ПРОГРАМНО-ТЕХНІЧНІ КОМПЛЕКСИ

С.Л. КРЫВЫЙ, С.Д. ПОГОРЕЛЫЙ, Н.Н. ГЛИБОВЕЦ, Ю.В. БОЙКО, Н.Н. СИДОРОВА

УДК 51.681.3

ПРОЕКТИРОВАНИЕ ИТ-ИНФРАСТРУКТУРЫ

Аннотация. Предложена методика проектирования ИТ-инфраструктуры в условиях ограниченных ресурсов. Методика базируется на свойствах алгебры графов, генетических алгоритмах, сетях Петри и особенностях ограничений. Рассмотрены два подхода к решению задачи балансирования нагрузок в ИТ-инфраструктуре.

Ключевые слова: алгебра графов, топология, ИТ-инфраструктура, генетические алгоритмы, сети Петри.

ВВЕДЕНИЕ

Проектирование ИТ-инфраструктуры, как правило, начинается с разработки математической модели ее топологии, т.е. модели организации внутренних и внешних связей между входящими в ее состав объектами [1]. Выбор топологии ИТ-инфраструктуры выполняется по результатам анализа различных моделей такой топологии. Наиболее подходящими моделями топологии ИТ-инфраструктуры в данном случае выступают графовые модели, в которых объекты ассоциируются с вершинами графа, а связи — с ребрами графа. При этом предпочтение отдается неориентированным графам, поскольку считается, что ребро неориентированного графа моделирует двустороннюю связь между объектами. Модель, обладающая необходимыми свойствами, выбирается для реализации описываемой ею топологии проектируемой ИТ-инфраструктуры. Основными свойствами топологии являются ее устойчивость к повреждениям линий связи и/или составляющих ее объектов. Правильно спроектированная топология должна быть представлена связным графом, поскольку его связность гарантирует доступ к каждому объекту ИТ-инфраструктуры. Если топология ИТ-инфраструктуры представлена связным графом, то такую структуру будем называть живой.

Сложность выбора подходящей топологии состоит в том, что эта задача носит противоречивый характер. Противоречивость заключается в том, что выбор наилучшей модели зависит от ряда ограничений, которые накладывают возможности реализации. Одни ограничения носят финансовый характер (ограничения финансирования, не позволяющие приобрести линии связи с хорошей пропускной способностью), другие ограничения имеют объективный характер и связаны с особенностью местности, по которой должны проходить линии связи (водные препятствия, городские коммуникации, не позволяющие протянуть линии связи, уровень качества имеющихся каналов и другие причины). В связи с этим наилучшую топологию нельзя реализовать в условиях имеющихся ограничений. Возникает вопрос: как в условиях данных ограничений выбрать наилучшую (оптимальную) топологию? В настоящей статье предлагаются подходы к решению этой задачи.

1. СТЕПЕНЬ СВЯЗНОСТИ СВЯЗНОГО ГРАФА

Неориентированным графом называется пара $G = (V, E)$, где V — множество вершин графа, E — множество ребер, состоящее из неупорядоченных пар элементов из V (т.е. пары (u, v) и (v, u) считаются одинаковыми). Граф G называется конечным, если конечно множество его вершин V. Если $(u, v) \in E$, то вершины u, v называются концами этого ребра. Ребро $e \in E$ называется инцидентным вершине $u \in V$, если эта вершина является концом ребра e. Степенью вершины $u \in V$ называется число ребер, инцидентных этой вершине. Степень вершины u обозначим $n(u)$. Если степень вершины равна нулю, то такая вершина называется изолированной, а если степень вершины равна единице, то такая вершина называется концевой или висячей. Две вершины $u, v \in V$ графа $G(V, E)$ связаны маршрутом, если существует такая последовательность его вершин $u_1, u_2, \ldots, u_k$, что $u = u_1$, $v = u_k$ и $(u_i, u_{i+1}) \in E$ для всех $i = 1, 2, \ldots, k-1$. Число $k-1$ называется длиной этого маршрута. Маршрут, первая и последняя вершины которого совпадают, называется циклом. Граф без циклов называется ациклическим.

Граф $G = (V, E)$ называется связным, если любые две вершины этого графа связаны маршрутом. Связный ациклический граф называется деревом. Вершина дерева называется внутренней, если степень этой вершины больше единицы. Проверка связности графа выполняется алгоритмом поиска в глубину (DFS-алгоритм), который подробно описан в [2].

1.1. Алгебра графов. Пусть U — некоторое универсальное множество, элементы которого используются для обозначения вершин. Обычно $U = \mathbf{N}$, где $\mathbf{N}$ — множество натуральных чисел.

Основные операции на конечных графах над множеством U:

- $\Lambda = (\varnothing, \varnothing)$ — абсолютно пустой граф;
- $G \ominus e = (V, E \setminus \{e\})$ — удаление ребра e из графа $G = (V, E)$;
- $G - v = (V \setminus \{v\}, E \setminus \{v, v_i\})$ — удаление вершины v из графа $G = (V, E)$;
- $G \oplus e = (V, E \cup \{e\})$ — введение ребра e в граф $G = (V, E)$;
- $G + v = (V \cup \{v\}, E)$ — введение вершины v в граф $G = (V, E)$.

Эти операции составляют полное множество операций алгебры конечных графов $AG = (B(U), \{\ominus, \oplus, -, +, \Lambda\})$ [3].

Поскольку операции $\oplus$ и $+$ сводятся к операциям объединения соответствующих множеств, то в силу ассоциативности и коммутативности операции объединения используем сокращения $G \oplus \{e_1, e_2, \ldots, e_k\}$ для последовательности операций $G \oplus \{e_1\} \oplus \{e_2\} \oplus \ldots \oplus \{e_k\}$. Аналогичное сокращение будем использовать и для операции введения вершин в граф. Заметим также, что операция $\ominus$ и операция $-$ являются коммутативными по второму аргументу, т.е. $(G \ominus \{e\}) \ominus \{e_1\} = (G \ominus \{e_1\}) \ominus \{e\}$ и $(G - \{v\}) - \{v_1\} = (G - \{v_1\}) - \{v\}$. Это означает, что порядок удаления ребер и вершин из графа несущественен.

Операция удаления ребра $e = (u, v)$ в графе $G = (V, E)$ может привести к графу, который не является связным. Аналогичная ситуация и относительно операции удаления вершины. Поскольку вместе с вершинами удаляются и все инцидентные ей ребра, то возникает вопрос, сколько ребер можно удалить из связного графа $G = (V, E)$, чтобы после выполнения этих операций полученный граф оставался связным.

Вершина графа называется точкой сочленения, если ее удаление приводит к несвязному графу. Ребро графа называется мостом или существенным ребром, если его удаление приводит к несвязному графу. Отсюда следует, что слабыми звеньями в любой топологии проектируемой ИТ-инфраструктуры являются мосты и точки сочленения. Введем такое определение.

Определение 1. Степенью связности связного графа называется максимальное число d ребер, удаление которых из графа не выводит его из класса связных графов, а удаление $(d+1)$-го ребра приводит к несвязному графу.

Известно, что удаление ребра, принадлежащего некоторому циклу графа $G=(V, E)$, не нарушает связности графа, а степень связности равна его цикломатическому числу $d=|E|-|V|+1$ (см. [2]). Следовательно, число возможных операций удаления ребра в связном графе $G=(V, E)$, не выводящих граф из класса связных, равно d, а операция удаления вершины в связном графе не нарушает его связности, если все ребра, инцидентные этой вершине, принадлежат некоторым циклам этого графа или эта вершина концевая.

Из сказанного следует, что наиболее уязвима топология, графом которой является дерево, а наименее уязвимая топология имеет граф полный. Действительно, в дереве каждая внутренняя вершина является точкой сочленения, а каждое ребро является мостом. Если в полном графе n-го порядка при удалении вершины он остается полным графом $n-1$-го порядка, то степень его связности равна $\frac{n^2-3n+2}{2}$. Так, для полного графа пятого порядка степень его связности равна шести.

1.2. Производные операции алгебры графов. Рассмотренные выше операции алгебры графов AG являются основными в смысле определения универсальной алгебры [3]. Кроме этих операций на графах определяются и такие, которые выражаются посредством основных операций. Рассмотрим те из них, которые далее используются в работе.

Пусть $G=(V, E)$ и $G_1=(V_1, E_1)$ — конечные графы, тогда

— объединением графов G и G_1 называется граф $G\cup G_1=(V\cup V_1, E\cup V_1)$;

— пересечением графов G и G_1 называется граф $G\cap G_1=(V\cap V_1, E\cap V_1)$;

— соединением графов G и G_1 таких, что $V\cap V_1=\varnothing$, называется граф $G*G_1=(V\cup V_1, E')$, где $E'=E\cup E_1\cup\{(u, v)\,|\,\forall u\in V, v\in V_1\}$;

— декартовым произведением графов G и G_1 называется граф $G\times G_1=(V\times V_1, E')$, где $((u, v), (u_1, v_1))\in E'\Leftrightarrow (u=u_1\wedge(v, v_1)\in E_1)\vee((u, u_1)\in\in E\wedge(v=v_1))$;

— дополнением графа G называется граф $\overline{G}=(V, V\times V\setminus\{E\cup i_V\})$, где $i_V=\{(u, u)\,|\,u\in V\}$ — диагональ множества V.

Непосредственно из определений операций следует справедливость таких свойств: $\forall G, H, Q\in AG$ имеет место

$$G\cup\Lambda=G,\ G\cup G=G,\ G\cup H=Y\cup G,\ G\cup(H\cup Q)=(G\cup H)\cup Q;$$

$$G\cap\Lambda=\Lambda,\ G\cap G=G,\ G\cap H=H\cap G,\ G\cap(H\cap Q)=(G\cap H)\cap Q;$$

$$G\cap(H\cup G)=G\cup(H\cap G)=G;$$

$$G*\Lambda=\Lambda,\ G*H=H*G,\ G*(H*Q)=(G*H)*Q;$$

$$G*(H\cup Q)=G*H\cup G*Q,\ G*(H\cap Q)=G*H\cap G*Q;$$

$$G\times\Lambda=\Lambda,\ G\times H=H\times G,\ G\times(H\times Q)=(G\times H)\times Q;$$

$$G\times(H\cup Q)=(G\times H)\cup(G\times Q),\ G\times(H\cap Q)=(G\times H)\cap(G\times Q).$$

Из первых трех групп свойств операций следует, что относительно операций пересечения и объединения алгебра AG является решеткой с нулем, в роли которого выступает абсолютно пустой граф Λ. Поскольку носитель решетки — частично упорядоченное множество, определяемое соотношением $G\le H\Leftrightarrow$ $\Leftrightarrow G\cap H=G$ или двойственным образом $G\le H\Leftrightarrow G\cup H=H$, то из $G\cap H=$ $=G=(V\cap V_1, E\cap E_1)=(V, E)$ следует, что граф G является подграфом графа H. Нетрудно понять, что атомами в данной решетке, т.е. минимальными связными графами, будут деревья с двумя вершинами, соединенными ребром.

Алгебру AG, сигнатура операций которой расширена введенными выше операциями, назовем расширенной алгеброй графов.

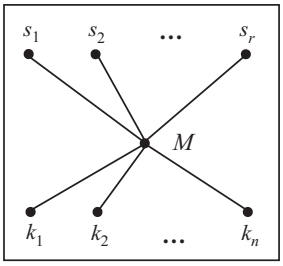

Рис. 1. Одноканальная магистраль

2. ХАРАКТЕРИСТИКИ ТОПОЛОГИЙ

Рассмотрим некоторые часто используемые топологии, в основе которых лежат определенные типы графов (см. [4]), и охарактеризуем эти топологии, исходя из определенной выше алгебры AG.

2.1. Одноканальная магистраль (дерево). Для представления ИТ-инфраструктуры с этой топологией магистраль будем рассматривать как объект, изображаемый вершиной графа. Тогда граф, лежащий в основе этой топологии, имеет вид звезды (а по сути дерева), показанной на рис. 1, где $s_1, s_2, \dots, s_r$, $k_1, k_2, \dots, k_n$ — объекты, а M — магистраль. Этот граф имеет одну внутреннюю вершину (магистраль) степени $n(M) = r+n$ и $r+n$ концевых вершин. Выражение этой топологии в алгебре графов AG имеет вид

$$G = (\Lambda + \{s_1, \dots, s_r, k_1, \dots, k_n\}) \oplus \{(M, s_1), \dots, (M, s_r), (M, k_1), \dots, (M, k_n)\}.$$

В терминах операций расширенной AG алгебраическое выражение принимает вид

$$G = M * \{s_1, s_2, \dots, s_r\} \cup M * \{k_1, k_2, \dots, k_n\} = M * \{s_1, s_2, \dots, s_r, k_1, k_2, \dots, k_n\}.$$

Как видно из приведенных выражений, важным свойством данной топологии является ее устойчивость относительно удаления любого числа конечных вершин, неустойчивость относительно удаления внутренней вершины (магистрали) и удаления хотя бы одного ребра. Следовательно, при применении этой топологии необходимо использовать надежные линии связи и надежную магистраль.

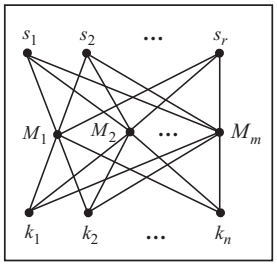

Рис. 2. Многоканальная магистраль

2.2. Многоканальная магистраль (полный трехдольный граф). Усовершенствованием одноканальной магистрали для повышения надежности ИТ-инфраструктуры является многоканальная магистраль. Граф, лежащий в основе этой топологии, есть трехдольный граф (рис. 2), где $s_1, s_2, \dots, s_r$, $k_1, k_2, \dots, k_n$ — объекты, $M_1, \dots, M_m$ — магистрали.

Степень связности данного графа составляет $d = |E| - |V| + 1 = rm + nm - r - n - m + 1 = (r+n-1)(m-1)$. Если эту топологию сравнивать с предыдущей, где $m = 1$ и $d = 0$, то степень надежности такой ИТ-инфраструктуры увеличивается в $m-1$ раз. Алгебраическое выражение для этой топологии имеет вид

$$G = (\Lambda + \{s_1, \dots, s_r, k_1, \dots, k_n, M_1, \dots, M_m\} \oplus \{(M_1, s_1), \dots, (M_1, s_r), \dots$$
$$\dots, (M_m, s_1), \dots, (M_m, s_r), (M_1, k_1), \dots, (M_1, k_n), \dots, (M_m, k_1), \dots, (M_m, k_n)\}.$$

В терминах операций расширенной алгебры графов эта топология имеет вид

$$G = \{M_1, M_2, \dots, M_m\} * \{s_1, \dots, s_r, k_1, \dots, k_n\}.$$

Отсюда следует, что выход из строя любой $m-1$ магистрали оставляет ИТ-инфраструктуру живой (в этом случае она редуцируется к одноканальной магистрали). Аналогичное утверждение справедливо и для ребер.

2.3. Шестигранник (куб). Граф, лежащий в основе этой топологии, является шестигранником (рис. 3). Степень связности для этого графа составляет $d = 12 - 8 + 1 = 5$. Алгебраическое выражение для этого графа имеет вид

$$G = (\Lambda + \{1, 2, 3, 4, 5, 6, 7, 8\}) \oplus \{(1, 2), (1, 4), (1, 5), (2, 3), (2, 6),$$
$$(3, 4), (3, 7), (4, 8), (5, 6), (5, 8), (6, 7), (7, 8)\}. \quad (1)$$

В терминах операций расширенной алгебры графов алгебраическое выражение этого графа принимает вид декартового произведения двух графов:

$$G = (V = \{1, 2\}, E = \{(1, 2)\}) \times (V_1 = \{3, 4, 5, 6\}, E_1 = \{(3, 4), (4, 5), (5, 6), (6, 3)\}). \quad (2)$$

Построенные по этим выражениям графы являются изоморфными, в чем можно убедиться непосредственно.

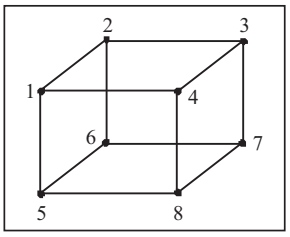

Рис. 3. Шестигранник

Из представления (1) следует, что данная топология устойчива к удалению любых двух вершин и любых двух ребер, а также что из первого аргумента можно удалить четыре ребра, а из второго аргумента — одно ребро и наоборот. Отсюда также вытекает, что из данного графа можно удалить все вершины, принадлежащие одной грани, и результирующая структура будет живой.

Из представления (2) для этого графа очевидным образом можно построить гипершестигранник.

2.4. Гипершестигранник. Граф, лежащий в основе этой топологии, называется гипершестигранником (рис. 4).

Более точную характеристику можно получить, проанализировав алгебраическое выражение для этого графа, которое имеет вид

$$(T_{1,2} \cup T_{2,3} \cup T_{3,4} \cup T_{4,1}) \times (T_{5,6} \cup T_{6,7} \cup T_{7,8} \cup T_{8,5}),$$

где $T_{i,j}$ — дерево с двумя вершинами i, j, соединенными ребром (минимальный связный граф).

Из этого представления следует, что удаление любых двух смежных вершин в любом аргументе приводит к связному графу, который изоморфен шестиграннику. Отсюда вытекает, что удаление любых двух граней (или вершин этих граней) и любых двух ребер не нарушает связности гипершестигранника. Кроме того, удаление ребра в любом аргументе в приведенном выше выражении оставляет граф связным. А это значит, что в результирующем графе можно удалить восемь ребер, которые соответствуют удаленному ребру.

Максимальное число ребер, которые можно удалить из этого графа, составляет $d = |E| - |V| + 1 = 36 - 16 + 1 = 21$. Отсюда следует, что этот граф более устойчив по отношению к операции удаления ребра, чем шестигранник.

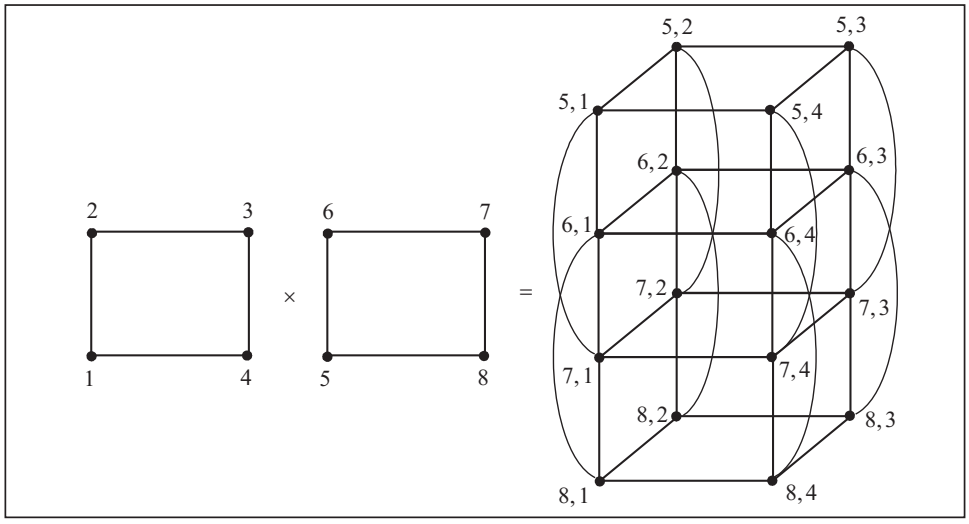

Рис. 4. Гипершестигранник

2.5. Решетка. В основе этой топологии лежит n-мерная решетка (рис. 5, двумерная решетка — наиболее простой вариант).

Алгебраическое выражение для этого графа имеет вид

$$[(T_{1,2} \cup T_{2,3}) \times (T_{4,5} \cup T_{5,6}),$$

где $T_{i,j}$ — деревья, аналогичные представленным в п. 2.4. Отсюда следует, что из аргументов нельзя удалить ни одного ребра, поскольку результирующий граф станет несвязным, т.е. аргументы этого произведения будут нередуцируемы к более простым связным графам, которые порождают такую решетку.

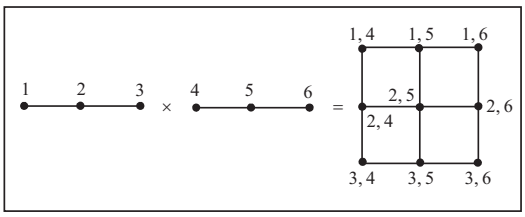

Рис. 5. Решетка

Из сказанного вытекает, что алгебра графов позволяет проектировать ИТ-инфраструктуру методом снизу–вверх, исходя из имеющихся средств и возможностей, в то время как операции расширенной алгебры позволяют проектировать топологию ИТ-инфраструктуры методом сверху–вниз. Это означает, что вначале выбирается максимально надежная топология, а затем на основании этой топологии удаляются те элементы, реализация которых в силу имеющихся ограничений невозможна. В результате проектирования будет выбрана топология, которая реализуема имеющимися средствами.

3. ПРИМЕР ПРОЕКТИРОВАНИЯ ТОПОЛОГИИ

Задача состоит в проектировании топологии ИТ-инфраструктуры некоторой организации, состоящей из десяти объектов: $o_1, o_2, \ldots, o_{10}$, географически расположенных в рамках одного города с максимальной отдаленностью на d объектов один от другого.

3.1. Проектирование методом снизу–вверх. На первом шаге проектирования топологии строим граф $G = \Lambda + \{o_1, o_2, \ldots, o_{10}\}$ (рис. 6, *а*). Из этих вершин выделяем основные, через которые должны проходить все или бóльшая часть информационных потоков, например вершины o_1, o_8, o_4. Такое выделение необходимо для представления возможной топологии.

Далее строим дерево из имеющихся вершин, что приводит, например, к графу $G_1 = G \oplus \{(o_1, o_8), (o_2, o_8), \ldots, (o_8, o_{10})\}$. Из рис. 6, *а* видно, что основная связь должна быть между объектами $o_1 - o_8 - o_4$. Тогда такая линия является моделью одноканальной магистрали, представленной вершиной o_8, к которой подключены все остальные объекты. Получаем граф, приведенный на рис. 6, *б*. При этом линии связи $o_1 - o_8$, $o_8 - o_4$ должны быть наиболее надежными в проектируемой инфраструктуре. Далее происходит оценка стоимости линий связи и возможности их реализации (что может изменить приоритеты), а затем определяются дополнительные линии связи, выбор которых продиктован внутренними потребностями с учетом имеющихся ограничений (например, связь $o_4 - o_9$).

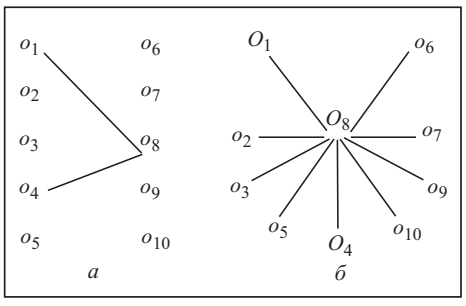

Рис. 6. Исходные графы

Если полученную топологию принять за основу, то для обеспечения ее надежности представим топологию колеса, наиболее близкую к полученной.

Степень связности данного графа равна девяти. Как видно из рис. 7, *а*, такая топология устойчива не только в случае выхода линий связи, но и при выходе из строя двух из основных вер-

шин. Если такую топологию можно реализовать в условиях имеющихся ограничений, то она принимается к реализации. После этого окончательно определяется стоимость затрат. Если эти ограничения не позволяют реализовать такую топологию, то она оптимизируется к виду, который удовлетворяет условиям ограничений.

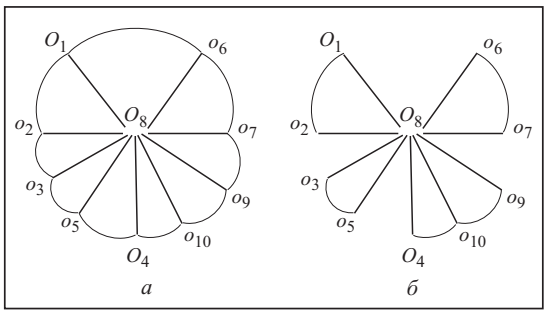

Рис. 7. Колесо и результат его оптимизации

3.2. Проектирование методом сверху–вниз. Проектирование целесообразно начать с топологии колеса или другой подходящей топологии (например, полного графа десятого порядка), а затем оптимизировать ее к виду, который удовлетворяет условиям ограничений. Если таким способом оптимизировать топологию колеса (см. рис. 7, *а*) с учетом ограничений, то приходим к топологии, приведенной на рис. 7, *б*. В результате получаем связный граф, степень связности которого равна пяти, а топология на этом графе является устойчивой к удалению любой вершины, кроме основной вершины o_8.

4. ПЕРЕРАСПРЕДЕЛЕНИЯ НАГРУЗОК В КАНАЛАХ

Следующей важной задачей является задача перераспределения нагрузок в каналах ИТ-инфраструктур. Будем решать эту задачу с помощью генетических алгоритмов (ГА), которые в общем случае дают приближенное решение, и сетевыми методами, которые дают точное решение задачи перераспределения нагрузки. Первые методы имеют полиномиальные оценки сложности, и это проявляется на графах больших размеров, а вторые методы эффективно работают на графах небольших размеров.

4.1. Генетические алгоритмы как метод балансирования нагрузки в каналах. Важность задачи распределения нагрузки в ИТ-инфраструктурах связана с обеспечением качества обслуживания пользователей. Возникновение новых подходов к распределенным вычислениям (облачные вычисления) и создание новых сервисов в ИТ-инфраструктурах не только накладывают дополнительные требования к надежности и скорости работы алгоритмов транспортировки данных, но и требуют создания эффективных методов балансирования нагрузок на ресурсы инфраструктуры (каналы связи, вычислительные элементы и др.). Классические алгоритмы решения задач транспорта данных оперируют только одним параметром оптимизации — весом (ценой) пути, который выражает совокупность его аддитивных характеристик. Однако, как правило, существует несколько параметров, характеризующих каждую ветвь сети (например, пропускная способность, задержка, скорость передачи, надежность), которые можно разделить на аддитивные и неаддитивные. В современных ИТ-инфраструктурах возникает необходимость решения задачи о кратчайших путях с несколькими критериями оптимизации. Классические алгоритмы неприменимы к задачам такого типа, поскольку они принадлежат классу NP-сложных проблем. Вычислительные затраты на решение таких задач экспоненциально возрастают с увеличением размерности обрабатываемых графов. Поэтому возникает необходимость формирования новых подходов и алгоритмов решения задач поиска оптимальных путей со многими критериями. Исследования последних десятилетий сформировали новую отрасль эволюционных вычислений, из которых важное место занимают генетические алгоритмы [8, 9].

4.1.1. Преимущества генетического алгоритма. Рассмотрим основные преимущества ГА в сравнении с классическими методами [8].

— ГА присущ параллелизм. Большинство других алгоритмов поиска и оптимизации строго последовательны и могут исследовать пространство решений задачи только в одном направлении. Если найденное решение окажется неоптимальным, единственным вариантом является отказ от всех полученных результатов и повтор работы алгоритма с самого начала. Поскольку ГА сохраняют информацию о своих потомках, они могут исследовать пространство решений в нескольких направлениях одновременно. Если один путь окажется тупиковым, ГА могут легко удалить решение и продолжить работу в более перспективных направлениях, что дает больше шансов найти оптимальное решение для каждого запуска алгоритма.

— Выполнимость ограничений задачи оптимизации происходит за счет включения их в метод кодирования хромосом.

— Существует возможность решения задач многоцелевой (многокритериальной) оптимизации. Многие реальные задачи не могут быть сведены к оптимизации одной величины, а только к комплексу взаимно зависимых параметров. Доказана эффективность применения ГА при решении такого класса задач.

— ГА дают возможность оптимизации топологии или структуры параллельно с оптимизацией параметров решения для конкретной задачи.

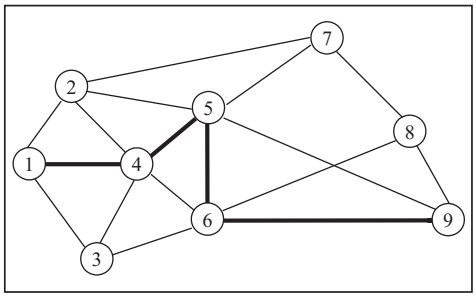

Рис. 8. Путь (1, 4, 5, 6, 9)

4.1.2. Формирование ГА решения задачи поиска кратчайших путей. Генетический алгоритм маршрутизации базируется на алгоритме поиска такого пути по многим критериям на графе $G = (V, E)$ [10].

Формулировка задачи поиска кратчайшего пути на графе. В общем случае имеем несколько весовых функций $w_1, \ldots, w_k : E \to R$, каждая из которых соответствует определенному критерию оптимизации (k — количество таких критериев). Произвольный путь $p = v_i \to v_j$ состоит из последовательности ребер $(v_i, v_l), \ldots, (v_k, v_j) \in E$ и может быть представлен в виде последовательности вершин графа, которые принадлежат пути $p = (v_i, v_l, \ldots, v_j)$. Каждая из вершин $v_i, v_l, \ldots, v_j \in V$ должна принадлежать пути только один раз. Например, на рис. 8 последовательность (1, 4), (4, 5), (5, 6), (6, 9) является путем из вершины 1 к вершине 9 и соответствует представлению (1, 4, 5, 6, 9).

Пусть индекс s соответствует начальной, а l — конечной вершинам искомого пути: $p = v_s \to v_l$. Определим x_{ij}:

$$x_{ij} = \begin{cases} 1, & \text{если ребро } (i, j) \text{ принадлежит пути,} \\ 0 & \text{в противном случае.} \end{cases}$$

Пусть также k — общее количество критериев оптимизации задачи. По каждому критерию можно вычислить определенный функционал — целевую функцию, которая соответствует качеству пути, исходя из алгоритма маршрутизации, и определяется как

$$C_m(p) = F_m(w_m(i, j), x_{ij}), \quad m = 1, \ldots, k, \quad (i, j) \in E,$$

где $w_m(i, j)$ — весовая функция ребра (i, j).

Для аддитивных характеристик пути (задержка, длина), используемых как метрику современных алгоритмов маршрутизации, F_m является суммой значений весовой функции ребер, составляющих путь. Для неаддитивной характеристики пути (нагрузки, пропускная способность, надежность) функционал является сложной функцией от многих параметров и может учитывать не только состояние соединений, но и состояние маршрутизаторов сети, изменение среды передачи данных и т.п.

Обозначим P множество всех возможных путей между вершинами v_i и v_j. В общем случае задачу поиска кратчайшего пути по многим критериям между двумя определенными вершинами на графе можно сформулировать следующим образом:

$$\min_P \overline{N}_m(p) = F_m(w_m(i,j)x_{ij}),\ \ m=1,\dots,k,\ \ (i,j)\in E; \tag{3}$$

$$\sum_{j=s,\, j\neq i}^{d} x_{ij} - \sum_{j=s,\, j\neq i}^{d} x_{ji} = \begin{cases} 1, & \text{если } i=s, \\ -1, & \text{если } i=l, \\ 0 & \text{иначе;} \end{cases} \tag{4}$$

$$\sum_{j=s,\, j\neq i}^{d} x_{ij} = \begin{cases} 1, & \text{если } i\neq l, \\ 0, & \text{если } i=l. \end{cases} \tag{5}$$

Условия (4) и (5) требуют, чтобы искомый путь не содержал циклов. Условие (3) требует, чтобы целевая функция по каждому критерию оптимизации по всем возможным путям $p=v_s\to v_d\in P$ достигала наименьшего значения на искомом пути. Задача (3) является NP-сложной; исходя из этого рассмотрим формирование ГА для ее решения.

Особенности метода кодирования решений алгоритма. Рассмотрим определенный путь $p=(v_s, v_i, v_j, \dots, v_d)$, где $v_s, v_i, v_j, \dots, v_d \in V$, который является промежуточным решением задачи. Хромосома генетического алгоритма, которая представляет решение p, состоит из последовательности целых положительных чисел, соответствующих идентификаторам вершин графа, через которые проходит путь p. Каждый локус (положение) хромосомы определяет порядковый номер вершины пути (ее идентификатор совпадает со значением соответствующего гена). Гены первого и последнего локусов хромосомы зарезервированы за начальной и конечной вершинами пути соответственно. Общий вид хромосомы изображен на рис. 9 (точка в локусе означает помеченный ген).

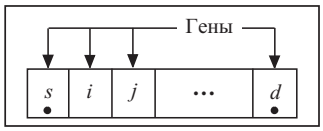

Рис. 9. Общий вид хромосомы

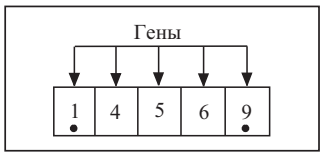

Рис. 10. Вид хромосомы, соответствующей пути (1, 4, 5, 6, 9)

Возвратимся к рис. 8. Пусть $s=1$, $d=9$. На рис. 10 изображена хромосома, которая соответствует пути (1, 4, 5, 6, 9). Длина хромосомы может варьироваться от двух до $|V|$. Такое представление решения задачи несколько затрудняет формирование генетических операций кроссовера и мутации, однако позволяет однозначно определить путь по его представлению и наоборот — по его представлению определить путь.

Методы генерации начальной популяции. Первым этапом работы ГА является выбор начальной популяции и ее размера. Начальная популяция, состоящая из большого количества хромосом, позволяет намного быстрее найти решение достаточной точности. Однако такой подход требует значительных вычислительных затрат для реализации алгоритма: чем больше размер популяции, тем больше памяти вычислительной системы будет затрачено и время выполнения всех генетических операций (кроссовер, мутация, отбор) будет возрастать. Размер популяции будет экспоненциально расти в зависимости от сложности поставленной задачи для получения достаточно хорошего результата. Однако последние исследования в области ГА показали, что хороший результат за приемлемое время можно получить, используя популяции значительно меньшего размера. Эксперимент показывает, что для задачи о кратчайшем пути в графе достаточный размер популяции составляет $N=2|V|$. Важным также является выбор способа формирования начальной популяции. Существуют два основных подхода решения этой задачи:

1) генерация случайных хромосом;

2) формирование начальной популяции с использованием классического метода [7].

Первый вариант позволяет значительно упростить процесс создания начальной популяции алгоритма, уменьшая, таким образом, время работы алгоритма в целом. Такой подход также целесообразен в случае отсутствия классического алгоритма решения исходной задачи.

Второй подход требует значительных вычислительных затрат, однако позволяет достичь желаемого результата работы ГА при значительно меньшем количество итераций.

Для описанной исходной задачи по количеству параметров оптимизации $k > 2$ в общем случае не существует не только классического алгоритма решения, но и точного решения в целом. Поэтому для задачи поиска кратчайших путей со многими параметрами оптимизации целесообразно формировать начальную популяции случайным образом. Однако применение классического алгоритма поиска кратчайших путей по одному из критериев позволит значительно сократить время работы алгоритма и улучшить параметры сходимости.

Возвратимся к формулировке первого подхода задачи. Для обозначения веса каждого критерия оптимизации введем коэффициенты δ_i, $i = 1, \ldots, k$. Для формирования начального поколения хромосом применяется алгоритм Дейкстры по одному из критериев оптимизации задачи, весовой коэффициент которого наибольший.

Осуществим выбор родительской пары хромосом. Существуют несколько подходов к выбору родительской пары хромосом. Наиболее простой из них — панмиксия. Этот подход представляет собой случайный выбор родительской пары. В этом случае особь может стать членом нескольких пар. Несмотря на простоту, такой подход считают универсальным для решения различных классов задач. Однако он достаточно критичен по отношению к численности популяции, поскольку эффективность алгоритма, реализующего такой подход, снижается с ростом численности популяции. При селективном способе отбора особей в родительскую пару следует учитывать, что «родителями» могут стать только те особи, значение приспособленности которых не меньше среднего значения приспособленности в популяции. Такой подход обеспечивает более быструю сходимость алгоритма. Однако ввиду быстрой сходимости селективный отбор родительской пары считается неприемлемым, если задача состоит в определении нескольких экстремумов, поскольку для таких задач алгоритм обычно быстро сходится к одному из решений. Кроме того, для некоторого класса задач со сложным ландшафтом приспособленности быстрая сходимость может привести к преждевременной сходимости к квазиоптимальному решению. Этот недостаток в предложенной модели скомпенсирован благодаря использованию соответствующего механизма отбора — турнирного.

Методика применения генетических операций и их особенности.

Операция кроссовер. Генетическая операция кроссовера формирует новое поколение решений задачи с использованием генов предыдущего поколения. В задаче о кратчайшем пути кроссовер представляет собой обмен отдельными путями избранных хромосом для формирования нового пути. Применение классической операции кроссовера невозможно, поскольку длина хромосомы изменяется, а каждый ген не соответствует определенной характеристике решения задачи.

Алгоритм кроссовера для описанного выше представления требует наличия хотя бы одного общего гена (вершины) в хромосомах, которые скрещиваются. Общие гены являются точками кроссовера, а отдельные пути, соединяющие их между собой, подлежат обмену между хромосомами-родителями. Применение операции кроссовера не требует, чтобы общие гены хромосом находились в одинаковых локусах хромосомы. В таком случае операция кроссовера не зависит от положения общих вершин в пути.

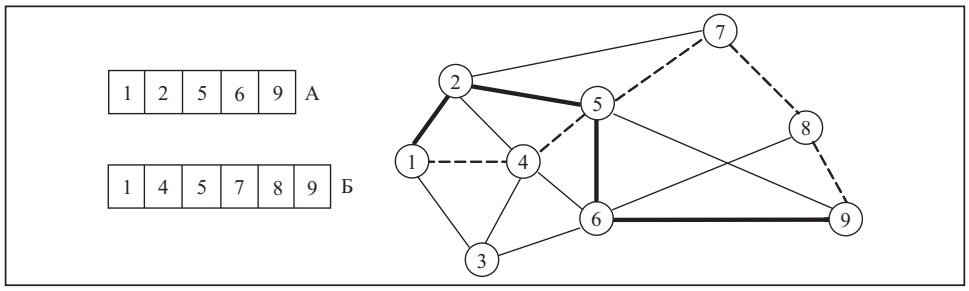

Рис. 11. Хромосомы А, Б и соответствующие пути на графе

Возвратимся к графу, изображенному на рис. 8. В качестве начальной и конечной вершин выберем вершины под номерами 1 и 9 соответственно. При этом случайным образом сформируем первоначальное поколение решений задачи. Для операции скрещивания (кроссовера) выберем два произвольных пути из множества решений. Рассмотрим хромосомы, которые подлежат операции кроссовера. Хромосома А соответствует пути (1, 2, 5, 6, 9) (рис. 11, жирная линия). Хромосома Б соответствует пути (1, 4, 5, 7, 8, 9) (рис. 11, штриховая линия). Как видно из рисунка, хромосомы А и Б имеют общий ген, отвечающий вершине под номером 5. Ген, отвечающий общей вершине, является точкой кроссовера.

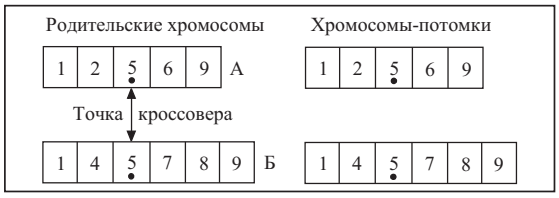

Рис. 12. Одноточечный кроссовер хромосом А и Б

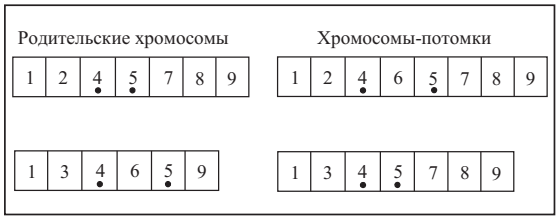

Рис. 13. Двухточечный кроссовер хромосом А и Б

Процедура обмена частичными путями и формирования следующего поколения решений изображена на рис. 12. В зависимости от количества совместных вершин в родительских хромосомах можно выделить одноточечный и многоточечный операторы кроссовера. Пример многоточечного кроссовера изображен на рис. 13. Пути (1, 2, 4, 5, 7, 8, 9) и (1, 3, 4, 6, 5, 9) имеют две общие вершины: 4 и 5 (кроме начальной и конечной вершин пути). Эти вершины являются точками кроссовера соответствующих хромосом.

В результате операции кроссовера возможно формирование путей, содержащих циклы. Поскольку такие пути не удовлетворяют условию (4) задачи, то их необходимо исключить из множества решений. Поэтому после операции кроссовера все хромосомы-потомки подлежат проверке на наличие циклов в соответствующих им путях на графе. Хромосомы, не прошедшие проверку, исключаются и не участвуют в операции отбора.

Операция мутации. Основная задача операции мутации в генетическом алгоритме — случайное изменение решений задачи в целях расширения множества поиска целевой функции. В классическом исполнении операция мутации представляет случайное изменение одного или нескольких генов. Ввиду особенности кодирования в задаче о кратчайшем пути классический механизм операции мутации неприменим.

Опишем алгоритм одноточечной мутации на примере графа, рассмотренного выше. Выберем произвольный путь $p = (1, 2, 4, 6, 8, 9)$ на графе, который является

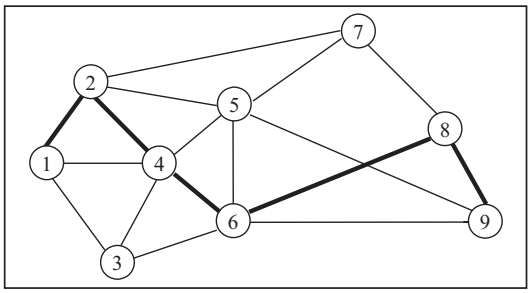

Рис. 14. Путь на графе, соответствующий хромосоме B

решением задачи на определенном этапе работы генетического алгоритма (рис. 14). Этот путь соответствует хромосоме B определенного поколения генетического алгоритма.

Для формирования операции мутации произвольным образом выбираются две вершины: v_i и v_j пути p. Пусть $P_{(i,j)}$ — множество всех возможных путей между вершинами v_i и v_j. Выберем произвольный путь $p_{i.j} \in P_{(i,j)}$, которым заменим промежуточный путь между вершинами v_i и v_j на пути p. Сформированное таким образом альтернативное решение удовлетворяет условиям задачи и может быть включено в следующее поколение генетического алгоритма. С точки зрения представления операция мутации соответствует замене последовательности генов в хромосоме между положениями v_i и v_j альтернативной последовательностью, соответствующей пути $p_{i,j} \in P(i, j)$. Выберем в хромосоме B произвольным образом точки мутации — вершины 4 и 8. Одним из альтернативных путей между этими вершинами будет путь (4, 5, 7, 8, 9). Применив его для операции мутации, получим путь (1, 2, 4, 5, 7, 8, 9). Общий вид операции мутации для хромосомы B изображен на рис. 15.

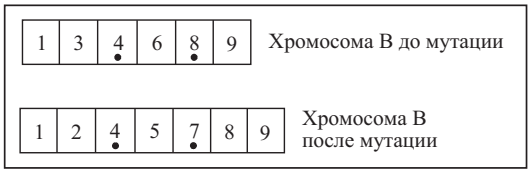

Рис. 15. Операции мутации хромосомы B

Как и в случае операции кроссовера, результат операции мутации может не удовлетворять условию (5) задачи. По аналогии с предыдущим случаем вводится операция проверки результата, а решения, содержащие циклы, исключаются.

Операция отбора. Операция отбора (воспроизведения) направлена на улучшение качества популяции генетического алгоритма. Такая возможность достигается за счет того, что наиболее приспособленные особи имеют больше шансов перехода к следующему поколению. Предложенный генетический алгоритм использует метод турнирного отбора.

Пусть N — количество особей популяции. Характеристика каждой i-й особи популяции (фенотип) выражается значением функции приспособленности $F(i)$ к условиям данной ситуации. Для отбора определенного индивида создается группа из $M > 2$ случайно выбранных особей. Индивид с наибольшей приспособленностью в группе отбирается, остальные — отвергаются. Преимущества такого подхода заключается в отсутствии:

— сходимости к локальному экстремуму;

— стагнации;

— глобального упорядочения всех особей поколения;

— явного расчета функции приспособленности.

Отклоненние решения ГА. Определим отклонение определенного решения задачи от точного решения как взвешенную L_h-норму:

$$r(\overline{C}; h, \overline{\delta}) = || \overline{C} - \overline{C}^* || = \left[\sum_{j=1}^{k} \delta_j^h | C_j - C_j^* |^h \right]^{1/h}, \qquad (6)$$

где $\overline{\delta} = (\delta_1, \ldots, \delta_k)$, $\overline{C} = (C_1, \ldots, C_k)$, $\overline{C}^* = (C_1^*, \ldots, C_k^*)$ — точное решение задачи. Параметр h определяет характер влияния отклонений по каждому отдельному критерию. При $h = 1$ на общее отклонение в основном влияет сумма отклонений решения по всем критериям, при стремлении значения параметра h к бесконечности ($h \to \infty$) повышается степень отклонения по каждому критерию отдельно. Такой подход позволяет гибко менять не только влияние каждого отдельного критерия, но и их корреляцию.

Для небольшого количества параметров оптимизации задачи точное решение найти можно, а в случае $k > 2$ целесообразно использовать лучшее промежуточное решение $\overline{C}' = (C_1', \ldots, C_k')$, т.е. точное решение, соответствующее текущей популяции генетического алгоритма, а не задачи в целом. Таким образом, поиск экстремума оптимизации происходит на отдельном множестве решений. В процессе развития популяции генетического алгоритма отдельное лучшее решение будет приближаться к точному, если таковое вообще существует.

Пусть P — множество особей текущей популяции. Значение лучшего промежуточного решения определяется следующим образом:

$$C_i' = \min \{C_i(p) \mid p \in P\}, \ \ i = 1, \ldots, k. \tag{7}$$

Подставив выражение (7) в формулу (6), получим значение отклонения особи p:

$$r(p) = \left[\sum_{j=1}^{k} \delta_j^h | C_j(p) - C_j |^h \right]^{1/h}.$$

Поскольку для лучших особей значение отклонения будет минимальным, то следует заменить значение отклонения на значение функции приспособленности. Обозначим $r_{\min}$ наименьшее и $r_{\max}$ наибольшее отклонение в текущей популяции. Определим значение функции приспособленности особи p:

$$F(p) = \frac{r_{\max} - r(p)}{r_{\max} - r_{\min}}.$$

Поскольку для операции отбора используется турнирный метод, то нет необходимости рассчитывать функции приспособленности в целом, что значительно упрощает непосредственно процедуру отбора. Для сравнения особей в группе достаточно сравнить значение отклонения каждой из них и оставить в группе одну хромосому с минимальным отклонением.

Отметим, что сформированные операции мутации и кроссовера генетического алгоритма целесообразно использовать в задаче поиска альтернативных путей между определенными парами вершин на графе.

4.2. Моделирование каналов связи сетью Петри. Предположим, что каналы связи распределены по блокам. Вначале рассмотрим блок с односторонним действием между каналами связи и представим его в виде графа (рис. 16, *а*). Вершинам данного графа поставим в соответствие места сети Петри (СП), а каждому ребру графа поставим в соответствие переходы t_i сети (рис. 16, *б*).

Пусть ИТ-инфраструктура состоит из блоков, каждый из которых включает n каналов, а каждый канал блока имеет пропускную способность k. Это означает, что каждый канал блока может обслуживать не более k заданий. Если блок имеет n каналов, то его пропускная способность ограничена величиной $n \cdot k$.

Рассмотрим случай, когда $m \le n \cdot k$. Пусть m — число поступивших от пользователей запросов, которое не превосходит пропускную способность блока, т.е. $m \le n \cdot k$. В этом случае перераспределение нагрузки можно выполнить внутри блока. Например, если блок имеет $n = 6$ каналов с пропускной способностью каждого канала $k = 3$, то пропускная способность такого блока равна

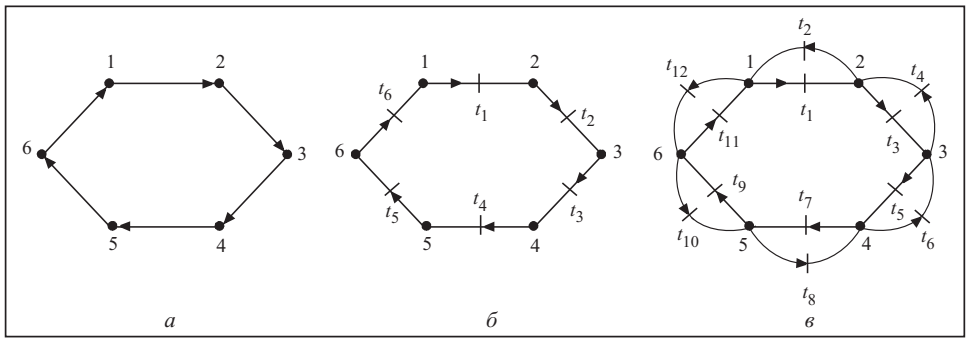

Рис. 16. Граф блока и его сеть Петри: орграф одного блока каналов связи (*а*); сеть односторонняя (*б*); сеть двусторонняя (*в*)

$n \cdot k = 3 \cdot 6 = 18$. Если каналы блока нагружены так, что имеют 5, 2, 5, 1, 0, 1 заданий (запросов) к первому, второму, ... , шестому узлам соответственно, то перераспределить нагрузки внутри блока можно следующим образом: 3, 3, 3, 3, 1, 1, поскольку $5+2+5+1+1+1=14<18$.

Необходимо определить, через какие каналы в блоке кратчайшим путем можно перераспределить нагрузку. Для этого воспользуемся СП, которая показана на рис. 16, *б*. Она представляет собой ординарную и ограниченную машину состояний, при этом все разметки в ней достижимы из начальной [5].

Для определения путей перераспределения нагрузки воспользуемся уравнением состояния, которое для первой СП имеет вид $A \cdot x$ (частный вид сети Петри не ограничивает общности рассмотрения):

$$A \cdot x = \left[\begin{array}{c|cccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \\ \hline 1 & -1 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & -1 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & -1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 1 & -1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 1 & -1 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & -1 \end{array}\right] \cdot x = d. \qquad (8)$$

Здесь $d = M - M_0$, M_0 — начальная разметка СП, а M — конечная разметка.

Подставляя в (8) значение для M_0, M и решая это уравнение, получаем последовательности срабатывания переходов, которые и составляют пути перераспределения нагрузки. Например, $n=6$, $k=3$, $M_0 = (5, 5, 2, 0, 1, 1)$, $M = (3, 3, 3, 3, 1, 1)$, то

$$A \cdot x = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \cdot x = \begin{bmatrix} -2 \\ -2 \\ 1 \\ 3 \\ 0 \\ 0 \end{bmatrix}.$$

Единственным решением этого уравнения является $x = (2, 4, 3, 0, 0, 0)$. Это решение показывает, что кратчайший путь перераспределения нагрузки состоит в срабатывании дважды перехода t_1, четыре раза перехода t_2 и три раза перехода t_3. Действительно,

$$M_0 = (5, 5, 2, 0, 1, 1) \to 2t_1 \to (3, 7, 2, 0, 1, 1) \to 4t_2 \to (3, 3, 6, 0, 1, 1) \to$$
$$\to 3t_3 \to (3, 3, 3, 3, 1, 1).$$

Это означает, что при последовательном соединении каналов перенаправляются из первого канала два задания во второй канал, а из второго канала четыре задания — в третий канал, из третьего канала три задания — в четвертый канал.

В дальнейшем будем предполагать, что распределение нагрузки по каналам (в случае, если некоторые из них имеют нагрузку большую, чем их пропускная способность) выполняется таким образом, что первые каналы получают максимальную нагрузку. В данном примере каналы 1, 2, 3, 4 получили максимальную нагрузку.

Таким образом, приходим к алгоритму перераспределения нагрузки для блока с n каналами и с пропускной способность k при начальной разметке M_0.

Алгоритм РАВНОМ-НАГР(k, n, M_0)

Вход: Блок из n каналов, пропускная способность узла k и начальная разметка M_0.

Выход: Множество минимальных решений уравнения состояния.

Метод:

1. Вычислить M исходя из M_0.
2. Построить уравнение состояния $A \cdot x = M - M_0$.
3. Решить уравнение состояния TSS-методом [6].
4. Выбрать среди решений оптимальное.
5. Перераспределить нагрузку в соответствии с оптимальным решением.
6. Стоп.

Если каналы в блоках могут работать в обоих направлениях (двусторонние) между соседними блоками, то ситуация анализируется тем же способом, что и рассмотренная выше.

Представим модель блока с двусторонним действием между каналами в блоке (рис. 16, *в*). Тогда сеть Петри остается машиной состояний, но при этом число неизвестных в уравнении состояния увеличивается в два раза. Однако алгоритм перераспределения нагрузки не изменяется. Например, если $n = 6$, $k = 3$, $M_0 = (5, 2, 5, 1, 5, 0)$, то отсюда $5+2+5+1+5 = 18$; тогда $M = (3, 3, 3, 3, 3, 3)$, так как $18 : 3 = 6$ для СП согласно рис. 16, *в*. Уравнение состояния для этой СП имеет вид

$$A \cdot x = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{bmatrix} \cdot x = \begin{bmatrix} -2 \\ 1 \\ -2 \\ 2 \\ -2 \\ 3 \end{bmatrix}.$$

Множество минимальных решений представляет

$$x_1 = (0, 1, 2, 0, 0, 0, 0, 2, 0, 0, 0, 3), \quad x_2 = (1, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 1),$$

$$x_3 = (0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 2), \quad x_4 = (2, 0, 0, 1, 0, 3, 1, 0, 3, 0, 0, 0).$$

Из них оптимальными являются решения x_2 и x_3 (сумма их координат равна шести, при этом суммы для x_1 и x_4 равны 8 и 10 соответственно). Любое из этих решений можно выбирать для задачи перераспределения нагрузки.

Сложность алгоритма и его правильность. При решении уравнения состояния в данном алгоритме используется TSS-алгоритм для вычисления множества T-инвариантов [6]. Этот алгоритм имеет экспоненциальную оценку временной сложности, однако специфика данного класса СП состоит в том, что матрица инцидентности СП является разреженной. В этом случае TSS-алгоритм работает полиномиальное время.

Для обоснования алгоритма перераспределения нагрузки заметим, что TSS-алгоритм обоснован, и поскольку он является основным при поиске маршрута перерас-

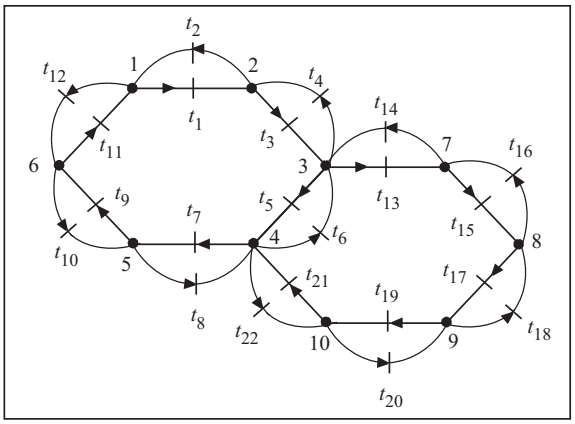

Рис. 17. Сеть Петри для двух блоков

пределения нагрузки, то этим подтверждается и правильность алгоритма РАВНОМ-НАГР. Кроме того, данный алгоритм находит минимальные решения, поскольку любое другое решение является неотрицательной линейной комбинацией этих решений.

Отметим, что вычисление разметки M исходя из $M_0 = (m_1, m_2, \dots, m_n)$ и k выполняется следующим образом. Находим разложение $S = \sum_{i=1}^{n} m_i = k \cdot b + b_1$ путем деления S на k; тогда M будет иметь b компонент, которые равны k, а остальные компоненты могут принимать произвольные значения, меньшие k.

Рассмотрим случай, когда $m > n \cdot k$. Приведенный выше алгоритм работает при ограничении, когда количество заданий в каналах блока не превышают его пропускной способности. Если это условие не выполняется, то перераспределение нагрузки осуществляется с учетом загруженности соседних блоков. Рассмотрим пример. Пусть имеется два соединенных между собой блока (рис. 17). Если один из них недогружен, а второй перегружен, то при $m \le (2n-2) \cdot k$ решение задачи перераспределения нагрузки выполняется алгоритмом РАВНОМ-НАГР. Например, если $n = 6$, $k = 3$, $M_0 = (7, 5, 4, 2, 1, 2, 0, 2, 1, 1)$, то следовательно, первый блок перегружен и необходимо перераспределение нагрузки за счет соседнего блока, который в данном случае недогружен. Принимаем $M = (3, 3, 3, 3, 3, 3, 3, 3, 0, 1)$ и получаем уравнение состояния для данной СП:

$$A \cdot x = \begin{bmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1
\end{bmatrix} \cdot x = \begin{bmatrix} -4 \\ -2 \\ -1 \\ 1 \\ 2 \\ 1 \\ 3 \\ 1 \\ -1 \\ 0 \end{bmatrix}.$$

Имеем множество решений данной системы:

$x_1 = (0\,0\,2\,0\,0\,0\,0\,1\,0\,3\,0\,4\,3\,0\,0\,0\,0\,1\,0\,0\,0\,0)$, $x_2 = (3\,0\,5\,0\,3\,0\,2\,0\,0\,0\,0\,1\,3\,0\,0\,0\,0\,1\,0\,0\,0\,0)$,

$x_3 = (0\,2\,0\,0\,0\,2\,0\,3\,0\,5\,0\,6\,3\,0\,0\,0\,0\,1\,0\,0\,0\,0)$, $x_4 = (1\,0\,3\,0\,1\,0\,0\,0\,0\,2\,0\,3\,3\,0\,0\,0\,0\,1\,0\,0\,0\,0)$,

$x_5 = (4\,0\,6\,0\,4\,0\,3\,0\,1\,0\,0\,0\,3\,0\,0\,0\,0\,1\,0\,0\,0\,0)$, $x_6 = (3\,0\,5\,0\,6\,0\,2\,0\,0\,0\,0\,1\,0\,0\,0\,3\,0\,4\,0\,3\,0\,3)$,

$x_7 = (0\,2\,0\,0\,1\,0\,0\,3\,0\,5\,0\,6\,0\,0\,0\,3\,0\,4\,0\,3\,0\,3)$, $x_8 = (0\,2\,0\,0\,0\,0\,0\,3\,0\,5\,0\,6\,1\,0\,0\,2\,0\,3\,0\,2\,0\,2)$,

$x_9 = (0\,3\,0\,1\,0\,0\,0\,4\,0\,6\,0\,7\,0\,0\,0\,3\,0\,4\,0\,3\,0\,3)$, $x_{10} = (0\,0\,2\,0\,3\,0\,0\,1\,0\,3\,0\,4\,0\,0\,0\,3\,0\,4\,0\,3\,0\,3)$,

$x_{11} = (1\,0\,3\,0\,4\,0\,0\,0\,0\,2\,0\,3\,0\,0\,0\,3\,0\,4\,0\,3\,0\,3)$, $x_{12} = (4\,0\,6\,0\,7\,0\,3\,0\,1\,0\,0\,0\,0\,0\,0\,3\,0\,4\,0\,3\,0\,3)$,

$x_{13} = (1\,0\,3\,0\,0\,0\,0\,0\,0\,2\,0\,3\,4\,0\,1\,0\,0\,0\,1\,0\,1\,0)$, $x_{14} = (0\,0\,2\,0\,0\,1\,0\,1\,0\,3\,0\,4\,4\,0\,1\,0\,0\,0\,1\,0\,1\,0)$,

$x_{15} = (0\,2\,0\,0\,0\,3\,0\,3\,0\,5\,0\,6\,4\,0\,1\,0\,0\,0\,1\,0\,1\,0)$, $x_{16} = (4\,0\,6\,0\,0\,0\,3\,0\,1\,0\,0\,0\,7\,0\,4\,0\,3\,0\,4\,0\,4\,0)$,

$x_{17} = (4\,0\,6\,0\,3\,0\,3\,0\,1\,0\,0\,0\,4\,0\,1\,0\,0\,0\,1\,0\,1\,0)$, $x_{18} = (3\,0\,5\,0\,0\,0\,2\,0\,0\,0\,0\,1\,6\,0\,3\,0\,2\,0\,3\,0\,3\,0)$,

$$x_{19} = (3\,0\,5\,0\,2\,0\,2\,0\,0\,0\,0\,1\,4\,0\,1\,0\,0\,0\,1\,0\,1\,0).$$

Из этих решений оптимальными будут, например, первое и четвертое. Выбирая первое решение в качестве оптимального и анализируя его, получаем оптимальный маршрут. В этом случае перенаправляются:

— два задания из канала 2 в канал 3 (двукратное срабатывание перехода t_3),
— одно задание из канала 5 в канал 4 (срабатывание перехода t_8),
— три задания из канала 6 в канал 5 (три срабатывания перехода t_{10}),
— четыре задания из канала 1 в канал 6 (четыре срабатывания перехода t_{12}),
— три задания из канала 3 в канал 7 (три срабатывания перехода t_{13}),
— одно задание из канала 9 в канал 8 (срабатывание перехода t_{18}).

Таким образом, получаем маршрут перераспределения нагрузки:

$$2 \xrightarrow{2t_3} 3 \xrightarrow{3t_{13}} 7,\ 1 \xrightarrow{4t_{12}} 6 \xrightarrow{3t_{10}} 5 \xrightarrow{t_8} 4,\ 9 \xrightarrow{t_{18}} 8.$$

Дальнейшее наращивание блоков и каналов в блоках не влияет на алгоритм перераспределения нагрузки. Однако при этом возрастает размер матрицы в уравнении состояния, что существенным образом влияет на сложность его решения. Поэтому если ограничить количество смежных блоков (например, четыре блока, в которых не более чем 20 каналов), то получим приемлемые временные ограничения сложности при вычислении оптимальных маршрутов перераспределения нагрузки в такой ИТ-инфраструктуре.

ЗАКЛЮЧЕНИЕ

Согласно предложенному методу проектирования ИТ-инфрастуктур необходимо выполнить последовательность этапов.

— Формирование СП в целях исследования свойств (как статического, так и динамического характера).

— Исправление ошибок проектирования и моделирования (в том числе ГА) на основе результатов анализа свойств, полученных на первом этапе.

— Выбор конкретной платформы для реализации разработанной ИТ-инфрастуктуры (в зависимости от существующих средств, потенциальных пользователей, условий эксплуатации и т.д.).

— Подход распространяется на модели, в которых явно фигурирует время (временные автоматы и временные СП, гибридные автоматы и гибридные СП и т.д.)

СПИСОК ЛИТЕРАТУРЫ

1. Бойко Ю.В., Волохов В.М., Глибовець М.М., Єршов С.В., Кривий С.Л. Погорілий С.Д., Ролік О.І., Теленик С.Ф., Ясочка М.В. Методи та новітні підходи до проектування, управління і застосування високопродуктивних ІТ-інфраструктур. Київ: Видавн.-поліграф. центр «Київський університет», 2016. 447 с.
2. Кривий С.Л. Дискретна математика. Чернівці; Київ: Букрек, 2017. 567 с.
3. Сергієнко І.В., Кривий С.Л., Провотар О.І. Алгебраїчні аспекти інформаційних технологій. Київ: Наук. думка, 2011. 399 с.
4. Kryvyi S., Hajder M., Dymora P., Mazurek M. Designing of the computer network topologies and coherent graphs algebra. *Annales Universitatis Mariae Curie-Sklodowska, Sectia A1. Informatica*. Lublin, 2006. Vol. 5. P. 379–391.
5. Murata T. Petri nets: Properties, analysis and applications. *Proc. of the IEEE*. 1989. Vol. 77, N 4. P. 541–580.

6. Кривий С.Л. Лінійні діофантові обмеження та їх застосування. Чернівці; Київ: Букрек, 2015. 224 с.
7. Глибовець М.М., Гулаєва Н.М. Еволюційні алгоритми. Київ: Києво-Могилянська академія, 2013. 826 с.
8. Погорілий С.Д., Білоус Р.В., Білоконь І.В. Застосування генетичних алгоритмів у комп'ютерних системах. За ред. Погорілого С.Д. Київ: Видавн.-поліграф. центр «Київський університет», 2014. 320 с
9. Погорілий С.Д., Білоус Р.В. Генетичний алгоритм розв'язання задачі маршрутизації в мережах. *Проблеми програмування*. 2010. № 2–3. С. 171–178.
10. Potebnia A., Pogorilyy S. Innovative GPU accelerated algorithm for fast minimum convex hulls computation. *Proc. of the Federated Conference on Computer Science and Information Systems* (*FedCSIS).* 2015. Vol. 5. P. 555–561.

Надійшла до редакції 30.01.2018

С.Л. Кривий, С.Д. Погорілий, М.М. Глибовець, Ю.В. Бойко, Н.М. Сидорова
ПРОЕКТУВАННЯ IT-ІНФРАСТРУКТУРИ

Анотація. Запропоновано методику проектування IT-інфраструктури в умовах обмежених ресурсів. Методика ґрунтується на властивостях алгебри графів, генетичних алгоритмів, мереж Петрі та особливостях обмежень. Розглянуто два підходи до розв'язання задачі балансування навантажень в IT-інфраструктурі.

Ключові слова: алгебра графів, топологія, IT-інфраструктура, генетичні алгоритми, мережі Петрі.

S.L. Kryvyy, S.D. Pogorilyy, M.M. Glybovets, Y.V. Boyko, N. Sydorova
DEVELOPMENT OF THE IT INFRASTRUCTURE

Abstract. A metodology for development of IT-infrastructure under conditions of constraints resources is proposed. This metodology is based on the properties of graph algebra, genetic algorithms, Petri nets, and specific features of the constraints. Two approaches to solving load-balancing problems in IT infrastructure are considered.

Keywords: graph algebra, topology, IT infrastructure, genetic algorithms, Petri nets.

Крывый Сергей Лукьянович,
доктор физ.-мат. наук, профессор, профессор кафедры Киевского национального университета имени Тараса Шевченко, e-mail: sl.krivoi@gmail.com.

Погорелый Сергей Демьянович,
доктор техн. наук, профессор, профессор кафедры Киевского национального университета имени Тараса Шевченко, e-mail: sdp77@i.ua.

Глибовец Николай Николаевич,
доктор физ.-мат. наук, профессор, профессор кафедры Национального университета «Киево-Могилянская академия», e-mail: glib@ukma.kiev.ua.

Бойко Юрий Владимирович,
кандидат техн. наук, доцент, доцент кафедры Киевского национального университета имени Тараса Шевченко, e-mail: Boyko@univ.kiev.ua.

Сидорова Ника Николаевна,
аспирантка Национального авиационного университета, Киев, e-mail: nika.sidorova@gmai.com.