма в секундах от числа вершин графа $n$ для значения устойчивости $L \in \{1, 2, 3\}$. В ходе эксперимента для каждого $n$ и каждого $L$ сгенерировано 75 примеров

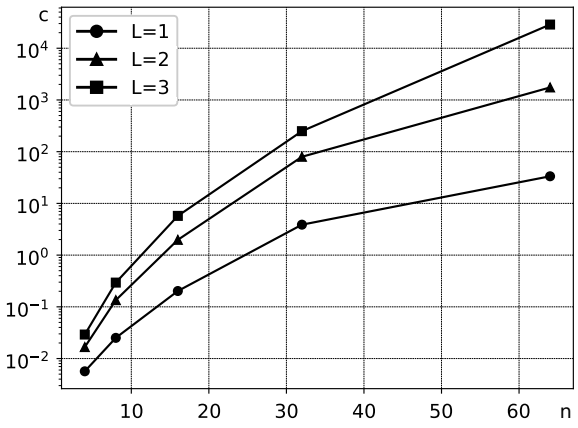



*Рис. 6.* Зависимость времени работы алгоритма от числа вершин $n$

вершинных покрытий различных графов с числом вершин $n$. Из этих покрытий удалены неустойчивые, а для остальных вычислено среднее время работы алгоритма. Заметим, что время проверки 3-устойчивости для графов размера до 32 вершин не превышает 16 минут.

**5.5. Распределение степеней устойчивости.** В этом пункте приведём сводные результаты численных экспериментов. В экспериментах рассмотрены графы с числом вершин до 64 и различной плотностью рёбер. Проверена $L$-устойчивость вершинных покрытий для значений $L$ от 2 до 512. При каждом эксперименте, состоящем в проверке $L$-устойчивости рассматриваемого вершинного покрытия и завершённом установлением фактической степени устойчивости $l$, $0 \leqslant l \leqslant L$, этого вершинного покрытия, вычислена величина $r = \frac{l}{L}$. Значение $r = 0$ означает, что проверенное покрытие неустойчиво, а $r = 1$ — что проверка $L$-устойчивости рассматриваемого покрытия завершилась успешно. Результат $0 < r < 1$ означает, что степень устойчивости покрытия меньше проверяемой.

Из 2714 случайно сгенерированных и проверенных вершинных покрытий, 1207 попали в категорию неустойчивых с $r = 0$, а 87 — в промежуточную категорию с $0 < r < 1$: степень устойчивости оказалась ниже проверяемой. У оставшихся 1420 покрытий степень устойчивости совпала с проверяемой. Таким образом, подавляющее число рассматриваемых вершинных покрытий либо оказались неустойчивыми, либо имели степень устойчивости не меньше той, которая проверялась.

## Заключение

В настоящей работе предложена процедура, позволяющая ответить на вопрос, существует ли выигрышная стратегия для Защитника, следуя которой он может защищать заданное вершинное покрытие на протяжении определённого конечного числа шагов. Для построения стратегии Защитника задача о вечном вершинном покрытии представляется как динамическая игра Штакельберга, на каждом шаге которой взаимодействие противоборствующих сторон формализуется в виде двухуровневой задачи математического программирования. Предложенная процедура протестирована на различных случайно сгенерированных графах и вершинных покрытиях.

Вычислительные эксперименты показали результаты, позволяющие предположить, что большая часть случайных покрытий либо неустойчивы, либо имеют проверяемую степень устойчивости. Таким образом, следующий шаг в исследовании рассматриваемой игры состоит в создании алгоритма, который за конечное число шагов сможет определить, является ли вершинное покрытие бесконечно устойчивым или нет.

**Финансирование работы**

Работа выполнена в рамках государственного задания Института математики им. С. Л. Соболева СО РАН (проект № FWNF–2022–0019).

**Конфликт интересов**

Авторы заявляют, что у них нет конфликта интересов.

**Литература**

1. **Klostermeyer W. F., Mynhardt C. M.** Protecting a graph with mobile guards // Appl. Anal. Discrete Math. 2016. V. 10, No. 1. P. 1–29. DOI: 10.2298/AADM151109021K.
2. **Klostermeyer W. F., Mynhardt C. M.** Edge protection in graphs // Australas. J. Comb. 2009. V. 45. P. 235–250.
3. **Fomin F. V., Gaspers S., Golovach P. A., Kratsch D., Saurabh S.** Parameterized algorithm for eternal vertex cover // Inf. Process. Lett. 2010. V. 110, No. 17. P. 702–706. DOI: 10.1016/j.ipl.2010.05.029.
4. **Babu J., Misra N., Nanoti S. G.** Eternal vertex cover in bipartite graphs // Computer science — Theory and applications. Proc. 17th Int. Comp. Sci. Symp. in Russia (St. Petersburg, Russia, June 29 – July 1, 2022). Cham: Springer, 2022. P. 64–76. (Lect. Notes Comput. Sci.; Vol. 13296). DOI: 10.1007/978-3-031-09574-0_5.
5. **Babu J., Prabhakaran V.** A new lower bound for the eternal vertex cover number of graphs // J. Comb. Opt. 2022. V. 44. P. 2482–2498. DOI: 10.1007/s10878-021-00764-8.
6. **Babu J., Prabhakaran V., Sharma A.** A substructure based lower bound for eternal vertex cover number // Theor. Comput. Sci. 2021. V. 890. P. 87–104. DOI: 10.1016/j.tcs.2021.08.018.
7. **Araki H., Fujito T., Inoue S.** On the eternal vertex cover numbers of generalized trees // IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 2015. V. E98-A, No. 6. P. 1153–1160. DOI: 10.1587/transfun.E98.A.1153.
8. **Paul K., Pandey A.** Some algorithmic results for eternal vertex cover problem in graphs // WALCOM: Algorithms and computation. Proc. 17th Int. Conf. and Workshops (Hsinchu, Taiwan, Mar. 22–24, 2023). Cham: Springer, 2023. P. 242–253. (Lect. Notes Comput. Sci.; Vol. 13973). DOI: 10.1007/978-3-031-27051-2_21.
9. **Beresnev V. L., Melnikov A. A., Utyupin S. Yu.** Representation of the eternal vertex cover problem as a dynamic Stackelberg game // Optimization and applications. Rev. Sel. Pap. 14th Int. Conf. (Petrovac, Montenegro, Sept. 18–22, 2023) Cham: Springer, 2023. P. 3–13. (Lect. Notes Comput. Sci.; V. 14395). DOI: 10.1007/978-3-031-47859-8_1.
10. **Bezanson J., Edelman A., Karpinski S., Shah V. B.** Julia: A fresh approach to numerical computing // SIAM Rev. 2017. V. 59, No. 1. P. 65–98. DOI: 10.1137/141000671.

**11.** COIN-OR Branch-and-cut solver. Towson: COIN-OR Found., 2023. URL: `github.com/coin-or/Cbc` (accessed Mar. 27, 2024).

*Береснев Владимир Леонидович*
*Мельников Андрей Андреевич*
*Утюпин Степан Юрьевич*

Статья поступила
26 февраля 2024 г.
После доработки —
14 марта 2024 г.
Принята к публикации
21 марта 2024 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2024. Vol. 31, No. 2. P. 28–45

UDC 519.8 DOI: 10.33048/daio.2024.31.797

# STABILITY OF VERTEX COVERS IN A GAME WITH FINITELY MANY STEPS

*V. L. Beresnev*[1, a], *A. A. Melnikov*[1, b], and *S. Yu. Utyupin*[2, c]

[1] Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk, Russia
[2] Novosibirsk State University,
2 Pirogov Street, 630090 Novosibirsk, Russia
E-mail: [a] beresnev@math.nsc.ru,
[b] melnikov@math.nsc.ru, [c] stepan.utyupin@gmail.com

**Abstract.** The eternal vertex cover problem is a version of the graph vertex cover problem that can be represented as a dynamic game between two players (the Attacker and the Defender) with an infinite number of steps. At each step, there is an arrangement of guards over the vertices of the graph forming a vertex cover. When the Attacker attacks one of the graph's edges, the Defender must move the guard along the attacked edge from one vertex to the other. In addition, the Defender can move any number of other guards from their current vertices to some adjacent ones to obtain a new vertex cover. The Attacker wins if the Defender cannot build a new vertex cover after the attack.

In this paper, we propose a procedure that allows us to answer the question, whether there exists a winning Defender strategy that permits protecting a given vertex cover for a given finite number of steps. To construct the Defender strategy, the problem is represented as a dynamic Stackelberg game in which at each step the interaction of the opposing sides is formalized as a two-level mathematical programming problem. The idea of the procedure is to recursively check the 1-stability of vertex covers obtained as a result of solving lower-level problems and to use some information about the covers already considered. Illustr. 6, bibliogr. 11.

**Keywords:** dynamic Stackelberg game, eternal vertex cover, graph edge protection, stability check algorithm.

English transl.: *Journal of Applied and Industrial Mathematics* **18** (2), 206–215 (2024), DOI: 10.1134/S1990478924020030.

## References


1. **W. F. Klostermeyer** and **C. M. Mynhardt,** Protecting a graph with mobile guards, *Appl. Anal. Discrete Math.* **10** (1), 1–29 (2016), DOI: 10.2298/AADM151109021K.
2. **W. F. Klostermeyer** and **C. M. Mynhardt,** Edge protection in graphs, *Australas. J. Comb.* **45**, 235–250 (2009).
3. **F. V. Fomin, S. Gaspers, P. A. Golovach, D. Kratsch,** and **S. Saurabh,** Parameterized algorithm for eternal vertex cover, *Inf. Process. Lett.* **110** (17), 702–706 (2010), DOI: 10.1016/j.ipl.2010.05.029.
4. **J. Babu, N. Misra,** and **S. G. Nanoti,** Eternal vertex cover in bipartite graphs, in *Computer Science — Theory and Applications* (Proc. 17th Int. Comp. Sci. Symp. in Russia, St. Petersburg, Russia, June 29 – July 1, 2022) (Springer, Cham, 2022), pp. 64–76 (Lect. Notes Comput. Sci., Vol. 13296), DOI: 10.1007/978-3-031-09574-0_5.
5. **J. Babu** and **V. Prabhakaran,** A new lower bound for the eternal vertex cover number of graphs, *J. Comb. Opt.* **44**, 2482–2498 (2022), DOI: 10.1007/s10878-021-00764-8.
6. **J. Babu, V. Prabhakaran,** and **A. Sharma,** A substructure based lower bound for eternal vertex cover number, *Theor. Comput. Sci.* **890**, 87–104 (2021), DOI: 10.1016/j.tcs.2021.08.018.
7. **H. Araki, T. Fujito,** and **S. Inoue,** On the eternal vertex cover numbers of generalized trees, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E98-A** (6), 1153–1160 (2015), DOI: 10.1587/transfun.E98.A.1153.
8. **K. Paul** and **A. Pandey,** Some algorithmic results for eternal vertex cover problem in graphs, in *WALCOM: Algorithms and Computation* (Proc. 17th Int. Conf. and Workshops, Hsinchu, Taiwan, Mar. 22–24, 2023) (Springer, Cham, 2023), pp. 242–253 (Lect. Notes Comput. Sci., Vol. 13973), DOI: 10.1007/978-3-031-27051-2_21.
9. **V. L. Beresnev, A. A. Melnikov,** and **S. Yu. Utyupin,** Representation of the eternal vertex cover problem as a dynamic Stackelberg game, in *Optimization and Applications* (Rev. Sel. Pap. 14th Int. Conf., Petrovac, Montenegro, Sept. 18–22, 2023) (Springer, Cham, 2023), pp. 3–13 (Lect. Notes Comput. Sci., Vol. 14395), DOI: 10.1007/978-3-031-47859-8_1.
10. **J. Bezanson, A. Edelman, S. Karpinski,** and **V. B. Shah,** Julia: A fresh approach to numerical computing, *SIAM Rev.* **59** (1), 65–98 (2017), DOI: 10.1137/141000671.
11. COIN-OR Branch-and-Cut solver (COIN-OR Found., Towson, 2023). URL: github.com/coin-or/Cbc (accessed Mar. 27, 2024).


*Vladimir L. Beresnev*
*Andrey A. Melnikov*
*Stepan Yu. Utyupin*

Received February 26, 2024
Revised March 14, 2024
Accepted March 21, 2024