электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51036. ISSN 2307-0595

Задача выбора средств защиты информации в автоматизированных системах на основе модели антагонистической игры

 # 04, апрель 2014
УДК: 004.056+519.83+519.854
авторы: Быков А. Ю., Алтухов Н. О., Сосенко А. С.

Россия, МГТУ им. Н.Э. Баумана
abykov@bmstu.ru
nik-altukhov@yandex.ru
sosenko1995@gmail.com

Введение

При проектировании подсистем в автоматизированных системах могут быть использованы математические постановки задач выбора программно-аппаратных средств различного целевого назначения [1-4]. Далее рассмотрим задачу выбора средств на примере выбора средств защиты информации (СЗИ).

В работе [2] предложена математическая постановка задачи выбора средств СЗИ для вычислительной сети с целью оптимизации следующих показателей: показателя стоимости средств защиты; показателя, задающего возможный предотвращенный ущерб при использовании выбранных средств защиты (показатель эффективности). Данные постановки задач являлись задачами булевого программирования [5]. В [6] рассмотрен один из методов решения подобного типа задач – метод вектора спада, являющийся приближенным методом. В данных работах предложены модели для принятия решения, большей частью, с учетом взгляда с одной стороны – стороны защиты информации.

При защите от различных угроз безопасности присутствуют две стороны: сторона нарушителя или сторона нападения, которая преследует свои цели, в качестве нарушителя можно рассматривать и «природу» («игра с природой»), в этом случае у нарушителя нет целей, рассматриваются различные случайные воздействия на систему, и сторона защиты от возможных реализаций этих угроз безопасности. Подобные модели принятия решения, когда существуют две или более противоборствующих сторон, рассматриваются в теории «игр» [7]. В [8] для выбора средств защиты от DDoS-атак предложено использовать матричную игру двух игроков с нулевой суммой. Решение по различным критериям оптимальности искалось на платежной матрице игры, но если число средств защиты и число возможных атак велико, то платежная матрица будет иметь большую размерность, искать решения напрямую на такой матрице затруднительно. В [9] рассмотрено применение игровой модели для систем обнаружения вторжений.

Цель данной статьи состоит в разработке игровой математической модели с двумя противоборствующими сторонами и алгоритмов, которые могут быть использованы для выбора СЗИ в автоматизированных системах, для обеспечения гарантированного результата по показателю ущерба от атак для стороны защиты. При этом число средств защиты и число атак могут измеряться десятками, в отдельных случаях сотнями.

В первом разделе представлены: исходные данные для постановки задачи, показатель качества игроков, ограничения игроков, а также модель игры, в которой каждый из игроков решает свою задачу булевого программирования. Во втором разделе рассмотрены критерии оптимальности, предложено использовать критерий гарантированного результата или минимаксный критерий, рассмотрены некоторые алгоритмы решения задачи выбора СЗИ: точные методы неявного перебора на решетках, а также приближенный метод. В третьем разделе представлены: результаты экспериментов по выявлению зависимости времени работы предложенных алгоритмов от размерности задачи, оценки погрешности работы приближенного алгоритма. В четвертом разделе представлены исходные данные и пример решения практической задачи.

1. Математическая постановка задачи

1.1. Исходные данные

1. $A = \{a_1, a_2, \dots, a_n\}$ – множество возможных угроз безопасности или средств проведения атак, которыми может воспользоваться нарушитель, $N = \{1, 2, \dots, n\}$ – множество индексов этих угроз (средств). В качестве возможных угроз здесь можно рассматривать конкретные технические, программные или организационные средства, которые может задействовать сторона нападения.
2. $B = \{b_1, b_2, \dots, b_m\}$ – множество средств защиты информации от угроз безопасности, $M = \{1, 2, \dots, m\}$ – множество индексов средств защиты соответственно.
3. $u_i, \forall\, i \in N$ – средний ущерб от непредотвращения i-ой угрозы за заданный период времени.
4. $c_i^{(н)} \geq 0, \forall\, i \in N$ – стоимость реализации i-ой угрозы стороной нападения;
5. $c_j^{(з)} \geq 0, \forall\, j \in M$ - стоимость j-го средства защиты;
6. $p_{ij} \in [0,\ 1], \forall\, i \in N, \forall\, j \in M$ – возможность, описываемая в рамках теории нечетких множеств, или вероятность (если есть статистика) предотвращения последствий i-ой угрозы с помощью j-го средства защиты, параметры образуют матрицу $P = \|p_{ij}\|$ размерности $n \times m$.

1.2. Показатель качества выбора игроков

Для стороны защиты введем булеву переменную $x_j \in \{0, 1\}, \forall\, j \in M$.

– $x_j = 1$, если j-ое средство защиты будет применяться в автоматизированной системе для защиты от тех или иных угроз;
– $x_j = 0$ в противном случае, то есть если j- ое средство не будет применяться.

Тогда $\vec{X}$ - вектор булевых переменных x_j.

По аналогии для стороны нападения введем булеву переменную $y_i \in \{0,1\}, \forall\, i \in N$:

– $y_i = 1$, если сторона нападения будет использовать i-ое средство атаки (реализовывать i-ую угрозу);
– $y_i = 0$ в противном случае.

Тогда $\vec{Y}$ - вектор булевых переменных y_i.

Ущерб от реализации атак без применения средств защиты для стороны защиты или максимально возможный ущерб определяется как:

$$U^{(max)}(\vec{Y}) = \sum_{i \in N} u_i y_i.$$

При использовании средств защиты некоторая часть этого ущерба будет предотвращена, предотвращенный ущерб в общем случае может быть описан:

$$U^{(\text{пред})}(\vec{X}, \vec{Y}) = \sum_{i \in N} u_i y_i F_i(P, \vec{X}),$$

где $F_i(P, \vec{X}), \forall\, i \in N$ – функции, определяющие степени предотвращения ущерба от каждой из угроз.

В качестве таких функций будем использовать:

$$F_i(P, \vec{X}) = \max_{j \in M}\{p_{ij} x_j\}, \forall\, i \in N.$$

Содержательно функция задает то, что для предотвращения ущерба учитывается только одно выбранное средство защиты, имеющее максимальную возможность (вероятность) предотвращения для i-ой угрозы. В этой функции не учитывается суммарный эффект от совместного использования двух или более средств защиты для предотвращения угрозы.

Тогда реальный ущерб для стороны защиты:

$$U(\vec{X}, \vec{Y}) = U^{(max)}(\vec{Y}) - U^{(\text{пред})}(\vec{X}, \vec{Y}) = \sum_{i \in N} u_i y_i - \sum_{i \in N} u_i y_i \max_{j \in M}\{p_{ij} x_j\}. \quad (1)$$

Данный показатель определяет реальный возможный ущерб для стороны защиты при использовании СЗИ. Сторона защиты старается этот ущерб минимизировать, а сторона нападения старается максимизировать, таким образом, получаем игру двух игроков с нулевой суммой.

1.3. Ограничения

Стоимость используемых средств защиты:

$$C^{(\text{з})}(\vec{X}) = \sum_{j \in M} c_j^{(\text{з})} x_j.$$

При этом сторона защиты ограничена в средствах и может потратить на защиту информации некоторую максимальную сумму $C_{max}^{(з)}$. Тогда ограничения на максимальную стоимость средств защиты определяется неравенством:

$$\sum_{j \in M} c_j^{(з)} x_j \leq C_{max}^{(з)}. \quad (2)$$

По аналогии для стороны нападения введем ограничения на максимальную стоимость используемых для атак ресурсов:

$$\sum_{i \in N} c_i^{(н)} y_i \leq C_{max}^{(н)}, \quad (3)$$

где $C_{max}^{(н)}$ - максимальная стоимость ресурсов, которые может выделить для проведения атак нарушитель.

1.4. Модель игры

Сторона защиты или первый игрок решает задачу минимизации показателя (1) с ограничениями (2):

$$U(\vec{X}, \vec{Y}) = \sum_{i \in N} u_i y_i - \sum_{i \in N} u_i y_i \max_{j \in M}\{p_{ij} x_j\} \longrightarrow \min_{\vec{X} \in \Delta_x^{(доп)}}, \quad (4)$$
$$\Delta_x^{(доп)}: \sum_{j \in M} c_j^{(з)} x_j \leq C_{max}^{(з)}.$$

где $\Delta_x^{(доп)}$ - множество допустимых альтернатив (значений компонент неизвестного вектора $\vec{X}$) для первого игрока. При фиксированных значениях компонент вектора $\vec{Y}$ получаем задачу булевого программирования.

Сторона нападения или второй игрок решает задачу максимизации показателя (1) с ограничениями (3):

$$U(\vec{X}, \vec{Y}) = \sum_{i \in N} u_i y_i - \sum_{i \in N} u_i y_i \max_{j \in M}\{p_{ij} x_j\} \longrightarrow \max_{\vec{Y} \in \Delta_y^{(доп)}}, \quad (5)$$
$$\Delta_y^{(доп)}: \sum_{i \in N} c_i^{(н)} y_i \leq C_{max}^{(н)}.$$

где $\Delta_y^{(доп)}$ - множество допустимых альтернатив (значений компонент неизвестного вектора $\vec{Y}$) для второго игрока. При фиксированных значениях компонент вектора $\vec{X}$ получаем задачу булевого программирования.

2. Критерии оптимальности и алгоритмы решения задачи

2.1. Критерий оптимальности

Представленную модель игры можно свести к игре, заданной платежной матрицей [7, 8]. Проблема в том, что размерность этой матрицы может быть достаточно велика: число строк будет равно числу допустимых значений вектора $\vec{X}$, удовлетворяющих ограничению (2), а число столбцов будет равно числу допустимых значений вектора $\vec{Y}$, удовлетворяющих ограничению (3). В случае использования платежной матрицы часто ищут седловую точку в чистых стратегиях или, если ее не существует, в смешанных стратегиях [7, 8]. При большой размерности платежной матрицы это затруднительно.

Можно использовать различные критерии оптимальности, такие как критерии Лапласа, Вальда, Гурвица или Сэвиджа [8]. С точки зрения обеспечения состояния защищенности информации чаще всего для стороны защиты используют критерий Вальда, так как при проектировании системы защиты необходимо обеспечить гарантированный результат в любых условиях. Критерий Вальда обеспечивает выбор осторожной или пессимистической стратегии, т.е. обеспечивает гарантированный результат при самых худших условиях. Из-за того, что при выборе средств защиты решается защита минимизации возможного ущерба, данный критерий превращается в минимаксный критерий:

$$\min_{\vec{X}\in\Delta_x^{(\text{доп})}} \max_{\vec{Y}\in\Delta_y^{(\text{доп})}} U(\vec{X},\vec{Y}). \qquad (6)$$

При решении задачи максимизации показателя $U(\vec{X},\vec{Y})$ по вектору $\vec{Y}\in\Delta_y^{(доп)}$ при фиксированных значениях вектора $\vec{X}$ или задачи минимизации показателя $U(\vec{X},\vec{Y})$ по вектору $\vec{X}\in\Delta_x^{(доп)}$ при фиксированных значениях вектора $\vec{Y}$ решаются задачи булевого программирования с линейным ограничением (2) или (3) и нелинейным показателем качества (1). Эти задачи относятся к классу *NP*-полных задач оптимизации [10]. Точное решение для таких задач может быть получено алгоритмом с экспоненциальной трудоемкостью. Рассмотрим некоторые точные и приближенные алгоритмы, подходящие для решения задачи по критерию (6).

2.2. Точные алгоритмы

Среди точных методов в булевом программировании всегда можно выделить метод полного перебора. Другие точные методы позволяют сократить некоторым образом полный перебор. Рассмотрим один из методов, который можно применить к данной задаче, – метод неявного перебора на векторной решетке.

Идея метода следующая: между булевыми векторами $\vec{X}$ (или $\vec{Y}$), имеющими различные значения компонент можно ввести отложение доминирования. Отношение доминирования устанавливается между любыми двумя векторами, отличающимися только значением одного элемента. Можно ввести данное отношение двумя способами: значение 0 доминирует значение 1 или наоборот – 1 доминирует 0. На первом уровне решетки располага-

ется недоминируемый вектор, состоящий из всех 0 (или всех 1), на втором уровне – вектора, доминируемые лишь вектором первого уровня и не доминируемые никакими другими векторами и т.д. Для векторов из трех булевых переменных решетка по правилу «0 доминирует 1» представлена на рис. 1, а решетка по правилу «1 доминирует 0» представлена на рис. 2. Решетка является отношением частичного порядка.

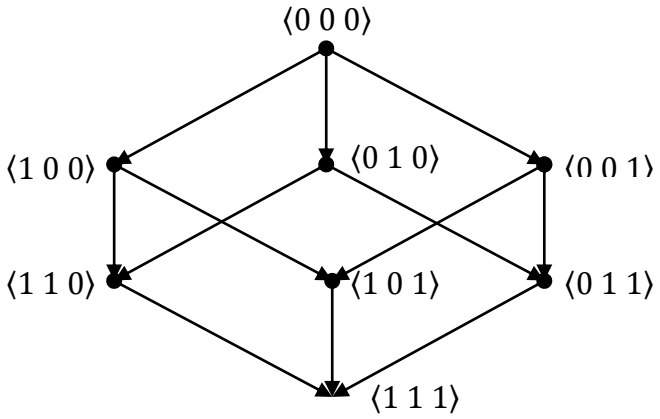

Рис. 1. Пример решетки по правилу «0 доминирует 1»

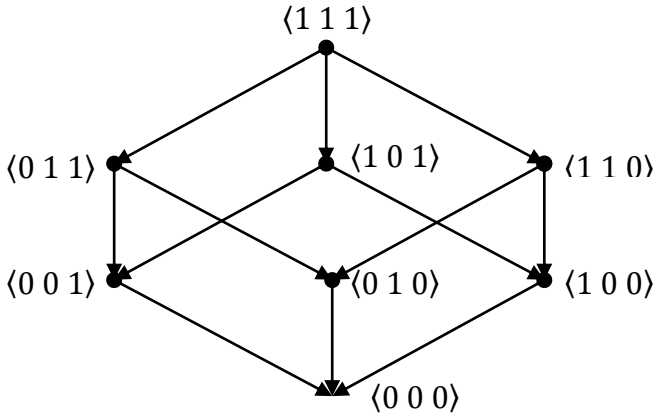

Рис. 2. Пример решетки по правилу «1 доминирует 0»

Элементы решетки в алгоритме получать достаточно легко, если используется правило «0 доминирует 1». Для получения новых, еще не полученных векторов следующего уровня для любого элемента используется правило 1.

Правило 1 получения новых векторов следующего уровня решетки в цикле: для всех последних элементов вектора, равных 0, до первой 1 (или до начала вектора, если вектор состоит из всех 0) последовательно, по одному, значение 0 превращаем в 1 в направлении от начала к концу вектора, каждый раз получаем новый вектор следующего уровня, если в векторе последний элемент 1, то векторов следующего уровня для элемента не существует.

Например, для вектора $\|101000\|^{\mathrm{T}}$ элементами следующего уровня решетки, которые не были получены при рассмотрении предшествующих решений, являются вектора: $\|101100\|^{\mathrm{T}}$, $\|101010\|^{\mathrm{T}}$, $\|101001\|^{\mathrm{T}}$.

Если используется правило «1 доминирует 0», то для получения новых векторов следующего уровня для любого элемента используется правило 2 (обратное правилу 1).

Правило 2 получения новых векторов следующего уровня решетки в цикле: для всех последних элементов вектора, равных 1, до первого 0 (или до начала вектора, если вектор состоит из всех 1) последовательно, по одному, значение 1 превращаем в 0 в направлении от начала к концу вектора, каждый раз получаем новый вектор следующего уровня, если в векторе последний элемент 0, то векторов следующего уровня для элемента не существует.

Процесс получения новых векторов нижнего уровня для текущего вектора (решения) будем называть зондированием этого вектора (решения).

Решетчатое упорядочивание позволяет, учитывая специфику задачи, не осуществлять полный перебор всех элементов. Если используется правило «0 доминирует 1», то элемент решетки, который не удовлетворяет ограничениям (2) или (3) в зависимости от решаемой задачи, можно далее не зондировать, так как эти вектора также будут недопустимыми. Это связано с тем, что коэффициенты в ограничениях (2) и (3) неотрицательные, и максимальная стоимость также неотрицательная (добавление в вектор новой единицы может только увеличить общую стоимость).

Если используется правило «1 доминирует 0», то элемент решетки, который удовлетворяет ограничениям (2) или (3) в зависимости от решаемой задачи, можно далее не зондировать, так как эти вектора будут иметь значение целевой функции (1) не лучше, чем полученное. Это связано с тем, что превращение в векторе 1 в 0, означает то, что некоторое средство (защиты или нападения, в зависимости от решаемой задачи) не будет применено, что не улучшит выигрыш соответствующего игрока. Для реализации алгоритмов перебора на решетке удобно использовать рекурсивные алгоритмы. Рассмотрим два алгоритма поиска решения на решетке.

Алгоритм неявного перебора на решетке, построенной по правилу «0 доминирует 1»

Стартовый алгоритм метода неявного перебора на решетке, построенной по правилу «0 доминирует 1», имеет следующие шаги:

1. Задаем начальное решение, состоящее из всех нулей $\vec{X} = \|0, 0, \dots, 0\|^{\mathrm{T}}$ (в программе массив *X* размерности *m*), полагаем $UMinMax = \infty$.

2. Выполнение рекурсивного алгоритма зондирования решения $\vec{X}$ по правилу «0 доминирует 1».

3. После работы алгоритма зондирования значение показателя для первого игрока, осуществляющего выбор средств защиты, найденное по критерию минимакс (6), будет

равно *UMinMax*, полученное решение будет записано в массиве *XRec*, решение второго игрока будет в массиве *YRec2*.

Рекурсивный алгоритм зондирования решения $\vec{X}$ (массива *X* размерности *m*) по правилу «0 доминирует 1»:

1. Задаем начальное решение $\vec{Y}$, состоящее из всех нулей $\vec{Y} = \|0, 0, \dots, 0\|^{\mathrm{T}}$, (в программе массив *Y* размерности *n*) полагаем *UMax=0*.

2. Выполнение рекурсивного алгоритма зондирования решения $\vec{Y}$ по правилу «0 доминирует 1».

3. Если *UMax<UMinMax*, то полагаем *UMinMax = UMax* и сохраняем решение *YRec* в массиве *YRec2* (рекордное решение по критерию минимакс), сохраняем значение массива *X* в массиве *XRec* (рекордное решение по критерию минимакс).

4. По правилу 1 в цикле получаем решение следующего уровня решетки – новый вектор $\vec{X}_{next}$ (массив *X* размерности *m*). Если решение $\vec{X}_{next}$ допустимое по ограничению (2), то для него запускаем рекурсивный алгоритм зондирования решения $\vec{\mathrm{X}}$ по правилу «0 доминирует 1». Если таких решений не существует, алгоритм прекращает свою работу.

Рекурсивный алгоритм зондирования решения $\vec{Y}$ (массива *Y* размерности *n*) по правилу «0 доминирует 1»:

1. Вычисляем значение показателя (1): $U = U(\vec{X}, \vec{Y})$.

2. Если *U* > *UMax*, то полагаем *UMax* = *U* и сохраняем решение *Y* в массиве *YRec* (рекордное значение по критерию максимум).

3. По правилу 1 в цикле получаем решение следующего уровня решетки – новый вектор $\vec{Y}_{next}$ (массив *Y* размерности *n*). Если решение $\vec{Y}_{next}$ допустимо по ограничению (3), то для него запускаем рекурсивный алгоритм зондирования решения $\vec{Y}$ по правилу «0 доминирует 1». Если таких решений не существует, алгоритм прекращает свою работу.

Алгоритм неявного перебора на решетке, построенной по правилу «1 доминирует 0»

Стартовый алгоритм метода неявного перебора на решетке, построенной по правилу «1 доминирует 0», имеет следующие шаги:

1. Задаем начальное решение, состоящее из всех единиц $\vec{X} = \|1, 1, \dots, 1\|^{\mathrm{T}}$ (в программе массив *X* размерности *m*), полагаем *UMinMax*= ∞.

2. Выполнение рекурсивного алгоритма зондирования решения $\vec{X}$ по правилу «1 доминирует 0».

3. После работы алгоритма зондирования значение показателя для первого игрока, осуществляющего выбор средств защиты, найденное по критерию минимакс (6), будет равно *UMinMax*, полученное решение будет записано в массиве *XRec*, решение второго игрока будет в массиве *YRec2*.

Рекурсивный алгоритм зондирования решения $\vec{X}$ (массива *X* размерности *m*) по правилу «1 доминирует 0»:

1. Если решение $\vec{X}$ допустимо в соответствии с ограничением (2), то задаем начальное решение $\vec{Y}$, состоящее из всех единиц $\vec{Y} = \|1, 1, \dots, 1\|^{\mathrm{T}}$, (в программе массив *Y* размерности *n*) полагаем *UMax=0*, переходим к следующему шагу, в противном случае переходим к шагу 5.

2. Выполнение рекурсивного алгоритма зондирования решения $\vec{Y}$ по правилу «1 доминирует 0».

3. Если *UMax<UMinMax*, то полагаем *UMinMax = UMax* и сохраняем решение *YRec* в массиве *YRec2* (рекордное решение по критерию минимакс), сохраняем значение массива *X* в массиве *XRec* (рекордное решение по критерию минимакс).

4. Алгоритм завершает работу.

5. По правилу 2 в цикле получаем решение следующего уровня решетки новый вектор $\vec{X}_{next}$ (массив *X* размерности *m*), для каждого $\vec{X}_{next}$ запускаем рекурсивный алгоритм зондирования $\vec{X}$ по правилу «1 доминирует 0». Если таких решений не существует, алгоритм завершает свою работу.

Рекурсивный алгоритм зондирования решения $\vec{Y}$ (массива *Y* размерности *n*) по правилу «1 доминирует 0»:

1. Если решение $\vec{Y}$ допустимо в соответствии с ограничением (3), то вычисляем значение показателя (1): $U = U(\vec{X}, \vec{Y})$ и переходим к следующему шагу, в противном случае переходим к шагу 3.

2. Если *U > UMax*, то полагаем *UMax = U* и сохраняем решение *Y* в массиве *YRec* (рекордное значение по критерию максимум), алгоритм завершает работу.

3. По правилу 2 в цикле получаем решение следующего уровня решетки – новый вектор $\vec{Y}_{next}$ (массив *Y* размерности *n*), для каждого $\vec{Y}_{next}$ запускаем рекурсивный алгоритм зондирования $\vec{Y}$ по правилу «1 доминирует 0». Если таких решений не существует, алгоритм завершает свою работу.

2.3. Приближенный алгоритм

Рассмотрим алгоритм приближенного метода решения задачи, некоторые аналоги которого рассмотрены в [11], алгоритм является эвристическим и не гарантирует получение приближенного решения с заданной априорно точностью, т.е. погрешность алгоритма может быть достаточно велика. Применение определенных критериев остановки предохраняет алгоритм от зацикливания. Алгоритм метода включает следующие шаги:

1. Задаем начальное решение, состоящее из всех нулей $\vec{X} = \|0, 0, \dots, 0\|^{\mathrm{T}}$ (в программе массив *X* размерности *m*), полагаем *UMinMax*= ∞.

2. Некоторым методом булевого программирования, возможно приближенным, решаем задачу максимизации (5) при фиксированном векторе $\vec{X}$ (массиве *X*), получаем решение $\vec{Y}$ (массив *Y*) и значение показателя для этого решения *UMax.*

3. Если *UMax<UMinMax*, то полагаем *UMinMax = UMax* и сохраняем решение *Y* в массиве *YRec* (рекордное решение по критерию минимакс), сохраняем значение массива *X* в массиве *XRec* (рекордное решение по критерию минимакс).

4. Проверяем критерий остановки, если он выполняется, алгоритм завершает свою работу, в противном случае переходим к следующему шагу.

5. Некоторым методом булевого программирования, возможно приближенным, решаем задачу минимизации (4) при фиксированном векторе $\vec{Y}$ (массиве *YRec*), получаем решение *X,* переходим к шагу 2.

Учитывая то, что множество решений конечно, то при работе алгоритма решения будут повторяться через некоторое число итераций. **В качестве критериев остановки** можно использовать следующие критерии в порядке очередности их проверки:

- получение повторного решения на шаге 3 со значением показателем качества *UMax,* равным значению сохраненном в переменной *UMinMax*, что не всегда происходит. Иногда решение с минимальным значением показателя получается на одной из первых итераций алгоритма и далее не повторяется, происходит повторение последовательности решений, среди которых нет решения с минимальным значением показателя, тогда используется следующий критерий;

- получение повторного решения на шаге 3, искомым решением будет одно из решений, полученных на предыдущих шагах с минимальным значением показателя, сохраненным в *UMinMax*.

Приближенный алгоритм может быть использован для быстрого поиска допустимого решения в случае задач большой размерности, параметры этого алгоритма, полученные в ходе экспериментов, представлены ниже.

3. Результаты экспериментов для исследования параметров алгоритмов

Представляет интерес исследование практической зависимости времени работы предложенных алгоритмов от размерности задачи, а также оценка погрешности работы приближенного алгоритма. Эксперименты выполнялись на ноутбуке с процессором Intel ® Core ™ i5-2410, тактовая частота 2,3 ГГц, для разработки использовался язык Си++, среда разработки Microsoft Visual Studio 2010.

В двух алгоритмах неявного перебора на решетках выигрыш по времени будет тем больше, чем меньше вершин решетки проверяются алгоритмом. Если используется правило «0 доминирует 1», то просмотр решетки начинается с допустимого решения по ограничению (2) или (3), число просмотренных вершин решетки будет меньше, если будет меньше число допустимых решений, определяемых ограничением (2) или (3). Если используется правило «1 доминирует 0», то просмотр решетки начинается с недопустимого

решения по ограничению (2) или (3), число просмотренных вершин решетки будет меньше, если будет меньше число недопустимых решений, определяемых ограничением (2) или (3). Соотношение чисел допустимых и недопустимых решений зависит от отношения суммы коэффициентов $\sum_{j \in M} c_j^{(з)}$ к $C_{max}^{(з)}$ для ограничения (2) и отношения $\sum_{j \in N} c_j^{(н)}$ к $C_{max}^{(н)}$ для ограничения (3). Это отношение является положительным значением, назовем его степенью недопустимости ограничения. Исследуем также зависимость времени работы точных алгоритмов от степени недопустимости ограничений. Исходные данные будем задавать с помощью генераторов псевдослучайных чисел (ПСЧ). В этом случае степень недопустимости ограничения определяется отношением средней суммы коэффициентов в ограничении к средней максимальной стоимости средств (правой части ограничения (2) или (3)) при использовании генераторов ПСЧ для получения чисел с равномерным распределением.

Результаты экспериментов по оценке времени работы разных алгоритмов в зависимости от числа переменных *m* представлены графиком на рис. 3. При этом *n=15*, степени недопустимости двух ограничений (2) и (3) одинаковые и равны 2.

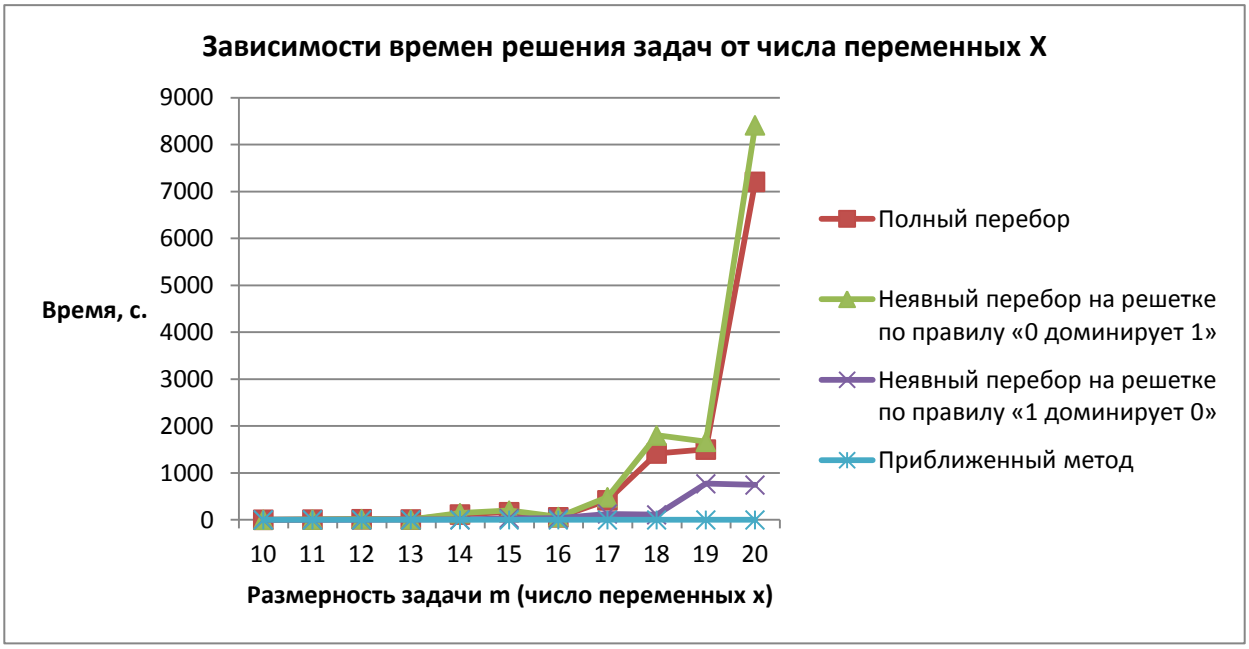

Рис. 3. График зависимости времен решения задач различными алгоритмами в зависимости от *m*

Результаты экспериментов по оценке времени работы разных алгоритмов в зависимости от числа переменных *n* представлены графиком на рис. 4. При этом *m=15*, степени недопустимости двух ограничений (2) и (3) одинаковые и равны 2.

Рис. 4. График зависимости времен решения задач различными алгоритмами в зависимости от *n*

Как видно из графиков, все алгоритмы, кроме приближенного, имеют экспоненциальную трудоемкость. Время работы приближенного алгоритма при решении различных задач не превысило во всех экспериментах 0,001 с. Метод неявного перебора на решетке по правилу «1 доминирует 0» при степени недопустимости равной 2 работает существенно быстрее, чем метод неявного перебора на решетке по правилу «0 доминирует 1». Причем метод неявного перебора на решетке по правилу «0 доминирует 1» уступает во многих экспериментах по времени решения даже методу полному. Это можно объяснить тем, что реализация рекурсивных вызовов функций требует существенных временных ресурсов.

На рис. 5 и рис. 6 представлены графики времен решения в зависимости от степени недопустимости ограничений, одинаковых для ограничений (2) и (3), при значениях *n=15, m=15*. На рис. 6 показан тот же график, что и на рис. 5, но на рис. 5 отсчет по оси абсцисс начинается со значения степени недопустимости 1, а на рис. 6 отсчет по оси абсцисс начинается со значения степени недопустимости 4, т.е. на оси ординат на рис. 6 использован более крупный масштаб, чем на рис. 5.

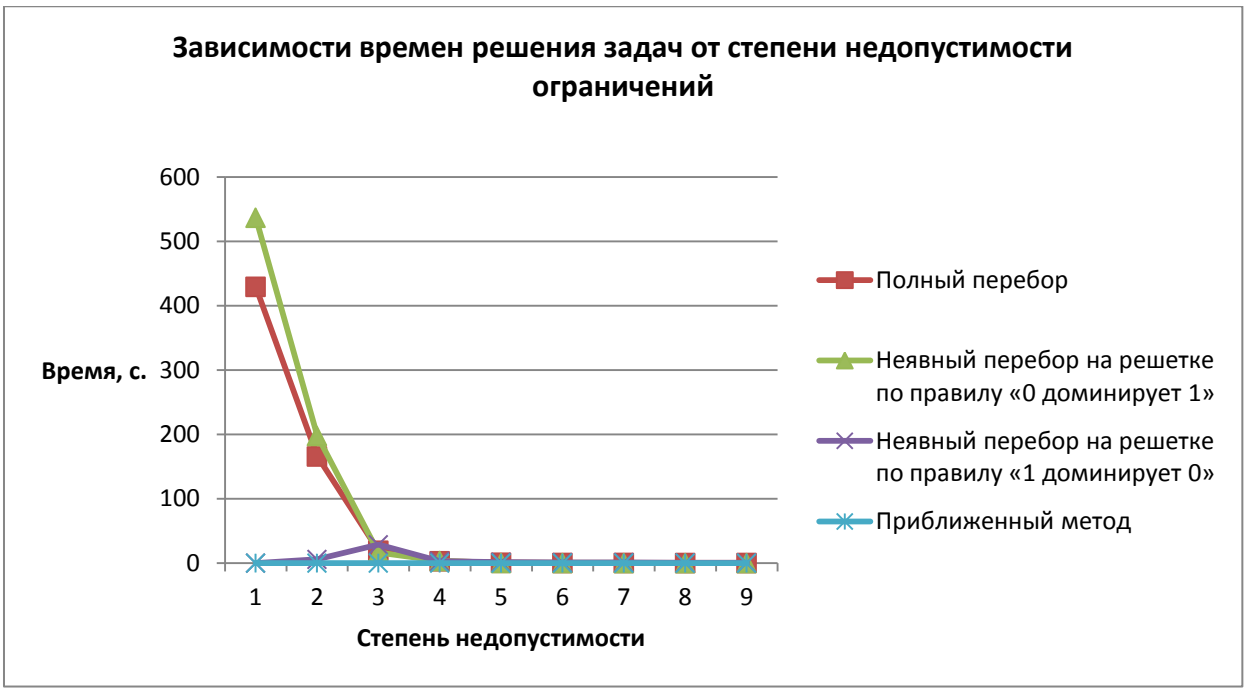

Рис. 5. График зависимости времен решения задач различными алгоритмами в зависимости от степени недопустимости ограничений

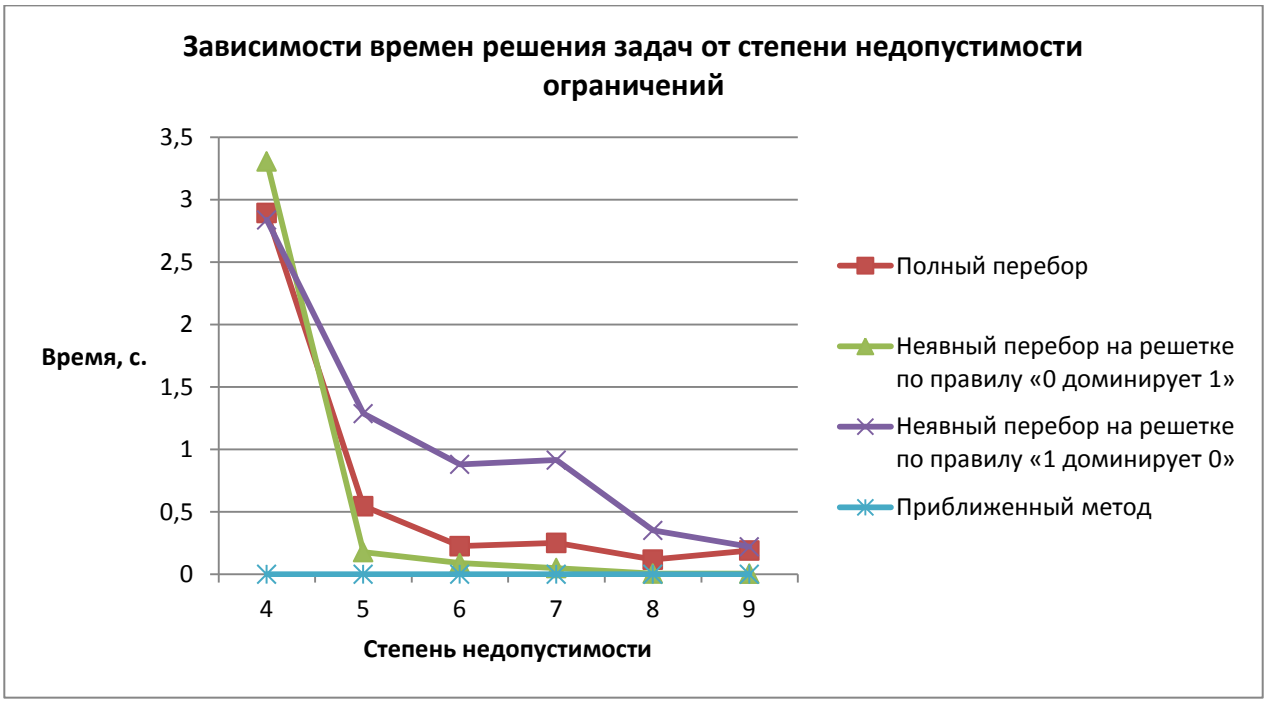

Рис. 6. График зависимости времен решения задач различными алгоритмами в зависимости от степени недопустимости ограничений

Из представленных на рис.5 и рис. 6 графиков можно сделать вывод, что при увеличении степени недопустимости метод неявного перебора на решетке по правилу «0 доминирует 1» начинает работать лучше, чем метод перебора по правилу «1 доминирует 0», начиная со значения отношения 5. Это естественно, так как уменьшается число допустимых решений. Кроме того, при уменьшении числа допустимых решений все алгоритмы

работают быстрее, так как во всех алгоритмах при определении того факта, что решение недопустимо, значение показателя не вычисляется, а осуществляется переход к следующему решению.

Приведем оценки точности получаемых решений приближенным методом. Примеры получаемых решений (10 решений) приближенным методом, число решаемых задач булевого программирования при работе алгоритма для случая, когда размерность вектора $\vec{X}$ равна 15, а размерность вектора $\vec{Y}$ равно 10, степень недопустимости равна 2, исходные данные сформированы с помощью генератора псевдослучайных чисел, а также оценка точности решения по сравнению с результатами, полученными точным алгоритмом, представлены в табл. 1. Для решения задач булевого программирования на каждой итерации алгоритма использовался приближенный метод вектора спада [6].

При проведении 500 подобных экспериментов средняя относительная погрешность решения задачи приближенным методом составила 28,837 %, процент полученных точных решений составил 19,600 %.

Таблица 1. Результаты экспериментов по решению задачи приближенным методом и оценка точности решения

№ п/п	Гарантированные значения показателя для 1- го «игрока», отвечающего за выбор средств защиты при выборе полученного вектора $\vec{X}$, усл. ед.	Числа решаемых задач булевого программирования	Гарантированные значения показателя для 1- го «игрока», полученное точным алгоритмом, усл. ед.	Относительные погрешности, %
1.	519612,569	9	406944,266	27,686
2.	248303,914	9	248303,914	0,000
3.	101839,163	9	101839,163	0,000
4.	449109,306	7	449109,306	0,000
5.	572839,217	9	212161,771	170,001
6.	214399,185	9	186151,112	15,175
7.	383668,375	5	383668,375	0,000
8.	510935,522	7	419980,403	21,657
9.	388943,277	11	388943,277	0,000
10.	441670,962	9	307767,555	43,508

4. Пример решения задачи

Для демонстрации использования приведенной математической постановки задачи антагонистической игры рассмотрим небольшой пример исходных данных для рассматриваемой задачи.

В табл. 2 представлены некоторые угрозы для автоматизированной системы, возможный ущерб от этих угроз за заданный период (1 год) и стоимости проведения этих атак для нарушителя за тот же период. Данные об ущербе сильно зависят от специфики деятельности компании, в которой используется информационная система, и заданы достаточно произвольно.

Таблица 2. Ущерб от непредотвращения атак и стоимость их реализации

№ п/п	Угрозы ($A = \{a_1, a_2, \dots, a_n\}$)	Ущербы от не предотвращения ($u_i, \forall\, i \in N$), руб.	Стоимости реализации угроз для нарушителя ($c_i^{(н)}, \forall\, i \in N$), руб.
1.	Утечка конфиденциальной информации из сети по каналам связи (e-mail, web, chat/IM и т.п.)	10 000 000	100 000
2.	Прослушивание внешних каналов связи злоумышленниками	1 000 000	100 000
3.	Нарушение конфиденциальности данных, передаваемых по линиям связи, проходящим вне контролируемой зоны, осуществляемое внешними нарушителями путем пассивного прослушивания каналов связи	10 000 000	50 000
4.	Перехват информации на линиях связи путем использования различных видов анализаторов сетевого трафика	1 000 000	50 000
5.	Замена, вставка, удаление или изменение данных пользователей в информационном потоке	5 000 000	90 000
6.	Перехват информации, например, пользовательских паролей, передаваемой по каналам связи, с целью ее последующего использования для обхода средств сетевой аутентификации	5 000 000	100 000
7.	Статистический анализ сетевого трафика (например, наличие или отсутствие определенной информации, частота передачи, направление, типы данных и т.п.)	1 000 000	50 000
8.	Внедрение несанкционированного, непроверенного или вредоносного программного кода (вирусов, троянских программ и т.п.).	1 000 000	40 000
9.	Анализ и модификация ПО	15 000 000	500 000
10.	Логические бомбы, пересылаемые по e-mail	1 000 000	50 000
11.	Атаки на отказ в обслуживании против внешних хостов компании.	1 000 000	20 000
	Всего выделено средств на реализацию угроз ($C_{max}^{(н)}$)		500 000

В табл. 3 представлены некоторые средства защиты от угроз безопасности. Данные о стоимости лицензий можно взять на сайтах многих организаций, занимающихся защитой

информации. Угрозы в табл. 3 заданы своими номерами в соответствии с табл. 2. В таблице не приведены названия конкретных производителей средств защиты из-за того, чтобы не делать им рекламу. Представленные примерные цены задают конфигурацию средств защиты для информационной системы на основе локальной вычислительной сети с одним сервером и двадцатью рабочими местами.

Таблица 3. Средства защиты от угроз безопасности, стоимости их реализации и возможности предотвращения угроз на интервале времени 1 год

№	Средства защиты ($\boldsymbol{B = \{b_1, b_2, \dots, b_m\}}$)	Стоимости реализации ($\boldsymbol{c_j^{(з)}, \forall\, j \in M}$), руб	Возможности (вероятности) предотвращения угрозы ($\boldsymbol{p_{ij},\ \forall\, i \in N, \forall\, j \in M}$)										
			Номера угроз по табл. 2										
			1	2	3	4	5	6	7	8	9	10	11
1	Простой антивирус	30 000	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,9	0,8	0,9	0,0
2	Программа, предназначенная для шифрования и дешифрования данных, электронной подписи файлов	60 000	0,6	0,7	0,9	0,0	0,8	0,6	0,0	0,0	0,0	0,0	0,0
3	Средство для защиты от сетевых вторжений, вредоносных программ и спама	40 000	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,7	0,6	0,8	0,0
4	Средство обнаружения вторжений и несанкционированного доступа к информации	20 000	0,0	0,0	0,0	0,6	0,5	0,4	0,8	0,1	0,0	0,0	0,4
5	Средство, включающее межсетевой экран, антивирус и средства обнаружения вторжений	50 000	0,0	0,0	0,0	0,7	0,5	0,4	0,7	0,1	0,6	0,7	0,6
6	Комплекс шифрования	600 000	0,7	0,8	0,99	0,0	0,7	0,1	0,0	0,0	0,0	0,0	0,0
7	Средство защиты информации от несанкционированного доступа	35 200	0,0	0,0	0,0	0,0	0,5	0,6	0,0	0,8	0,8	0,0	0,0
8	Электронный замок	200 000	0,0	0,0	0,0	0,0	0,6	0,5	0,0	0,9	0,9	0,0	0,0
9	Средство защиты данных при взломе или краже дисков, а также при утере ноутбука или флеш-накопителя	28 000	0,9	0,0	0,0	0,0	0,0	0,8	0,0	0,0	0,0	0,0	0,0
10	Средство защиты от DDoS	60 000	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,7
	Выделено средств ($\boldsymbol{C_{max}^{(з)}}$)	500 000											

При заданных в табл. 2 и 3 исходных данных получено одно из возможных решений (решений существует несколько) точным методом для первого игрока: $\vec{X} = \|1, 1, 0, 1, 1, 0, 0, 0, 1, 1\|^{\mathrm{T}}$, это означает, что выбраны средства защиты с номерами: 1, 2, 4, 5, 9, 10 из табл. 3. В самом худшем случае, если второй игрок использует решение $\vec{Y} = \|1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1\|^{\mathrm{T}}$, ущерб будет составлять 4 900 000 руб., при других решениях второго игрока ущерб может быть меньшим. Следует отметить, что приближенный алгоритм выдал решение для первого игрока по критерию минимакс с ущербом 6 000 000 руб.

Из полученного решения можно сделать вывод о том, что предложенный ограниченный набор СЗИ, представленный в табл. 3, является недостаточным для существенного уменьшения возможного ущерба. Так, при фактическом снятии ограничения (2), для этого достаточно задать $C_{max}^{(з)} = 1123200$, что позволит использовать все СЗИ из табл. 3, ущерб будет составлять 4 000 000 руб. Таким образом, для более эффективной защиты требуется расширение списка СЗИ, представленных в табл. 3. Если же не использовать СЗИ, тогда $C_{max}^{(з)} = 0$, ущерб может достигать 34 000 000 руб., что почти на порядок больше, чем с использованием СЗИ.

Заключение

В статье рассмотрена математическая постановка задачи антагонистической игры двух игроков с нулевой суммой. Перечислены критерии оптимальности для принятия решения стороной защиты при выборе СЗИ: Лапласа, Вальда, Гурвица и др. Для обеспечения гарантированного результата по показателю ущерба при выборе СЗИ стороной защиты необходимо использовать критерий Вальда, которой при минимизации ущерба стороной защиты превращается в минимаксный критерий.

Для решения задачи выбора СЗИ стороной защиты в соответствии с критерием Вальда необходимо решать задачи булевого программирования. Рассмотрены некоторые алгоритмы для решения подобных задач: точный алгоритм, основанный на методе неявного перебора на решетке, и приближенный алгоритм. Проведены исследования практической зависимости времени работы предложенных алгоритмов от параметров задачи, даны рекомендации по выбору методов в конкретных условиях. Проведена практическая оценка относительной погрешности приближенного метода. Представлен пример решения конкретной задачи выбора СЗИ стороной защиты.

Представленные модели и алгоритмы могут быть применены для решения задач выбора СЗИ различных классов при проектировании элементов системы защиты информации для автоматизированных систем различного целевого назначения.

Список литературы

1. Домарев В.В. Безопасность информационных технологий: Методология создания систем защиты. Киев: Диасофт, 2002. 688 с.

2. Овчинников А.И., Журавлев А.М., Медведев Н.В., Быков А.Ю. Математическая модель оптимального выбора средств защиты от угроз безопасности вычислительной сети предприятия // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2007. № 3. С. 115- 121.
3. Быков А.Ю., Панфилов Ф.А., Шмырев Д.В. Задача выбора средств защиты в автоматизированных системах с учетом классов защищенности от несанкционированного доступа к информации // Инженерный журнал: наука и инновации. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 1. Режим доступа: http://engjournal.ru/catalog/it/hidden/85.html (дата обращения 21.04.2014).
4. Быков А.Ю., Гуров А.В. Задача выбора средств защиты информации от атак в автоматизированных системах при нечетких параметрах функции цели // Инженерный журнал: наука и инновации. МГТУ им. Н.Э. Баумана. Электрон. журн. 2012. № 1. Режим доступа: http://engjournal.ru/catalog/it/hidden/86.html (дата обращения 21.04.2014).
5. Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование. М.: ФИЗМАТЛИТ, 2007. 304 с.
6. Овчинников А.И., Медведев Н.В., Быков А.Ю. Применение метода вектора спада для решения задачи поиска вариантов защиты от угроз безопасности вычислительной сети предприятия // Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение. 2008. № 2. С. 73- 82.
7. Петросян Л.А., Зенкевич Н.А., Семина Е.А. Теория игр: Учеб. пособие для ун-тов. М.: Высшая школа, Книжный дом «Университет», 1998. 304 с.
8. Абденов А.Ж., Заркумова Р.Н. Выбор средства эффективной защиты с помощью методов теории игр // Вопросы защиты информации. 2010. № 2. С. 26- 31.
9. Лаврентьев А. В., Зязин В. П. О применении методов теории игр для решения задач компьютерной безопасности // Безопасность информационных технологий. 2013. № 3. С. 19- 24.
10. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
11. Стрекаловский А.С., Орлов А.В. Биматричные игры и билинейное программирование. М.: ФИЗМАТЛИТ, 2007. 224 с.