ИНФОРМАТИКА

2006 **июль-сентябрь** **№ 3**

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

УДК 519.8

С.В. Балтак[1], Ю.Н. Сотсков[2]

ПОСТРОЕНИЕ РАСПИСАНИЙ УЧЕБНЫХ ЗАНЯТИЙ НА ОСНОВЕ РАСКРАСКИ ВЕРШИН ГРАФА

Приводится описание алгоритмов и программ для построения расписаний учебных занятий. Процесс составления расписания представляется как раскраска вершин графа с дополнительными ограничениями на множества цветов. Для оптимизации раскраски графа разрабатываются эвристические алгоритмы, которые позволяют строить практически приемлемые расписания учебных занятий в школах, колледжах или высших учебных заведениях. Система апробирована на реальной информации.

Введение

До настоящего времени расписания в большинстве базовых и средних школ, колледжей и высших учебных заведений Беларуси составлялись вручную. При этом построение расписания требует значительных трудозатрат и времени, а качество полученного расписания существенно зависит от опыта его составителя и часто не удовлетворяет как преподавателей, так и учащихся. При составлении учебных расписаний вручную сложно обеспечивать оперативное внесение необходимых изменений в действующее расписание в связи с изменением тех или иных условий в процессе его реализации. Например, в конце учебного полугодия возникает необходимость проведения дополнительных занятий по некоторым предметам и, наоборот, программы полугодия по другим предметам могут быть исчерпаны. В случае болезни преподавателя или при проведении внеклассных мероприятий в школе также возникает необходимость оперативной корректировки действующего расписания учебных занятий. С математической точки зрения задача построения оптимального расписания учебных занятий является достаточно сложной, поскольку она принадлежит классу так называемых NP-трудных задач [1]. Для решения задачи построения учебных расписаний разработано множество алгоритмов (как правило, эвристических), учитывающих те или иные особенности постановки задачи [2–4]. Существуют также коммерческие пакеты программ составления учебных расписаний – как отечественные [5], так и зарубежные [6, 7]. Эти программы позволяют составлять расписания занятий в интерактивном режиме. При этом расписание составляется, как правило, путем прямых назначений уроков (лекций) на возможные интервалы времени с последующим разрешением возникающих конфликтов. В данной статье показано, как раскраска вершин графа может применяться для построения расписания учебных занятий. Используемая терминология по теории расписаний соответствует монографии [8], а по теории графов – монографии [9]. Вводимые понятия выделяются курсивом. Для определенности в разд. 1–4 речь идет в основном о задаче составления расписаний уроков в общеобразовательной школе. После соответствующей настройки разработанная система может быть использована и для составления расписаний учебных занятий в училищах, колледжах, институтах и университетах.

1. Постановка задачи построения расписания учебных занятий

При составлении расписания занятий (уроков) используются следующие исходные данные: множество классов (*групп*) учеников, множество преподавателей и множество классов (*аудиторий*), предназначенных для проведения занятий. Множество аудиторий может быть разбито на непересекающиеся подмножества, каждое из которых состоит из аудиторий определенного *типа*, предназначенных для проведения соответствующего множества занятий (уроков). На-

пример, в школе могут быть оборудованы специализированные классы для проведения уроков информатики, физики, химии или математики. Для проведения уроков физкультуры используется спортивный зал. Уроки труда проводятся, как правило, в специализированных классах или мастерских. В дальнейшем вместо понятия «класс учеников» будем использовать понятие «группа учеников», поскольку при проведении некоторых занятий класс учеников делится на группы (например, при проведении занятий по иностранному языку или по труду). В состав исходных данных включается также множество занятий, которые следует провести в течение планового периода (рабочей недели, учебной четверти, полугодия). Предполагается, что преподавание в школе проводится по фиксированным учебным программам, причем заранее известно, какое занятие у какой группы учеников должно быть проведено и какой преподаватель должен провести данное занятие. Поскольку за классами учеников, как правило, закрепляются аудитории, то заранее известно, либо в какой аудитории должно быть проведено занятие, либо какой тип аудиторий требуется для проведения того или иного занятия.

В процессе решения задачи построения школьного расписания необходимо назначить каждое занятие на конкретный промежуток времени рабочего дня недели. Поскольку длительность урока фиксирована и равняется 45 мин., то заранее известны все возможные интервалы времени, на которые могут быть назначены уроки. Обозначим через W множество всех таких 45-минутных *интервалов* времени в рассматриваемом при составлении расписания *плановом периоде* (например, в течение шестидневной рабочей недели). Пронумеруем интервалы множества W начиная с первого по времени интервала и будем считать элементы множества W *упорядоченными* в соответствии с номерами его интервалов: $N = \{1, 2, \ldots, n\}$. Здесь n равно мощности $|W|$ множества W. Если для проведения какого-либо урока аудитория не была назначена заранее (до составления расписания), то в процессе составления расписания необходимо также назначить аудиторию для проведения такого урока. Расписания занятий в школе являются *периодическими* (по крайней мере, в большей части учебного полугодия), поэтому достаточно составить расписание учебных занятий на один плановый период. Обычно периодом является одна неделя или же две последовательные недели (если расписания для них существенно отличаются одно от другого). Нарушения периодичности, возникающие в процессе реализации школьного расписания, можно учитывать в *режиме корректировки* основного расписания, составленного на один плановый период. При построении расписания учебных занятий в школе необходимо выполнить следующие условия 1–4, а также предусмотреть возможности 5–8 и обеспечить требование 9.

1. Если два занятия проводятся с одной и той же группой учеников, то они должны быть назначены на различные интервалы времени.

2. Если два занятия проводятся одним преподавателем, то они должны быть назначены на различные интервалы времени.

3. Если два занятия проводятся в одной и той же аудитории, то они должны быть назначены на различные интервалы времени.

4. Пусть V_k обозначает множество всех занятий, которые должны быть проведены в аудиториях типа k. Тогда мощность подмножества занятий множества V_k, которые могут быть назначены на один и тот же интервал времени из множества W, не должна превышать общего числа имеющихся в школе аудиторий типа k.

5. Необходимо предусмотреть возможность запрещать назначение уроков для конкретной группы учеников на тот или иной интервал времени из множества W (например, в связи с пятидневкой для учеников младших классов).

6. Необходимо предусмотреть возможность запрещать назначение уроков для преподавателя на тот или иной интервал времени (например, в связи с методическими днями или в связи с личными пожеланиями преподавателя).

7. Необходимо обеспечить возможность проводить некоторые занятия непосредственно одно за другим (например, лабораторные занятия, требующие два последовательных урока) с помощью *правила первого типа*, согласно которому необходимо определить *основное занятие*, непосредственно назначаемое на интервал времени из множества W, и *зависимое занятие*, сле-

дующее непосредственно за основным занятием. В постановке задачи может допускаться несколько зависимых занятий, которые должны следовать одно за другим.

8. Необходимо обеспечить возможность проводить какие-либо занятия в разные дни (например, три занятия по математике не должны проводиться в один день с одной группой учеников). Такая необходимость обеспечивается с помощью *правила второго типа*, когда задается подмножество занятий, никакие два из которых не должны проводиться в один и тот же день.

9. Для учеников в построенном расписании не должно быть так называемых *форточек*.

Необходимость соблюдения условий 1–4 достаточно очевидна, поскольку нарушение хотя бы одного из них делает построенное расписание практически *недопустимым*. Возможности 5–8 и требование 9 хотя и не связаны с допустимостью расписания, однако без их учета *качество* построенного расписания существенно снижается. Поэтому при построении приемлемого расписания (близкого к оптимальному) условия 5–9 или часть из них целесообразно включить в целевую функцию с соответствующими коэффициентами, определяющими их относительную важность. Для представления исходных данных задачи построения оптимального школьного расписания, а также для представления процесса ее решения предлагается использовать сетевую модель. При этом процесс построения расписания сводится к раскраске вершин V графа $G = (V, E)$ с множеством ребер E и *дополнительными ограничениями* на множества цветов, которые разрешается использовать для различных вершин из множества V.

2. Сведение задачи построения расписания к раскраске вершин графа

Выбор теоретико-графовой модели определялся тем, что процесс обучения в общеобразовательной школе можно считать детерминированным, поскольку для каждого класса учеников заранее определен список предметов (курсов), и этот список не изменяется в образовательном процессе в течение длительного периода времени. Преподавательский коллектив и классы учеников, как правило, остаются неизменными во время всего планового периода. В силу такой детерминированности задача построения школьного расписания может быть сформулирована в терминах раскраски вершин графа. Функцию φ называют *раскраской* (вершин) графа $G = (V, E)$, если для каждой вершины $v_i \in V$ она определяет натуральное число (называемое *цветом*) $\varphi(v_i) \in N$ так, что из включения $[v_i, v_j] \in E$ следует соотношение $\varphi(v_i) \neq \varphi(v_j)$. Здесь N обозначает подмножество множества натуральных чисел. В задаче оптимальной раскраски графа требуется построить раскраску $\varphi : V \to N$ вершин графа $G = (V, E)$, которая содержит минимальное число цветов. Покажем, каким образом можно представить задачу построения школьного расписания в терминах раскраски графа $G = (V, E)$, который строится согласно приведенным далее правилам.

Предполагается, что каждая *нефиктивная* вершина графа $G = (V, E)$ соответствует определенному занятию, которое проводится конкретным преподавателем с конкретной группой учеников в конкретной аудитории (если аудитория задана заранее) или в аудитории конкретного типа (если аудитория заранее не задана). Раскраска $\varphi : V \to N$ интерпретируется следующим образом: если нефиктивная вершина $v_i \in V$ окрашена в цвет $\varphi(v_i) \in N = \{1, 2, \ldots, n\}$, то занятие, определенное вершиной $v_i \in V$, должно быть проведено в $\varphi(v_i)$-й по порядку следования интервал времени из упорядоченного множества W интервалов, предназначенных для проведения уроков в плановом периоде. Граф $G = (V, E)$ может содержать *фиктивные* вершины, включение которых во множество V определяется правилами представления условий 5 и 6 в терминах раскраски графа.

Ребра множества E графа $G = (V, E)$ используются для определения условий и ограничений 1–9. В частности, ребро $[v_i, v_j] \in E$, инцидентное нефиктивным вершинам $v_i \in V$ и $v_j \in V$, может обозначать, что занятия, соответствующие данным вершинам, проводятся у одной и той же группы учеников (условие 1 из разд. 1). Поскольку в раскраске φ смежные вершины графа G должны быть окрашены в различные цвета, то занятия с одной и той же группой учеников должны быть проведены в различные интервалы времени из множества W. Точно так же следует ввести в граф G ребра $[v_i, v_j] \in E$ в связи с условием 2 для преподавателей и в связи с условием 3 для аудиторий. Если в процессе построения графа возникают кратные ребра, то их следует

удалить, оставив в графе G по одному ребру для каждого множества кратных ребер, инцидентных одной и той же паре вершин. Вершины, представляющие занятия одного и того же преподавателя (или одной и той же группы учеников, или проводимые в одной и той же аудитории), должны быть попарно соединены между собой ребрами. В результате множество таких вершин определяет полный подграф графа G. Следовательно, в раскраске φ графа G все вершины полного подграфа будут окрашены в различные цвета. Таким образом, раскраска $\varphi : V \to N$ построенного графа $G = (V, E)$ определяет расписание занятий, для которого выполнены условия 1, 2 и 3 из разд. 1.

Следующее ограничение, которое необходимо учитывать при составлении школьного расписания, связано с количеством имеющихся в школе аудиторий каждого типа (условие 4). Это ограничение учитывается путем введения дополнительного условия на использование одного и того же цвета для ограниченного числа вершин из заданного подмножества множества V. Конкретный цвет вершин означает, что занятия, представленные вершинами одного цвета, проводятся в один и тот же интервал времени из множества W. Следовательно, учет при построении раскраски $\varphi : V \to N$ заданной границы на число вершин одного цвета позволяет ограничить количество аудиторий соответствующего типа, которые могут быть использованы в одном и том же интервале времени из множества W. Такая интерпретация условия 4 предполагает применение нетрадиционного понятия раскраски вершин графа. Для соответствующего *обобщения* раскраски (оптимальной раскраски) будем использовать термин *допустимая* раскраска $\varphi : V \to N$ (оптимальная *допустимая* раскраска).

При составлении школьного расписания следует учитывать необходимость или пожелания преподавателей или группы учеников не назначать для них занятия на какие-то конкретные промежутки времени из множества W (или не проводить занятия в каких-то аудиториях в конкретные промежутки времени). Для задания такого требования или пожелания требуется сопоставить преподавателю (группе учеников или аудиториям определенного типа) конкретные цвета (номера интервалов времени из множества W), запрещенные для них по тем или иным причинам (условия 5 и 6). В терминах сетевой модели такие требования могут быть реализованы следующим образом.

При построении допустимой раскраски необходимо раскрасить вершины графа $G = (V, E)$ так, чтобы определенные вершины графа G не получили запрещенные для них цвета. Для представления такого условия в граф G вводится соответствующее множество *фиктивных* вершин (по одной фиктивной вершине для каждого интервала времени из множества W, который может оказаться *запрещенным*). Например, если составляется расписание на шесть рабочих дней недели по четыре занятия каждый день, то требуется рассмотреть 24 интервала времени для проведения занятий: $|W| = n = 24$. Если каждый из этих интервалов может быть запрещенным (для преподавателя, для группы учеников или для какого-то типа аудиторий), то необходимо добавить 24 фиктивные вершины в построенный граф, заранее окрасив их последовательно в различные цвета из множества $N = \{1, 2, \ldots, 24\}$. Затем следует добавить в полученный граф ребра, инцидентные фиктивным вершинам и тем вершинам графа G, для которых следует запретить соответствующие цвета в допустимой раскраске. Например, фиктивная вершина v_i, заранее окрашенная в цвет $\varphi(v_i) \in N = \{1, 2, \ldots, 24\}$, должна быть соединена ребром с каждой вершиной, определяющей занятие, которое нельзя проводить в $\varphi(v_i)$-м по порядку следования интервале времени из упорядоченного множества W. В процессе последующей раскраски графа G соответствующие нефиктивные вершины не могут окрашиваться в запрещенные для них цвета, и, следовательно, соответствующие занятия не будут проводиться в запрещенные для них интервалы времени из множества W. В отличие от условия 4 приведенная выше интерпретация условий 5 и 6 не требует обобщения традиционного понятия раскраски графа. Для учета условий 5 и 6 достаточно лишь дополнить множество вершин графа соответствующим множеством фиктивных вершин и соединить их ребрами с вершинами, для которых требуется запретить использование соответствующих цветов. Следует заметить, однако, что построение допустимой раскраски графа G следует начинать при условии, что части его вершин (а именно всем фиктивным вершинам) заранее приписаны конкретные цвета, причем эти

цвета не могут быть изменены в дальнейшем, даже если это исключает возможность получения оптимальной раскраски графа G.

Условие 7 учитывается в процессе построения допустимой раскраски $\varphi : V \to N$ следующим образом. При окрашивании подмножества вершин, связанных условием 7, цвет необходимо выбирать только для основной вершины из этого подмножества. При этом неосновным вершинам подмножества цвета назначаются исходя из их расположения относительно окрашенной основной вершины. Аналогично условиям 4 и 5 такая интерпретация условия 7 также приводит к нетрадиционному понятию раскраски графа, состоящему в *принудительном* окрашивании неосновных вершин графа G согласно цветам, которые назначены его основным вершинам. Для учета этого обобщения понятия раскраски будем использовать тот же термин «допустимая раскраска» графа G.

Условия 8 и 9 отражаются непосредственно в целевой функции, которая используется при построении оптимальной допустимой раскраски графа G, и учитываются как тенденция к запрещению цветов, нарушающих эти условия уже в самом алгоритме построения допустимой раскраски. Таким образом, получено следующее соответствие между допустимыми раскрасками $\varphi : V \to N$ графа G и расписаниями учебных занятий, удовлетворяющими условиям 1–9 из разд. 1:

$$\{\text{вершина } v_i \in V\} \leftrightarrow \{\text{урок – преподаватель – группа учеников}\};$$

$$\{\text{ребро } [v_i, v_j] \in E\} \leftrightarrow \{\text{преподаватель проводит оба урока, соответствующие } v_i \text{ и } v_j\};$$

$$\{\text{ребро } [v_i, v_j] \in E\} \leftrightarrow \{\text{соответствующие } v_i \text{ и } v_j \text{ уроки проводятся с одной группой учеников}\};$$

$$\{\text{ребро } [v_i, v_j] \in E\} \leftrightarrow \{\text{уроки, соответствующие } v_i \text{ и } v_j, \text{ проводятся в одной аудитории}\};$$

$$\{\text{число вершин множества } V_k \text{ цвета } r\} \leftrightarrow \{\text{число аудиторий типа } k, \text{ используемых в интервале } r\};$$

$$\{\text{ограничения на цвета вершины } v_i\} \leftrightarrow \{\text{ограничения на интервалы для проведения занятия } v_i\};$$

$$\{\text{цвет } \varphi(v_i) \text{ вершины } v_i \in V\} \leftrightarrow \{\text{номер } \varphi(v_i) \text{ интервала из множества } W \text{ для проведения занятия } v_i\}.$$

3. Алгоритмы построения допустимой раскраски графа

Задача раскраски вершин графа является NP-трудной [1], и, следовательно, ее обобщение, описанное в разд. 2, также является NP-трудной задачей. Для решения классической задачи раскраски вершин графа разработано множество точных, приближенных и эвристических алгоритмов [2–4, 9], однако воспользоваться известными алгоритмами раскраски непосредственно для построения школьного расписания не представляется возможным, поскольку они не учитывают дополнительные ограничения на допустимость раскраски, связанные с условиями 4–9. Кроме того, порядок $|V|$ графа $G = (V, E)$, который возникает при моделировании практических задач составления расписаний занятий в общеобразовательной школе, оказывается слишком большим, чтобы можно было строить оптимальные допустимые раскраски за практически приемлемое время на персональных компьютерах, которыми, в основном, оснащены базовые и средние школы Беларуси. Следует отметить также, что при составлении школьного расписания трудно выбрать единственный критерий оптимальности. При построении школьного расписания (на основе описанного в разд. 2 сведения) классический критерий, связанный с минимизацией числа различных цветов, которые используются в раскраске φ графа G, неприемлем, поскольку это число является одним из исходных параметров задачи. Действительно, число допустимых цветов ограничено множеством номеров $N = \{1, 2, \ldots, n\}$ интервалов времени из множества W, на которые могут быть назначены занятия. Отметим также, что ряд практически важных критериев оптимальности школьного расписания трудно представить в формализованном виде. По этим причинам для построения допустимой раскраски φ графа G были разработаны эвристические алгоритмы, которые в общем случае не гарантируют оптимальность полученной раскраски. В качестве критерия оптимальности рассматривалась взвешенная сумма за нарушение ограничений 1–9 из разд. 2.

Далее опишем алгоритм, который реализуется перед началом раскраски вершин графа. На этом этапе определяются вершины, цвета которым назначаются заранее, а также множества допустимых цветов, которые могут быть назначены остальным вершинам. Если явно не огово-

рено, какие вершины просматриваются, то предполагается, что просматриваются нефиктивные вершины графа G.

Алгоритм разбиения множества занятий по интервалам времени из множества W

1. Фиктивные вершины окрашиваются в соответствующие цвета из множества N.

2. Для всех нефиктивных вершин анализируются ребра, инцидентные фиктивным вершинам. Если существует ребро, инцидентное фиктивной и нефиктивной вершинам, то для нефиктивной вершины запрещается цвет из множества N, в который окрашена фиктивная вершина.

3. Определяются нефиктивные вершины, цвета которых можно определить до начала реализации основного алгоритма раскраски. Для каждой нефиктивной вершины просматриваются смежные с ней вершины и запрещаются цвета, в которые они окрашены. Если в результате для какой-либо вершины остается только один допустимый цвет из множества N, то после просмотра всех окрашенных вершин выполняется шаг 4, иначе – шаг 5.

4. Просматриваются все вершины до тех пор, пока удается найти вершину v_i, которую можно окрасить только в один цвет из множества N, но которая еще не окрашена. Такую вершину красим в единственно возможный для нее цвет и просматриваем все смежные с ней вершины, запрещая для них цвет, в который окрашена вершина v_i.

5. Если в результате для какой-либо вершины остается один возможный цвет из множества N, то повторяется шаг 4, иначе находим подмножества (*блоки*) вершин, связанных условием 7 (т. е. подмножества вершин, цвета которых связаны правилом первого типа). Определяется порядок следования зависимых вершин блока, и составляются два вспомогательных массива. В первый массив (usedVertices) помещаются все вершины, которые используются в указанных блоках. Во второй массив (coloredVertices) помещаются вершины, которые определяют цвета вершин внутри блока, т. е. вершины, на основании цвета которых раскрашиваются все вершины блока.

6. В массив (m_IsType2) помещаются вершины, цвета которых связаны правилом второго типа, а в массив usedVertices – вершины, цвета которых связаны двумя или более правилами второго типа.

7. Для каждого блока проверяем, есть ли в нем вершины, цвет которых зафиксирован. Если такие вершины есть, то проверяем, можно ли раскрасить остальные вершины блока в цвета из множества N. Если можно, то раскрашиваем остальные вершины в соответствующие цвета. При этом для каждой окрашенной вершины просматриваем смежные с ней вершины и запрещаем для них данный цвет. Если остальные вершины блока нельзя окрасить в допустимые цвета из множества N, то для вершины, определяющей цвета вершин блока, находим допустимый цвет, такой, что остальные вершины блока можно окрасить в допустимые для них цвета из множества N.

8. Просматриваем все вершины до тех пор, пока будем находить неокрашенные вершины, которые можно окрасить только в один цвет из множества N. Если находим такую вершину v_i, то красим ее в единственно возможный для нее цвет из множества N. Просматриваем все смежные с ней вершины и запрещаем для них цвет, в который окрашена вершина v_i. Если для какой-либо вершины остается только один возможный цвет, то после просмотра всех вершин шаг 8 повторяется снова.

9. Предварительная расстановка цветов в соответствии с правилом второго типа.

Находим вершины, которые можно окрасить только в один цвет из множества N. Если найдена такая вершина v_i, то запрещаем этот цвет для всех вершин, связанных правилом второго типа, а вершину v_i помечаем как просмотренную. Для непросмотренных вершин проверяем, есть ли среди них вершины из массива usedVertices. Если такая вершина найдена, то помечаем ее как просмотренную и запрещаем соответствующий цвет для непросмотренных вершин. Для непомеченных вершин находим все допустимые цвета. Среди непросмотренных вершин находим группы *идентичных* вершин: они соответствуют одной и той же группе учеников, одному и тому же преподавателю и требуют для своего проведения либо один и тот же тип аудитории, либо одну и ту же аудиторию. Идентичные вершины помечаются как просмотренные и для них находятся цвета, в которые они могут быть окрашены. Затем вы-

полняются действия, аналогичные шагу 8. Определяются списки допустимых цветов для всех вершин, которые еще не окрашены.

Алгоритм раскраски вершин графа G

1. Назначаются последовательно цвета из множества N всем вершинам, которые необходимо раскрасить. Назначаются цвета вершинам блока в соответствии с цветом, который получила вершина из массива coloredVertices.

2. Вычисляются оценки качества полученного назначения цветов. Для этого формируется массив conflictVertex, в который помещаются смежные вершины с одинаковыми цветами.

Для каждой вершины v_i просматриваются смежные с ней вершины. Если какая-либо из смежных вершин имеет цвет, такой же, как и вершина v_i, то вершина v_i помещается в массив conflictVertex, а штраф за назначение цветов увеличивается на величину соответствующего штрафа.

Для каждого интервала времени и каждого *ресурса* проверяется, превышено ли число имеющихся ресурсов. Если оно превышено, то все вершины, которые запрашивают данный ресурс в данный интервал времени, помечаются как *конфликтные*. Штраф за полученное назначение цветов увеличивается на произведение штрафа за данное нарушение на число таких нарушений.

Определяются конфликты при нарушении правил первого и второго типа. Если нарушено первое правило (т. е. соответствующие занятия не следуют непосредственно одно за другим), то обе вершины помечаются как конфликтные, а штраф за данное назначение цветов увеличивается на величину штрафа за это нарушение. Если нарушено правило второго типа (т. е. соответствующие занятия проводятся в один день), то одна из вершин помечается как конфликтная, а цена решения увеличивается на величину штрафа за это нарушение.

Определяются конфликты, приводящие к появлению форточек для учеников. Для каждого дня проверяется, есть ли у группы учеников в этот день форточки. Если есть, то подсчитывается их количество, а штраф за данное назначение цветов увеличивается на произведение штрафа за такое нарушение на количество форточек. Если форточка одна, то конфликтной считается либо первая вершина v_i, либо последняя вершина v_j, окрашенная в цвет из множества цветов для данного дня; если имеется больше одной форточки, то обе вершины v_i и v_j считаются конфликтными.

3. Если штраф за полученное назначение цветов равняется нулю, то получена допустимая раскраска, которая определяет расписание учебных занятий. В противном случае выполняется шаг 4.

4. Разрешение конфликтов.

Выделяются блоки вершин, которые содержат конфликтные вершины. Все вершины такого блока помечаются как конфликтные. (Если блок помечен как имеющий фиксированный цвет, то такой блок не может содержать конфликтных вершин.)

Все конфликтные вершины считаются неокрашенными и соответствующие ресурсы освобождаются. Случайным образом выбираем блок с конфликтными вершинами. Вычисляются возможные цвета для определяющей вершины выбранного блока с учетом уже окрашенных вершин. Если количество допустимых цветов отлично от нуля, то случайным образом выбирается один из них. Если нет допустимых цветов во множестве N, то выбирается один из возможных цветов без учета уже окрашенных вершин. Этот цвет назначается определяющей вершине блока, а также остальным его вершинам. Ресурсы, запрашиваемые вершинами блока, определяются как используемые.

Среди конфликтных вершин, связанных правилом второго типа, выбирается случайным образом вершина v_i. Просматриваются все вершины, смежные v_i, и если им приписан цвет, то он помечается как запрещенный для вершины v_i. Для каждого цвета из множества N, который остался допустимым, определяем, можно ли назначить его вершине v_i с учетом использованных ресурсов. Если нельзя, то помечаем этот цвет как недопустимый. Затем удаляем форточки из списка допустимых цветов. Для каждой группы учеников, связанной с вершиной v_i, находим количество еще не окрашенных вершин, а также количество форточек в имеющемся на данный момент расписании для рассматриваемой группы учеников. Если количество форточек равно

количеству неокрашенных вершин, то среди всех допустимых цветов из множества N разрешаем те цвета, которые соответствуют форточкам. Если имеются допустимые цвета, то выбираем один из них. Если нет допустимых цветов, то выбираем один из первоначально допустимых цветов. Окрашиваем вершину v_i в выбранный цвет и помечаем ресурсы, запрашиваемые вершиной v_i, как используемые.

Среди оставшихся конфликтных вершин случайным образом выбирается вершина v_j, для которой повторяются все действия предыдущего шага. Затем выполняется шаг 2.

4. Распределение занятий по аудиториям

В результате реализации описанных в разд. 3 алгоритмов получается разбиение занятий по интервалам времени из множества W, которое гарантирует существование допустимого разбиения занятий по аудиториям, а значит, и существование допустимого расписания.

Для разбиения учебных занятий по аудиториям, удаленным одна от другой (например, расположенным в различных учебных корпусах), используются алгоритмы, описанные в данном разделе. Этот этап особенно полезен при составлении расписаний занятий для высших учебных заведений, которые имеют обычно несколько зданий. Разбиение занятий по аудиториям состоит из двух частей, первая из которых представляет собой *генетический алгоритм*, а вторая – алгоритм последующего улучшения решения. Второй алгоритм реализуется после выполнения операций *скрещивания* и *мутации* в генетическом алгоритме, а также после удаления локального минимума, в который может попасть решение, полученное в результате применения генетического алгоритма.

При описании генетического алгоритма будем использовать терминологию монографии [10]. Данные генетического алгоритма кодировались следующим образом. В качестве *хромосомы* рассматривалось занятие (урок), а в качестве *информации*, которую несет хромосома, – временной интервал из множества W. Целевая функция представляет собой сумму штрафов за нарушение заданных условий при некорректном назначении занятий на интервалы из множества W (это взвешенная сумма штрафов за нарушение ограничений 4–9). Целью работы алгоритма является минимизация целевой функции от числа нарушений ограничений.

Первоначально определяется набор хромосом, над которыми будут производиться действия алгоритма. В данном случае это первые занятия из непрерывно следующих одно за другим занятий для каждой группы учеников и для каждого дня из планового периода. Соответственно, значение хромосомы – это аудитория, в которой будет проходить данное занятие. Следующие одна за другой аудитории для группы учеников строятся в расписании с целью минимизации расстояния, которое необходимо преодолеть учащимся при переходе в очередную аудиторию согласно построенному расписанию. Такие построения реализуются в алгоритме улучшения решения. Оценка качества решения вычисляется в результате суммирования расстояния всех переходов между аудиториями для всех групп учеников. Для этого вводится расстояние между аудиториями, которое соответствует длительности перехода. Например, если группа учеников остается на следующее занятие в той же самой аудитории, то цена перехода равняется нулю. Если группа учеников переходит в соседнюю аудиторию, то штраф полагается равным единице, а если в аудиторию на этаж выше или ниже – двум. Если группа учащихся разбивается на две подгруппы, то расстояние переходов вычисляется для каждой подгруппы. При решении данной задачи на реальных примерах были выбраны следующие параметры генетического алгоритма: $N_{pop} = 100$, $N_{keep} = 50$, $\mu = 0{,}1$, $N_{gen} = 503$, где N_{pop} – размер популяции, N_{keep} – число первоначальных решений (*особей*), μ – коэффициент мутации и N_{gen} – количество учебных занятий.

Генетический алгоритм

1. Определяются числа N_{pop}, N_{keep}, μ, N_{gen}.

2. Случайным образом строятся N_{keep} первоначальных решений задачи (особей). После построения каждого решения к нему применяется алгоритм улучшения решения.

3. Выполняется операция скрещивания. Для этого выбираются два *родителя* из первых N_{keep} элементов (особей), происходит скрещивание их *генов*, после чего над полученным результатом выполняется алгоритм улучшения решения. Данный шаг выполняется до тех пор, пока размер *популяции* не достигнет N_{pop}.

4. Полученная популяция сортируется по порядку возрастания штрафа за качество решения (чем меньше штраф, тем лучше решение).

5. Над элементами популяции (кроме первого) выполняется операция мутации. Количество мутаций (*mutation*) зависит от коэффициента мутации μ и вычисляется по следующей формуле: $mutation = \mu \times (N_{pop} - 1) \times N_{gen}$. Определяются случайным образом особи, подверженные мутации, гены, которые будут мутировать, и сами значения генов, которые получаются после мутации. После мутации над каждой особью выполняется алгоритм улучшения решения.

6. Полученная популяция сортируется по возрастанию штрафа за качество решения.

7. Удаляются все решения, начиная с $N_{keep} + 1$.

8. Если получено решение с нулевым штрафом, то алгоритм завершает свое выполнение и возвращает это решение, которое является оптимальным.

9. Если достигнуто предельно допустимое количество итераций, в течение которых можно искать решение, то алгоритм завершает свое выполнение и возвращает наилучшее из полученных решений.

10. Если алгоритм достиг локального минимума, т. е. какое-либо решение не улучшается на протяжении заранее определенного числа итераций генетического алгоритма, то происходит спонтанное изменение данного решения, чтобы выбраться из области локального минимума. Затем переходим на выполнение шага 3.

Следующий алгоритм выполняется после каждого изменения решения в генетическом алгоритме. Исходными данными для него являются назначенные аудитории на первые занятия из непрерывно следующих одно за другим занятий для каждой группы учеников и для каждого дня.

Алгоритм улучшения решения

1. Для всех первых занятий определяем те из них, которые проходят в одно и то же время и которые требуют для своего проведения одну и ту же аудиторию. Для таких занятий отменяем назначение аудиторий, а сами занятия помечаем как конфликтные.

2. Для каждого конфликтного занятия находим все доступные аудитории, в которых может проходить данное занятие, и случайным образом выбираем одну из них. (Это заведомо можно сделать, поскольку расстановка занятий по интервалам времени проводилась так, чтобы существовало допустимое распределение занятий по аудиториям.)

3. Назначение аудиторий для проведения последующих занятий в каждой цепи занятий.

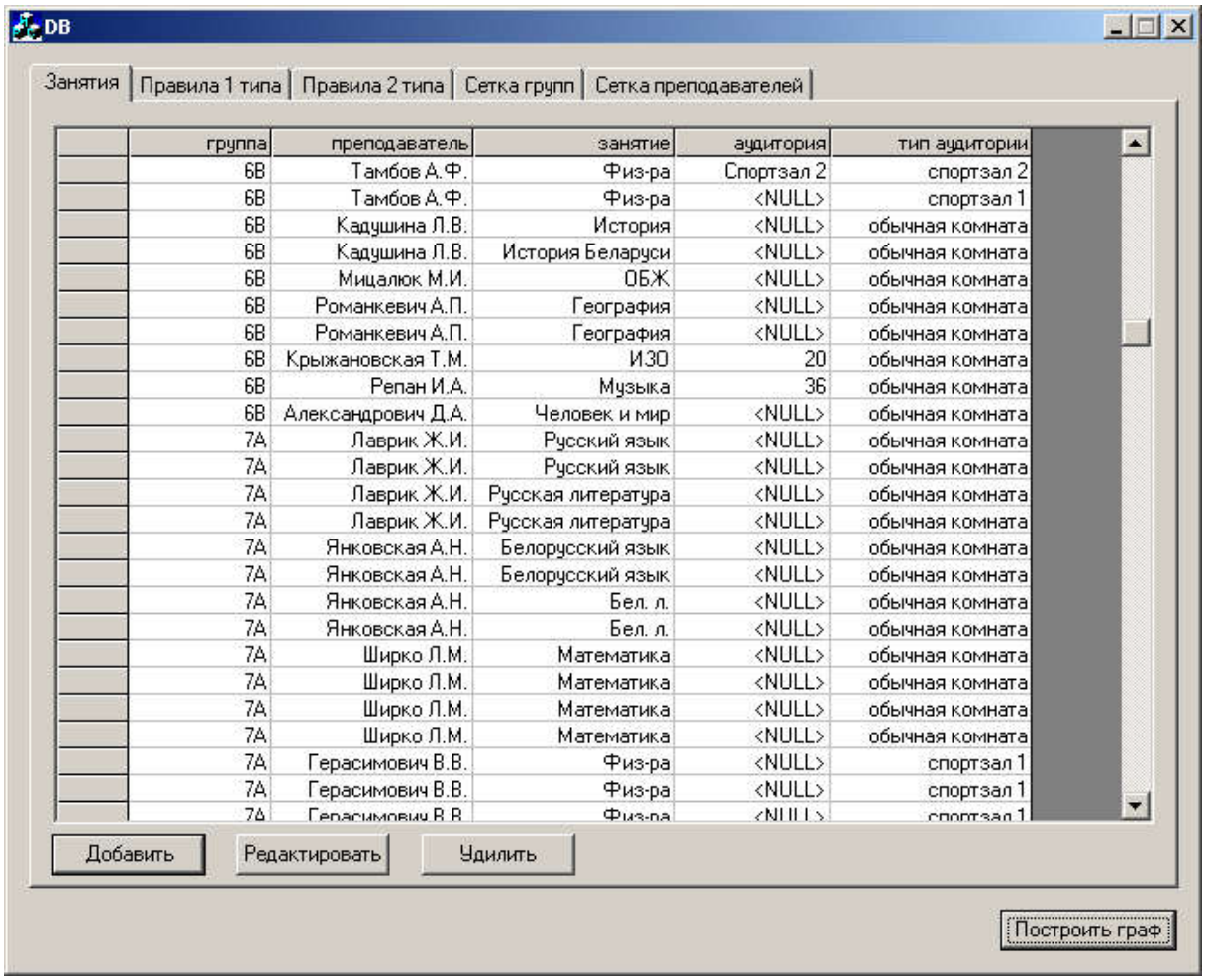
Выбирается очередная цепь занятий и для каждого последующего занятия (если для него явно не задана аудитория) выбирается одна из имеющихся в наличии аудиторий таким образом, чтобы сумма штрафов за переходы групп учеников была наименьшей.

5. Описание системы построения расписания учебных занятий

На основании описанной в разд. 2 сетевой модели и эвристических алгоритмов, описанных в разд. 3 и 4, была разработана интерактивная система построения и корректировки расписания, предназначенная для генерации учебных расписаний. Интерфейс системы представлен в виде диалоговых окон, что существенно облегчает работу пользователя. В основном диалоге (рис. 1) на экран персонального компьютера (ПК) программа выводит сводную таблицу занятий. В каждой строке таблицы содержится информация, необходимая для проведения занятия (о группе учеников, с которыми проводится занятие; о преподавателе, проводящем занятие; о названии предмета урока; об аудитории, в которой проводится занятие, или о типе необходимой для проведения занятия аудитории). Если аудитория не указывается, то в таком случае используется ключевое слово NULL. Все остальные поля таблицы являются обязательными. Пользователь имеет возможность добавить в таблицу новое занятие (кнопкой Добавить), ре-

дактировать таблицу (кнопкой Редактировать) либо удалять занятия из таблицы (кнопкой Удалить). На основе данных таблицы программа генерирует граф $G = (V, E)$ по команде Построить граф. При нажатии на кнопку Добавить либо Редактировать на экране ПК появляется окно для диалога, в котором можно либо создавать новое занятие, либо редактировать занятие, уже представленное в таблице. В режиме диалога можно задавать правила первого типа, определяя основное занятие и зависимые от него занятия, задавать правила второго типа и запрещенные интервалы времени для групп учеников. После ввода всей информации, необходимой для построения расписания, следует нажать кнопку Построить граф. В результате система начинает процесс построения графа. На основе полученного графа строится расписание для проведения указанных занятий. Готовое расписание представляется в виде сводной Excel-таблицы для групп учеников и отдельной Excel-таблицы для преподавателей (рис. 2).

Система была наполнена реальными данными, основанными на расписании занятий средней школы № 21 Минска. В школе занималось 17 классов (с 5-го по 11-й), занятия проводили 34 преподавателя в 27 аудиториях, разбитых на шесть подмножеств аудиторий различных типов. Занятия проводились шесть раз в неделю по семь занятий в день. Всего в неделю проводилось 503 занятия (у младших классов 24–28 занятий в неделю, у старших классов 30–35 занятий в неделю).



группа	преподаватель	занятие	аудитория	тип аудитории
6В	Тамбов А.Ф.	Физ-ра	Спортзал 2	спортзал 2
6В	Тамбов А.Ф.	Физ-ра	<NULL>	спортзал 1
6В	Кадушина Л.В.	История	<NULL>	обычная комната
6В	Кадушина Л.В.	История Беларуси	<NULL>	обычная комната
6В	Мицалюк М.И.	ОБЖ	<NULL>	обычная комната
6В	Романкевич А.П.	География	<NULL>	обычная комната
6В	Романкевич А.П.	География	<NULL>	обычная комната
6В	Крыжановская Т.М.	ИЗО	20	обычная комната
6В	Репан И.А.	Музыка	36	обычная комната
6В	Александрович Д.А.	Человек и мир	<NULL>	обычная комната
7А	Лаврик Ж.И.	Русский язык	<NULL>	обычная комната
7А	Лаврик Ж.И.	Русский язык	<NULL>	обычная комната
7А	Лаврик Ж.И.	Русская литература	<NULL>	обычная комната
7А	Лаврик Ж.И.	Русская литература	<NULL>	обычная комната
7А	Янковская А.Н.	Белорусский язык	<NULL>	обычная комната
7А	Янковская А.Н.	Белорусский язык	<NULL>	обычная комната
7А	Янковская А.Н.	Бел. л.	<NULL>	обычная комната
7А	Янковская А.Н.	Бел. л.	<NULL>	обычная комната
7А	Ширко Л.М.	Математика	<NULL>	обычная комната
7А	Ширко Л.М.	Математика	<NULL>	обычная комната
7А	Ширко Л.М.	Математика	<NULL>	обычная комната
7А	Ширко Л.М.	Математика	<NULL>	обычная комната
7А	Герасимович В.В.	Физ-ра	<NULL>	спортзал 1
7А	Герасимович В.В.	Физ-ра	<NULL>	спортзал 1
7А	Герасимович В.В.	Физ-ра	<NULL>	спортзал 1

Рис. 1. Общий вид окна для диалога пользователя с системой

Во время тестирования системы практически приемлемые расписания строились за 200–7000 поколений генетического алгоритма. Для построения графа и приемлемого расписания требовалось от 1 до 10 мин. работы процессора PC 1700 МГц. Генерируемые расписания не имеют *форточек* для учеников, распределяют занятия равномерно по всей неделе по пять-шесть занятий в день. Си-

туации, когда в один день у одного класса есть только четыре занятия, а в другой день – шесть, встречаются редко. Почти во всех классах занятия начинаются с первого урока (первого интервала, предназначенного для проведения уроков), но встречаются и такие дни, когда занятия начинаются со второго, а иногда и с третьего урока. Эти нарушения нетрудно скорректировать вручную.

Microsoft Excel - Book1

File Edit View Insert Format Tools Data Window Help

Arial

A1 =

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1			1							2							
2	№	Преподаватель	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1
3	1	Александрович Д.	11Б		8Г		8Б			9А		6В	9Б	11Б	11А		6Б
4	2	Бакар Л.Е.								8А							
5	3	Бережная В.В.			6В	6Б	8А	6А									
6	4	Вайдугова Н.В.											8В				5А
7	5	Вербило Л.М.	8Г	5Б	5А	8Б		8В		8Г	8В		5Б	8В	8Г		
8	6	Воскресенский Н.С.								10В	11А			10Б	11Б		
9	7	Герасимович В.В.	9Б		7А	5А	5Б	11Б		5Б	9Б		5А	7А	9А		
10	8	Долгушова А.Д.	10В	10В	10А	10А	10Б	10Б									
11	9	Жданович Н.В.		11Б	8Б	10В	8В	11А	9А	11А	10В	10А	8Г	9Б	5А		5Б
12	10	Захаренко Н.С.	8А	10А							10А		10В			10Б	
13	11	Зыбина С.Л.										8Б			8В		
14	12	Измайлович Л.Р.		8Б	8В	8Г	9Б				9А				8Б		8В
15	13	Кадушина Л.В.		8А	11А	11Б				6В	11Б	11А	8А	6А	6Б		
16	14	Кендыш О.Д.	5Б		10Б	7А	10А	10В									7А
17	15	Клышко И.А.		8В	8А		10В	8Г									9А
18	16	Крыжановская Т.М.		6Б	6А		5А	5Б									
19	17	Кузьмич А.О.				6В	6В						8В				5А
20	18	Паврик Ж.И.	7А	9Б	11Б	9Б	11А	9А		9Б	7А	9А		9А			
21	19	Любинский Д.В.								10Б	8А				10А		
22	20	Ляндрес А.И.	6Б	6В	6Б	6А		8А		6Б	6А	8А					6А
23	21	Маргевич Л.М.									5Б	5Б	8Б	8Г			8Б
24	22	Мицалюк М.И.	8Б	7А	5Б	8В	8Г	5А		10В	11А	6Б	6А	10Б	11Б		
25	23	Обухович Т.В.		8Г		11А		8Б		8В	8Б	8Г	11Б	11А			11Б
26	24	Омельчук Л.М.		11А	9Б	9А	11Б	10А		10А		5А	10Б	10В		9А	10В
27	25	Песецкая Т.Г.									8Г	8В	6Б	8Б			
28	26	Пилько А.И.			8Б	8А	8В		9А		8Г	8В	8Г	8Б	9Б		9Б
29	27	Репан И.А.										7А	6В	5А			
30	28	Романкевич А.П.									6В	10Б	10А		7А	11А	
31	29	Рябушко С.Л.	6А	6А			6Б	6Б		6А	6Б	11Б	11А	6В			11А
32	30	Симач О.А.	10А		10В	10Б	9А	9Б				9Б	9А	10А	10Б		10Б
33	31	Стжалковская О.Е.		11Б		10В		11А		11А	10В	10А	6Б	8А	5А		5Б
34	32	Тамбов А.Ф.	8В	10А			6А	6В		8Б	10А	6А	10В	6Б		10Б	6В
35	33	Ширко Л.М.		5А		5Б				7А	5А			5Б			
36	34	Янковская А.Н.									10Б	10В	7А		10В		10А
37																	

Рис. 2. Расписание занятий для преподавателей

В дальнейшем предполагается реализовать возможность достраивать расписания при уже имеющемся частичном первоначальном назначении некоторых занятий на интервалы времени для их проведения и с заранее назначенными для некоторых занятий аудиториями.

Заключение

Компьютеры все шире используются в общеобразовательных школах Беларуси как на уроках информатики, математики, физики, так и для организации учебного процесса в школе. В данной работе предлагается к использованию автоматизированная система составления школьных и вузовских расписаний. Система готова к практическому внедрению в школах, оснащенных ПК не слабее по мощности чем 1000 МГц с операционными системами версий Windows 2000 и выше.

Разработанная система может использоваться также при составлении расписаний учебных занятий в высших учебных заведениях. При этом наряду с рассмотренными ограничениями, которые следует учитывать при построении расписаний учебных занятий, требуется иметь в виду, что различные аудитории могут располагаться в разных зданиях высшего учебного за-

ведения, в связи с чем возникает задача оптимального распределения занятий между зданиями. Эта задача также представляется в терминах раскраски вершин графа с критерием минимизации переходов групп студентов и преподавателей из одного здания в другое, что соответствует запрещению некоторых последовательностей цветов для некоторых подмножеств вершин.

Данная работа выполнена при частичной поддержке Белорусского республиканского фонда фундаментальных исследований (проект Ф06МС-002).

Список литературы

1. Гэри, М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон. – М.: Мир, 1982. – 416 с.

2. Leighton, F.T. A graph coloring algorithm for large scheduling problems / F.T. Leighton // Journal of Research of the National Bureau of Standards. – Vol. 84. – 1979. – P. 489–506.

3. Werra, D. de. An introduction to timetabling / D. de Werra // European Journal of Operations Research. – Vol. 19. – 1985. – P. 151–162.

4. Tuza, Z. Graph colorings with local constraints: a survey / Z. Tuza // Discussions Mathematical Graph Theory. – Vol. 17. – 1997. – P. 101–228.

5. Расписание Про 2.3. – Mode of access: http://www.softsearch.ru/programs/28-705-raspisanie-pro-download.shtml.

6. Magic Timetable Master. – Mode of access: http://www.imagictimetablesoftware.com/

7. TimeTabler. – Mode of access: http://www.timetabler.com/

8. Танаев, В.С. Теория расписаний. Многостадийные системы / В.С. Танаев, Ю.Н. Сотсков, В.А. Струсевич. – М.: Наука, 1989. – 328 с.

9. Харари, Ф. Теория графов / Ф. Харари – М.: Мир, 1973. – 300 с.

10. Haupt, R.L. Practical Genetic Algorithms / R.L. Haupt, S.E. Haupt. – NJ: John Wiley & Sons, 2004. – 261 p.

Поступила 31.03.06

[1] *СП ЗАО «Научсофт»,*
Минск, Беды, 2
e-mail: sergey-111@tut.by

[2]*Объединенный институт проблем*
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: sotskov@newman.bas-net.by

S.V. Baltak, Yu.N. Sotskov

GENERATING COURSE TIMETABLING ON THE BASIS OF VERTEX GRAPH COLORING

The model, algorithms and software are briefly described for course timetabling generating. Process of timetabling generating is reduced to vertices coloring of the graph with additional restrictions on the set of used colors. Heuristic algorithms are developed that allow to construct suitable timetabling for schools, colleges, and universities. Software is tested on real input data.