e-ISSN 1607-3274 Радіоелектроніка, інформатика, управління. 2019. № 3
p-ISSN 2313-688X Radio Electronics, Computer Science, Control. 2019. № 3

УДК 004.94

СОСТАВЛЕНИЕ РАСПИСАНИЯ ЗАНЯТИЙ УНИВЕРСИТЕТА НА ОСНОВЕ КОНСТРУКТИВНОГО МОДЕЛИРОВАНИЯ

Шинкаренко В. И. – д-р техн. наук, профессор, заведующий кафедрой «Компьютерных информационных технологий» Днепровского национального университета железнодорожного транспорта им. академика В. Лазаряна, Днепр, Украина.

Жеваго А. А. – аспирант кафедры «Компьютерных информационных технологий» Днепровского национального университета железнодорожного транспорта им. академика В. Лазаряна, Днепр, Украина.

АННОТАЦИЯ

Актуальность. Составление расписания учебных занятий является одной из важнейших задач управления учебным процессом. Рост требований к качеству обучения, сокращение материальной базы и увеличение количества специальностей приводят к необходимости оптимизации процесса использования кадрового потенциала, аудиторного фонда и экономии энергетических ресурсов. Поскольку все факторы, влияющие на расписание, практически невозможно учесть, а интересы участников учебного процесса многообразны, задача составления расписания является многокритериальной с нечетким множеством факторов. В связи с этим проблема автоматизации составления расписаний учебных занятий в образовательных системах по-прежнему остается одной из актуальных проблем организации учебного процесса.

Цель. Повышение качества расписания занятий университета и уменьшение времени его составления. Качество расписания определяется тем насколько соблюдаются необходимые и желательные требования к нему. Необходимо предусмотреть возможность расширения желательных требований без изменения оптимизирующего механизма.

Метод. Для описания процессов составления расписания занятий применена методология конструктивно-продукционного моделирования. С точки зрения конструктивизма: на основе ряда конструкций с заданной структурой и содержанием необходимо завершить конструирование расписания с заданной структурой и частично содержанием. Разработка конструктора предусматривает определение расширяемого носителя, сигнатуры отношений и операций, а также информационного обеспечения. Наиболее сложной и существенной частью является создание множества правил подстановки, определяющих процесс вывода соответствующих конструкций.

Результаты. Разработан конструктор составления расписания занятий и программное обеспечение, реализующее предложенный метод.

Выводы. Проведенные эксперименты подтвердили эффективность предложенного подхода и программного обеспечения, реализующие предложенный метод. Это позволяет рекомендовать его для использования на практике при решении задач составления расписания занятий.

КЛЮЧЕВЫЕ СЛОВА: расписание занятий, конструктор, генетический алгоритм, моделирование.

АББРЕВИАТУРЫ

КПМ – конструктивно-продукционное моделирование.

НОМЕНКЛАТУРА

C – обобщенный конструктор;

M – неоднородный носитель конструктора;

Σ – сигнатура, состоящая из множеств операций связывания, подстановки и вывода, операций над атрибутами, а также из отношений подстановки;

Λ – информационное обеспечение конструирования;

$S\mapsto$ – операция специализации конструктора;

C_{Sch} – конструктор случайного допустимого расписания;

M_{Sch} – неоднородный носитель случайного допустимого расписания;

Σ_{Sch} – сигнатура случайного допустимого расписания;

Λ_{Sch} – информационное обеспечение конструирования случайного допустимого расписания;

T – множество терминалов;

N – множество нетерминалов;

Ξ – множество операций связывания;

Θ – множество операций подстановки и вывода;

Φ – множество операций над атрибутами;

$\Rightarrow$ – операция подстановки;

$|\Rightarrow$ – операция частичного вывода;

$\|\Rightarrow$ – операция полного вывода;

F_K – коэффициент значимости;

c_i – условие;

$\underset{c}{count}$ – функция определения количества элементов множества, удовлетворяющие условию c;

$class$ – занятие (терминал);

$idWD$ – идентификатор дня недели (атрибут);

NC – номер занятия (атрибут);

$idTW$ – идентификатор типа недели (атрибут);

$idTC$ – идентификатор типа занятия (атрибут);

$idStr$ – идентификатор потока (атрибут);

idD – идентификатор дисциплины (атрибут);

idT – идентификатор преподавателя (атрибут);

idA – идентификатор аудитории (атрибут);

$schedules$ – массив расписаний занятий (терминал);

DOI 10.15588/1607-3274-2019-3-17

allClasses – упорядоченное по индексу множество занятий (терминал);

classes – массив занятий, копия *allClasses* (терминал);

doneClasses – массив распределенных занятий (терминал);

index – индекс записи во множестве (атрибут);

S_N – размер упорядоченного по индексу множества (атрибут);

countDoneSchedules – количество составленных расписаний (терминал);

count $\mathrm{Re}\, q$ – количество желательных требований;

$countFouls_i$ – количество нарушений желательных требований;

value – значение (атрибут);

groupClass – занятие группы (терминал);

idGroup – идентификатор группы (атрибут);

groupSchedule – массив занятий групп (терминал);

listGroupSchedule – массив множеств занятий групп (терминал);

teacherClass – занятие преподавателя (терминал);

teacherSchedule – массив занятий преподавателей (терминал);

listTeacherSchedule – массив множеств занятий преподавателей (терминал);

auditoryClass – аудиторное занятие (терминал);

auditorySchedule – массив аудиторных занятий (терминал);

listAuditorySchedule – массив множеств аудиторных занятий (терминал);

auditory – аудитория (терминал);

id – идентификатор (атрибут);

auditories – упорядоченное по индексу множество аудиторий (терминал);

time – время проведения занятия (терминал);

classTimes – массив времен проведения занятий (терминал);

group – группа (терминал);

countStudent – количество студентов (атрибут);

groups – упорядоченное по индексу множество групп (терминал);

streamGroup – связь группы с потоком (терминал);

listStreamGroup – упорядоченное по индексу множество связей групп с потоками (терминал);

sizePopulation – размер популяции (терминал);

counter – счетчик популяций с одинаковыми оценками (терминал);

fine – оценка популяции (терминал);

newFine – измененная оценка популяции (терминал);

population – популяция (терминал);

countEqualsFine – количество популяций с одинаковыми оценками (терминал);

i, j, k, t – счетчики (терминалы);

$\alpha, \beta, \chi, \delta, \phi, \gamma, \eta, \kappa, \lambda, \mu, \nu, o, \pi, \vartheta, \tau$ – нетерминалы;

$\triangleright (L_1, L_2, \mathrm{arg}_1 .. \mathrm{arg}_n)$ – операция поиска в списке L_1 по условиям $\mathrm{arg}_1 .. \mathrm{arg}_n$ и запись результата в список L_2;

$\#(L, res)$ – операция получения количества элементов в списке L и запись целочисленного результата в res;

$\oplus(L, R)$ – операция добавления записи R в список L;

$\vee(L, res)$ – операция случайного выбора записи из списка L в res;

$\wedge(L, R)$ – операция удаления записи R из списка L;

$\nabla(L, i, res)$ – операция выбора из списка L i-ого элемента и запись в res;

$\equiv (L_G, class, res)$ – операция проверки возможности добавления занятия $class$ в список занятий групп L_G и запись в res, значения $true$ или $false$;

$\cong (L_T, class, res)$ – операция проверки возможности добавления занятия $class$ в список занятий преподавателей L_T и запись в res;

$\approx (L_A, class, res)$ – операция проверки возможности добавления занятия $class$ в список занятий аудиторий L_A и запись в res;

$\div(L)$ – операция удаления из L всех записей;

$\propto (L)$ – операция перемешивания записей в L;

$A|_X^Y$ – алгоритм над данными из входного множества X со значениями из множества Y;

$I\mapsto$ – операция интерпретации конструктора;

$K\mapsto$ – операция конкретизации конструктора;

s_i – отношение подстановки;

σ – начальный нетерминал;

g_i – набор операций над атрибутами;

d_i – вектор доступности правила (правило доступно, если $d_i = 1$ и не доступно, если $d_i = 0$);

!= – операция сравнения на неравенство;

C_{G_Sch} – конструктор оптимизации расписания, с помощью генетического алгоритма;

M_{G_Sch} – неоднородный носитель оптимизации расписания;

Σ_{G_Sch} – сигнатура оптимизации расписания;

DOI 10.15588/1607-3274-2019-3-17

Λ_{G_Sch} – информационное обеспечение оптимизации расписания;

$\prec(L, res)$ – операция оценки популяции L и запись в res;

$\exists(L)$ – операция оценки особей популяции L и выбор половины лучших;

$\times(L)$ – операция выполнения генетических операторов (селекция, кроссовер, мутация) над популяцией L. В результате популяция пополняется новой особью;

$\mp(L)$ – операция формирования случайного допустимого расписания L.

ВВЕДЕНИЕ

Составление расписания занятий университета представляет собой трудоемкий и сложный процесс. Размерность задач составления оптимальных расписаний настолько велика, что решить их простым перебором вариантов не представляется возможным.

На сегодняшний день существует много методов и алгоритмов для решения задачи составления расписания [1]. Однако еще до сих пор нет совершенного решения в этом вопросе.

Целью данной работы являлось создание конструктора формирования расписания занятий. На основе, которого можно создать программное обеспечение для автоматизации процесса составления расписания занятий и повышения его качества.

1 ПОСТАНОВКА ЗАДАЧИ

Имея учебную нагрузку с информацией о занятиях и данные, об аудиторном фонде необходимо найти такой вариант расписания, который удовлетворяет обязательным требованиям, а также минимизирует штрафные показатели за невыполнение желательных.

Из сказанного выше вытекает следующая задача.

Пусть заданы упорядоченные (по индексу) множества, элементы которых определены атрибутами в виде соответствующих кортежей:

– аудиторий $auditories$, элементы
$\langle idA, countStudent\rangle \leftarrow\!\lrcorner auditory_i$;

– групп $groups$, элементы
$\langle idGroup, countStudent\rangle \leftarrow\!\lrcorner group_i$;

– потоков $listStreamGroup$, элементы
$\langle idStr, idGroup\rangle \leftarrow\!\lrcorner streamGroup_i$;

– занятий $allClasses$, элементы
$\langle idWD, NC, idTW, idTC,$
$idStr, idT, idD, idA\rangle \leftarrow\!\lrcorner class_i$.

Для каждого занятия заданы значения атрибутов $idTC, idStr, idT, idD$.

Необходимо найти значения $idWD, NC, idTW, idA$ всех элементов $class_i$, такие что:

$$F = \sum_{i=1}^{count\,\mathrm{Re}\,q} F\_K_i \cdot countFouls_i \to \min,$$

где коэффициенты значимости F_K_i и количество желательных требований $count\,\mathrm{Re}\,q$ известны заранее.

$$countFouls_1 = \sum_{c_1}\sum_{c_2}\sum_{c_3} (\max_{c_4}(NC \leftarrow\!\lrcorner class_k) -$$
$$- \min_{c_4}(NC \leftarrow\!\lrcorner class_k) - count_{c_4}(class_k) + 1),$$
$$c_1 : \forall j = idT \in \{idT \leftarrow\!\lrcorner class_i\},$$
$$c_2 : \forall p = idWD \in \{idWD \leftarrow\!\lrcorner class_i\},$$
$$c_3 : \forall q = idTW \in \{idTW \leftarrow\!\lrcorner class_i\},$$
$$c_4 : j = idT \leftarrow\!\lrcorner class_k \;\&\; p = idWD \leftarrow\!\lrcorner class_k \;\&$$
$$q = idTW \leftarrow\!\lrcorner class_k;$$

$$countFouls_2 = \sum_{c_5}\sum_{c_6}\sum_{c_7} (\max_{c_8}(NC \leftarrow\!\lrcorner class_k) -$$
$$\min_{c_8}(NC \leftarrow\!\lrcorner class_k) - count_{c_8}(class_k) + 1),$$
$$c_5 : \forall j = idGroup \in \{idGroup \leftarrow\!\lrcorner group_m\},$$
$$c_6 : \forall p = idWD \in \{idWD \leftarrow\!\lrcorner class_i\},$$
$$c_7 : \forall q = idTW \in \{idTW \leftarrow\!\lrcorner class_i\},$$
$$c_8 : j = idGroup \leftarrow\!\lrcorner group_m = idGroup \leftarrow\!\lrcorner streamGroup_j \&,$$
$$idStr \leftarrow\!\lrcorner streamGroup_j = idStr \leftarrow\!\lrcorner class_k \&,$$
$$p = idWD \leftarrow\!\lrcorner class_k \;\&\; q = idTW \leftarrow\!\lrcorner class_k;$$

$$countFouls_3 = \sum_{c_1}\sum_{c_2}\sum_{c_3} \begin{cases} 1, \text{если } count_{c_4}(class_k) > 4, \\ 0, \text{ в противном случае;} \end{cases}$$

$$countFouls_4 = \sum_{c_5}\sum_{c_6}\sum_{c_7} \begin{cases} 1, \text{если } count_{c_8}(class_k) > 4, \\ 0, \text{ в противном случае.} \end{cases}$$

$countFouls_1, countFouls_2$ – количество окон у преподавателя и у группы, соответственно; $countFouls_3, countFouls_4$ – количество дней в которых у преподавателя и у группы больше 4 занятий.

При этом на расписание накладываются ограничения, нарушение которых считается недопустимым. А именно:

– одновременно у преподавателя одно занятие

$$\neg\exists i, j : \langle idWD, NC, idTW, idT\rangle \leftarrow\!\lrcorner class_i =$$
$$= \langle idWD, NC, idTW, idT\rangle \leftarrow\!\lrcorner class_j;$$

– одновременно у группы одно занятие

DOI 10.15588/1607-3274-2019-3-17

$$\neg\exists i, j, k, p, q : idStr{\downarrow}class_i = idStr{\downarrow}streamGroup_j \&,$$
$$idGroup{\downarrow}streamGroup_j = idGroup{\downarrow}group_k \&,$$
$$idStr{\downarrow}class_p = idStr{\downarrow}streamGroup_q \&,$$
$$idGroup{\downarrow}streamGroup_q = idGroup{\downarrow}group_k \&,$$
$$idTW{\downarrow}class_i = idTW{\downarrow}class_p \&,$$
$$idWD{\downarrow}class_i = idWD{\downarrow}class_p \&,$$
$$idNC{\downarrow}class_i = idNC{\downarrow}class_p;$$

– в аудитории проводится не более одного занятия одновременно

$$\neg\exists i, j : \langle idWD, NC, idTW, idA\rangle{\downarrow}class_i = \langle idWD, NC, idTW, idA\rangle{\downarrow}class_j;$$

– размер потока не должен быть больше количества мест в аудитории

$$\forall class_i : \sum_{c_5} countStudent{\downarrow}group_k \le countStudent{\downarrow}auditory_m,$$
$$c_5 : \forall k : idA{\downarrow}class_i = idA{\downarrow}auditory_m \&,$$
$$idStr{\downarrow}class_i = idStr{\downarrow}streamGroup_j \&,$$
$$idGroup{\downarrow}streamGroup_j = idGroup{\downarrow}group_k.$$

2 ОБЗОР ЛИТЕРАТУРЫ

Проблемой разработки методов и алгоритмов для решения задачи составления учебного расписания занимаются уже достаточно давно. Первые работы появились еще в 60-е гг. XX в. [2]. Математической основой таких работ является теория расписаний.

Общим для всех исследователей является признание NP-трудности задачи составления расписания и необходимости нахождения различных эвристик, понижающих порядок операций полного перебора.

Разнообразие методов и подходов при решении говорит, о том, что все еще нет универсального способа.

Первый этап формализации задачи – формирование набора требований, которые необходимо или желательно соблюдать при составлении расписания, и оценка важности каждого из этих требований. Сформирован набор из двадцати четырех показателей, необходимых для выбора наиболее приемлемого варианта учебного расписания. Проведена оценка этих показателей и после обработки получены результирующие оценки. Оценка проводилась на основе метода иерархий Саати [3].

Важное замечание, что задача составления расписания зависит от других смежных задач, которые необходимо решать при организации учебного процесса, например от задачи составления учебной нагрузки, т. к. варианты ее распределения могут по-разному влиять на качество итогового расписания [3].

В 2007 году был организован «Второй международный конкурс по составлению расписаний» [4], и было дано формальное определение проблемы составления расписаний по учебным планам с учетом ряда реальных ограничений. Анализ методов приведен в [1].

Применение методов целочисленного программирования к решению задачи составления расписания занятий описаны в [5] на примере системы образования Кувейта. В работе представлен двухэтапный подход. На начальном этапе определяется время проведения занятий. На втором этапе назначаются преподаватели.

Архитектура и работа автоматических систем основанных на эвристических алгоритмах со специализированными генетическими операторами и оценочной функцией на основе штрафов представлена в [6].

В последнее время все чаще для решения задачи составления расписания, используют различные эвристические методы. Обзор и критический анализ применения эвристических методов для составления расписания занятий приведен в [7]. Этот обзорный документ охватывает все аспекты учебного расписания, включая расписание экзаменов и школьное расписание, а также проблемы с расписанием в университете. К числу таких методов относятся генетические алгоритмы, которые являются методами эволюционного поиска и сочетают компьютерные методы моделирования генетических процессов в природных и искусственных системах [8].

Аспекты применения генетического алгоритма рассмотрены в работах [9,10], доказана целесообразность применения генетического алгоритма, который характеризуется устойчивостью к попаданию в локальные оптимумы поверхности возможных решений и гарантирует получение некоторого варианта решения за конечное время.

3 МАТЕРИАЛЫ И МЕТОДЫ

Для решения задачи формирования расписания использован двухэтапный подход, включающий формирование начального расписания и его последующую оптимизацию, с помощью генетического алгоритма.

Для описания процессов составления расписания занятий применена методология конструктивно-продукционного моделирования (КПМ).

Исследования по теме КПМ представлены в работах [11–12].

Разработка конструктора предусматривает определение расширяемого носителя, сигнатуры отношений и операций, а также информационного обеспечения.

Информационное обеспечение конструирования включает: онтологию, цель, правила, ограничения, начальные условия, условия завершения конструирования.

Основы онтологии обобщенного конструктора приведены в работах [11–12]. Онтология предметной области организации учебного процесса университета

DOI 10.15588/1607-3274-2019-3-17

интуитивно понятна. В работе приведены только те составляющие, которые необходимы для изложения материала.

Формирование расписания выполняется двумя конструкторами. Первый – формирует корректное, но не оптимальное расписание, второй – выполняет оптимизацию.

Первый этап конструирования это специализация обобщенного конструктора. На этапе специализации определяется предметная область.

Цель конструктора – построение корректного расписания занятий университета, сформированного случайным образом.

Начальные условия – предоставленные данные о учебной нагрузке преподавателей кафедр, аудиторный фонд.

Условия завершения – все предусмотренные занятия добавлены в расписание.

Определим специализацию обобщенного конструктора для составления расписания занятий:

$$C = \langle M, \Sigma, \Lambda \rangle_S \mapsto C_{Sch} \langle M_{Sch}, \Sigma_{Sch}, \Lambda_{Sch} \rangle,$$
$$\Lambda_{Sch} = \Lambda \bigcup \Lambda_1, \Lambda_1 = \{M_{Sch} \supset \mathrm{T} \bigcup \mathrm{N}\},$$
$$\Sigma_{Sch} = \{\Xi, \Theta, \Phi\}, \Xi = \{\bullet\}, \Theta = \{\Rightarrow, |\Rightarrow, \|\Rightarrow\}.$$

Терминальный алфавит содержит элементы и их атрибуты:

$idWD,NC,idTW,idTC,idStr,idT,idD,idA^{class};$

$class^{index,S\_N\,allClasses};$

$allClasses^{index,S\_N\,schedules};$

$value^{countDoneSchedules}; value^{counter}; value^{fine};$

$idStr,idGroup,idWD,NC,idTW,idTC^{groupClass};$

$groupClass^{index,S\_N\,groupSchedule};$

$idGroup,countStudent^{group};$

$groupSchedule^{index,S\_N\,listGroupSchedule};$

$idT,idWD,NC,idTW,idTC^{teacherClass};$

$teacherClass^{index,S\_N\,teacherSchedule};$

$teacherSchedule^{index,S\_N\,listTeacherSchedule};$

$idA,idStr,idWD,NC,idTW,idTC^{auditoryClass};$

$auditoryClass^{index,S\_N\,auditorySchedule};$

$auditorySchedule^{index,S\_N\,listAuditorySchedule};$

$idWD,NC,idTW^{time}; value^{sizePopulation};$

$time^{index,S\_N\,classTimes};$

$auditory^{index,S\_N\,auditories};$

$group^{index,S\_N\,groups}; idGroup,idStr^{streamGroup};$

$streamGroup^{index,S\_N\,listStreamGroup};$

$idA,countStudent^{auditory};$

$allClasses^{index,S\_N\,population};$

$value^{countEqualsFine}.$

Вводятся следующие операции над атрибутами:

$$\triangleright(L_1, L_2 \arg_1 .. \arg_n); \#(L,res); \oplus(L,R); \vee(L,res);$$
$$\wedge(L,R); \nabla(L,i,res); \equiv(L_G, class, res);$$
$$\cong(L_T, class, res); \approx(L_A, class, res); \div(L); \propto(L);$$
$$\prec(L,res); \mp(L); \exists(L); \times(L).$$

Для интерпретации C_{Sch} строится модель исполнителя в виде базовой алгоритмической структуры [11, 12]:

$$C_{A,Sch} = \langle M_{A,Sch}, V_{A,Sch}, \Sigma_{A,Sch}, \Lambda_{A,Sch} \rangle.$$

Вводятся следующие алгоритмы, реализующие операции над атрибутами:

$$A_1|_{l_h, l_q, f_i}^{f_j}; A_2|_{f_i, \psi}^{f_j}; A_3|_{\sigma, \psi}^{\bar{\Omega}}; A_4|_{L_1, \arg_1, .., \arg_n}^{L_2};$$
$$A_5|_{L}^{res}; A_6|_{L,R}^{L}; A_7|_{L}^{L}; A_8|_{L}^{res}; A_9|_{L,R}^{L}; A_{10}|_{L,i}^{res};$$
$$A_{11}|_{L_G, class}^{res}; A_{12}|_{L_T, class}^{res}; A_{13}|_{L_A, class}^{res}; A_{14}|_{a,b}^{a};$$
$$A_{15}|_{a,b}^{c}; A_{16}|_{a,b}^{c}; A_{17}|_{a,b}^{c}; A_{18}|_{L}^{L}; A_{19}|_{L}^{res}; A_{20}|_{L}^{L};$$
$$A_{21}|_{L}^{L}; A_{22}|_{L}^{L}.$$

Интерпретация заключается в связывании операций с алгоритмами [11, 12].

Выполнена интерпретация конструктора для составления расписания занятий университета:

DOI 10.15588/1607-3274-2019-3-17

$$\langle C_{Sch} = \langle M_{Sch}, \Sigma_{Sch}, \Lambda_{Sch} \rangle,$$
$$C_{A,Sch} = \langle M_{A,Sch}, V_{A,Sch}, \Sigma_{A,Sch}, \Lambda_{A,Sch} \rangle\rangle$$
$$I \mapsto {}_I C_{Sch} = \langle M_{I,Sch}, \Sigma_{I,Sch}, \Lambda_{I,Sch} \rangle,$$
$$\Lambda_{I,Sch} = \Lambda_{Sch} \cup \Lambda_3, \Lambda_3 = \{(A_1|_{l_h, l_q, f_i}^{f_j} \hookleftarrow \Rightarrow),$$

$$(A_2|_{f_i,\psi}^{f_j} \hookleftarrow | \Rightarrow), (A_3|_{\sigma,\psi}^{\bar{\Omega}} \hookleftarrow || \Rightarrow), (A_4|_{L_1, \arg_1, ..., \arg_n}^{L_2} \hookleftarrow \triangleright),$$
$$(A_5|_L^{res} \hookleftarrow \#), (A_6|_{L,R}^{L} \hookleftarrow \oplus), (A_7|_L^L \hookleftarrow \propto), (A_8|_L^{res} \hookleftarrow \vee),$$
$$(A_9|_{L,R}^{L} \hookleftarrow \wedge), (A_{10}|_{L,i}^{res} \hookleftarrow \nabla), (A_{11}|_{L_G, class}^{res} \hookleftarrow \equiv),$$
$$(A_{12}|_{L_T, class}^{res} \hookleftarrow \cong), (A_{13}|_{L_A, class}^{res} \hookleftarrow \approx), (A_{14}|_{a,b}^{a} \hookleftarrow :=),$$
$$(A_{15}|_{a,b}^{c} \hookleftarrow !=), (A_{16}|_{a,b}^{c} \hookleftarrow >), (A_{17}|_{a,b}^{c} \hookleftarrow +), (A_{18}|_L^L \hookleftarrow \div),$$
$$(A_{19}|_L^{res} \hookleftarrow \prec); (A_{20}|_L^L \hookleftarrow \mp); (A_{21}|_L^L \hookleftarrow \exists); (A_{22}|_L^L \hookleftarrow \times)\}.$$

Конкретизация заключается в расширении информационного обеспечения множеством правил.

Правила подстановки имеют вид $\psi_r : \langle s_r, g_r \rangle \in \Psi$, где s_r – отношение подстановки, g_r – набор операций над атрибутами, r – номер правила [11, 12].

Конкретизация конструктора для составления расписания занятий:

$$I\, C_{Sch} = \langle M_{I,Sch}, \Sigma_{I,Sch}, \Lambda_{I,Sch} \rangle,$$
$$K \mapsto C_{Sch} = \langle M_{K,Sch}, \Sigma_{K,Sch}, \Lambda_{K,Sch} \rangle.$$

Приведем правила подстановки.

Правило s_1 содержит отношение для инициализации значений атрибутов терминалов:

$$s_1 = \langle \sigma \to \alpha \rangle,$$
$$g_1 = \langle S\_N \hookleftarrow schedules := value \hookleftarrow sizePopulation,$$
$$S\_N \hookleftarrow listGroupSchedule := value \hookleftarrow sizePopulation,$$
$$S\_N \hookleftarrow listTeacherSchedule := value \hookleftarrow sizePopulation,$$
$$S\_N \hookleftarrow listAuditorySchedule := value \hookleftarrow sizePopulation,$$
$$value \hookleftarrow countDoneSchedules := 0 \rangle.$$

Правило s_2 проверяет, сформирована ли начальная популяция для генетического алгоритма, если да, выполнится правило s_3, которое заключается в выводе сформированных конструкций расписания, иначе формирование случайного допустимого расписания продолжится.

$$s_2 = \langle \alpha\, d_2 \to \beta \rangle,$$
$$g_2 = \langle d_2 := value \hookleftarrow countDoneSchedules! = value \hookleftarrow sizePopulation \rangle,$$
$$s_3 = \langle \alpha \to \{listGroupSchedule, listTeacherSchedule, listAuditorySchedule, schedules\} \rangle.$$

Правило s_4 очищает результаты предыдущего формирования, и инициализирует список занятий, данными взятыми из нагрузок кафедр.

$$s_4 = \langle \beta\, d_4 \to \delta \rangle,$$
$$g_4 = \langle \div(groupSchedule), \div(teacherSchedule),$$
$$\div(auditorySchedule), \div(doneClasses),$$
$$classes := allClasses,\ d_4 := \#(classes) > 0 \rangle.$$

Правило s_5 выполнится, когда случайное допустимое расписание будет сформировано. Расписание добавится в популяцию и увеличит счетчик количества сформированных расписаний на один. После чего выполнится s_2.

$$s_5 = \langle \beta \to \alpha \rangle,$$
$$g_5 = \langle value \hookleftarrow countDoneSchedules := value \hookleftarrow countDoneSchedules + 1,$$
$$\oplus(listGroupSchedule, groupSchedule),$$
$$\oplus(listTeacherSchedule, teacherSchedule),$$
$$\oplus(listAuditorySchedule, auditorySchedule),$$
$$\oplus(schedules, doneClasses) \rangle.$$

В s_6 случайным образом выбирается занятие для распределения. Из занятия достаются группы, и записываются в *groups*. Затем выполняется s_7, которое занимается поиском времени проведения занятия. Если такое время не удается найти, выполнится s_8, которое заново начнет формировать расписание.

$$s_6 = \langle \delta \to \phi \rangle, g_6 = \langle \vee(classes, class),$$
$$\wedge(classes, class), \triangleright(listStreamGroup,$$
$$idStr \hookleftarrow streamGroup \hookleftarrow i \hookleftarrow listStreamGroup$$
$$= idStr \hookleftarrow class, groups), value \hookleftarrow i := -1 \rangle$$
$$s_7 = \langle \phi\, d_7 \to \gamma \rangle, g_7 = \langle value \hookleftarrow i := value \hookleftarrow i + 1,$$
$$d_7 := \#(classTimes) > value \hookleftarrow i \rangle,$$
$$s_8 = \langle \phi \to \delta \rangle.$$

DOI 10.15588/1607-3274-2019-3-17

Правила s_{9-13} осуществляют поиск времени проведения занятия, проверяя, что у групп это время не занято.

$s_9 = \langle \gamma \to \eta \rangle,$
$g_9 = \langle \nabla(classTimes, value\text{↵}i, time),$
$idT\text{↵}teacherClass := idT\text{↵}class,$
$idTW\text{↵}teacherClass := idTW\text{↵}time,$
$idWD\text{↵}teacherClass := idWD\text{↵}time,$
$NC\text{↵}teacherClass := NC\text{↵}time,$
$value\text{↵}j := -1\rangle$
$s_{10} = \langle \eta d_{10} \to \kappa \rangle,$
$g_{10} = \langle value\text{↵}j := value\text{↵}j + 1, d_{10} := \#(groups) > value\text{↵}j \rangle$
$s_{11} = \langle \eta \to \chi \rangle.$
$s_{12} = \langle \kappa d_{12} \to \lambda \rangle,$
$g_{12} = \langle \nabla(groups, value\text{↵}j, group),$
$idStr\text{↵}groupClass := idStr\text{↵}class,$
$idTW\text{↵}groupClass := idTW\text{↵}time,$
$idWD\text{↵}groupClass := idWD\text{↵}time,$
$NC\text{↵}groupClass := NC\text{↵}time,$
$idGroup\text{↵}groupClass := idGroup\text{↵}group,$
$d_{12} := \equiv (groupSchedule, groupClass)\rangle,$
$s_{13} = \langle \kappa \to \phi \rangle.$

Правило s_{14} проверяет, что выбранное время проведения занятия свободно у преподавателя. Если у всех групп потока и преподавателя нет занятия в найденное время, выполнится s_{16}. Если время занято, выполнится s_{15}, которое заново будет искать время проведения занятия.

$s_{14} = \langle \lambda d_{14} \to \eta \rangle,$
$g_{14} = \langle d_{14} := \cong (teacherSchedule, teacherClass)\rangle,$
$s_{15} = \langle \lambda \to \phi \rangle.$

Правила s_{16-20} подбирают аудиторию для проведения занятия.

$s_{16} = \langle \chi \to \mu \rangle,\ g_{16} = \langle$
$idTW\text{↵}class = idTW\text{↵}time,$
$idWD\text{↵}class = idWD\text{↵}time,$
$NC\text{↵}class = NC\text{↵}time,$
$\propto (auditories), value\text{↵}k := -1\rangle$
$s_{17} = \langle \mu d_{17} \to \pi \rangle, g_{17} = \langle value\text{↵}k := value\text{↵}k + 1,$
$d_{17} := \#(auditories) > k\rangle$
$s_{18} = \langle \mu \to \nu \rangle, g_{18} = \langle value\text{↵}t := -1\rangle$
$s_{19} = \langle \pi d_{19} \to o \rangle, g_{19} = \langle,$
$\nabla(auditories, value\text{↵}k, auditory),$
$idA\text{↵}auditoryClass := idA\text{↵}auditory,$
$idTW\text{↵}auditoryClass := idTW\text{↵}time,$
$idWD\text{↵}auditoryClass := idWD\text{↵}time,$
$NC\text{↵}auditoryClass := NC\text{↵}time,$
$d_{19} := \approx (auditorySchedule, auditoryClass)\rangle,$
$s_{20} = \langle \pi \to \mu \rangle.$

Правила s_{21-25} добавляют полученное занятие в список распределенных занятий, после чего снова выполняется s_6.

$s_{21} = \langle o \to \nu \rangle,$
$g_{21} = \langle \oplus(auditorySchedule, auditoryClass), value\text{↵}t := -1\rangle,$
$s_{22} = \langle \nu d_{22} \to \vartheta \rangle, g_{22} = \langle value\text{↵}t := value\text{↵}t + 1,$
$d_{22} := \#(groups) > value\text{↵}t\rangle,$
$s_{23} = \langle \nu \to \tau \rangle,$
$s_{24} = \langle \vartheta \to \nu \rangle, g_{24} = \langle \nabla(groups, t, group),$
$idStr\text{↵}groupClass := idStr\text{↵}class,$
$idTW\text{↵}groupClass := idTW\text{↵}time,$
$idWD\text{↵}groupClass := idWD\text{↵}time,$
$NC\text{↵}groupClass := NC\text{↵}time,$
$idGroup\text{↵}groupClass := idGroup\text{↵}group,$
$\oplus(groupSchedule, groupClass)\rangle,$
$s_{25} = \langle \tau \to \delta \rangle, g_{25} = \langle,$
$idA\text{↵}class := idA\text{↵}auditoryClass,$
$\oplus(teacherSchedule, teacherClass),$
$\oplus(doneClasses, class)\rangle.$

В результате реализации конструктора формируется случайное допустимое расписание занятий университета, начальная популяция для генетического алгоритма.

После формирования случайного допустимого расписания следует этап преобразования с помощью генетического алгоритма.

Цель конструктора – построение оптимального расписания занятий университета.

Начальные условия – корректные расписания занятий университета, сформированные случайным образом, которые являются начальной популяцией.

Условия завершения – количество итераций без изменения штрафов не равно значению атрибута *value* терминала *countEqualsFine* (качество расписания не улучшается на протяжении заданного количества поколений).

Специализированной формальной структурой для преобразования случайного допустимого расписания

DOI 10.15588/1607-3274-2019-3-17

с помощью генетического алгоритма, C_{G_Sch}, будет:

$$\mathrm{C} = \langle \mathrm{M}, \Sigma, \Lambda \rangle_S \mapsto$$
$$C_{G\_Sch}\langle \mathrm{M}_{G\_Sch}, \Sigma_{G\_Sch}, \Lambda_{G\_Sch} \rangle.$$

Отношение и операции подстановки, частичного и полного вывода, конкатенации, присваивания аналогичны одноименным операциям конструктора случайного допустимого расписания.

Алгоритмы операций, использованы в рассматриваемой структуре, аналогичны алгоритмам одноименных операций структуры для построения случайного допустимого расписания.

Выполним конкретизацию структуры для преобразования случайного допустимого расписания с помощью генетического алгоритма

$$I\, C_{G\_Sch} = \langle M_{I,G\_Sch}, \Sigma_{I,G\_Sch}, \Lambda_{I,G\_Sch} \rangle,$$
$$K \mapsto C_{G\_Sch} =$$
$$\langle M_{K,G\_Sch}, \Sigma_{K,G\_Sch}, \Lambda_{K,G\_Sch} \rangle.$$

Приведем правила подстановки.

Правило s_1 выполняет начальную инициализацию значений атрибутов терминалов и расчет штрафа начальной популяции.

$$s_1 = \langle \sigma \to \alpha \rangle, g_1 = \langle value\text{↵}counter := 0,$$
$$value\text{↵}fine := \prec (\mp(schedules)),$$
$$value\text{↵}populaion := value\text{↵}schedules \rangle.$$

Правило s_2 проверяет что количество итераций без изменения штрафов не равно значению атрибута *value* терминала *countEqualsFine*, которое задается внешним исполнителем. Невыполнение условия означает, что расписание готово, и выполнится s_3. Иначе выполнится s_4.

$$s_2 = \langle \alpha_{d_2} \to \beta \rangle,$$
$$d_2 := value\text{↵}counter != value\text{↵}countEqualsFine,$$
$$s_3 = \langle \alpha \to population \rangle.$$

Правило s_4 формирует новую популяцию, половина новой популяции это особи старой, остальная часть это результат выполнения генетических операторов, а именно особи полученные путем селекции, скрещивания и мутации.

$$s_4 = \langle \beta_{d_4} \to \chi \rangle,$$
$$d_4 := \#(populaion) != value\text{↵}sizePopulaion * 2.$$

Правило s_5 выполняется, когда новая популяция удвоенного размера сформирована. Выполняет оценку новых особей, выбор половины лучших и оценку всей популяции.

$$s_5 = \langle \beta \to \delta \rangle, g_5 = \langle \exists(population),$$
$$value\text{↵}newFine := \prec (population) \rangle.$$

Правило s_6 выполняет генетические операторы и добавляет новые особи в популяцию.

$$s_6 = \langle \chi \to \beta \rangle, g_6 = \langle \times(population) \rangle.$$

Правила s_{7-9} сравнивает новую оценку штрафов популяции с оценкой, полученной на предыдущей итерации. Если оценка не изменилась, выполняется s_2, при этом увеличив счетчик повторений оценок. Иначе счетчик обнуляется и выполняется s_6.

$$s_7 = \langle \delta_{d_7} \to \varphi \rangle, d_7 := value\text{↵}newFine == value\text{↵}fine,$$
$$s_8 = \langle \delta \to \chi \rangle,$$
$$g_8 = \langle value\text{↵}counter := 0, value\text{↵}fine := value\text{↵}newFine \rangle,$$
$$s_9 = \langle \varphi \to \alpha \rangle, g_9 = \langle value\text{↵}counter := value\text{↵}counter + 1 \rangle.$$

В результате реализации конструктора формируется расписание занятий университета.

4 ЭКСПЕРИМЕНТЫ

Для исследования практической применимости предложенного метода он был программно реализован и экспериментально изучен при решении задачи построения расписания занятий Днепровского национального университета железнодорожного транспорта имени академика В. Лазаряна.

Эксперимент включал два этапа: на первом этапе происходил сбор входной информации, на втором загрузка входных данных и автоматизированное составление расписания.

Традиционно трудоемкая и рутинная работа по формированию сетки расписания характеризуется отсутствием единого источника входной информации и, как следствие, с необходимостью тщательной подготовки, структурирования, сбора и обработки большого объема исходной информации из разных структурных подразделений университета.

Весь объем документации вести вручную очень сложно. Поэтому была разработана система автоматизации формирования первичной документации, а именно рабочих учебных планов и распределение учебной нагрузки по кафедрам. Формат исходной документации был составлен таким образом, чтобы он совпадал с форматом, в настоящее время используется учебным отделом.

DOI 10.15588/1607-3274-2019-3-17

Было собрано 227 рабочих учебных планов для 227 групп. Загрузив их в систему автоматического формирования документации, было получено 18 документов «Нагрузка кафедры», для 18 кафедр.

Когда документы «Нагрузки кафедры» сформированы, они отправляются на кафедры, где заполняются ответственными за распределением нагрузки. В документе для каждого вида занятий указываются преподаватели, которые будут их вести. На этом закончился первый этап эксперимента.

После заполнения, документы возвращаются в учебный отдел, они загружаются в систему и на их основе формируется расписание занятий.

Непосредственно перед формированием расписания задаются все внешние параметры, такие как размер популяции, веса для каждого параметра по которым назначаются штрафы, количество лучших вариантов для принятия решения внешним исполнителем и другие (табл. 1).

Таблица 1 – Настройки формирования расписания

Настройка	Значение
Размер популяции	100
Количество лучших расписаний для выбора	3
Квадратный расчет штрафа	нет
Количество популяций с одинаковыми оценками	5
Штраф за нарушение сбалансированности расписания групп	1
Штраф за нарушение сбалансированности расписания преподавателей	1
Штраф за то что аудитория не на кафедре преподавателя	3
Штраф если у группы в день более четырех занятий	10
Штраф если у преподавателя в день более четырех занятий	7
Штраф если у группы в день менее двух занятий	10
Штраф если у преподавателя в день менее двух занятий	3
Штраф за «окна» в расписании группы	10
Штраф за «окна» в расписании преподавателя	5

5 РЕЗУЛЬТАТЫ

Результатом эксперимента, является три сформированных расписания (число три было указано диспетчером учебного отдела на втором этапе эксперимента). Каждое расписание оценено по следующим характеристикам, количество:

– нарушений сбалансированности расписания у групп и преподавателей;

– аудиторий не на кафедре преподавателя;

– дней, в которых занятий у преподавателя или группы больше четырех;

– дней, в которых занятий у преподавателя или группы меньше двух;

– окон в расписании групп или преподавателей;

– штрафов.

Для настроек указанных во время эксперимента, были получены следующие результаты (табл. 2).

Таблица 2 – Характеристики полученных расписаний

Характеристика	Расписание		
	1	2	3
Нарушение сбалансированности расписания у групп	95	118	127
Нарушение сбалансированности расписания у преподавателей	46	34	23
Аудитория не на кафедре преподавателя	53	15	40
Дни, в которых занятий у преподавателя больше четырех	16	12	10
Дни, в которых занятий у группы больше четырех	14	29	23
Дни, в которых занятий у преподавателя меньше двух	41	11	20
Дни, в которых занятий у группы меньше двух	5	18	12
Окна в расписании групп	19	12	20
Окна в расписании преподавателей	15	15	9
Сумма штрафов	990	979	995

Диспетчером учебного отдела будет выбрано расписание, которое более других удовлетворяет его требованиям. Если такого нет, то можно изменить настройки и повторно составить расписание.

6 ОБСУЖДЕНИЕ

Во время экспериментов для сравнения расписаний использовалось небольшое количество показателей, это количество может быть расширено, например как в [3], что приведет к более явному отличию разных расписаний и что увеличит область поиска. Но, это приведет к увеличению времени составления расписания, и увеличит сложность выставления приоритетности этих показателей для внешнего исполнителя. Поэтому очень важно выбрать оптимальное количество показателей.

Кроме того, во время экспериментов, было выявлено, что размер популяции и количество поколений с одинаковыми оценками влияют на качество расписания в большей степени, чем изменение приоритетности показателей. Следовательно, важно выбрать оптимальные значения для этих показателей, экспериментальным путем.

Проведенные эксперименты подтвердили работоспособность предложенных методов и реализующих их программных средств.

Использование предложенного метода значительно уменьшило время формирования расписания, а главное улучшило его качество по сравнению с ручным составлением.

ЗАКЛЮЧЕНИЕ

Во время выполнения работы было построено два конструктора для составления расписания занятий. Первый конструктор формирует корректное, но не оптимальное расписание. Второй конструктор на основе результатов первого проводит оптимизацию расписания. Для этого генетический алгоритм был адаптирован для решения задачи составления расписания.

DOI 10.15588/1607-3274-2019-3-17

На основании двух конструкторов было создано программное обеспечение средствами языка программирования C#, которое составляет расписание занятий университета. Для контроля входной информации были разработаны запрограммированные на VBA шаблоны (файлы Microsoft Excel).

БЛАГОДАРНОСТИ

Работа выполнена по научно-исследовательской теме Днипровского национального университета железнодорожного транспорта «Прикладное конструктивное моделирование программных сущностей» (№ гос. регистрации – 0116U006841). Авторы выражают благодарность ректорату университета за финансовую поддержку данных исследований.

ЛИТЕРАТУРА / ЛІТЕРАТУРА

1. An overview of curriculum-based course timetabling / [A. Bettinelli, V. Cacchiani, R. Roberti et al.] // Top. – 2015. – № 2. – P. 37. DOI: 10.1007/s11750-015-0366-z
2. Conway R. W. Theory of Scheduling / R. W. Conway, W. L. Maxwell, L. W. Miller. – New York : Addison-Wesley, 1967. – 128 p.
3. Хасухаджиев А. С. Формирование системы показателей для автоматизации учебного расписания типового вуза / А. С. Хасухаджиев // Вестник АГТУ. Серия : Управление, вычислительная техника и информатика. – 2017. – № 3. – С. 117–127. DOI: 10.24143/2072-9502-2017-3-117-127
4. Di Gaspero L. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3) : Technical Report : QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0 / L. Di Gaspero, B. McCollum, A. Schaerf / Queen's University. – Belfast, 2007. – 12 p.
5. Al-Yakoob S. M. Mathematical models and algorithms for a high school timetabling problem / S. M. Al-Yakoob, H. D. Sherali // Computers & Operations Research. – 2015. – P. 56–68. DOI: 10.1016/j.cor.2015.02.011
6. Lukas S. Solving timetable problem by genetic algorithm and heuristic search case study: Universitas pelita harapan timetable. In O. Roeva / S. Lukas, A. Aribowo, M. Muchri // Real-World Applications of Genetic Algorithms. – 2012. – Vol. 378. – P. 303–316. DOI: 10.1109/ICADIWT.2009.5273979
7. Pillay N. A review of hyper-heuristics for educational timetabling / N. Pillay // Annals of Operations Research. – 2014. – P. 36. DOI: 10.1007/s10479-014-1688-1
8. Субботін С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей / С. О. Субботін, А. О. Олійник, О. О. Олійник. – Запоріжжя : ЗНТУ, 2009. – 375 с.
9. Akkan C. A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem / C. Akkan, A. Gülcü // Computers & Operations Research. – 2018. – P. 22–32. DOI: 10.1016/j.cor.2017.09.007
10. Abdelhalim E. A. A utilization-based genetic algorithm for solving the university timetabling problem (uga) / E. A. Abdelhalim, G. A. El Khayat // Alexandria Engineering Journal. – 2016. – № 2. – P. 1395–1409. DOI: 10.1016/j.aej.2016.02.017
11. Shynkarenko V. I. Constructive-Synthesizing Structures and Their Grammatical Interpretations. I. Generalized Formal Constructive-Synthesizing Structure / V. I. Shynkarenko, V. M. Ilman. // Cybernetics and Systems Analysis. – 2014. – Vol. 50, Issue 5. – P. 655–662. DOI: 10.1007/s10559-014-9655-z
12. Shynkarenko V. I. Structural Models of Algorithms in Problems of Applied Programming. I. Formal Algorithmic Structures / V. I. Shynkarenko, V. M. Ilman, V. V. Skalozub // Cybernetics and Systems Analysis. – 2009. – Vol. 45, Issue 3. – P. 329–339. DOI: 10.1007/s10559-009-9118-0

Received 06.03.2019.
Accepted 03.06.2019.

УДК 004.94

СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ УНІВЕРСИТЕТУ НА ОСНОВІ КОНСТРУКТИВНОГО МОДЕЛЮВАННЯ

Шинкаренко В. І. – д-р техн. наук, професор, завідувач кафедрою «Комп'ютерних інформаційних технологій» Дніпровського національного університету залізничного транспорту ім. академіка В. Лазаряна, Дніпро, Україна.

Жеваго О. О. – аспірант кафедри «Комп'ютерних інформаційних технологій» Дніпровського національного університету залізничного транспорту ім. академіка В. Лазаряна, Дніпро, Україна.

АНОТАЦІЯ

Актуальність. Складання розкладу навчальних занять є одним із найважливіших завдань управління навчальним процесом. Зростання вимог до якості навчання, скорочення матеріальної бази і збільшення кількості спеціальностей призводять до необхідності оптимізації процесу використання кадрового потенціалу, аудиторного фонду та економії енергетичних ресурсів. Оскільки всі фактори, що впливають на розклад, практично неможливо врахувати, а інтереси учасників навчального процесу різноманітні, завдання складання розкладу є багатокритеріальною з нечіткою множиною факторів. У зв'язку з цим проблема автоматизації складання розкладів навчальних занять в освітніх системах навчання як і раніше залишається однією з актуальних проблем організації навчального процесу.

Мета. Підвищення якості розкладу занять університету і зменшення часу на його складання. Якість розкладу визначається тим наскільки дотримуються необхідні і бажані вимоги до нього. Необхідно передбачити можливість розширення бажаних вимог без зміни оптимізуючого механізму.

Метод. Для опису процесів складання розкладу занять застосована методологія конструктивно-продукційного моделювання. З точки зору конструктивізму: на основі ряду конструкцій із заданою структурою і змістом необхідно завершити конструювання розкладу із заданою структурою та частково змістом. Розробка конструктора передбачає визначення розширюваного носія, сигнатури відносин і операцій, а також інформаційного забезпечення. Найбільш складною і істотною частиною є створення множини правил підстановки, що визначають процес виведення відповідних конструкцій.

DOI 10.15588/1607-3274-2019-3-17

Результати. Розроблено конструктор складання розкладу занять та програмне забезпечення, що реалізує запропонований метод.

Висновки. Проведені експерименти підтвердили ефективність запропонованого підходу і програмного забезпечення, що реалізує запропонований метод. Це дозволяє рекомендувати його для використання на практиці при вирішенні завдань складання розкладу занять.

КЛЮЧОВІ СЛОВА: розклад занять, конструктор, генетичний алгоритм, моделювання.

UDC 004.94

GENERATING UNIVERSITY COURSE TIMETABLE USING CONSTRUCTIVE MODELING

Shinkarenko V. I. – Dr. Sc., Professor, Head of the Department of Computer Information Technologies, Dnipropetrovsk National University of Railway Transport named after academician V. Lazaryan, Dnipro, Ukraine.

Zhevago O. O. – Post-graduate student of the Department of Computer Information Technologies, Dnipropetrovsk National University of Railway Transport named after academician V. Lazaryan, Dnipro, Ukraine.

ABSTRACT

Context. Generating university course timetable is one of the most important tasks of managing the educational process. Growing requirements for the quality of education, a reduction in the material base and an increase in the number of specialties lead to the need to optimize the process of using human resources, the classroom fund and to save energy resources. Since all the factors influencing the schedule are almost impossible to take into account and the interests of the participants in the educational process are diverse, the task of creating a schedule is multicriteria with a fuzzy set of factors. In this regard, the problem of automating the timetabling is still one of the urgent problems of the organization of the educational process.

Objective. The goal of the work is the improving the quality of university timetables and reducing the time it takes. The quality of the schedule is determined by the extent to which the necessary and desirable requirements are met. It is necessary to provide possibility of expanding the desired requirements without changing the optimizing mechanism.

Method. To describe the processes of scheduling classes applied the methodology of constructive-production modeling. From the point of view of constructivism: on the basis of a number of structures with a given structure and content, it is necessary to complete the construction of a schedule with a given structure and partly content. The development of the designer provides for the definition of expandable carrier, signatures of relations and operations, as well as information support. The most difficult and essential part is the creation of a set of substitution rules, which define the output process of the corresponding constructions.

Results. Developed a university course timetabling constructor and software that implements the proposed method.

Conclusions. The experiments confirmed the effectiveness of the proposed approach and the software that implements the proposed method. This allows us to recommend it for use in practice in solving problems of scheduling classes.

KEYWORDS: timetabling, constructor, genetic algorithm, modeling.

REFERENCES

1. Bettinelli A. , Cacchiani V., Roberti R. et al. An overview of curriculum-based course timetabling, *Top*, 2015, No. 2, P. 37. DOI: 10.1007/s11750-015-0366-z
2. Conway R. W., Maxwell W. L., Miller L. W. Theory of Scheduling. New York, Addison-Wesley, 1967, 128 p.
3. Hasuhadzhiev A. C. Formirovanie sistemy pokazatelej dlja avtomatizacii uchebnogo raspisanija tipovogo vuza, *Vestnik AGTU. Serija: Upravlenie, vychislitel'naja tehnika i informatika*, 2017, No. 3, pp. 117–127. DOI: 10.24143/2072-9502-2017-3-117-127
4. Gaspero L. Di, McCollum B., Schaerf A. The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3) : Technical Report : QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0, Queen's University. Belfast, 2007, 12 p.
5. Al-Yakoob S. M., Sherali H. D. Mathematical models and algorithms for a high school timetabling problem, *Computers & Operations Research*, 2015, pp. 56–68. DOI: 10.1016/j.cor.2015.02.011
6. Lukas S., Aribowo A., Muchri M. Solving timetable problem by genetic algorithm and heuristic search case study: Universitas pelita harapan timetable. In O. Roeva, *Real-World Applications of Genetic Algorithms*, 2012, Vol. 378, pp. 303–316. DOI: 10.1109/ICADIWT.2009.5273979
7. Pillay N. A review of hyper-heuristics for educational timetabling, *Annals of Operations Research,* 2014, P. 36. DOI: 10.1007/s10479-014 -1688-1
8. Subbotin S. O., Olijnyk A. O., Olijnyk O. O. Neiteratyvni, evoljucijni ta mul'tyagentni metody syntezu nechitkologichnyh i nejromerezhnyh modelej. Zaporizhzhja, ZNTU, 2009, 375 p.
9. Akkan C., Gülcü A. A bi-criteria hybrid Genetic Algorithm with ro-bustness objective for the course timetabling problem, *Computers & Operations Research,* 2018, pp. 22–32. DOI: 10.1016/j.cor.2017.09.007
10. Abdelhalim E. A., El Khayat G. A. A utilization-based genetic algorithm for solving the university timetabling problem (uga), *Alexandria Engineering Journal*, 2016, No. 2, pp. 1395–1409. DOI: 10.1016/j.aej.2016.02.017
11. Shynkarenko V. I., Ilman V. M. Constructive-Synthesizing Structures and Their Grammatical Interpretations. I. Generalized Formal Constructive-Synthesizing Structure, *Cybernetics and Systems Analysis,* 2014, Vol. 50, Issue 5, pp. 655–662. DOI: 10.1007/s10559-014-9655-z
12. Shynkarenko V. I., Ilman V. M., Skalozub V. V. Structural Models of Algorithms in Problems of Applied Programming. I. Formal Algorithmic Structures, *Cybernetics and Systems Analysis*, 2009, Vol. 45, Issue 3, pp. 329–339. DOI: 10.1007/s10559-009-9118-0

DOI 10.15588/1607-3274-2019-3-17