# Генетическая оптимизация и визуальный анализ при формировании схем доступа в ВЛВС[1]

И.Б. Саенко, И.В. Котенко

**Аннотация.** В статье рассматривается подход к проектированию виртуальной локальной вычислительной сети (ВЛВС), основанный на использовании программных средств генетической оптимизации и визуального анализа схемы доступа ВЛВС. Излагается формальная постановка задачи оптимизации схемы доступа ВЛВС, решение которой повышает надежность и безопасность функционирования корпоративной вычислительной сети. Показано, что рассматриваемая задача относится к одной из форм булевой матричной факторизации и является NP-полной. В разработанном генетическом алгоритме, предложенном для решения поставленной задачи, реализован ряд усовершенствований, касающихся формирования начальной популяции, вида функции пригодности, кодирования хромосом и выполнения операций скрещивания и мутации. Разработанные программные средства реализуют генетический алгоритм, формируют визуальное отображение хода решения задачи и обеспечивают оценку решения задачи. Экспериментальные результаты показали высокую эффективность разработанного генетического алгоритма.

**Ключевые слова:** виртуальная локальная вычислительная сеть, булева матричная факторизация, генетический алгоритм оптимизации, визуальный анализ.

## Введение

Обеспечение устойчивого, надежного и безопасного обмена данными между компьютерами в локальных компьютерных сетях является достаточно важной задачей, для решения которой постоянно проводятся исследования и разрабатываются новые средства и методы. К числу возможных угроз, оказывающих существенное влияние на устойчивость информационного обмена в локальной сети, можно отнести множество различных факторов. В частности, достаточно опасным фактором являются несанкционированные действия внутренних пользователей (инсайдеров), направленные на доступ к соседним узлам. Эти угрозы способны привести не только к утечке конфиденциальной информации на отдельных узлах, но и к отказу функционирования всей компьютерной сети (например, за счет реализации DoS-атаки). Известные средства разграничения доступа, применяемые в локальной компьютерной сети, не всегда способны предотвратить реализацию таких угроз. Поэтому желательно, а ряде случаев, например, в критических инфраструктурах, необходимо иметь дополнительный уровень защиты локальной сети, на котором возможна строгая регламентация потоков данных между компьютерами, при условии, что его реализация не приводит к большим издержкам по производительности и стоимости.

[1] Работа выполнена при финансовой поддержке РФФИ (13-01-00843, 14-07-00697, 14-07-00417, 15-07-07451) и программы фундаментальных исследований ОНИТ РАН (контракт №1.5).

Одним из возможных способов создания такого уровня защиты в локальной компьютерной сети является реализация технологии виртуальной локальной вычислительной сети (ВЛВС) [1, 2], позволяющей разделить компьютерную сеть на фрагменты логически связанных компьютеров, которые определяются как *виртуальные компьютерные сети* или, для простоты, *виртуальные подсети*. Схема распределения компьютеров по виртуальным подсетям называется *схемой ВЛВС*. В локальной сети, в которой реализована технология ВЛВС, обмен данными между двумя компьютерами возможен тогда и только тогда, когда они принадлежат к одной и той же виртуальной подсети. При этом количество компьютеров, входящих в одну и ту же виртуальную подсеть, не ограничивается. В предельном случае все компьютеры локальной сети могут входить в одну и ту же виртуальную подсеть. Однако тогда отпадает необходимость использования ВЛВС, так как между каждой парой компьютеров обмен данными будет разрешен.

Кроме ВЛВС, известны другие подходы к разграничению обмена данными на сетевом уровне. В частности, можно использовать списки доступа в операционных системах рабочих станций. Однако этот подход является менее гибким по сравнению с ВЛВС, так как если возникает необходимость изменения схемы разграничения доступа, то тогда сетевой администратор должен выполнить соответствующие действия на удаленных рабочих станциях, что не всегда возможно. В отличие от этого подхода, все действия по конфигурированию ВЛВС осуществляются централизованно, так как схема ВЛВС хранится в одном месте, например, на коммутаторе или маршрутизаторе Ethernet.

Несмотря на несомненные достоинства использования виртуальных подсетей для повышения устойчивости информационных потоков в локальной компьютерной сети, задача проектирования схемы ВЛВС, на наш взгляд, является недостаточно обсуждаемой в научной литературе. В настоящее время для проектирования схемы ВЛВС используются эмпирические рекомендации, согласно которым в локальной сети следует формировать виртуальные подсети в соответствии с имеющимися в организации функциональными подсистемами. В соответствии с этими рекомендациями виртуальные подсети обычно образуются вокруг серверов различного назначения и охватывают компьютеры, имеющие право работать с этими серверами. Однако такой подход делает невозможным обмен данными между компьютерами, работающими с различными серверами. Во многом это объясняется отсутствием формальной постановки задачи проектирования схемы ВЛВС в форме задачи оптимизации и, соответственно, средств и методов ее решения.

Одной из целей настоящей работы является разработка нового подхода к проектированию схемы доступа ВЛВС, основанного на формальной постановке оптимизационной задачи и ее последующем решении. В качестве исходных данных предлагается использовать матрицу требуемой связности компьютеров в локальной сети. Роль переменных задачи играют элементы булевой матрицы принадлежности компьютеров к виртуальным подсетям. Критерием поиска решения является требование минимизации количества виртуальных подсетей при условии полного достижения матрицы требуемой связности. Анализ известных работ показывает, что данная задача может быть отнесена к одной из разновидностей задач булевой матричной факторизации (БМФ). По этой причине она является NP-полной и требует эвристических методов решения.

Известно, что одним из достаточно эффективных и универсальных методов решения таких задач являются генетические алгоритмы. Они в наибольшей степени приспособлены для решения NP-полных оптимизационных задач, в которых имеются булевы переменные. Известны работы, в которых генетические алгоритмы применялись для задач БМФ. Тем не менее, прямое применение этих методов для оптимизации схемы ВЛВС является невозможным. Необходима разработка и применение в генетическом алгоритме некоторых усовершенствований, получение которых также является одной из целей настоящей работы.

Генетические алгоритмы относятся к настраиваемым методам решения оптимизационных задач, эффективность которых во мно-

гом зависит от параметров, используемых алгоритмами в ходе своей работы. К их числу относятся размер популяции, вероятность скрещивания, вероятность мутации и некоторые другие. Для того чтобы определить, как зависит эффективность применения генетического алгоритма от его параметров при проектировании схемы ВЛВС, был разработан специальный инструментальный стенд, включающий средства проектирования схемы доступа ВЛВС. Данные средства не только находят решение задачи, но и обладают возможностями визуализации решения, помогающими в оценке качества решения задачи.

*Основной теоретический вклад* статьи может быть определен как дальнейшее развитие теории построения виртуальных сетей и теории генетической оптимизации. В рамках развития первой теории формулируется постановка задачи проектирования схемы доступа ВЛВС и показывается, что она является одной из форм БМФ. Развитие теории генетической оптимизации определяется рядом усовершенствований, предложенных в статье, необходимых для учета в генетическом алгоритме критерия минимизации количества виртуальных подсетей при условии полного достижения матрицы требуемой логической связности.

Статья организована следующим образом. В разделе 1 представлен обзор существующих работ. Раздел 2 раскрывает математическую постановку задачи. В разделе 3 обсуждается разработанный генетический алгоритм и его усовершенствования. В разделе 4 рассматриваются разработанные инструментальные средства и результаты экспериментов. Заключение содержит выводы и направления дальнейших исследований.

## 1. Обзор существующих работ

Для того чтобы определить область, к которой относится задача формирования схем доступа виртуальных подсетей, и сформировать ее математическую постановку, были в первую очередь проанализированы работы, связанные с вопросами декомпозиции булевых матриц. В [3, 4, 5] были рассмотрены некоторые виды задач матричной факторизации, к которым относятся неотрицательная матричная факторизация (НМФ) и БМФ. В этих работах доказано, задачи НМФ и БМФ являются NP-полными. Для некоторых видов этих задач предложены математические методы их решения.

В [6, 7] к задачам БМФ были сведены некоторые задачи из области информационной безопасности. В частности, в этих работах было показано, что к классу задач БМФ относится «задача извлечения ролей» (Role mining problem, RMP).

Эти идеи в дальнейшем были развиты в [8, 9], в которых предложено использовать для решения задачи RMP генетические алгоритмы. В [8] для кодирования решений задачи RMP предложен мульти-хромосомный подход. Согласно этому подходу каждая особь популяции генетического алгоритма имеет три хромосомы. Одна из этих хромосом является управляющей. Она помогает осуществлять скрещивание, когда родительские хромосомы имеют разную длину. В [9] был сделан отказ от мульти-хромосомного кодирования, но предложен этап предварительной обработки для скрещивания. Тем не менее, оба эти новшества не могут применяться для решения задачи создания виртуальных подсетей, так как эта задача требует поиска только одной логической матрицы вместо двух. По этой причине этап обработки, который предлагается в [9], следует рассматривать как избыточный. Кроме того, как будет показано ниже, рассматриваемая задача не предполагает, что хромосомы будут иметь различную длину. Предлагается подход для расчета длины хромосом и использования так называемых тривиальных решений для генерации начальной популяции.

Высокая сложность задач НМФ и БМФ и возможность поиска эффективных математических методов их решения только для частных случаев определили необходимость поиска эвристических методов их решения. В [10] для решения задач НМФ исследуется возможность использования био-инспирированных алгоритмов, основанных на популяциях. Среди алгоритмов, рассматриваемых в этой работе, присутствуют генетические алгоритмы, алгоритмы оптимизации на основе роя частиц, алгоритмы дифференциальной эволюции, алгоритмы поиска «рыбным косяком» и алгоритмы поиска на основе фейер-

верка. Сравнительная оценка этих алгоритмов, проведенная в этой работе, показала, что для решения задач НМФ наибольшей эффективностью обладают генетические алгоритмы.

В [11, 12] был предложен и исследован генетический алгоритм решения задачи БМФ. Этот алгоритм позволяет находить булевы матрицы **W** и **H**, на которые декомпозируется заданная булева матрица **A**. В этом алгоритме в качестве генов использовались строки матрицы **H**, а операция скрещивания применялась к столбцам матрицы **W**. Функция пригодности была сформирована на основе евклидова расстояния между исходной и результирующей матрицами. Однако, как легко заметить, этот алгоритм не гарантирует, что между матрицами **W** и $\mathbf{H}^{\mathrm{T}}$ соблюдается строгое равенство. По этой причине алгоритм, предложенный в этих работах, не может быть применен для задачи, рассматриваемой в настоящей работе.

Имеется несколько работ, посвященных применению методов интеллектуального анализа данных (data mining) к формированию виртуальных подсетей. Так, в [13] предложено для формирования схемы виртуальных подсетей применять методы кластерного анализа. Однако этот подход имеет определенные ограничения, так как ориентирован на реализацию в мобильных самоорганизующихся (ad hoc) сетях. Генетический алгоритм был предложен для оптимизации схемы доступа виртуальных сетей в [14]. Однако этот алгоритм ориентирован только на поиск матрицы **A**, причем в контексте удовлетворения заданных полномочий по доступу пользователей к ресурсам сети при условии, что некоторые ресурсы могут иметь «общий доступ». Следовательно, он не позволяет находить матрицы **X** и $\mathbf{X}^{\mathrm{T}}$, на которые исходная матрица **A** должна быть декомпозирована. Кроме того, он не учитывает критерий минимизации числа виртуальных подсетей.

Таким образом, анализ существующих работ показывает, что генетические алгоритмы следует считать достаточно эффективным методом решения рассматриваемой задачи. В то же время генетические алгоритмы, предложенные в известных работах, не могут быть напрямую использованы для этой цели.

## 2. Математическая постановка задачи

Положим, что в компьютерной сети находится $n$ компьютеров. Пусть схема разрешенных информационных потоков между этими компьютерами определяется булевой матрицей $\mathbf{A}[n, n]$. Если $a_{ij} = 1$ ($i, j = 1, \ldots, n$), то обмен между компьютерами $i$ и $j$ разрешен. В противном случае этот обмен невозможен.

Положим далее, что в компьютерной сети сформировано $k$ виртуальных подсетей, каждая из которых объединяет два и более компьютера. Зададим распределение компьютеров по подсетям с помощью булевой матрицы $\mathbf{X}[n, k]$. Если $x_{ij} = 1$ ($j = 1, \ldots, k$), то компьютер $i$ принадлежит подсети $j$, в противном случае подсеть $j$ не охватывает компьютер $i$.

Матрица **A** играет роль исходных данных задачи. Матрица **X** является искомой переменной задачи. Покажем, что булевы матрицы **A** и **X** связывает следующая зависимость:

$$\mathbf{A} = \mathbf{X} \otimes \mathbf{X}^{\mathrm{T}}, \qquad (1)$$

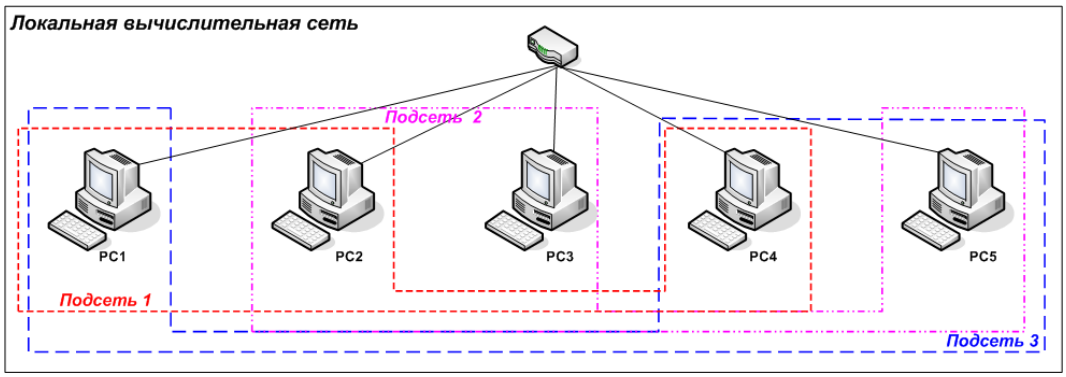
где $\mathbf{X}^{\mathrm{T}}$ – транспонированная матрица **X**, символ $\otimes$ обозначает булево матричное умножение, которое является формой матричного умножения, основанной на правилах булевой алгебры. Булево матричное умножение позволяет получать элементы матрицы **A** согласно следующему выражению: $a_{ij} = \vee_{j=1}^{n} \left( x_{ij} \wedge x_{ji} \right)$.

Рассмотрим следующий пример. Пусть $n = 5$, $k = 3$, и в компьютерной сети существуют подсети, как показано на Рис. 1. В подсеть 1 вошли рабочие станции PC1, PC2 и PC4, в подсеть 2 – PC2, PC3 и PC5, в подсеть 3 – PC1, PC4 и PC5.

Легко заметить, что матрица «узлы – подсети», представленная на Рис. 1, играет роль матрицы **X**, а матрица «узлы – узлы» является матрицей **A**.

Матрицы **A** и **X** связаны друг с другом следующим образом:

$$\mathbf{X} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad \mathbf{X}^{\mathrm{T}} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix},$$




Подсети

| Узлы | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 0 |
| 3 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 |
| 5 | 0 | 1 | 1 |

Матрица «узлы - подсети»

Узлы

| Узлы | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 |
| 3 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 |

Матрица «узлы - узлы»

Рис. 1. Пример распределения виртуальных подсетей по рабочим станциям

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Легко заметить, что матрица **A** является симметричной.

Теперь покажем, что рассматриваемая задача является разновидностью задач БМФ. Как известно, задача БМФ сводится к нахождению булевых матриц **W** и **H**, связанных с заданной булевой матрицей **A** следующим уравнением:

$$\mathbf{A} = \mathbf{W} \otimes \mathbf{H}, \qquad (2)$$

где **A** = **A** [$n$, $m$], **W** = **W** [$n$, $k$] и **H** = **H** [$k$, $m$].

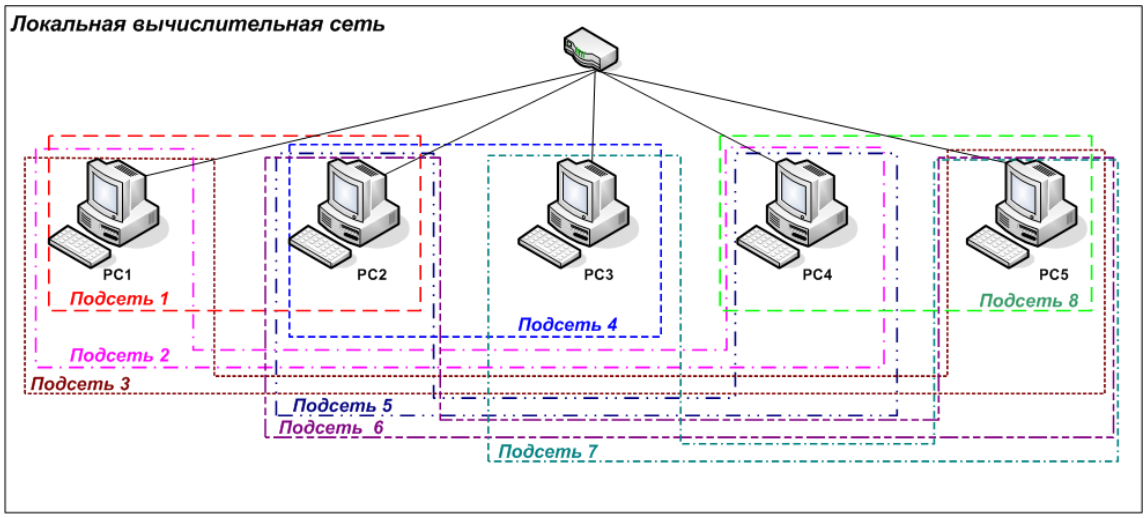
Сравнивая (1) и (2), можно заметить, что задача (1) может рассматриваться как частный случай (2) при выполнении двух условий. Первое условие есть равенство $m = n$. Второе условие имеет следующий вид:

$$w_{ij} = h_{ji} \text{ для любых } i = 1, \ldots, n \text{ и } j = 1, \ldots, k. \qquad (3)$$

Из того, что рассматриваемая задача является разновидностью задач БМФ, следует, что она является NP-полной. Однако в силу (3) прямое применение к ней методов решения БМФ-задачи является затруднительным. Этим объясняется необходимость решения этой задачи эвристическими методами, а именно генетическими алгоритмами.

Однако для этой цели необходимо ввести критерий оптимизации. Чтобы это сделать, обратим внимание на тот факт, что в задаче (1) значение $k$ может быть произвольным. Покажем, каким образом можно ограничить его сверху. Для этого сформируем разбиение сети на подсети таким образом, чтобы в каждую подсеть входили только два компьютера $i$ и $j$, если $a_{ij} = 1$. Такое распределение рабочих станций по виртуальным подсетям будем называть бинарным.

Для примера, приведенного выше, бинарное распределение рабочих станций по виртуальным подсетям имеет вид, показанный на Рис. 2. В этом случае матрица **X** имеет пять строк и восемь столбцов в соответствии с матрицей «узлы – подсети», представленной на этом рисунке. Матрица $\mathbf{X}^{\mathrm{T}}$, соответственно, имеет во-




*Подсети*

| Узлы | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 5 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |

**Матрица «узлы - подсети»**

*Узлы*

| Узлы | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 | 1 |
| 3 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 | 1 | 1 |

**Матрица «узлы - узлы»**

Рис. 2. Пример бинарного распределения рабочих станций по виртуальным подсетям

семь строк и пять столбцов. Тем не менее, булево умножение этих матриц дает в результате матрицу **A** (это не трудно проверить). Порядок столбцов может быть произвольным. Матрица **A** соответствует матрице «узлы – узлы», представленной на Рис. 2.

Будем называть такую матрицу *тривиальным решением* $\mathbf{X}_0$. Иными словами, тривиальное решение $\mathbf{X}_0$ соответствует распределению компьютеров по виртуальным подсетям, которое показано на Рис. 2, и отвечает требованиям матрицы **A**, представленной на Рис. 1. Иными словами, тривиальное решение $\mathbf{X}_0$ имеет вид

$$\mathbf{X}_0 = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

и отвечает уравнению $\mathbf{A} = \mathbf{X}_0 \otimes \mathbf{X}_0^{\mathrm{T}}$.

Легко заметить, что тривиальное решение обладает следующими свойствами:

- количество столбцов в матрице $\mathbf{X}_0$ равно количеству единичных элементов в матрице **A**, лежащих выше главной диагонали (обозначим это количество как $M$);
- только два единичных элемента находятся в каждом столбце матрицы $\mathbf{X}_0$;
- количество тривиальных решений равно числу $P$ перестановок столбцов в матрице $\mathbf{X}_0$ (очевидно, что $P = 2^M$).

Анализ вышеприведенных свойств позволяет сделать следующие выводы:

- тривиальные решения не могут рассматриваться в качестве хороших или рациональных решений. Это связано с тем, что количество $M$ виртуальных подсетей в тривиальном решении достаточно велико. Поэтому количество тривиальных решений, равное $P = 2^M$, является очень большим, а рациональных решений не может быть так много;
- количество столбцов в матрице $\mathbf{X}_0$ в ходе решения задачи (1) можно ограничить сверху значением $M$;

- видно, что для примера, приведенного на Рис. 1, существует решение, когда количество виртуальных подсетей меньше, чем $M$ (здесь $k = 3$). Если считать, что матрица $\mathbf{X}_0$ имеет $M$ столбцов, то $k = 3$ соответствует количеству ненулевых столбцов в этой матрице;
- следует полагать, что чем меньше $k$, тем лучше решение задачи. Чем меньше $k$, тем меньше сложность администрирования и выше доступность сети.

Таким образом, предлагается в качестве *критерия оптимизации* задачи использовать *минимум значения k при полном совпадении* матриц $\mathbf{A}$ и $\mathbf{X} \otimes \mathbf{X}^{\mathrm{T}}$.

Формально этот критерий может быть представлен следующим образом:

$$\begin{cases} k \to \min, \\ k = 1,..,M;\ M = \left| \left\{ a_{ij} \mid a_{ij} = 1, i > j \right\} \right|, \\ \mathbf{X}[n,k] \otimes \mathbf{X}[n,k]^{\mathrm{T}} = \mathbf{A}[n,n];\ A = \left\| a_{ij} \right\|. \end{cases}$$

## 3. Генетический алгоритм решения задачи

Генетические алгоритмы являются хорошо известным методом био-инспирированной оптимизации. Тем не менее, существуют различные взгляды на последовательность и содержание шагов генетических алгоритмов [16-18]. Предложим следующую последовательность шагов.

1. *Определение функции пригодности*, которая показывает, почему одно из возможных решений задачи мы считаем лучше, чем другое решение.

2. *Кодирование возможных решений задачи*. Закодированное решение в терминологии генетического алгоритма называется *особью*. Обычно решения кодируются с помощью символьных или числовых строк. Отдельный символ этого кода называется *геном*. Совокупность генов в строке называется *хромосомой.*

3. *Создание начального множества особей*, которое называется *популяцией*. Как правило, этот процесс проходит случайным образом, однако количество особей в популяции $N$ является постоянным. Этот шаг также включает оценку всех особей в популяции с помощью функции пригодности и сортировку их по убыванию ее значения.

4. Выбор пар особей, которые называются *родителями*, для формирования новых особей, называемых *потомками*, путем *скрещивания*. Хромосомы родителей разбиваются на фрагменты. Затем происходит обмен фрагментами родительских хромосом, чтобы сформировать хромосомы потомков. Новые особи оцениваются с помощью функции пригодности и добавляются в текущую популяцию.

5. Выбор особей для *мутации* их хромосом. При мутации особи изменяют свои гены.

6. *Селекция популяции*, которая заключается в оставлении в ней $N$ особей, обладающих самыми высокими значениями функции пригодности. Остальные особи (самые "плохие") удаляются из популяции.

7. Если выполняются критерии завершения алгоритма, то в качестве решения оптимизационной задачи выбирается особь с максимальным значением функции пригодности. Иначе происходит возврат к шагу 3.

Для того чтобы использовать генетический алгоритм для решения рассматриваемой задачи, необходимо определить функцию пригодности и порядок кодирования решений.

В известных работах по применению генетических алгоритмов для решения задачи БМФ функция пригодности строилась только на основании эвклидова расстояния между матрицами $\mathbf{A}$ и $\mathbf{W} \otimes \mathbf{H}$ [10, 12]:

$$F = \left( \sqrt{\sum_i \sum_j \left( a_{ij} - w_{ij} h_{ji} \right)^2} \right)^{-1}.$$

При полном совпадении этих матриц евклидово расстояние равно 0, и функция пригодности принимает максимальное значение.

Однако в нашем случае учет в функции пригодности только евклидова расстояния между $\mathbf{A}$ и $\mathbf{X} \otimes \mathbf{X}^{\mathrm{T}}$ не является достаточным, так как по такому критерию могут быть получены тривиальные решения, а они не могут считаться рациональными.

Необходимо дополнительно учитывать требование того, чтобы в матрице $\mathbf{X}$ количество столбцов $k$ было минимальным.

В результате предлагается следующий вид функции пригодности:

$$F = \left( \alpha k + \beta \sqrt{\sum_i \sum_j \left( a_{ij} - x_{ij} x_{ji} \right)^2} \right)^{-1} \qquad (4)$$

где $\alpha$ и $\beta$ являются весовыми коэффициентами, которые определяют направление поиска решений. Условие $\alpha << \beta$ между этими коэффициентами (например, $\alpha = 1$ и $\beta = 10$) гарантирует, что в первую очередь мы хотим получить решения с полным совпадением матриц **A** и $\mathbf{X} \otimes \mathbf{X}^{\mathrm{T}}$, а затем мы хотим найти решения с меньшими значениями *k*. Противоположное условие $\alpha >> \beta$ обеспечивает первоочередной поиск решений, в которых вначале уменьшается значение *k*, в затем выполняется полное совпадение матриц **A** и $\mathbf{X} \otimes \mathbf{X}^{\mathrm{T}}$.

Теперь рассмотрим порядок формирования хромосом. Для этого предложим подход, согласно которому в качестве гена хромосомы будет рассматриваться не отдельный символ или число, а вектор $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{jn})$.

Вектор $\mathbf{x}_i$ играет роль столбца матрицы **X**. Количество генов в хромосоме примем равным *M*. В самом деле, число *M* определяет количество столбцов в тривиальном решении. Поэтому иметь в хромосоме большее число генов не имеет смысла.

Наконец, в целях увеличения сходимости генетического алгоритма сделаем еще одно предложение, которое касается формирования начальной популяции. Будем использовать для ее формирования возможные тривиальные решения.

Тем самым в начальную популяцию искусственным образом будут вноситься особи с достаточно высоким значением функции пригодности, равным $F = 1 / k$**.** Если получится, что $P \geq N / 2$, то тогда половина начальной популяции полностью будет состоять из тривиальных решений. В противном случае в качестве *P* особей начальной популяции будут использоваться тривиальные решения, а остальные особи (их количество равно $N - P$) будут формироваться случайным образом.

Операции скрещивания и мутации будем выполнять стандартным образом. Однако то, что в качестве гена хромосомы используется столбец матрицы **X**, накладывает некоторые особенности на их проведение. В частности, при выполнении операции скрещивания предлагается выполнять разделение родительских хромосом на части не в одномерном, а в двумерном режиме.

Идея реализации двумерного режима при выполнении операции скрещивания наглядно иллюстрируется на Рис. 3 применительно к двум тривиальным решениям.

Двумерное скрещивание повышает вероятность появления новых единичных или нулевых элементов в генах дочерних хромосом и, следовательно, повышает скорость работы генетического алгоритма.

Другая особенность выполнения операции скрещивания заключается в следующем. При ее

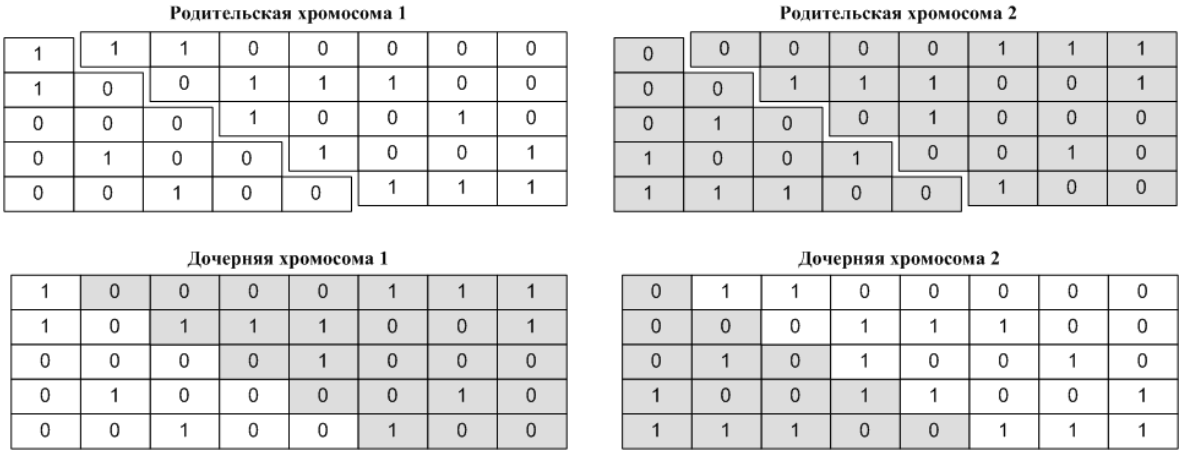



Рис. 3. Пример двумерного скрещивания

выполнении возможно появление потомков с одинаковыми генами. Такое возможно, например, при скрещивании двух тривиальных решений, находящихся в начальной популяции. Это означает, что в матрице **X**, которую имеет такой потомок, одна и та же подсеть отображена два раза. В этом случае один из столбцов необходимо полностью обнулить. В результате количество ненулевых столбцов, соответствующее общему количеству подсетей, уменьшается на единицу.

Мутация выполняется в два этапа. На первом этапе определяются гены (столбцы матрицы **X**), подлежащие модификации. Отбор генов осуществляется с вероятностью $W_{\text{ген}}$. Затем с вероятностью $W_{\text{эл}}$ происходит инвертирование значения соответствующего элемента в выбранном столбце матрицы **X**.

## 4. Экспериментальная оценка

Для решения поставленной задачи, а также оценки скорости и точности предложенного генетического алгоритма, были разработаны программные *средства проектирования схемы доступа* (СПСД) для ВЛВС.

Архитектура СПСД представлена на Рис. 4.

В состав СПСД входят следующие модули:

- *ввода исходных данных*, в качестве которых выступают общее количество компьютеров в локальной сети ($n$), параметры генетического алгоритма (размерность популяции $N$, вероятность скрещивания $W_{\text{cross}}$, вероятность мутации $W_{\text{mut}}$ и др.), а также количество виртуальных подсетей в оптимальной схеме доступа ($k$);
- *генерации эталонного решения* ($\mathbf{X}_{et}$), которое используется для формирования матрицы **A** и оценки точности решения;
- *формирования матрицы требуемой логической связности* $\mathbf{A} = \mathbf{X}_{et} \otimes \mathbf{X}_{et}^{\mathrm{T}}$;
- *поиска решения задачи* оптимизации схемы доступа для ВЛВС, являющегося решением уравнения $\mathbf{A} = \mathbf{X} \otimes \mathbf{X}^{\mathrm{T}}$, с помощью предложенного генетического алгоритма;
- *визуализации* хода поиска решения и *оценки* генетического алгоритма на основе сравнения **X** и $\mathbf{X}_{et}$.

В состав СПСД также входит датчик псевдослучайных чисел (ДПСЧ) [18], который используется для формирования эталонного решения и работы генетического алгоритма.

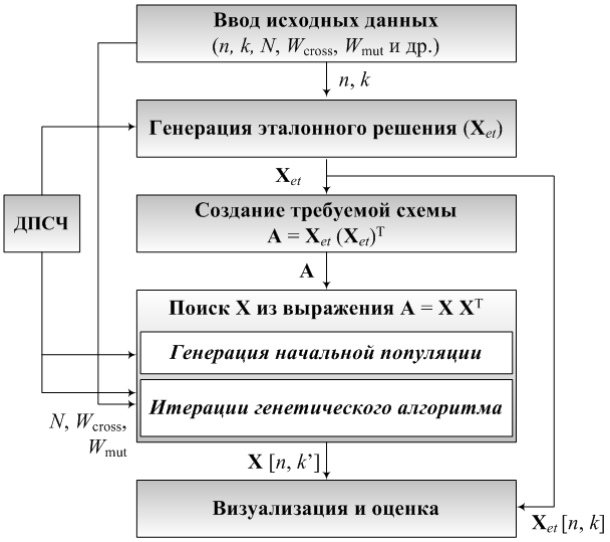



Рис. 4. Архитектура СПСД

Исходные данные СПСД вводятся через диалоговую форму, общий вид которой представлен на Рис. 5.

Значения элементов матрицы **A**, отображаемые на этом рисунке, соответствуют примеру, приведенному на Рис. 1.

К числу исходных данных относятся:

- параметры, характеризующие особи в популяции (объединены в блок **Individual**). К их числу относятся $n$, $k$, $\alpha$ (**alpha**), $\beta$ (**beta**), а также вероятность формирования значения «1» в матрице **A**, обозначенная как **gamma**. Значение **gamma** по умолчанию равно 0.5, что соответствует равновероятностному закону. Однако для такого закона при большом $n$ ($n > 20$) в матрице **A** практически все элементы принимают значение «1». Чтобы увеличить количество в **A** нулевых элементов, необходимо выбирать **gamma** меньше, чем 0.5;
- параметры, характеризующие популяцию (объединены в блок **Population**). К их числу относятся размер популяции $N$, вероятности $W_{\text{cross}}$ и $W_{\text{mut}}$, а также предельное число итераций $T$.

Формирование матрицы **A** происходит при нажатии кнопки **Random A**. Результат формирования матрицы отображается в окне **Matrix A**.

При нажатии кнопки **Start Algorithm** запускается модуль поиска решения. Как показано на Рис. 4, данный модуль включает блок генера-

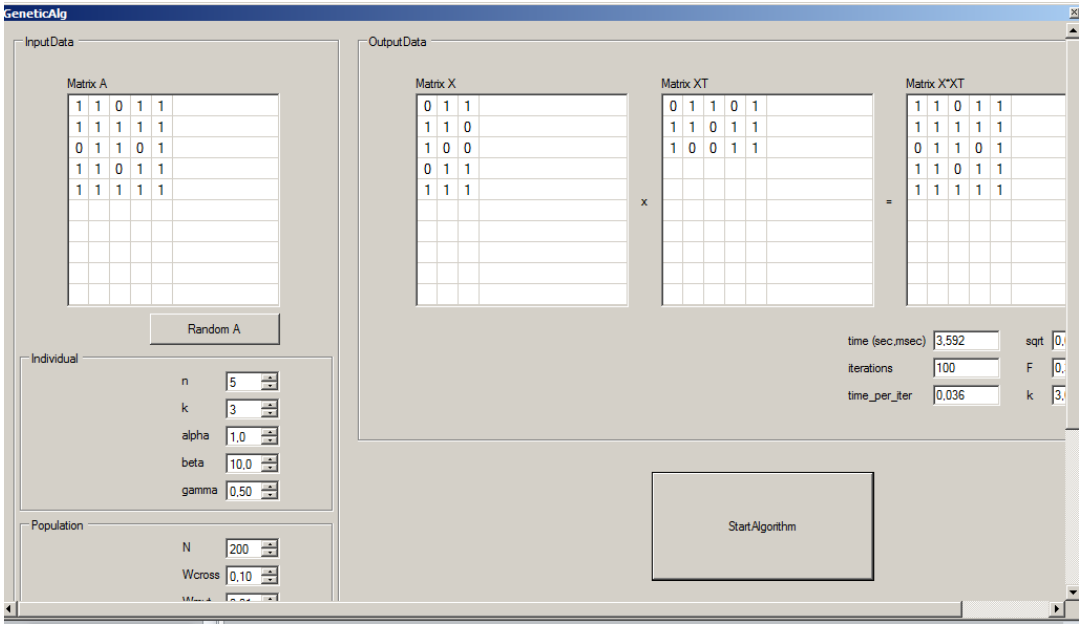



Рис. 5. Диалоговая форма СПСД

ции начальной популяции и блок выполнения итераций генетического алгоритма. Размер популяции в алгоритме является постоянным и равен $N$ (по умолчанию $N = 200$). Одна половина начальной популяции заполняется тривиальными решениями (пример показан на Рис. 2). Вторая половина заполняется решениями, полученными с использованием ДПСЧ.

Отбор особей популяции для выполнения над ними операции селекции осуществляется с вероятностью $W_{\text{cross}}$, которая по умолчанию равна $W_{\text{cross}} = 0.1$. Отбор особей для мутации осуществляется с вероятностью $W_{\text{mut}}$ (по умолчанию $W_{\text{mut}} = 0.01$). Значения $N$, $W_{\text{cross}}$ и $W_{\text{mut}}$, принятые по умолчанию, взяты из [11]. Однако они могут быть изменены в модуле ввода исходных данных.

Также подлежат изменению значения весовых коэффициентов $\alpha$ и $\beta$, которые используются для расчета функции пригодности. По умолчанию эти коэффициенты принимают значения $\alpha = 1$ и $\beta = 10$.

Окончание выполнения итераций и останов работы алгоритма происходит, если достигнуто предельное число итераций $T$ (по умолчанию $T = 1000$).

Определение номера итерации, на которой было получено оптимальное решение, осуществляется путем визуального анализа по графикам изменения значений функции пригодности и ее компонентов, которые формируются в модуле визуализации.

На выходе модуля поиска решения после окончания работы генетического алгоритма будет находиться матрица $\mathbf{X}$ [$n$, $k$']. В диалоговой форме СПСД это решение отображается в блоке **Output Data**. Кроме того, в этом блоке отображаются следующие значения:

- общее время работы алгоритма (**time**);
- среднее время одной итерации (**time_per_iter**);
- достигнутое значение функции пригодности (**F**) и ее составляющих – количества ненулевых столбцов в решении (**k**) и евклидова расстояния (**sqrt**).

Характер поведения функции пригодности и ее составляющих в ходе работы генетического алгоритма для восьми наилучших решений отображается в виде графиков зависимости от номера итерации. Эти графики формируются в модуле визуализации и оценки.

Пример визуализации хода решения задачи приведен на Рис. 6.

В этом примере $n = 25$, $k = 5$. Как видно из рисунка, оптимальное решение задачи было найдено на 220-й итерации. При этом функция пригодности достигла максимального значения 0.2, а евклидово расстояние стало равным нулю.

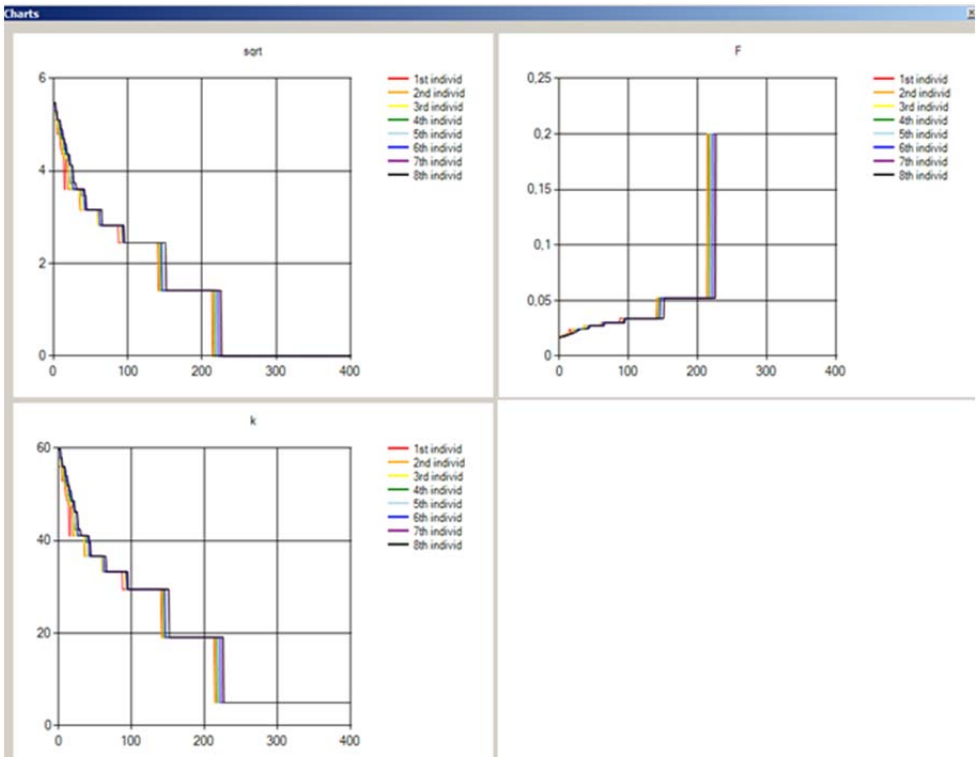



Рис. 6. Визуальное представление хода решения задач

В общем случае, после окончания работы генетического алгоритма количество столбцов $k'$ в матрице $\mathbf{X}$ не равно числу $k$ в эталонном решении $\mathbf{X}_{et}$. Сравнение $\mathbf{X}$ и $\mathbf{X}_{et}$, выполняемое в модуле визуализации и оценки, позволяет оценить точность решения задачи, производимого с помощью генетического алгоритма.

Разработанное СПСД было использовано для проектирования схемы доступа ВЛВС в корпоративной локальной сети, которая физически размещалась на трех этажах здания. В эту сеть входило около 100 компьютеров (серверов и рабочих станций). Для хранения схемы ВЛВС были использованы коммутаторы серий Cisco Catalyst 5000. Поддержка виртуальных подсетей осуществлялась на основе MAC-адресов.

Матрица требуемой логической связности формировалась на основе списков доступа, которые были сформированы сетевым администратором для каждой рабочей станции. Каждый список доступа формировал одну строку этой матрицы. С помощью СПСД осуществлялось формирование схемы ВЛВС как для всей сети, так и для ее фрагментов, состоящих из 10, 25, 50 и 75 рабочих станций.

Полученные результаты затем были реализованы в реальной сети. При этом снижение пропускной способности сети пользователями замечено не было. Тем не менее, потоки данных были полностью защищены от несанкционированного доступа к соседним компьютерам. Тем самым можно утверждать, что повысилась безопасность и надежность функционирования сети.

Используя СПСД, также были проведены эксперименты по оценке эффективности предложенного генетического алгоритма для указанных выше размерностей сети. Оценка производилась по следующим показателям:

- среднее количество итераций алгоритма, потребовавшихся для нахождения оптимального решения ($\overline{T}$);
- средняя длительность одной итерации ($\overline{\tau_{it}}$) для различных размерностей задачи;
- точность алгоритма ($\delta$).

Первые два показателя характеризуют скорость работы генетического алгоритма.

При определении $\overline{T}$ и $\overline{\tau_{it}}$ для каждой пары ($n$, $k$) было проведено пять тестов. Значение $k$

имели следующие величины: 0.2$n$, 0.3$n$, 0.5$n$ and 0.7$n$. Компьютер, на котором было установлено СПСД, имел следующую конфигурацию: процессор Intel(R) Atom(TM) CPU N2800, оперативная память 2 Гб, операционная система Windows 7.

Программный код для СПСД был выполнен на C#.

Точность $\delta$ алгоритма оценивалась путем сравнения $\mathbf{X}[n, k]$ и $\mathbf{X}_{et}[n, k']$ на основе следующего утверждения: «если $k' \le k$, тогда считается, что алгоритм нашел точное решение задачи, иначе – приближенное». Максимальное количество итераций было равно 1000. Для оценки точности алгоритма использовалась следующая формула:

$$\delta = \left[\left(n - \max\left(0; k' - k\right)\right)/ n\right] \cdot 100\% . \qquad (5)$$

В этом случае, если выполнялось условие $k' = k$, то независимо от значения $n$ точность составляет 100%. Если $k' > k$, то тогда точность зависит от соотношения между $(k' - k)$ и значением $n$. Так, если $n = 50$, $k = 10$ и $k' = 11$, то тогда $\delta = 98$.

Результаты оценки скорости и точности предложенного генетического алгоритма представлены в Табл. 1.

Анализируя данные, представленные в Табл. 1, можно сделать следующие выводы.

Во-первых, при малых размерностях задачи ($n = 10$ и $n = 25$) предложенный алгоритм позволяет решать задачу с максимальной точностью $\delta = 100\%$ в реальном или близком к реальному масштабу времени.

Во-вторых, при больших размерностях ($n = 75$ и $n = 100$) не удается получить максимальную точность за отведенное время, однако точность решения задачи остается достаточно большой и находится в диапазоне от 97 до 78 процентов.

Наконец, сложность решения задачи, при заданном значении $n$, возрастает при уменьшении значения $k$. Это хорошо видно на примере $n = 50$. Если при $k = 15$ и $k = 10$ получены максимально точные решения, то при $k = 10$ получено решение с точностью 98 процентов.

Кроме того, Табл. 1 раскрывает характер зависимости средней длительности одной итерации $\overline{\tau_{it}}$ от размерности компьютерной сети ($n$).

Табл.1 Экспериментальные результаты

| **Конфигурация** | | **Результаты оценки** | | |
|---|---|---|---|---|
| **Количество узлов в сети** ($n$) | **Количество виртуальных подсетей** ($k$) | $\overline{T}$ | $\overline{\tau_{it}}$, сек | $\delta$, % |
| 10 | 2 | 48.3 | 0,023 | 100 |
| 10 | 3 | 35.6 | 0,026 | 100 |
| 10 | 5 | 25.8 | 0,028 | 100 |
| 10 | 7 | 22,9 | 0,031 | 100 |
| 25 | 5 | 255.8 | 0,096 | 100 |
| 25 | 8 | 183.7 | 0,13 | 100 |
| 25 | 13 | 127.5 | 0,19 | 100 |
| 25 | 18 | 107.2 | 0,24 | 100 |
| 50 | 10 | 1000.0 | 0,96 | 98 |
| 50 | 15 | 811.4 | 1,19 | 100 |
| 50 | 25 | 534.6 | 1,33 | 100 |
| 50 | 35 | 483.4 | 1,65 | 100 |
| 75 | 15 | 1000.0 | 2,27 | 91 |
| 75 | 23 | 1000.0 | 4,58 | 95 |
| 75 | 38 | 1000.0 | 5,55 | 97 |
| 75 | 53 | 1000.0 | 6,65 | 98 |
| 100 | 20 | 1000.0 | 5,28 | 78 |
| 100 | 30 | 1000.0 | 6,98 | 85 |
| 100 | 50 | 1000.0 | 10,75 | 90 |
| 100 | 70 | 1000.0 | 14,64 | 92 |

Эта зависимость представлена на Рис. 7-*a* для различных значений отношения $k/n$. Видно, что при больших размерностях сети рост $\overline{\tau_{it}}$ имеет экспоненциальный характер.

На Рис. 7-*b* показана зависимость средней длительности одной итерации от отношения $k/n$. Видно, что при любых размерностях эта зависимость имеет линейный характер.

В целом из анализа экспериментальных результатов, показанных в таблице, можно сделать вывод, что предложенный генетический алгоритм обладает достаточно высокой эффективностью для решения задачи проектирования виртуальных компьютерных сетей.

## Заключение

В статье предложен подход к проектированию схемы доступа ВЛВС с использованием разработанного СПСД, который позволяет решать задачу поиска решения уравнения (1) с минимальным количеством ненулевых столбцов методом генетической оптимизации, дает визуальное отображение хода решения задачи и обеспечивает оценку генетического алгоритма.

Выбор генетического алгоритма в качестве метода решения был обусловлен следующими

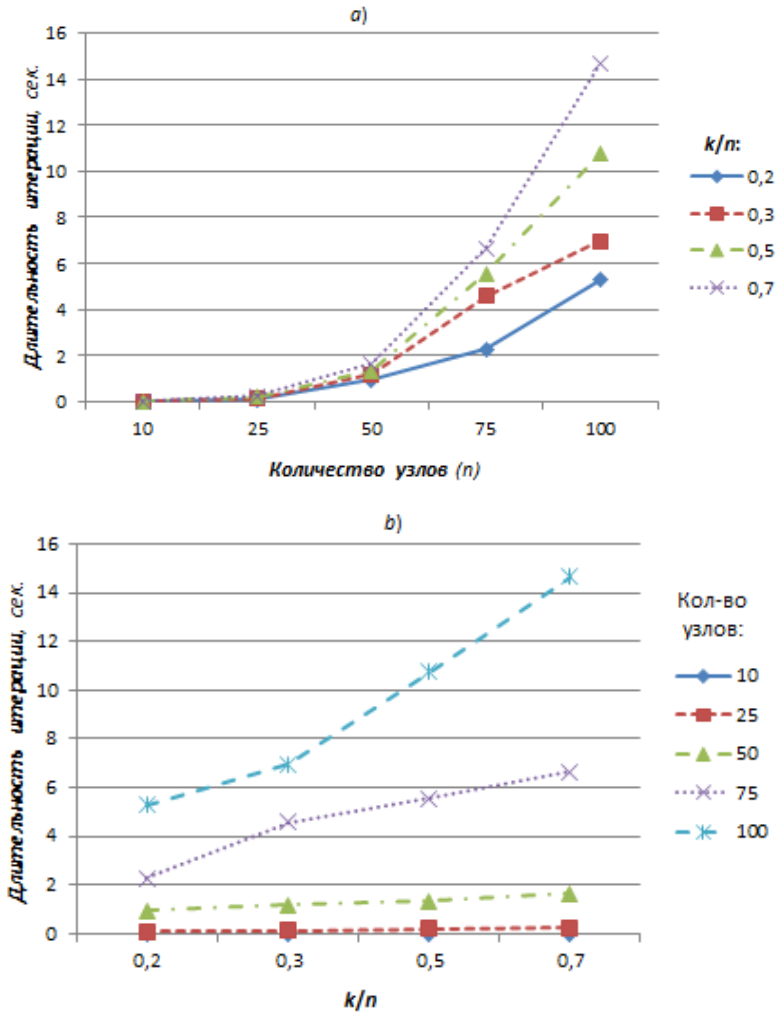



Рис. 7. Зависимость средней длительности одной итерации от размерности компьютерной сети (*a*) и отношения *k*/*n* (*b*)

факторами. Во-первых, было показано, что рассматриваемая задача является разновидностью задач БМФ, в которой обе искомые матрицы совпадают. По этой причине проблема относится к *NP*-полным задачам, требующим разработки и использования эвристических методов решения. Во-вторых, известны работы, в которых генетические алгоритмы успешно использовались для решения задач класса БМФ. Однако они не учитывают особенности рассматриваемой разновидности задачи. Поэтому генетический алгоритм потребовал разработки некоторых усовершенствований, учитывающий данные особенности.

Ключевыми усовершенствованиями предложенного генетического алгоритма являются:

- использование тривиальных решений для генерации начальной популяции;
- учет в функции пригодности критерия минимального числа виртуальных подсетей;
- использование столбцов матрицы связности в качестве генов хромосом, кодирующих решения;
- двумерный режим выполнения операции скрещивания в алгоритме;
- двухфазное выполнение операции мутации в алгоритме;
- обнуление дублирующих столбцов в матрице дочерних решений, получающихся при скрещивании или мутации.

Разработанное СПСД обладает следующими возможностями:

- ввод и корректировка исходных данных с помощью диалоговой формы;
- генерация матрицы требуемой логической связности с помощью ДПСЧ;
- визуальное отображение хода работы генетического алгоритма с помощью графиков изменения значений функции пригодности и ее компонентов для ряда наилучших решений;
- оценка временных параметров работы генетического алгоритма.

Экспериментальная оценка предложенного алгоритма по оперативности и точности показала его достаточно высокую эффективность.

Будущие исследования связываются с вопросами применения предложенного подхода к задаче реконфигурирования виртуальных подсетей.

## Литература


1. Catalyst 2900 Series XL and Catalyst 3500 Series XL Software Configuration Guide. Cisco IOS Release 12.0(5) WC(1), Cisco Systems, San Jose, 2001.
2. Oracle Real Application Clusters (RAC) and Oracle Clusterware Interconnect Virtual Local Area Networks (ВЛВСs) Deployment Considerations, An Oracle White Paper, June 2012.
3. Miettinen P., Vreeken J. Model Order Selection for Boolean Matrix Factorization // Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, ACM, New York, 2011, pp. 51-59.
4. Miettinen P. Dynamic Boolean Matrix Factorizations // Proceedings of the 2012 IEEE 12th International Conference on Data Mining, ACM, New York, 2012, pp. 519-528.
5. Cergani E., Miettinen P. Discovering Relations using Matrix Factorization Methods // Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM, New York, 2013, pp. 1549-1552.
6. Lu H., Vaidya J., Atluri V., Hong Y. Extended Boolean Matrix Decomposition // Proceedings of the Ninth IEEE International Conference on Data Mining, IEEE Press, New York, 2009, pp. 317-326.
7. Lu H., Vaidya J., Atluri V. Optimal Boolean Matrix Decomposition. Application to Role Engineering // Proceedings of the 24th IEEE International Conference on Data Engineering, IEEE Press, New York, 2008, pp. 297-306.
8. Saenko I., Kotenko I. Genetic Algorithms for Role Mining Problem // Proceedings of the 19th International Euromicro Conference on Parallel, Distributed and Network-based Processing, IEEE Press, New York, 2011, pp. 646-650.
9. Saenko I., Kotenko I. Design and Performance Evaluation of Improved Genetic Algorithm for Role Mining Problem // Proceedings of the 20th International Euromicro Conference on Parallel, Distributed and Network-based Processing, IEEE Press, New York, 2012, pp.269-274.
10. Janecek A., Tan Y. Using Population Based Algorithms for Initializing Nonnegative Matrix Factorization // Advances in Swarm Intelligence. LNCS, vol. 6729, 2011, pp. 307-316.
11. Snasel V., Platos J., Kromer P. On Genetic Algorithms for Boolean Matrix Factorization // Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications, vol. 2, IEEE Press, New York, 2008, pp. 170-175.
12. Snasel V., Platos J., Kromer P., Husek D., Neruda R., Frolov A.A. Investigating Boolean Matrix Factorization // Proceedings of the Workshop on Data Mining using Matrices and Tensors, 2008.
13. Tai Ch.-F., Chiang Tz.-Ch., Hou T.-W. A Virtual Subnet Scheme on Clustering Algorithms for Mobile Ad Hoc Networks // Expert Systems with Applications. 38(3), 2011, pp. 2099-2109.
14. Saenko I., Kotenko I. Genetic Optimization of Access Control Schemes in Virtual Local Area Networks // Computer Network Security, LNCS, vol. 6258, 2010, pp. 209-216.
15. Goldberg D.E. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Longman Publishing, Boston, 1989.
16. Mitchell M. An Introduction to Genetic Algorithms, MIT Press, Massachusetts, 1998.
17. Eiben A.E., Smith J.E. Introduction to Evolutionary Computing, Springer, Heidelberg, 2007.
18. Barker E., Kelsey J. Recommendation for Random Number Generation Using Deterministic Random Bit Generators, NIST Special Publication, NIST, 2012.


**Саенко Игорь Борисович.** Ведущий научный сотрудник СПИИРАН. Окончил Белорусский государственный университет в 1981 году и Военную академию связи в 1989 году. Доктор технических наук, профессор. Автор более 200 научных работ. Область научных интересов: защита информации в компьютерных сетях, искусственный интеллект, базы данных, поддержка принятия решений при проектировании и планировании использования информационных систем. E-mail: ibsaen@comsec.spb.ru

**Котенко Игорь Витальевич.** Заведующий лабораторией СПИИРАН. Окончил Военно-космическую академию им. А.Ф. Можайского в 1983 году и в. Военную академию связи в 1987 году. Доктор технических наук, профессор. Автор более 500 научных работ. Область научных интересов: безопасность компьютерных сетей, в том числе анализ защищенности, обнаружение компьютерных атак, межсетевые экраны, защита от вирусов и сетевых червей, технологии моделирования и визуализации для противодействия кибер-терроризму. E-mail: ivkote@comsec.spb.ru