ЗАДАЧА СОСТАВЛЕНИЯ РАСПИСАНИЙ: РЕШЕНИЕ НА ОСНОВЕ МНОГОАГЕНТНОГО ПОДХОДА

Т.С. Бабкина,
старший преподаватель кафедры информационных систем и технологий Нижегородского филиала Государственного университета — Высшей школы экономики

Разработана оригинальная математическая модель составления раписаний учебных заведений с учётом индивидуальных предпочтений и проанализированы методы многоагентной оптимизации.

Общая постановка задачи и математическая модель

Проведённый анализ показал большую перспективность применении многоагентного подхода к решению задачи составления учебного расписания. Использование парадигмы взаимодействия большого числа независимых рациональных сущностей для поиска оптимального расписания позволяет учитывать предпочтения индивидуальных пользователей о времени и месте проведения занятий, повысить качество получаемого расписания и полностью учесть свойственный этой задаче распределённый характер; составлять расписание аудиторного фонда и любых других видов ресурсов. В этом важное отличие многоагентного подхода от других известных алгоритмов, где составляются только расписания по времени.

Основа построения собственного многоагентного алгоритма составления учебного расписания — точная постановка задачи в виде математической модели. В целях упрощения изложения в такой модели будем называть лиц, заинтересованных в результатах процесса составления расписания, пользователями расписания. К ним относятся преподаватели и учебные группы. Роль первых — провести лекционное или практическое занятие; роль последних — присутствовать на занятиях. Введём следующие обозначения.

Учебные группы и потоки.

$g \in \mathbf{G}$ — номер учебной группы.

$\mathbf{G}$ — множество номеров всех учебных групп.

$|\mathbf{G}| = \gamma$ — количество групп.

Каждая группа входит в один или несколько потоков. При объединении групп в один поток используются следующие принципы:

1. Группы в потоке используют один и тот же аудиторный фонд. Например, поток «Гуманитарные предметы, ФИСТ, 4-ый курс», состоящий из групп 99-ПМ, 99-РРТ, 99-Р-1, 99-Р-2, 99-ССК, использует аудиторный фонд из двух аудиторий на 120 мест.
2. Лекции читаются всему потоку одновременно.
3. У каждого потока есть хотя бы одно занятие.
4. Поток может состоять из одной учебной группы. Например, поток «Прикладная математика, предметы по специальности, 4-ый курс» состоит из одной группы 99-ПМ.

$\mathbf{R}$ — множество номеров учебных потоков.

$|\mathbf{R}| = \rho$ — количество потоков.

$r \in \mathbf{R}$ — номер потока. Каждая группа самостоятельный поток, поэтому $\rho \geq \gamma$.

$\mathbf{C}_r \subset \mathbf{G}$ — поток. Потоком называется любое подмножество множества групп.

$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2,..., \mathbf{C}_\rho$ — множество всех потоков. Потоки могу пересекаться между собой.

Преподаватели.

$p \in \mathbf{P}$ — номер преподавателя;

$\mathbf{P}$ — множество всех преподавателей.

$|\mathbf{P}| = \pi$ — количество преподавателей.

Пользователи расписания.

Объединение множества всех групп с множеством всех преподавателей: $\mathrm{M} = \mathbf{G} \cup \mathbf{P}$, $m \in \mathbf{M}$ — уникальный номер пользователя расписания.

Время.

$w \in \mathbf{W}_g$ – номер учебного дня недели; $\mathbf{W}_g \in \mathbf{W} = \{1, 2,..., 7\}$ – множество учебных дней группы; g, $\mathbf{W}$ – множество всех дней недели. $j \in \mathbf{J} = \{1, 2,..., 8\}$ – номер учебной пары. $\mathbf{T} = \{(w, j) \mid w \in \mathbf{W}, j \in \mathbf{J}\}$ – множество *таймслотов*. Таймслотом называется элементарная единица времени в задаче составления расписания. Например, таймслот (1, 2) означает: понедельник, вторая пара. Для каждого пользователя m известно множество таймслотов, когда он свободен, то есть в это время занятие может быть проведено $\mathbf{T}^+_m \subset \mathbf{T}$, и множество недоступных таймслотов $\mathbf{T}^-_m \subset \mathbf{T}$ ($\mathbf{T}^+_m \cup \mathbf{T}^-_m = \mathrm{T}$; $\mathbf{T}^+_m \cap \mathbf{T}^-_m = \varnothing$), например, выходные дни, когда занятия не могут быть проведены.

Занятия.

Преподаватели проводят лекционные и практические занятия. Лекционные занятия проводятся у всего потока сразу. Практические занятия – только у одной группы; требуют другого аудиторного фонда. Например, практические занятия по информатике должны проходить в компьютерном классе. Введём следующие обозначения для занятий.

$\mathbf{S}_r = \{1, 2, ..., \sigma_r\}$ – множество номеров лекционных занятий, читаемых на потоке r; $s_r \in \mathbf{S}_r$ – номер лекционного занятия;

$\mathbf{Q}_r = \{1, 2, ..., \theta_r\}$ – множество номеров практических занятий, проводимых на потоке r; $q_r \in \mathbf{Q}_r$ – номер практического занятия;

Глобально уникальным идентификатором занятия будет пара «номер потока – номер занятия на потоке». Занятия существенно зависят от потока. Так, «физика на 2-м курсе» отличается по содержанию от «физика на 3-м курсе».

Лекционное занятие однозначно идентифицируется парой значений «номер потока – номер лекционного занятия»: (r, s_p), где

$$\mathbf{RS} = \{(r, s_r) \mid r \in \mathbf{R}, s_r \in \mathbf{S_r}\} \quad (1)$$

есть множество всех лекционных занятий.

Количество лекционных занятий:

$$|\mathbf{RS}| = \sum_{r=1}^{\rho} \sigma_r \quad (0.1)$$

Практическое занятие однозначно идентифицируется тройкой значений «номер потока – номер практического занятия – номер группы»: $(r, g, q_r) \in \mathbf{RQG}$, где

$$\mathbf{RQG} = \{(r, g_r, q_r) \mid r \in \mathbf{R},\ q_r \in \mathbf{Q}_r,\ g_r \in \mathbf{C}_r\} \quad (2)$$

есть множество всех практических занятий.

Количество практических занятий:

$$|\mathbf{RQG}| = \sum_{r=1}^{\rho} |\mathbf{C_r}| \cdot \theta_r ,$$

где: $|\mathbf{C}_r|$ – количество групп в потоке .

При дальнейшем описании задачи нам в большинстве случаев будет неважно, какое занятие рассматривается – лекционное или практическое. Поэтому под занятием будем понимать элемент из объединения двух множеств – множества лекционных и множества практических занятий:

$$\mathbf{E} = \mathbf{RS} \cup \mathbf{RQG} \quad (0.2).$$

Учебный план закрепляет за каждым преподавателем предметы, которые он должен будет провести в течение семестра. Назначение лекционных занятий:

$$\delta_1 : \mathbf{RS} \to \mathbf{P},$$

где: $\mathbf{P}$ – множество преподавателей;

$\mathbf{RS}$ – множество лекционных занятий.

Например, $\delta_1(1, 2) = 4$ означает, что преподаватель номер 4 ведет занятие номер 2 из списка лекционных занятий потока номер 1.

Учебный план практических занятий:

$$\delta_2 : \mathbf{RQG} \to \mathbf{P}$$

где: $\mathbf{RQG}$ – множество троек: «номер потока – номер практического занятия – номер группы».

Например, $\mathbf{C}_1 = \{3, 4, 5\}$ – поток 1 состоит из групп 3, 4, 5; $\mathbf{Q}_1 = \{1, 2\}$ – у потока 1 всего два практических занятия; запись $\delta 2(1, 2, 4) = 7$ означает, что у группы с номером 4, которая входит в поток 1, практическое занятие номер 2 ведёт преподаватель 7.

В общем случае учебный план можно записать так:

$$\delta : \mathbf{E} \to \mathbf{P} \quad (3)$$

где:

$$\delta(e) = \begin{cases} \delta_1(e), e \in \mathbf{RS} \\ \delta_2(e), e \in \mathbf{RQG} \end{cases}$$

Зная учебный план, можно вычислить следующую величину: множество занятий, проводимых пользователем-преподавателем (или для пользователя-группы) с номером m:

$$\begin{aligned} \mathbf{E}_m = \{e \mid m \in \mathbf{P} \wedge \delta(e) = m\} \cup \\ \{e = (r, s) \mid m \in \mathbf{C}_r \wedge s \in \mathbf{S}_r\} \cup \\ \{e = (r, q, m) \mid m \in \mathbf{C}_r \wedge q \in \mathbf{Q}_r\} \end{aligned} \quad (4)$$

Аудиторный фонд – множество аудиторий, где могут быть проведены занятия.

A – множество всех аудиторий и помещений.

Для каждого занятия *e* выделяется некоторое подмножество аудиторий $\mathbf{A}_e \subset \mathbf{A}$; аудитория для проведения занятия выбирается только из этого подмножества.

Неизвестные функции задачи. Неизвестные в данной задаче – *расписание времени* и *расписание аудиторий*. Расписание времени – отображение множества предметов на множество таймслотов:

$$\tau : \mathbf{E} \to \mathbf{T}, \tag{5}$$

где: **E** – множество всех занятий;

T – множество всех таймслотов.

Пример 1. $\tau(1, 2) = (4, 4)$, означает, что лекция номер 2 на потоке 1 будет проводиться в четверг на четвертой паре.

Пример 2. $\mathbf{C}_1 = \{3, 4, 5\}$ – поток 1 состоит из групп 3, 4, 5; $\mathbf{G}_1 = \{1, 2\}$ – у потока 1 всего два практических занятия; запись означает, что у группы с номером 4, которая входит в поток 1, практическое занятие номер 2 будет проводиться в четверг на четвертой паре.

Расписание аудиторий – это отображение множества занятий на множество аудиторий:

$$\alpha : \mathbf{E} \to \mathbf{A} \tag{6}$$

где: **E** – множество всех занятий;

A – множество всех аудиторий.

Пример 1. $\alpha(1, 2) = 101$ означает, что занятие 2 потока 1 проводится в аудитории 101.

Пример 2. $\mathbf{C}_1 = \{3, 4, 5\}$ – поток 1 состоит из групп 3, 4, 5; $\mathbf{G}_1 = \{1, 2\}$ – у потока 1 всего два практических занятия; запись означает, что у группы с номером 4, которая входит в поток 1, практическое занятие номер 2 будет проводиться в аудитории 101.

Ограничения. Введём в задаче следующие ограничения.

1. Один преподаватель в каждый момент времени может проводить не более одного занятия.

$$\forall p \in \mathbf{P}, \forall e_1, e_2 \in \mathbf{E} : e_1 \neq e_2 \wedge \delta(e_1) = \delta(e_2) = p \Rightarrow \tau(e_1) \neq \tau(e_2) \tag{7}$$

2. В одной аудитории в каждый момент времени может проводиться не более одного занятия.

$$\forall a \in \mathbf{A}, \forall e_1, e_2 \in \mathbf{E} : e_1 \neq e_2 \wedge \alpha(e_1) = \alpha(e_2) = a \Rightarrow \tau(e_1) \neq \tau(e_2) \tag{8}$$

3. У одной группы в каждый момент времени может проводиться не более одного занятия.

$$\begin{gathered}\forall g \in \mathbf{G} : \big(e_1 = (r_1, \ldots), e_2 = (r_2, \ldots) \in \mathbf{E} \wedge g \in \mathbf{Cr_1} \wedge g \in \mathbf{Cr_2} \wedge e_1 \neq e_2\big) \vee \\ \vee \big(e_1 = (\ldots, g), e_2 = (\ldots, g) \in \mathbf{E} \wedge e_1 \neq e_2\big) \Rightarrow \tau(e_1) \neq \tau(e_2)\end{gathered} \tag{9}$$

Другими словами, для каждых двух пересекающихся потоков в каждый момент времени не может быть разных лекционных занятий в одно и то же время и для каждой группы не может быть разных практических занятий в одно и то же время. Это ограничение удобно проверять, предварительно разбив все занятия на γ классов. В класс

$$\mathbf{K_g} \subset \mathbf{E},\ g \in \mathbf{G}\left(g_1 \neq g_2 \Rightarrow \mathbf{K_{g_1}} \cap \mathbf{K_{g_2}} = \varnothing\right)$$

входят все занятия, которые читаются группе *g* персонально или в каком-либо потоке. Ограничение 3 будет выполнено тогда и только тогда, когда все занятия групп одного класса проводятся в разное время.

Приоритеты занятий. Очевидно, что не все занятия имеют одинаковую важность в рамках учебного процесса. Для них можно задать порядок по приоритету. Более приоритетные занятия будут в первую очередь занимать время и место; менее приоритетные занятия уже не смогут претендовать на эти ресурсы. Итак, приоритеты занятий – отношение частичного порядка на множестве занятий.

$$e_1 \succ e_2 \Leftrightarrow U(e_1) \geq U(e_2), \tag{10}$$

где: $e_1, e_2 \in \mathbf{E}$;

$$U(e) = |\mathbf{M}_e| + k_1(e) + k_2(e) + k3(p_e)$$

– «полезность» занятия е ;

$$\mathbf{M}_e = \{m \mid \big(e = (r, s) \in \mathbf{RS} \wedge m \in \mathbf{C}_r\big) \vee \big(e = (r, q, m) \in \mathbf{RQG}\big)\}$$

– множество групп, у которых проводится занятие *e*;

$k_1(e) \in \{0, 5, 10\}$ – характеризует степень важности занятия для специальности (0 – «ненужные занятия», 5 – важное, но не по специальности, 10 – занятие по специальности);

$k_2(e) \in \{0, 2\}$ – 0 – младшие курсы, 2 – старшие курсы;

$p_e = p$, $\delta(e) = p$ – номер преподавателя, который ведет занятие *e*;

$k_3(p_e) \in \{0\ldots5\}$ – оценка преподавателя как научного сотрудника; больше – значит, что преподаватель более ценный научный сотрудник. В случае равенства занятия упорядочиваются в лексикографическом порядке.

Предпочтения пользователей. Важная часть предлагаемой модели — предпочтения пользователей. Каждое предпочтение — это числовая оценка в диапазоне от 0 до 1. 0 соответствует наименьшему уровню предпочтения; 1 — наибольшему. В модели имеют место два вида предпочтений:

- ✧ предпочтения пользователя $m \in \mathbf{M}$ о времени проведения занятия:

$$f_m^1 : \mathbf{E}_m \times \mathbf{T}_m^+ \to [0,1] \qquad (11)$$

- ✧ предпочтения пользователя $m \in \mathbf{M}$ о месте проведения занятия:

$$f_m^2 : \mathbf{EA}_m \to [0,1], \qquad (12)$$

где: $\mathbf{EA}_m = \{(e,a) \mid a \in \mathbf{A}_e \wedge e \in \mathbf{E}_m\}$

— множество допустимых пар «занятие — аудитория».

Графически предпочтения можно представить в виде таблиц (*см. табл. 1, 2*). Более тёмный цвет отмечает более предпочтительное время (место) для проведения занятия.

Таблица 1

Предпочтения пользователя *m* о времени проведения занятия, f_m^1

		Номер пары, *j*							
		1	2	3	4	5	6	7	8
	ПН								
	ВТ								
	СР								
	ЧТ								
	ПТ								
	СБ								
	ВС								

Таблица 2

Предпочтения пользователя *m* о месте проведения занятия, f_m^2

Аудитория, a	1	2	3	4	5
Оценка, f_m^2					

Критерий качества решения. Критерий качества — обобщённый критерий, состоящий из нескольких частных критериев — оценивает найденное решение, т.е. пару функций .

Частный критерий

$$F_e^1(\tau) = \sum_{m \in \mathbf{M}_e} f_m^1\big(e, \tau(e)\big) \to \max \qquad (13)$$

определяет сумму предпочтений пользователей о времени занятия *e*. Частный критерий

$$F_e^2(\alpha) = \sum_{m \in \mathbf{M}_e} f_m^2\big(e, \alpha(e)\big) \to \max \qquad (14)$$

определяет сумму предпочтений пользователей о месте проведения занятия *e*. Обобщённый критерий

$$F(\tau, \alpha) = \sum_{e \in \mathbf{E}} \big(F_e^1 + F_e^2\big) \to \max \qquad (15)$$

в начале максимизирует пару частных критериев для наиболее приоритетного занятия, затем для следующего по приоритету и т.д. В паре критериев вначале максимизируется первый критерий (время), затем второй (место). Такая структура гарантирует, что приоритетные занятия получат лучшее время и место. Считается, что время проведения занятия важнее при составлении расписания, поэтому вначале определяется время, а затем место проведения занятия.

Решение. Решением задачи являются неизвестные функции при условии выполнения всех ограничений и максимальном значении критерия качества.

Анализ возможностей многоагентного подхода

В информатике и программной инженерии агент — самостоятельный объект системы, обладающий свойством коммуникабельности и наделённый собственной системой принятия решений. Коммуникабельность — способность агента обмениваться сообщениями с другими агентами и прочими объектами системы.

Суть применения агентного подхода к какой-либо задаче: задача разбивается на несколько более мелких задач. Для решения каждой мелкой задачи выделяется агент. Цель агента — найти решение своей задачи такое, чтобы оно согласовалось с решениями других агентов. Агенты добиваются согласования друг с другом путем обмена информационными сообщениями. В итоге агенты находят решения для своих задач, значит и для исходной задачи; либо выясняется, что решения нет.

Идея применения агентного подхода к задаче составления табличного расписания состоит в том, чтобы заменить агентами каждого пользователя расписания. Цель каждого агента — найти такое расписание для пользователя, чтобы оно:

- ✧ соответствовало ограничениям задачи и ограничениям самого пользователя;

- ✧ имело бы наибольший уровень пользы для пользователя;
- ✧ не противоречило расписаниям других пользователей.

Все агенты запускаются в рамках некоторого контейнера (агентной системы). Агенты выполняют локальные вычисления, обмениваются сообщениями, что в итоге приводит к решению задачи, то есть построению расписания занятий.

Основные достоинства агентного подхода (по сравнению с классическими централизованными методами решения задачи составления расписаний) представлены в *табл. 3*.

В идеале пользователь расписания может один раз запрограммировать своего агента-представителя,

Таблица 3

Преимущества агентных алгоритмов по сравнению с централизованными

Название	Описание
Сильная индивидуальность предпочтений	Каждый пользователь имеет возможность высказать свои предпочтения (в централизованной системе речь идет, как правило, только об абстрактном "общем благе"), может заложить любую программу принятия решений в своего агента
Распределенность	Агент – это достаточно самостоятельный программный модуль. Все агенты могут исполняться на разных компьютерах, объединенных в сеть
Отсутствие предварительного сбора заявок	В общем случае агенты могут без ограничений подключаться к процессу решения задачи. Предварительный сбор информации о предпочтениях каждого агента не требуется
Динамическое изменение предпочтений	Каждый агент может поменять свои предпочтения в любой момент времени. Это приведет к обновлению расписания
Существование частичного решения	В каждый момент времени существует частичное решение задачи

который будет замещать его в дальнейшем по всем вопросам составления расписания проведения занятий и встреч, общаясь напрямую с агентами других агентов. Кроме того, отпадает необходимость в централизованном сборе и хранении предпочтений и пожеланий пользователей расписания, так как все эти данные хранит агент и другим агентам достаточно обратиться к нему с запросом на получение необходимых данных. Пользователь может в любой момент перепрограммировать своего агента и через некоторое время увидеть результат – новое расписание, соответствующее его новым требованиям.

Многоагентные алгоритмы можно разделить не несколько типов.

Первый тип алгоритмов – это алгоритмы, использующие экономическую модель в том или ином виде [1–3]. Как правило, в моделях таких алгоритмов агенты взаимодейтсвуют в рамках некоторого аукциона. Чтобы получить какой-либо ресурс, они должны выиграть торги за этот ресурс у других агентов. Торги осуществляются с помощью «денег». Агенты делятся на продавцов и покупателей. Продавцы контролируют распределение ресурсов; покупатели пытаются, используя свои ограниченные денежные ресурсы, максимизировать свою функцию полезности путем приобретения у продавцов ресурсов. Такие алгоритмы предназначены для применения в автоматических электронных аукционах и электронной коммерции. Основной недостаток таких алгоритмов – в необходимости адаптации исходной задачи к рыночной модели, что не всегда удаётся сделать.

Второй тип алгоритмов предлагает методы для решения распределённых задач удовлетворения ограничений (Distributed Constraint Satisfaction Problem - DisCSP) [4, 5]. Обычная нераспределённая задача удовлетворения ограничений (Constraint Satisfaction Problem – CSP) состоит из n переменных x_1, $x_2, ..., x_n$, чьи значения берутся из конечных дискретных множеств $D_1, D_2, ..., D_n$ и набора ограничений на переменные. Ограничение задается предикатом. То есть ограничение $p_k(x_{k1}, x_{k2}, ..., x_{kj})$ – предикат, определ`нный на декартовом произведении $D_{k1} \times D_{k2} \times D_{kj}$. Этот предикат истинен тогда и только тогда, когда все его переменные удовлетворяют его ограничению. Решить CSP означает найти значения всех переменных такие, что все ограничения выполнены.

Распределенная CSP (DisCSP) – это CSP, в которой переменные и ограничения распределены между автоматическими агентами. Мы предполагаем, что имеет место следующая модель общения между агентами:

- ✧ агенты общаются путём посылки сообщений. Агент может посылать сообщения другим агентам тогда и только тогда, когда он знает адреса других агентов;
- ✧ время доставки сообщения – конечное случайное число. Передача сообщений между любыми двумя агентами идет последовательно, т.е. сообщения получаются именно в том порядке, в котором они были посланы.

Каждый агент имеет несколько переменных и его задача – определить значения этих переменных. Но существуют межагентные ограничения и значения переменных должны им тоже удовлетворять. Формально, существует m агентов 1, 2, ..., m. Каждая переменная принадлежит одному агенту i (это отношение выражается предикатом $belongs(x_j, i)$). Если x_j принадлежит агенту i, мы можем назвать x_j *локальной переменной* агента i. Ограничения также, как и переменные, распределены между агентами. Факт того, что агент k знает предикат ограничения p_l представлен с помощью отношения $known(p_l, k)$. Ограничение, заданное только на локальных переменных, будем называть *локальным ограничением*.

Мы говорим, что DCSP решена тогда и только тогда, когда выполнены следующие условия:

$\forall i$, $\forall x_j$, $belongs(x_j, i)$ значение x_j равно d_j и $\forall k$, $\forall p_l$, $known(p_l, k)$, $p_l = true$.

Без потери общности мы делаем следующие предположения. Эти ограничения достаточно просто могут быть сняты при рассмотрении общего случая:

- Каждый агент знает все предикаты ограничений, касающиеся его переменных.
- Все ограничения двоичные, т.е. определены на двух переменных.

Мы можем представить DCSP с двоичными ограничениями как сеть, где переменные являются вершинами, а ограничения – ребрами (*рис. 1*).

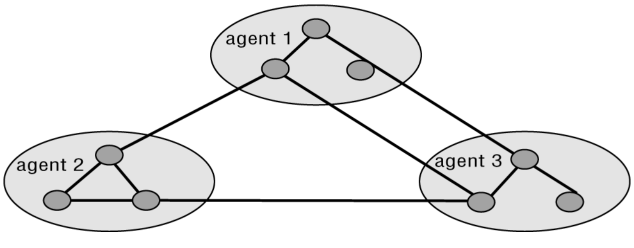

Рис. 1. Пример сети ограничений

Агент может быть представлен как множество переменных, которое обведено на рисунке кругом.

Идея методов решения задач DisCSP заключается в применении распределённых алгоритмов поиска в пространстве решений задачи [4, 5]. Наиболее эффективный алгоритм, способный обрабатывать случай нескольких локальных переменных, предложен в работе [4] Это Multi-AWS (Multi Asynchronous Weak-commitment Search – асинхронный алгоритм со слабой фиксацией) – полный распределённый алгоритм эвристического поиска. Универсальность предложенных алгоритмов приводит к тому, что они не используют дополнительной информации о модели решаемой задачи, что приводит к увеличению перебора в пространстве решений.

Наиболее эффективными алгоритмами оказываются алгоритмы третьего типа, использующие всю информацию о структуре конкретной задачи и изначально разрабатывались для её решения. В работах [6–9] предлагаются решения задачи многагентного составления расписания встреч. Алгоритмы в этих статьях хорошо вписываются в модель составления расписания, и имеют хорошие оценки производительности.

Выводы

Построенная математическая модель – основа для практической реализации многоагентного алгоритма составления расписания учебного заведения. Анализ существующих методов многоагентной оптимизации и планирования показал: наибольший интерес представляет разработка алгоритмов, основанных на парадигме динамического планирования расписания встреч многих участников, предложенной в [9]. ■

Литература

1. Cheng J.Q., Wellman M.P. The WALRAS Algorithm: A Convergent Distributed Implementation of General Equilibrium Outcomes // Journal of Computational Economics, 12. 1998. С. 1–23.
2. Wellman M.P., Walsh W.E., Wurman P.R., MacKie-Mason J.K. Auction Protocols for Decentralized Scheduling // Games and Economic Behavior 35. 2001. С. 271–303.
3. Walsh W.E., Yokoo M., Hirayama K., Wellman M.P. On Market-Inspired Approaches to Propositional Satisfiability. Proceedings of Int. Conf. IJCAI-2000. 2001. С. 1152–1160.
4. Yokoo M., Hirayama K. Distributed Constraint Satisfaction Algorithm for Complex Local Problems. Proceedings of the Int. Conf. ICMAS-98. 1998. С.372–379.
5. Yokoo M., Hirayama K. Algorithms for Distributed Constraints Satisfaction: A Review. Proceedings of Int. Conf. AAMAS-02. Vol.3. No.2. 2000. С. 185 – 207.
6. Garrido L., Sycara K. Multi-Agent Meeting Scheduling: Preliminary Experimental Results. Proceedings of the Int. Conf. ICMAS-96. Kyoto, Japan.1996.
7. Franzin M.S., Freuder E.C., Rossi F., and Wallace R. Multiagent Meeting Scheduling with Preferences: Efficiency, Privacy Loss, and Solution Quality. Proceedings of Intrl. Conf. AAAI-02 (workshop on preference in AI and CP). 2002.
8. BenHassine A., Ito T., Ho T.B. A New Distributed Approach to Solve Meeting Scheduling Problems. Proceedings of IEEE/WIC Int. Conf. IAT, 2003. С. 588–591.
9. BenHassine A., D?efago X., Ho T.B. Agent-Based Approach to Dynamic Meeting Scheduling Problems. Proceedings of Int. Conf.AAMAS-04.V.3. 2004. С. 1132–1139.