СОВРЕМЕННЫЕ ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

УДК 004.89, 004.832.2 ISSN 1995-5499
DOI: https://doi.org/10.17308/sait/1995-5499/2022/4/156-179
Поступила в редакцию 08.07.2022
Подписана в печать 05.12.2022

ПАРАДИГМА ПРОГРАММИРОВАНИЯ В ОГРАНИЧЕНИЯХ ПРИ РЕШЕНИИ ЗАДАЧ СОСТАВЛЕНИЯ РАСПИСАНИЙ: АНАЛИТИЧЕСКИЙ ОБЗОР

© 2022 А. А. Зуенко, О. В. Фридман✉, О. Н. Зуенко

Институт информатики и математического моделирования – обособленное подразделение ФИЦ «Кольский научный центр Российской академии наук»
ул. Ферсмана, 24А, 184209 Апатиты, Мурманская область, Российская Федерация

Аннотация. В статье рассмотрен подход к решению задач составления расписаний на основе парадигмы программирования в ограничениях. Любой метод удовлетворения ограничений включает две обязательные части: a) компоненту, отвечающую за поиск; b) компоненту, осуществляющую вывод на ограничениях (распространение ограничений). При реализации первой компоненты обычно применяется определенный вариант поиска в глубину с возвратами с использованием эвристик для выбора преемника текущего узла в дереве поиска. Вторая компонента реализуется с применением механизма глобальных ограничений. Глобальные ограничения можно рассматривать как составные ограничения, представляющие собой набор более простых однотипных ограничений. Алгоритмы распространения глобальных ограничений, как правило, подкреплены соответствующими развитыми теориями, позволяющими организовывать высокопроизводительные вычисления. Технология программирования в ограничениях позволяет реализовывать общие методы решения сложных комбинаторных задач, а также дает возможность интегрировать с помощью механизма глобальных ограничений существующие методы теории исследования операций, направленные на решение узких классов задач. При решении задач составления расписаний основные ограничения могут быть типизированы следующим образом: ограничения на порядок выполнения операций, ограничения на унарные (дистрибутивные) ресурсы, ограничения на кумулятивные ресурсы. В настоящей работе произведен обзор основных классов задач составления расписаний и сделан акцент на типах ограничений, которые в них используются. Для решения упомянутых задач могут быть использованы методы удовлетворения ограничений общего назначения, однако более перспективным считается применение специализированных эвристик и глобальных ограничений. В работе произведен обзор наиболее популярных эвристик, используемых при решении задач составления расписаний, а также глобальных ограничений таких, как: глобальное ограничение на дизъюнктивные ресурсы, глобальное ограничение на кумулятивные ресурсы, глобальное ограничение на мощность множеств. В статье обсуждаются особенности реализации различных классов задач составления расписаний с использованием современных сред и библиотек программирования в ограничениях.

Ключевые слова: составление расписаний, унарные ресурсы, кумулятивные ресурсы, методы составления расписаний, задача удовлетворения ограничений, распространение ограничений, программирование в ограничениях.

✉ Фридман Ольга Владимировна
e-mail: ofridman@iimm.ru

ВВЕДЕНИЕ

Следует различать два направления исследований: интеллектуальное планирование (*AI-planning, automated planning*) и составление расписаний (*scheduling*). В сообществе искусственного интеллекта (*AI-community*) принято считать, что задача составления расписаний является одной из разновидностей задач интеллектуального планирования [1].

Методы планирования и составления расписаний в последнее время значительно продвинулись благодаря применению моделей и инструментов удовлетворения ограничений [2].

Составление расписаний — это задача назначения множества операций (действий) множеству рабочих ресурсов, на которые (как на операции, так и на ресурсы) накладывается ряд ограничений, но последовательность действий, в отличие от задач планирования, заранее известна и требуется отыскать лишь их порядок и распределение по рабочим ресурсам. Расписание определяет, кто и чем занимается, как правило, с привязкой ко времени.

В зависимости от ситуации, ресурсами могут быть машины, люди, взлетно-посадочные полосы, процессоры и т. д., действия могут быть производственными операциями, обязанностями, посадками и взлетами, компьютерными программами и т. д., а в качестве целей могут выступать: минимизация длины расписания, максимизация использования ресурсов, минимизация задержек и другие. Известно, что многие задачи составления расписаний являются NP-трудными. В настоящее время повышенное внимание уделяется аппроксимационным и стохастическим алгоритмам, которые могут дать некоторые решения даже для сложных задач. В теории исследования операций основное внимание почти всегда уделяется решению конкретной задачи составления расписаний или определению ее сложности, а не разработке общего подхода к составлению расписаний. В результате разработано огромное количество алгоритмов составления расписаний для большого количества конкретных задач. Это сильно отличает составление расписаний в рамках традиционного подхода от подхода к составлению расписаний с использованием технологии программирования в ограничениях, где основное внимание уделяется разработке общих методов решения сложных комбинаторных задач, а не специальных методов для конкретных задач.

Программирование в ограничениях (*Constraint Programming* — CP) — мощная парадигма для решения комбинаторных задач. CP зародилась как междисциплинарная область исследований, включающая в себя методы и понятия из многих других областей, среди которых важную роль играют искусственный интеллект, компьютерные науки, базы данных, программирование [3, 4]. CP в настоящее время успешно применяется во многих областях, таких как планирование, составление расписаний, конфигурация сети и биоинформатика [5–7].

1. ЗАДАЧИ СОСТАВЛЕНИЯ РАСПИСАНИЙ

В работе [8] введены некоторые базовые термины теории расписаний. Как правило, задача составления расписаний состоит из набора n заданий (требований), состоящих из операций (или иначе — работ), которые могут выполняться на m машинах (приборах). Каждая операция j требует некоторого времени обработки p_{ij} на конкретной машине i. Время запуска операции j может быть ограничено r_j — самым ранним временем, когда операция j может начать выполняться. Для операции j может быть указан крайний срок d_j, то есть самое позднее время, к которому операция j должна быть выполнена. Операция может завершиться позже (или раньше) срока, но тогда могут быть наложены штрафы [9]. Задача составления расписаний состоит в том, чтобы найти расписание, удовлетворяющее определенным ограничениям и оптимизирующее заданную целевую функцию.

Между операциями могут быть установлены ограничения приоритета, которые выражают тот факт, что одни операции должны быть завершены до того, как некоторые другие операции смогут быть обработаны. В наи-

более общем случае, ограничения предшествования представляются в виде ориентированного ациклического графа, каждая вершина которого представляет операции и, если операция i предшествует операции j, то существует направленная дуга от i до j.

Каждая операция требует определенных ресурсов для своего выполнения. Операции могут быть предварительно назначены на определенные ресурсы, и тогда необходимо найти только время обработки. Может быть несколько альтернативных ресурсов, с использованием которых производится операция, выделение ресурсов может рассматриваться как часть задачи составления расписаний. Такие альтернативные ресурсы либо идентичны, либо объединены в так называемый *кумулятивный* ресурс, который может обрабатываться несколькими операциями параллельно. Ресурс, который может выполнять не более одной операции в любой момент времени, называется *унарным* или *дизъюнктивным* ресурсом. Если обработка операции на машине может быть прервана на выполнение другой операции, а затем возобновлена, возможно, на другой машине, то операция называется *вытесняемой*. В работе [10] проведен анализ методов учета возобновляемых и невозобновляемых ресурсов при календарном планировании.

Поскольку в последнее время для решения задач составления расписаний активно применяются модели и инструменты удовлетворения ограничений [2], перейдем к описанию их особенностей.

2. МЕТОДЫ УДОВЛЕТВОРЕНИЯ ОГРАНИЧЕНИЙ, ПРИМЕНЯЕМЫЕ ПРИ СОСТАВЛЕНИИ РАСПИСАНИЙ

При решении задач составления расписаний с использованием парадигмы программирования в ограничениях успех в реальных приложениях достигается благодаря объединению преимуществ двух направлений, а, именно: исследования операций (ИО) и искусственного интеллекта (ИИ). Традиционный подход к составлению расписаний, применяемый в теории исследования операций, фокусируется на использовании комбинаторной природы относительно простой математической модели задачи составления расписаний. Это приводит к высокому уровню эффективности при решении подобных задач. Недостатком данного подхода является то, что при отображении реальной задачи на математическую модель обычно необходимо отказаться от некоторых степеней свободы и принять упрощающие допущения. Отбрасывание степеней свободы может исключить некоторые решения, а отказ от дополнительных ограничений может привести к неприемлемым решениям [8].

Подход ИИ традиционно фокусируется на общих методах решения задач, поэтому все степени свободы и дополнительные ограничения сохраняются, что может привести к низкой производительности по сравнению с традиционными алгоритмами решения. CP обеспечивает хорошую основу для интеграции методов исследования операций и более общих схем решения комбинаторных задач. Подобная интеграция основана на понятии *глобального ограничения*. Глобальные ограничения инкапсулируют определенную часть задачи удовлетворения ограничений и, вместо того, чтобы использовать набор ограничений для моделирования конкретной подзадачи используется специальное «более крупное» глобальное ограничение, которое позволяет лучше анализировать структуру подзадачи. Глобальные ограничения вместе со сложными методами поиска являются ключевыми для достижения успеха при составлении расписаний в рамках парадигмы программирования в ограничениях. Глобальные ограничения обеспечивают эффективные алгоритмы для решения четко определенных подзадач. Их можно комбинировать с другими ограничениями, моделирующими некоторые дополнительные особенности задачи.

При использовании парадигмы CP любая решаемая задача должна быть представлена как задача удовлетворения ограничений (Constraint Satisfaction Problem — CSP).

Согласно [6] CSP заключается в поиске решений для сети ограничений (*Constraint Network*). Сеть ограничений задается тремя ком-

понентами: $< X, D, C >$: X — множество переменных $\{X_1, X_2, \ldots, X_n\}$, D — множество доменов переменных $\{D_1, D_2, \ldots, D_n\}$, C — множество ограничений $\{C_1, C_2, \ldots, C_m\}$, которые регламентируют допустимые сочетания значений переменных. Каждый домен D_i описывает множество допустимых значений $\{v_1, \ldots, v_k\}$ для переменной X_i.

Ограничение C_j со схемой $S_j = \{X_{j_1}, X_{j_2}, \ldots, X_{j_k}\} \subseteq X$ будем обозначать $C_j[S_j]$.

Решением задачи CSP является полное присваивание, которое удовлетворяет всем ограничениям. В некоторых случаях необходимо получить все решения. Иногда требуется найти такое решение, в котором значения переменных оптимизировали бы некоторый заданный функционал.

Опишем базовую модель ограничений задачи составления расписаний. Задания обычно состоят из связанных во времени операций (работ). Для каждой операции (работы) вводятся три переменные, указывающие ее положение во времени, а именно время начала, время окончания, и время обработки (выполнения) т. е. длительность.

Для операции A обозначим эти переменные как $start(A)$, $end(A)$ и $p(A)$. Домены этих переменных должны быть дискретными (например, натуральные числа, представляющие время), где r_j и d_j представляют собой границы этих доменов. Можно дополнительно ограничить домен, предполагая временные окна, когда операция может быть обработана (временные окна — типичный пример дополнительного ограничения). Часто считается, что время обработки операции является постоянным числом, поэтому времени начала операции достаточно, чтобы полностью указать распределение операций во времени, но предпочтительно использовать все три переменные. Для операций без прерываний три вышеуказанные переменные связывает следующее ограничение: $start(A) + p(A) = end(A)$ [8]. В [11] подробно описаны более сложные задачи, в которых допускаются прерывания операций. Если распределение ресурсов является частью задачи составления расписаний, то требуется переменная для описания того, какой ресурс будет обрабатывать операция: $resource(A)$. Домен этой переменной состоит из набора чисел, где номера однозначно присвоены ресурсам.

По сути, в модели есть две группы ограничений: временные и ресурсные ограничения. Временные ограничения описывают временные отношения между операциями, такие как отношения предшествования. Отношение, что операция A должна быть выполнена до операции B можно моделировать с использованием ограничения: $end(A) \le start(B)$. Обозначим это ограничение как $A \ll B$. Это ограничение легко обобщить для моделирования временной связи с минимальными и максимальными задержками между операциями. Тогда ограничение примет вид:

$$min\_delays\ (A, B) \le start(B) - end(A) \le$$
$$\le max\_delays(A, B).$$

Одной только модели ограничений недостаточно для решения задачи составления расписаний. Модель ограничений должна сопровождаться методом удовлетворения ограничений.

Под методом удовлетворения ограничений будем понимать любой метод, который находит решение задачи CSP или, по крайней мере, сужает область возможных значений переменных.

Основные методы общего назначения для решения задач CSP могут быть разбиты на три класса [12]. Первый класс содержит различные варианты алгоритмов *поиска в глубину с возвратами*, которые строят решение путем расширения частичного присваивания шаг за шагом, используя различные эвристики и применяя разумные стратегии возврата из тупиковых вершин. Ко второму классу относятся алгоритмы *распространения ограничений*, которые исключают из пространства поиска некоторые элементы, не входящие в решение, обеспечивая снижение размерности задачи. Эти алгоритмы не строят сами по себе решение, поскольку исключают не все элементы, не входящие в решение. Они применяются или для препроцессинга задачи до использования алгоритмов другого типа, или перемежаются с шагами алгоритма другого

типа (например, поиска с возвратами) для повышения производительности последнего. Методы распространения ограничений осуществляют вывод значений одних переменных на основе известных значений других переменных. Наконец, *структурные алгоритмы* используют информацию о структуре первичного или двойственного графа ограничений задачи. Алгоритмы этого класса производят декомпозицию исходной задачи CSP на слабо связанные подзадачи.

Любой метод удовлетворения ограничений должен содержать две обязательные части: a) компоненту, реализующую распространение ограничений, и b) компоненту, реализующую некоторую стратегию поиска с использованием эвристик для выбора переменной и ее значения на каждом шаге поиска. Для решения задач составления расписаний можно применять любую стратегию поиска общего назначения. Например:

• простой алгоритм поиска с возвратами (*Backtracking Search* — BT) [13];

• алгоритмы обратного просмотра (*Look-Back Algorithms*), которые выполняют проверку согласованности в обратном направлении (между текущей переменной и уже просмотренными переменными) [14–18].

• алгоритмы прямой проверки (*Forward-checking Algorithms* — FC) [19]. При использовании этих алгоритмов проверяется выполнимость ограничений, и удаляются значения еще не рассмотренных переменных, которые несовместны с экземпляром текущей переменной;

• алгоритмы неполного поиска (*Incomplete Search Algorithms*) [20], которые не гарантируют ни получения всех решений, ни нахождения единственного решения, но их время вычислений может быть намного меньше по сравнению с систематическими методами поиска.

Среди методов распространения ограничений наиболее распространены методы достижения различного рода совместностей (*Consistency Techniques*) [21]. Тот или иной вид совместностей определяет глубину процесса вывода на ограничениях. Наиболее известными методами вынуждения совместностей являются: достижение совместности в вершинах (*Node-consistency*), достижение совместности по дугам (*Arc-consistency*), достижение совместности по путям (*Path-consistency*) [22].

Однако специализированные стратегии поиска и методы распространения ограничений для задач составления расписаний часто дают лучшие результаты. Например, при выборе наиболее перспективной ветви дерева поиска вместо присваивания переменной конкретного значения, могут быть применены другие стратегии ветвления. Поскольку составление расписаний в основном заключается в поиске последовательности операций, ветвление обычно основано на способе выбора следующей операции для обработки. В [23] представлены и эмпирически оценены новые эвристики для решения задачи составления расписаний для рабочих мест с нерелаксируемыми временными окнами.

Теперь опишем методы распространения ограничений, используемые при составлении расписаний.

Обычно сеть временных ограничений разрежена и для распространения используется алгоритм *arc-B-consistency* [13]. Для плотных сетей более целесообразно использовать алгоритмы вынуждения совместности по дугам (*Arc Consistency*).

Две операции A и B, обрабатываемые на унарном (дизъюнктивном) ресурсе, не могут перекрываться во времени, поэтому либо A предшествует B, либо наоборот: $(A \ll B) \vee (B \ll A)$. Унарный ресурс может быть полностью смоделирован набором таких дизъюнктивных ограничений. К сожалению, дизъюнктивные ограничения плохо распространяются (алгоритмы достижения дуговой совместности не сильно обрезают домены переменных). Как упомянуто выше, набор дизъюнктивных ограничений может быть заключен в одно глобальное ограничение, и тогда может быть достигнуто более глубокое распространение.

В [24] описывается обработка глобальных ограничений, моделирующих унарные ресурсы (*disjunctive global constraint*), которая основывается на технике распространения ограничений с поиском границ (*edge-finding*

technique). В [25] представлен новый квадратичный алгоритм для составления расписаний для дизъюнктивных ресурсов без вытеснения. Этот алгоритм можно использовать в сочетании с традиционным алгоритмом поиска границ, для того, чтобы обновить самое раннее время начала (или самое позднее время окончания) некоторой операции A [25]. Среди алгоритмов распространения ограничений с поиском границ выделяют алгоритмы с временной сложностью не хуже $O(n \times log\, n)$, где n — количество операций [26]. В [27] представлен метод распространения ограничений с поиском границ, который называют «не первый/не последний» (*Not First\Not Last*), в ходе его выполнения делается вывод, что операция не может быть обработана первой или последней.

Существует большое количество методов распространения глобальных кумулятивных ограничений (*cumulative global constraint*). Ниже приводятся некоторые из них [2]:

1) *time table propagation*,

2) *edge finding*, строящий рассуждения о временных интервалах, а не об отдельных дискретных моментах времени,

3) *energy based reasoning*,

4) *time table edge finding* (TTEF).

В [27] предложены алгоритмы распространения для кумулятивных ограничений, которые анализируют отношения приоритета между операциями, а не их абсолютное положение во времени. Они эффективны, даже когда набор операций не полностью определен и когда временное окно велико. В работе [28] предложен алгоритм фильтрации с временной сложностью $O(n \times log\, n)$. В статье [11] показана версия метода поиска границ для случая с кумулятивными ресурсами.

Преимуществом моделей, основанных на ограничениях, является их гибкость при объединении ограничений, описывающих различные аспекты задачи. Кроме того, пользователь может определять любые дополнительные ограничения на переменные.

Для программирования в ограничениях используются как отдельные среды программирования с поддержкой специализированных языков ограничений, например, *MiniZinc*, *IBM ILOG CPLEX*, *ECLiPSe Constraint Programming System*, так и подключаемые библиотеки на обычных языках программирования, например, *Google Optimization Tools* (*C++*, *Python*, *.Net, Java*), *Choco* (*Java*), *Gecode* (*C++*). В рамках библиотек программирования в ограничениях, как правило, реализованы специализированные глобальные ограничения для различных формулировок задач составления расписаний. Также некоторые инструменты дискретной оптимизации, в частности *Google Optimization Tools*, содержат специальные программные классы для решения типовых задач теории расписаний.

Теперь рассмотрим классы задач теории расписаний, которые выделены в [29 30], и особенности их моделирования с применением технологии программирования в ограничениях.

3. БАЗОВАЯ ЗАДАЧА СОСТАВЛЕНИЯ РАСПИСАНИЙ

На простом примере сборки автомобилей опишем базовую задачу теории расписаний [28]. Пусть имеются два задания: собрать два автомобиля. Каждое задание состоит из трех операций: установка двигателя (Уд), установка колес (Ук) и проверка (П). Двигатель должен устанавливаться в первую очередь, т.к. в данной модели автомобиля с установленными передними колесами затрудняется доступ к двигательному отсеку, а проверка должна осуществляться в последнюю очередь. Таким образом, операции упорядочены. Необходимо определить времена начала и конца каждой операции, с учетом заданной продолжительности операций, а именно, составить расписание выполнения операций, такое, чтобы суммарное время их выполнения было минимальным.

На рис. 1 представлены отношения предшествования, описывающие условия задачи. Каждый блок соответствует операции, в нижней части каждого блока приведена продолжительность каждой операции, заданная заранее.

Для поиска оптимального расписания в данной задаче можно применить метод *критического пути* (*Critical Path Method* — CPM).

Путем называется линейно упорядоченная последовательность операций, начинающаяся в состоянии *Start* и оканчивающаяся в состоянии *Finish* (на рис. 1, есть два пути). На рис. 1 критический путь показан жирными линиями.

Операции, лежащие вне критического пути, имеют определенный резерв времени – временное окно, в течение которого они могут быть выполнены.

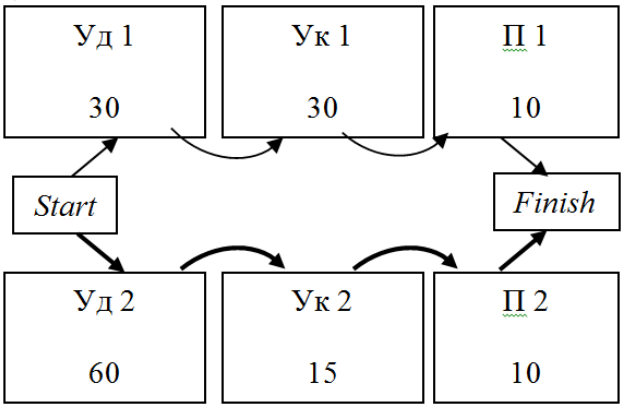


Рис.1. Отношения предшествования
[Fig.1. Precedence Relations]

На рис. 2 показано решение задачи.

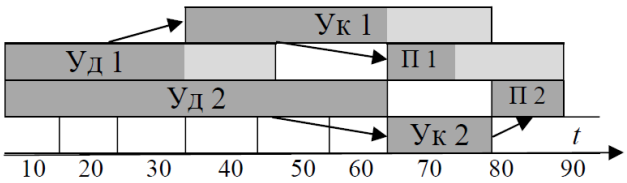


Рис.2. Решение задачи
[Fig.2. The solution of the problem]

На рис. 2 прямоугольники показывают интервалы времени, в течение которых может быть выполнена некоторая операция, при условии, что соблюдаются ограничения упорядочения. Более светлая часть прямоугольника обозначает резерв времени. Более темная соответствует конкретному варианту выполнения операции в рамках временного окна. Как показано на рис. 2, для выполнения всех операций потребуется 85 минут.

Средства моделирования отношений предшествования между операциями (заданиями) в рамках различных библиотек программирования в ограничениях могут сильно отличаться. Например, в пакете *Google OR tools* [31] их предлагается моделировать с помощью глобального ограничения *LinearExp*, поскольку там отсутствует специализированное глобальное ограничение для рассматриваемого случая. А в среде *IBM CPLEX* [32], подобные ограничениями на порядок выполнения для пары операций моделируются с помощью глобального ограничения *EndBeforeStart*.

В следующих разделах будут рассмотрены более сложные задачи теории расписаний, в которых анализируются не только ограничения предшествования, но и ограничения на используемые ресурсы.

Выделяют задачи составления расписаний с дизъюнктивными (унарными) ресурсами, а также задачи с кумулятивными ресурсами. Остановимся подробнее на реализации этих понятий в рамках парадигмы программирования в ограничениях. Для обработки этих видов ресурсов используются соответствующие глобальные ограничения: глобальное дизъюнктивное ограничение (*disjunctive global constraint*) и глобальное кумулятивное ограничение (*cumulative global constraint*).

Ниже представлены два предиката, записанные с помощью псевдокода и служащие для описания глобального дизъюнктивного ограничения. Первый из данных предикатов — предикат *nonoverlap*:

bool nonoverlap(int:s_1*, int:*p_1*, int:*s_2*, int:* p_2*)*
begin
  return $((s_1 + p_1 \le s_2) \vee (s_2 + p_2 \le s_1))$*;*
end.

Он предписывает, что для пары заданий (операций) интервалы времени, когда данные задания (операции) исполняются, не могут пересекаться. Причем, здесь каждое из двух заданий (операций) представлено двумя параметрами: временем начала — s_i и продолжительностью исполнения задания (операции) — p_i.

Само глобальное ограничение *disjunctive global constraint* запрещает подобное пересечение для каждой пары заданий (операций) из рассматриваемого набора:

bool disjunctive(array of int: s, array of int: p)
begin
  return forall(i_1, i_2 *in index(*s*)* *where* $i_1 < i_2$*)*
nonoverlap $(s[i_1], p[i_1], s[i_2], p[i_2])$*;*
end;

Массивы s и p содержат моменты начала и продолжительности заданий (операций) из рассматриваемого набора.

Для глобального кумулятивного ограничения соответствующий предикат приводится ниже:

bool cumulative(array of int: s, array of int: p, array of int: r, int: L)
begin
return forall(i_1 *in index(s) where* $i_1 < i_2$)
sum(i_2 *in index(s))*(((*s*[i_1] ≤ *s*[i_2])
∧(*s*[i_1] + *p*[i_1] > *s*[i_2])) * *r*[*t*]) ≤ *L*;
end.

Предикат формализует тот факт, что во время исполнения каждого из заданий (операций), которые описываются массивами s и p, суммарно используемое всеми заданиями (операциями) количество единиц ресурса не должно превышать порог L. При этом, i-й элемент массива r задает количество ресурса, необходимое для выполнения i-го задания (операции).

Программные функции (процедуры), реализующие описанные глобальные ограничения, как правило, основываются на том или ином методе распространения ограничений, описанном в предыдущем разделе. Другими словами, реализуют определенный метод редукции доменов переменных (достижения совместности).

В различных средах и библиотеках программирования в ограничениях глобальные дизъюнктивные и кумулятивные ограничения так же, как и в случае с ограничениями на порядок выполнения заданий (операций), могут быть реализованы различными способами.

Для случая унарного ресурса в одних библиотеках, например в *Google OR Tools for Java*, глобальное ограничение может соответствовать предикату *nonoverlap*, в других же библиотеках — предикату *disjunctive*, как, в частности, в *MiniZinc* [33]. Глобальное кумулятивное ограничение, как правило, во всех инструментальных средствах, соответствует описанному выше предикату *cumulative*.

В дальнейшем при рассмотрении отдельных классов задач теории расписаний будут сделаны акценты на том, какого типа ресурсы в них используются. Примеры решения задач, в основном, приводятся для среды MiniZinc, которая позволяет описывать задачи на метаязыке ограничений и в дальнейшем транслировать это описание в код на конкретном языке программирования.

4. ПОСТРОЕНИЕ РАСПИСАНИЙ ДЛЯ ПРОЕКТА

Рассмотрим задачу построения расписания выполнения работ проекта с учетом отношений предшествования и ограничений на необходимые/доступные ресурсы (*Resource-Constrained Project Scheduling Problem* — RCPSP) [29].

Пусть дано множество заданий $N = \{1,...,n\}$ и K возобновляемых ресурсов $k = 1,...,K$. В каждый момент времени t доступно L_k единиц ресурса k. Заданы продолжительности обслуживания $p_i \geq 0$ для каждого задания $i = 1,...,n$. Во время обслуживания задания i требуется $r_{ik} \leq L_k$ единиц ресурса $k = 1,...,K$. После завершения обслуживания задания, освобожденные ресурсы в полном объеме могут быть мгновенно назначены на обслуживание других заданий. На некоторые пары заданий накладываются ограничения предшествования: $i \to j$ означает, что обслуживание задания j начинается не раньше окончания обслуживания задания i. Обслуживание заданий начинается в момент времени $t = 0$. Прерывания при обслуживании заданий запрещены. Необходимо определить моменты времени начала обслуживания заданий S_i, $i = 1,...,n$, так, чтобы минимизировать время выполнения всего проекта, т. е. минимизировать значение $C_{\max} = \max\{C_i\}$ $i = 1,...,n$, где $C_i = S_i + p_i$.

Таким образом, основными ограничениями задачи являются:

1) задания в процессе своего обслуживания в каждый момент времени $t \in [0, C_{\max})$ должны быть полностью обеспечены ресурсами;

2) не должны нарушаться отношения предшествования между заданиями.

Рассмотрим конкретный пример задачи RCPSP. Проект состоит из 4-х заданий. Зада-

ния 1–3 выполняются строго последовательно, а задание 4 может выполняться одновременно с любым из первых трех. Всего доступно 4 единицы ресурса. В табл. 1. представлены характеристики заданий.

Таблица 1. Исходные данные для задачи
[Table 1. Initial data for the problem]

№ задания	1	2	3	4
Продолжительность p_i	2	3	4	4
Количество ресурса r_{i1}	2	3	2	2

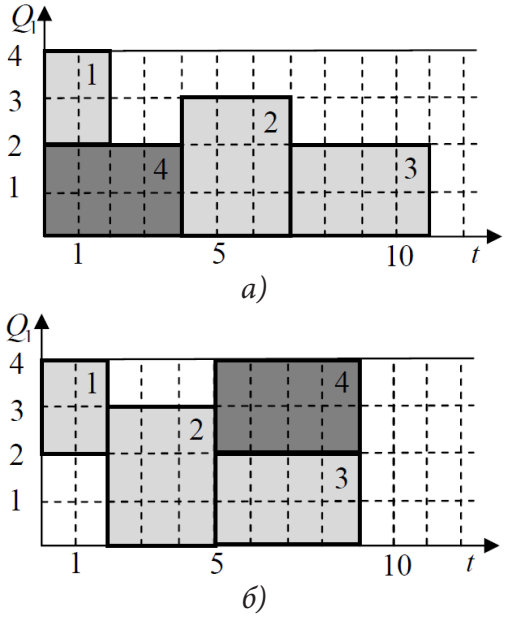


Рис. 3. Пример задачи RCPSP:
а) допустимое расписание — $C_{\max} = 4+3+4 = 11$,
б) оптимальное расписание — $C_{\max} = 2+3+4 = 9$
[Fig. 3. RCPSP example:
a) admissible schedule — $C_{\max} = 4+3+4 = 11$,
b) optimal schedule — $C_{\max} = 2+3+4 = 9$ *]*

На рис. 3а представлено допустимое расписание, при котором соблюдены отношения предшествования, ограничения на ресурсы и продолжительность обслуживания заданий. На рис. 3b показано оптимальное расписание.

При моделировании данной задачи в рамках CP, помимо ограничений на порядок выполнения заданий, используется глобальное кумулятивное ограничение.

На рис. 4 приведен пример описания и решения данной задачи в среде *MiniZinc*.

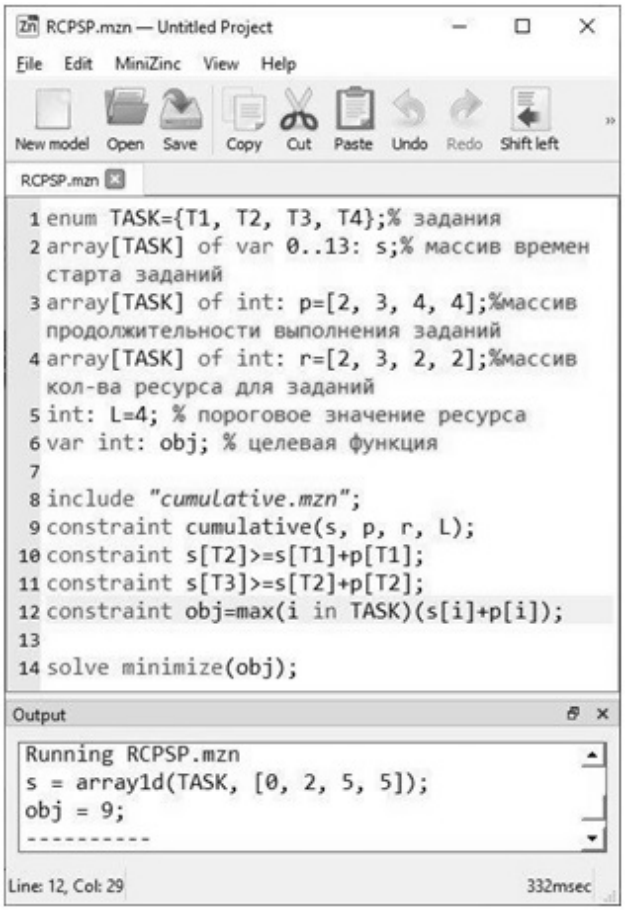


Рис.4. Пример описания и решения задачи RCPSP в среде MiniZinc
[Fig.4. An example of describing and solving the RCPSP problem in the MiniZinc]

Другие примеры задач этого класса рассматриваются в работах [34–37].

5. ПОТОКИ В СЕТЯХ

В рамках сетевого планирования выделяют следующие задачи [29]:

1) построение сетевого графика и расчет его временных характеристик (метод критического пути);

2) расчет вероятностных показателей для трехпараметрической или двухпараметрической сетевой модели;

3) оптимизация стоимости выполнения проекта [38].

К последней категории относятся потоковые задачи:

– задача о максимальном потоке, которая состоит в том, чтобы найти такой поток, для

которого сумма потоков для всей сети будет как можно больше;

– задача минимальной стоимости потока, в которой каждая дуга графа имеет удельную стоимость транспортировки материала по ней. Задача состоит в том, чтобы найти поток с наименьшими суммарными затратами.

В качестве примера задачи рассматриваемого класса приведем задачу проектирования открытых карьеров с максимальной прибылью, которая сводится к поиску максимального потока в сети [39].

Рассмотрим двухмерную блочную модель карьера с заданными значениями прибыли, представленную на рис. 5 [40]. В каждом прямоугольнике значения сверху показывают прибыль, получаемую при извлечении блока, а снизу приводятся обозначения соответствующих булевых переменных задачи.

+1 x_1	+1 x_2	–1 x_3	–1 x_4
	+2 x_5	–1 x_6	

Рис. 5. Двухмерная блочная модель карьера с заданными значениями прибыли
[Fig. 5. 2D block model of a pit with specified profit values]

В рассматриваемом примере безопасные углы наклона считаются равными 45°, поэтому при извлечении, например, блока 5 должны быть извлечены также и блоки 1, 2, 3. Описанные ограничения на порядок выемки блоков представлены на рис. 6.

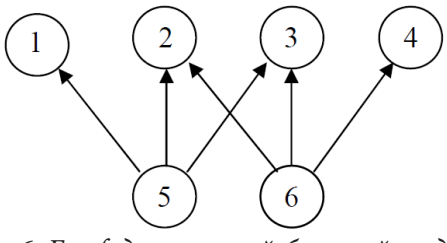


Рис. 6. Граф двухмерной блочной модели
[Fig. 6. Graph of 2D block model]

Требуется принять решение о том, какие блоки следует добывать, чтобы максимизировать прибыль с учетом описанных ограничений.

Задачу максимизации прибыли на языке линейных ограничений можно сформулировать следующим образом:

$$\max(x_1 + x_2 - x_3 - x_4 + 2x_5 - x_6),$$

при условиях:

$$x_5 \le x_1,$$
$$x_5 \le x_2,$$
$$x_5 \le x_3,$$
$$x_6 \le x_2,$$
$$x_6 \le x_3,$$
$$x_6 \le x_4,$$
$$x_i \in \{1,0\}, i = 1\ldots6.$$

Для решения задачи проектирования карьера как потоковой задачи сначала формируется сеть N (рис. 7), связанная с графом карьера, а затем решается задача о максимальном потоке в сети [41].

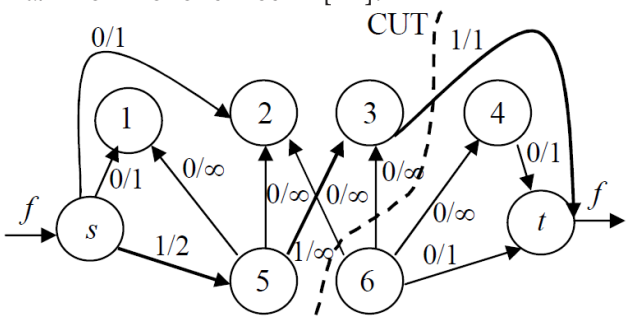


Рис. 7. Сеть для задачи максимального потока
[Fig. 7. Network for the maximum flow problem]

На рис. 7 с истоком s связаны узлы 1, 2, 5, из которых исходят дуги с положительным значением прибыли, а непосредственно к стоку t ведут дуги из узлов 3, 4, 6 с отрицательным значением прибыли.

Решая задачу о максимальном потоке (минимальном разрезе), получаем результат, показанный на рис. 7, где f — общий поток, s — исток (исходный узел), t — узел-приемник, а на каждой дуге показаны оптимальный поток и пропускная способность. Жирными стрелками показано направление потока. Минимальный разрез показан пунктирной линией.

Таким образом, чтобы максимизировать прибыль, при разработке карьера нужно вынуть блоки 1, 2, 3 и 5 как показано на рис. 8.

Максимальная прибыль равна 3, тогда как $p_1 + p_2 + p_5 = 4$, а минимальная пропускная способность равна 1.

+1 x_1	+1 x_2	−1 x_3	−1 x_4
	+2 x_5	−1 x_6	

Рис. 8. Окончательная форма карьера
[Fig. 8. The final design of a pit]

В пакете *Google OR tools* для различных языков программирования представлены программные классы-солверы, позволяющие решать задачу о максимальном потоке, указывая в качестве параметров конкретные наборы вершин и дуг с весами. Так, например, для языка *Java* подобный класс называется *MaxFlow*.

6. КАЛЕНДАРНОЕ ПЛАНИРОВАНИЕ ИЛИ ПОСТРОЕНИЕ РАСПИСАНИЯ ДЛЯ ПРИБОРОВ

В отличие от задач *Project Scheduling*, где для выполнения одной работы может потребоваться одновременное участие нескольких исполнителей, в задачах составления расписаний для приборов (*machine scheduling* — MS) каждое задание обычно выполняется (обслуживается) единовременно только на одной машине (приборе). Различают:

• *Задачи для одного прибора*. Характерной особенностью этих задач является то, что одновременно машина (прибор) может обслуживать (выполнять) только одно задание.

• *Задачи для параллельных приборов*. Для этих задач вместо одного прибора (машины) доступно m приборов (машин) $M_1, M_2, \ldots, M_m$. Между заданиями могут быть заданы отношения предшествования. Каждое задание может выполняться на любом приборе. Если приборы идентичны, то время обслуживания p_j требования j не зависит от выбора машины, на которой задание будет выполнено.

Приведем пример задачи *Machine scheduling* (MS) для *параллельных приборов* [29]. Пусть требуется выполнить восемь заданий на трех идентичных машинах. Различные задания могут выполняться одновременно на разных машинах. Времена обслуживания заданий: $p_1 = 1$, $p_2 = 3$, $p_3 = 4$, $p_4 = 2$, $p_5 = 2$, $p_6 = 3$, $p_7 = 1$, $p_8 = 5$. Необходимо составить оптимальное расписание, не нарушающее перечисленных выше условий.

На рис. 9 представлено оптимальное расписание, для которого максимальное время выполнения всех заданий равно 7.

На рис. 10 приведен пример описания и решения этой задачи среде *MiniZinc*.

В рамках технологии CP, каждая машина в данном случае рассматривается как унарный ресурс, используемый в ходе выполнения заданий, и для его моделирования применяется глобальное дизъюнктивное ограничение.

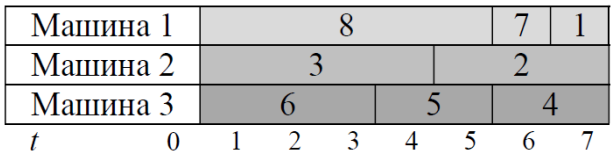


Рис .9. Оптимальное расписание для идентичных машин
[Fig. 9. Optimal schedule for identical cars]

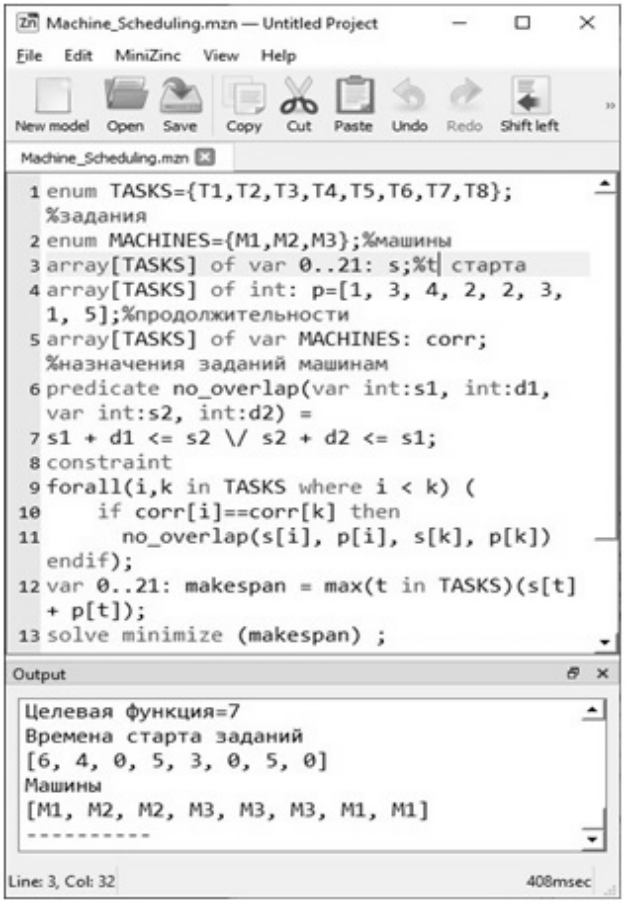


Рис. 10. Пример описания и решения задачи Machine scheduling для параллельных приборов в среде MiniZinc
[Fig. 10. An example of describing and solving the Machine scheduling problem for parallel devices in the MiniZinc]

Помимо идентичных приборов, могут рассматриваться приборы с разной производительностью. Для каждого задания j и машины k может быть задано свое время p_{jk} обслуживания задания j на приборе k. Задачи этого класса представлены в работах [42–44].

7. ЗАДАЧИ ЦЕХА

В задачах цеха (*shop scheduling*) каждое задание (требование) состоит из операций (работ), выполнение которых может назначаться только на определенные машины (приборы). В общем случае дано m машин $M_1, M_2, \ldots, M_m$ и каждое задание j содержит операции $O_{1j}, \ldots, O_{nj}$. Между операциями могут быть заданы отношения предшествования (маршрут обработки детали). Две операции одного и того же задания не могут выполняться одновременно, и каждая машина может выполнять единовременно только одну операцию.

Если задание состоит из цепочки операций, где каждая операция назначена на конкретную машину (на некоторых машинах задание может быть выполнено более одного раза или не выполняться никогда), то говорят о составлении расписаний для рабочих мест (*job-shop scheduling* — JSS). При этом, количество операций у разных заданий может быть различным.

Если машины упорядочены линейно и все операции следуют по одному и тому же маршруту (от первой машины до последней машины) тогда задача называется задача поточного цеха (*flow-shop problem*).

Если из *flow-shop problem*, удалить ограничения приоритета, то есть операции каждого задания могут быть обработаны в произвольном порядке, то получаем задачу открытого цеха (*open-shop problem*).

Приведем пример задачи *Job Shop Scheduling* [31]. Пусть необходимо выполнить три задания, каждое из которых представляет собой набор операций, которые могут быть выполнены тремя машинами. В табл. 2 приведены исходные данные.

Основные ограничения формализуют следующие требования:

Таблица 2. Исходные данные
[Table 2. Initial data]

Задание	Операция	Время обработки	Номер машины
Job_1	O_1	3	1
	O_2	2	2
	O_3	2	3
Job_2	O_4	2	1
	O_5	1	3
	O_6	4	2
Job_3	O_7	4	2
	O_8	3	3

• *Ограничения приоритета*: заключаются в том, что для любых двух последовательных операций в одном и том же задании первая должна быть завершена до того, как можно будет начать вторую. Например, операции O_1 и O_2 являются последовательными для задания Job_1.

• *Ограничения отсутствия перекрытия*: предписывают, что машина не может обрабатывать две операции одновременно. Например, операция O_2 и операция O_6 обрабатываются на машине 2, а следовательно интервалы их обработки не могут перекрываться.

• *Операция, однажды начатая, должна выполняться до завершения.*

В данном случае решением задачи составления расписаний является назначение времени начала для каждой операции, удовлетворяющей приведенным выше ограничениям.

Задача состоит в том, чтобы свести к минимуму промежуток времени от самого раннего времени начала работ до самого позднего времени окончания.

На рис. 11 приведено оптимальное решение.

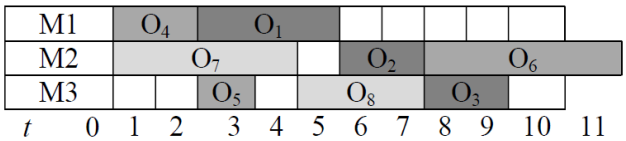


Рис. 11. Оптимальное решение задачи Job Shop Scheduling
[Fig. 11. The optimal solution to the Job Shop Scheduling problem]

Темным цветом помечены операции задания Job_1, более светлым — операции задания Job_2, а самым светлым — операции задания Job_3.

На рис. 12 приведен пример описания и решения задачи *Job Shop* в среде *MiniZinc*.

При моделировании данного класса задач составления расписаний каждая машина рассматривается как унарный ресурс, используемый в ходе выполнения операций, назначенных на определенную машину. При моделировании унарных ресурсов применяется глобальное ограничение *disjunctive*. В случае, если на операциях задан порядок их выполнения, то используются средства для моделирования отношений предшествования [45].

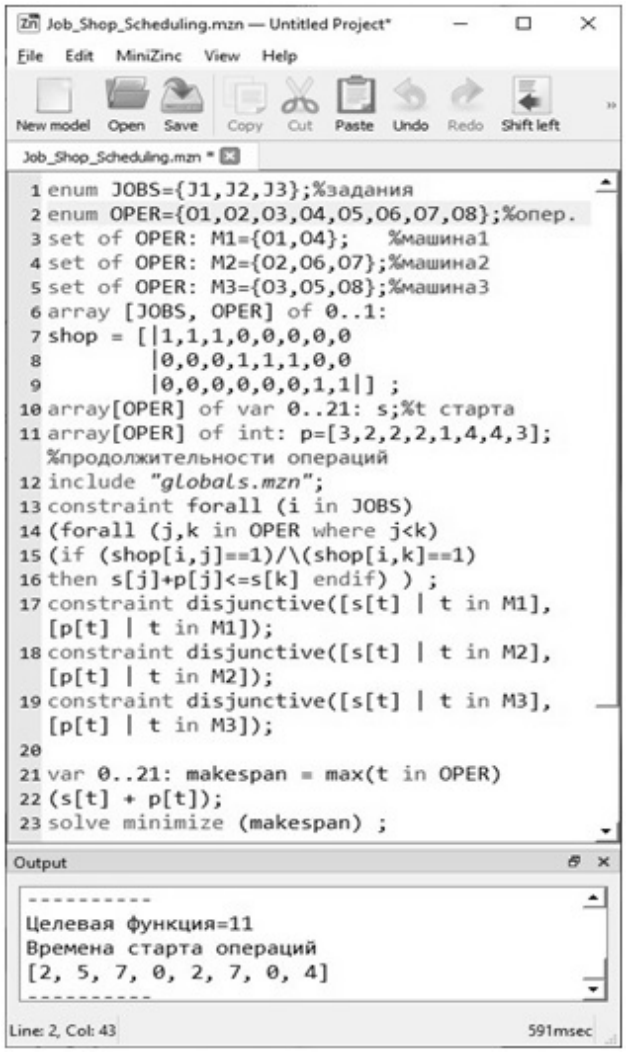


Рис. 12. Пример описания и решения задачи Job Shop в среде MiniZinc
[Fig. 12. An example of describing and solving the Job Shop problem in the MiniZinc]

8. СОСТАВЛЕНИЕ ВРЕМЕННЫХ ТАБЛИЦ

Задачи составления временных таблиц (*Time Tabling*) возникают при планировании занятости персонала, при согласовании времени различных встреч и т. д. Обобщенная задача формулируется следующим образом. Даны множества ресурсов $R_1, R_2, ...R_k$ и множество операций $J_1, J_2, ...J_n$, которые нужно выполнить. Для каждой операции задан набор ресурсов, необходимых для ее выполнения, причем ресурсы могут быть однозначно определены (фиксированные ресурсы), или может быть задано подмножество ресурсов, из которых нужно выбрать фиксированное их количество (свободные ресурсы). Каждый ресурс может быть назначен только на одну операцию в каждый момент времени [46].

Требуется для каждой операции выбрать «свободные ресурсы» и определить время начала ее выполнения. Полученное расписание должно быть или допустимым (не нарушать ограничения на ресурсы) или оптимальным, т. е. допустимым расписанием, при котором минимизирована или максимизирована некоторая целевая функция.

Задачи данного класса еще называют задачами составления графиков (дежурств). Зачастую задачи *Time Tabling* можно свести к задачам *Project Scheduling*.

В отличие от других уже рассмотренных классов задач теории расписаний, здесь в качестве переменных рассматриваются ячейки некоторой таблицы, а в качестве значений переменных — т. е., значения, которые могут располагаться в ячейках. Как правило, все переменные определены на одном домене. Соответственно, условия задачи обычно формулируются как ограничения на количество тех или иных значений в том или ином столбце\строке, части столбца\строки.

Для моделирования и обработки подобных количественных ограничений часто применяется *глобальное ограничение на мощность множеств* (*global cardinality constraint* — GCC) [47]. В некоторых библиотеках программирования в ограничениях данное ограничение может называться *count*, *atleast\atmost*.

Псевдокод, описывающий предикат, соответствующий глобальному ограничению GCC приведен ниже:

bool GCC (array of int: x, array of int: v, array of int: lb, array of int: ub)
begin
return forall(j in index(v))
(sum(i in index(x)) (x[i] == v[j]) ≥ lb[j]) ∧
(sum(i in index(x)) (x[i] == v[j]) ≤ ub[j]);
end.

Первый параметр — это массив переменных, второй — массив тех значений данных переменных, на которые накладывается ограничение кардинальности.

Следующие два массива задают нижние и верхние пороги на встречаемость в векторе решения задачи для значений из второго массива.

Другим глобальным ограничением, часто используемым в рассматриваемых задачах составления временных таблиц, является ограничение *alldifferent*, предписывающее, что все входящие в него переменные попарно должны принимать различные значения.

Псевдокод, описывающий предикат *alldifferent* приведен ниже:

bool alldifferent(array x)
begin
return for all (i, j in index(x) where i < j)
x[i] ≠ x[j];
end.

При использовании данного ограничения все переменные массива x должны принимать различные значения.

Приведем пример задачи [48].

Пусть в течение последних четырех лет работники E_1, E_2, E_3 и E_4 получают очередные отпуска в мае (M), июне (JN), июле (JL) и августе (A). Причем если один из них отдыхает в мае, то другой — в июне, третий — в июле, а четвертый — в августе. Каждый из них получает отпуск в разное время.

В первый год работник E_3 отдыхал в июле, во второй год — в августе, а работник E_1 — в мае. В третий год E_4 отдыхал в июне, а E_2 на четвертый год — в июле. Необходимо узнать время отдыха каждого работника в каждом году.

Перечислим ограничения задачи:

• Ежегодно отдыхают все работники в разные месяцы.

• В разные годы каждый из работников отдыхает в разные месяцы.

В табл. 3 представлены начальные данные и решение задачи.

При моделировании данной задачи в парадигме программирования в ограничениях каждой ячейке тела таблицы сопоставляется переменная, принимающая значение из множества $\{M, JN, JL, A\}$.

Таблица 3. Расписание получения отпусков для четырех работников за четыре года [Table 3. Vacation schedule for four employees for four years]

Y / E	E_1	E_2	E_3	E_4
Y_1	Июнь	Май	**Июль**	Август
Y_2	**Май**	Июнь	**Август**	Июль
Y_3	Июль	Август	Май	**Июнь**
Y_4	Август	**Июль**	Июнь	Май

Основные ограничения задачи могут быть переформулированы следующим образом:

a) все значения переменных, соответствующие строке данной таблицы, должны быть различными;

b) все значения переменных, соответствующие столбцу данной таблицы, должны быть различными.

Данные ограничения моделируются с помощью глобального ограничения *alldifferent*.

Применяя имеющиеся ограничения, получаем решение, представленное в табл. 3, где начальные данные отображены жирным шрифтом.

На практике приходится решать гораздо более трудные и объемные задачи. Примером подобных задач является задача о графике дежурств для медсестер (*nurse scheduling problem*) [49].

В подобных задачах возникают дополнительные условия на порядок дневных, ночных смен, выходных, например: после ночного дежурства должен быть выходной, две смены не могут идти подряд и т. д. Такие условия описываются с помощью глобального огра-

ничения *regular*, которое позволяет задавать детерминированные конечные автоматы [47].

На рис. 13 приведены описание и решение рассмотренной в примере задачи *Time Tabling* в среде *MiniZinc*.

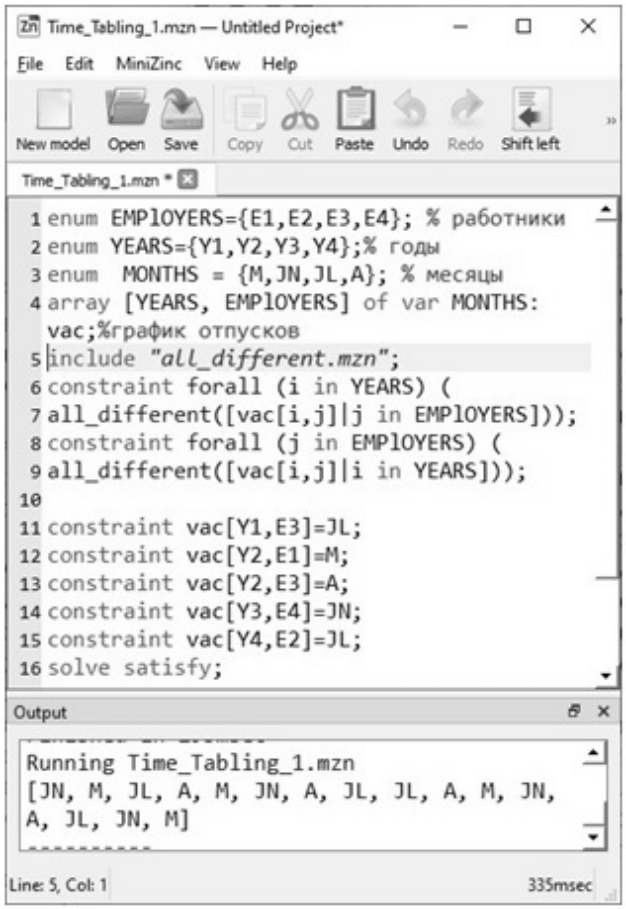


Рис. 13. Пример описания и решения задачи Time Tabling в среде MiniZinc
[Fig. 13. An example of describing and solving the Time Tabling problem in the MiniZinc]

Примеры задач этого класса приведены в работах [50, 51].

9. СОСТАВЛЕНИЕ РАСПИСАНИЙ ДВИЖЕНИЯ ТРАНСПОРТНЫХ СРЕДСТВ

Составление расписаний для транспортных средств (*transport scheduling*) описывает процесс назначения транспортных средств для движения по расписанию.

Целью составления расписаний движения транспортных средств является поиск наилучших маршрутов для парка транспортных средств, посещающих набор местоположений. Обычно под «лучшими» подразумеваются маршруты с наименьшим общим расстоянием или стоимостью. Обычно требуется составить расписание транспортировки, которое должно учитывать ряд ограничений [52]. Самая известная задача маршрутизации — это *задача коммивояжера* (*Traveling Salesperson Problem* — TSP). В задаче коммивояжера рассматривается n городов и матрица попарных расстояний между ними. Требуется найти такой порядок посещения городов, чтобы суммарное пройденное расстояние было минимальным, каждый город посещался ровно один раз и коммивояжер вернулся в тот город, с которого начал свой маршрут. Другими словами, во взвешенном полном графе требуется найти гамильтонов цикл минимального веса. Для формирования оптимального маршрута объезда n городов необходимо выбрать один лучший из $(n-1)!$ вариантов по критерию времени, стоимости или длине маршрута. Задача TSP может быть представлена графом, в котором узлы соответствуют локациям, а ребра (или дуги) обозначают прямое перемещение между локациями. В работах [53–56] описаны различные модификации задачи TSP, здесь мы покажем простейший случай этой задачи.

На рис. 14 показана простейшая задача TSP всего с четырьмя местоположениями, помеченными A, B, C и D. Расстояние между любыми двумя местоположениями определяется числом рядом с соединяющим их ребром.

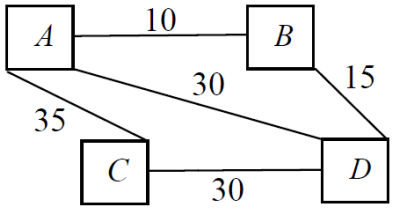


Рис. 14. Схема местоположений в задаче TSP с четырьмя местоположениями
[Fig. 14. Location scheme in the TSP with four locations]

На рис. 15 представлен пример описания и решения задачи коммивояжера в среде *MiniZinc*.

Подсчитав расстояния всех возможных маршрутов, можно увидеть, что кратчайшим

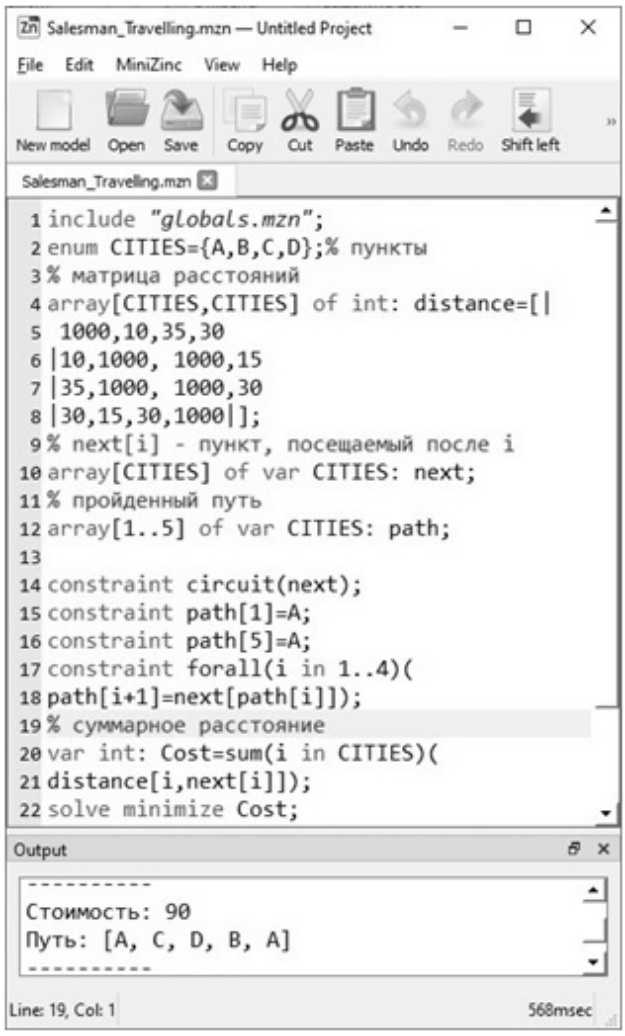


Рис. 15. Пример описания и решения задачи TSP в среде MiniZinc
[Fig. 15. An example of describing and solving the TSP problem in the MiniZinc]

маршрутом является *ACDBA*, для которого общее расстояние равно 35 + 30 + 15 + 10 = 90. Пример более сложной задачи TSP можно найти в [52].

При моделировании задачи коммивояжера, а также других задач маршрутизации транспортных средств, в виде задачи удовлетворения ограничений применяется глобальное ограничение circuit [57].

10. ЦИКЛИЧЕСКИЕ РАСПИСАНИЯ ДЛЯ ТРАНСПОРТНЫХ СРЕДСТВ

В задаче маршрутизации транспортных средств (*Vehicle routing problem* — VRP) требуется оптимизировать маршруты для нескольких транспортных средств, посещающих набор местоположений (когда есть только одно транспортное средство, это сводится к задаче коммивояжера). Под «оптимальными маршрутами» для VRP подразумеваются, например маршруты с наименьшим общим расстоянием. Если требуется завершить все поставки как можно скорее, то для определения оптимального маршрута требуется минимизировать длину самого длинного одиночного маршрута среди всех транспортных средств.

Выделяют следующие классы задач:

• VRP с ограничениями по грузоподъемности, где транспортные средства имеют максимальную грузоподъемность для предметов, которые они могут перевозить.

• VRP с временными окнами, где транспортные средства должны посещать локации в указанные промежутки времени.

• VRP с ограничениями по ресурсам, таким как пространство или персонал для погрузки и разгрузки автомобилей в депо (начальная точка маршрутов).

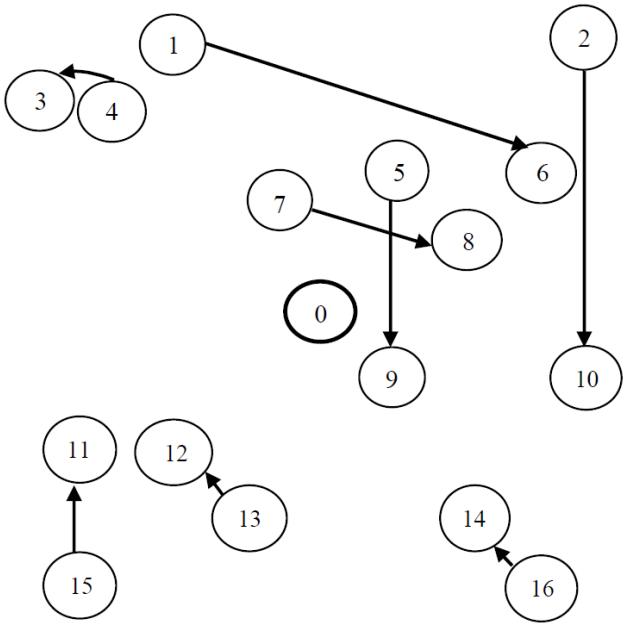
Приведем пример задачи маршрутизации транспортных средств с самовывозом и доставкой (*Vehicle Routing with Pickups and Deliveries*) [58].

Каждое транспортное средство забирает предметы в разных местах и выгружает их в других пунктах. Задача состоит в том, чтобы назначить маршруты для транспортных средств так, чтобы забрать и доставить все предметы, минимизировав при этом длину самого длинного маршрута.

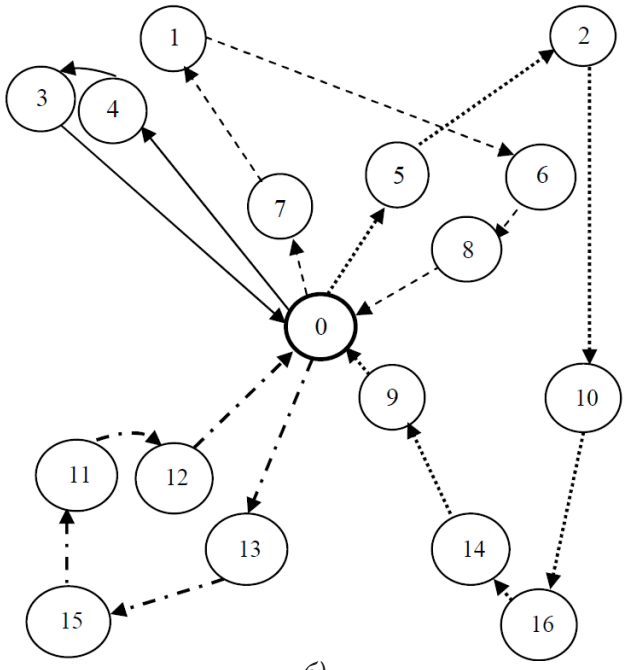
На рис. 16a показана схема города, стрелками показаны связи между узлами, которые должны быть реализованы оптимальным образом.

На рис. 16b представлена схема движения по оптимальным маршрутам для четырех машин.

В пакете *Google OR Tools* имеется специальный класс *RoutingModel*, реализующий решатель для задач построения расписаний для транспортных средств (в документации *Google OR Tools* [49, 52] приводятся различные параметры методов упомянутого класса для описания и решения задач, продемонстрированных в примерах данного подраздела, а также для задачи коммивояжера из предыдущего подраздела).



а)



б)

Рис. 16. Задача VRP с самовывозом и доставкой, а) схема города, б) схема движения по оптимальным маршрутам для четырех машин
[Fig. 16. VRP with pickup and delivery a) city scheme, b) traffic scheme along optimal routes for four cars]

ЗАКЛЮЧЕНИЕ

В статье рассмотрен подход к решению задач составления расписаний на основе парадигмы программирования в ограничениях. Существенным преимуществом использования данной технологии по сравнению с традиционными методами теории исследования операций является то, что набор ограничений задачи может быть легко дополнен новыми типами глобальных ограничений без изменения используемого механизма поиска. Это позволяет с одной стороны избежать излишних упрощений при моделировании задачи, а, с другой стороны, поэтапно усложнять постановку задачи, постепенно дополняя ее новыми деталями (условиями).

В целом, применение технологии программирования в ограничениях позволяет решать практически значимые экземпляры задач составления расписаний.

БЛАГОДАРНОСТИ

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-07-00708а.

КОНФЛИКТ ИНТЕРЕСОВ

Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.

СПИСОК ЛИТЕРАТУРЫ

1. *Ghallab Malik, Nau Dana, Traverso Paolo*. Automated Planning: Theory and Practice. – Cambridge University Press, 2016.

2. *Baptiste Ph., Le Pape C., Nuijten W.* Constraint-based scheduling: applying constraint programming to scheduling problems. – Kluwer Academic Publishers, 2001. – 198 p.

3. *Apt K. R.* Principles of Constraint Programming. – Cambridge University Press. – 2003.

4. *Dechter R.* Constraint Processing. – Morgan Kaufmann. – 2003.

5. *Tsang E.* Foundation of Constraint Satisfaction. – Academic Press. – 1993.

6. *Rossi F., Van Beek P., Walsh T.* Handbook of constraint programming. – Elsevier. – 2006.

7. *Bartak R.* Constraint Satisfaction for Planning and Scheduling // In Vlahavas, Vrakas (eds.): Intelligent Techniques for Planning. – 2005. – P. 320–353.

8. *Roman Bartak, Miguel A. Salido, Francesca Rossi*. Constraint Satisfaction Techniques in Planning and Scheduling. // Constraints. – 2011. – Vol. 16, Issue 3. – P. 223–227. doi: 10.1007/s10601-011-9109-4.

9. *Lazarev A. A., Lemtyuzhnikova D. V., Werner F.* A metric approach for scheduling problems with minimizing the maximum penalty // Applied Mathematical Modelling. – 2021. – Vol. 89, No 2. – P. 1163–1176, doi: https://doi.org/10.1016/j.apm.2020.07.048.

10. *Antonova A. S. Aksyonov K. A.* Analysis of the methods for accounting the renewable and non-renewable resources in scheduling // Journal of Physics: Conference Series. – 2020. – Vol. 1694, Issue 1. 012005, doi: 10.1088/1742-6596/1694/1/012005.

11. *Baptiste P., Laborie P., Le Pape C., Nuijten W.* Constraint-Based Scheduling and Planning // Handbook of Constraint Programming. – 2006. – P. 761–799, doi:10.1016/S1574-6526(06)80026-X.

12. *Ruttkay Z.* Constraint Satisfaction – a Survey // CWI Quarterly. – 1998. – 11(2&3). – P. 123–162.

13. *Lhomme O.* Consistency techniques for numeric CSPs // Proceedings of 13th International Joint Conference on Artificial Intelligence. – 1993. – P. 232–238.

14. *Bitner J. R., Reingold E. M.* Backtracking Programming Techniques. // Communications of the ACM. – 1975. – Vol. 18. – P. 651–655.

15. *Gaschnig J.* Performance measurement and analysis of certain search algorithms. – Carnegie-Mellon University : Technical Report CMUCS, 1979. – P. 79–124.

16. *Prosser P.* Hybrid Algorithm for the Constraint Satisfaction Problem // Computational Intelligence. – 1993. – № 9. – P. 268–299.

17. *Gaschnig J.* A general backtrack algorithm that eliminates most redundant tests // Proccedings of IJCAI. – 1977.

18. *Frost D., Dechter R.* Deadend driven learning // Procceedings of the National Conference on Artificial Intelligence. – 1994. – P. 294–300.

19. *Haralick R., Elliot G.* Increasing Tree Efficiency for Constraint Satisfaction Problems // Artificial Intelligence. – 1980. – № 14. – P. 263–314.

20. *Michalewicz Z., Fogel D. B.* How to Solve It: Modern Heuristics. SpringerVerlag. – 2000.

21. *Roman Barták*. Artificial Intelligence for Advanced Problem Solving. – 2008. – 44 p.

22. *Baptiste P., Le Pape C., Nuijten W.* Constraint-Based Optimization and Approximation for Job-Shop Scheduling // Proceedings of the AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, IJCAI-95. – 1995.

23. *Smith S. F., Cheng Ch.-Ch.* Slack-Based Heuristics For Constraint Satisfaction Scheduling // Proceedings of the National Conference on Artificial Intelligence (AAAI). – 1993. – P. 139–144.

24. *Mackworth A. K.* Consistency in networks of relations // Artificial Intelligence. – 1977. – 8 (1). – P. 99–118.

25. *Baptiste P., Le Pape C., Nuijten W.* Constraint-Based Scheduling. – Springer. – 2001.

26. *Baptiste P., Le Pape C.* Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling // Proceedings of PLANSIG. – 1996.

27. *Vil´ım P., Bartak R., Cepek O.* Extension of O(n log n) filtering algorithms for the unary resource constraint to optional activities // Constraints. – 2005. – № 10(4). – P. 403–425. doi: 10.1007/s10601-005-2814-0.

28. *Torres P., Lopez P.* On Not-First/Not-Last conditions in disjunctive scheduling // European Journal of Operational Research. – 2000. – 127. P. 332–343.

29. *Лазарев А. А., Гафаров Е. Р.* Теория расписаний задачи и алгоритмы. – М. : Московский государственный университет им. М. В. Ломоносова, 2011. – 222 с.

30. *Russel S., Norvig P.* Artificial Intelligence: A Modern Approach. 3rd edition. – Prentice Hall, 2010. – 1132 p.

31. The Job Shop Problem. Режим доступа:https://developers.google.com/optimi-

zation/scheduling/job_shop. (Дата обращения: 01.06.2022).

32. IBM CP LEX. Режим доступа: https://www.ibm.com/analytics/cplexoptimizer. (Дата обращения: 01.06.2022).

33. MiniZinc. Режим доступа: https://www.minizinc.org/doc2.5.5/en/index.html. (Дата обращения: 01.06.2022).

34. *Arkhipov D. I., Lazarev A. A., Tarasov G. V.* Estimating Maximum Resource Load for Resource-Constrained Project Scheduling Problem // Proceedings of the 8th International Conference on Optimization Methods and Applications "OPTIMIZATION AND APPLICATINS" (OPTIMA-2017). Moscow : CC RAS, 2017. – Vol. 1987. – P. 356–363.

35. *Gimadi E. Kh., Goncharov E. N., Shtepa A. A.* A fast algorithm for finding a lower bound of the solution of the Resource-Constrained Project Scheduling Problem tested on PSPLIB instances // Proceedings of the Institute of Mathematics and Mechanics UrO RAN, 2021. – Vol. 27, № 1. – P. 22–36. doi: 10.21538/0134-4889-2021-27-1-22-36.

36. *Nattaf M., Artigues C., Lopez P.* Cumulative scheduling with variable task profiles and concave piecewise linear processing rate functions // Constraints. – 2017. – 22. – P. 530–547, doi: https://doi.org/10.1007/s10601-017-9271-4.

37. *Kreter S., Schutt A., Stuckey P. J.* Using constraint programming for solving RCPSP/max-cal // Constraints. – 2017. – 22. – P. 432–462. doi: https://doi.org/10.1007/s10601-016-9266-6.

38. *Плескунов М. А.* Задачи сетевого планирования: учебное пособие. – Екатеринбург : Изд-во Урал. ун-та, 2014. – 92 с.

39. *Henry Amankwah*. Mathematical Optimization Models and Methods for Open-Pit Mining. LiU-Tryck, Linköping, Sweden, 2011.

40. *Lerchs H., Grossmann I. F.* Optimum Design of Open-Pit Mines, Transactions // Canadian Institute of Mining and Metallurgy. – 1965. – Vol. LXVIII. – P. 17–24.

41. *Picard J.-C., Smith B. T.* Parametric Maximum Flows and the Calculation of Optimal Intermediate Contours in Open Pit Mine Design. // INFOR Journal. – 2004. – Vol. 42, № 2. – P. 143–153, doi: 10.1080/03155986.2004.11732697.

42. *Pikies T., Kubale M., Turowski K.* Scheduling with complete multipartite incompatibility graph on parallel machines: complexity and algorithms // Artificial intelligence. – 2022. – Vol. 309. – P. 103711, doi: 10.1016/j.artint.2022.103711.

43. *Margaux Nattaf, Stéphane Dauzère-Pérès, Claude Yugma, Cheng-Hung Wu*. Parallel Machine Scheduling with Time Constraints on Machine Qualifications // Computers and Operations Research, Elsevier. – 2019. – 107. – P. 61–76. doi:10.1016/j.cor.2019.03.004. hal-02067750.

44. *Hua Gong, Yuyan Zhang, Puyu Yuan*. Scheduling on a Single Machine and Parallel Machines with Batch Deliveries and Potential Disruption // Complexity in Economics and Business. – Vol. 2020, Article ID 6840471, 2020. doi: https://doi.org/10.1155/2020/6840471.

45. *Ek A., Garcia de la Banda M., Schutt A., Stuckey P. J., Tack G.* Modelling and Solving Online Optimisation Problems // Proceedings of the AAAI Conference on Artificial Intelligence. – 2020. – 34(02). – P. 1477–1485. doi: https://doi.org/10.1609/aaai.v34i02.5506.

46. 1С: Автоматизированное составление расписания. Возможности продукта. Режим доступа: https://solutions.1c.ru/asp_spo/features. (Дата обращения: 01.06.2022).

47. *R´egin J.* Generalized arc consistency for global cardinality constraint // Proceedings of the Thirteenth National Conference on Artificial Intelligence. Portland. – 1996. – P. 209–215.

48. *Лихтарников Л. М.* Задачи мудрецов. – М. : Просвещение, АО «Учеб. лит.», 1996. – 112 с.

49. Employee Scheduling. Режим доступа: https: //developers.google.com/optimization/scheduling/employee_scheduling. (Дата обращения: 01.06.2022).

50.*Mushfiqur Rahman M., Binte Noor S., Hasan Siddiqui F.* Automated Large-scale Class Scheduling in MiniZinc // Proceedings of the 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI). – 2020. – P. 1–6. doi: 10.1109/STI50764.2020.9350485.

51. *Demirović E., Stuckey P. J.* Constraint Programming for High School Timetabling: A Scheduling-Based Model with Hot Starts // In: van Hoeve, WJ. (eds) Integration of Constraint

Programming // Artificial Intelligence and Operations Research. CPAIOR 2018. Lecture Notes in Computer Science, Vol. 10848. Springer, Cham. doi: https://doi.org/10.1007/978-3-319-93031-2_10.

52. Traveling Salesperson Problemhttps Режим доступа: https://developers.google.com/optimization/routing/tsp. (Дата обращения: 01.06.2022).

53. *Joshi C. K., Cappart Q., Rousseau L. M. [et al.]* Learning the travelling salesperson problem requires rethinking generalization // Constraints. – 2022. – 27. – P. 70–98. doi: https://doi.org/10.1007/s10601-022-09327-y.

54. *Van Cauwelaert S., Schaus P.* Efficient filtering for the Resource-Cost AllDifferent constraint // Constraints. – 2017. – 22. – P. 493–511. doi: https://doi.org/10.1007/s10601-017-9269-y.

55. *Koehler J., Bürgler J., Fontana U. [et al.]* Cable tree wiring-benchmarking solvers on a real-world scheduling problem with a variety of precedence constraints // Constraints. – 2021. – 26. – P. 56–106. doi: https://doi.org/10.1007/s10601-021-09321-w.

56. *Fages J. G., Lorca X., Rousseau L. M.* The salesman and the tree: the importance of search in CP // Constraints. – 2016. – 21. – P. 145–162. doi: https://doi.org/10.1007/s10601-014-9178-2.

57. *Nicolas Isoart.* The traveling salesman problem in constraint programming. Data Structures and Algorithms [cs.DS]. Université Côte d'Azur. – 2021. – 191 p.

58. Resource Constraints. Режим доступа: https://developers.google.com/optimization/routing/cvrptw_resources. (Дата обращения: 01.06.2022).

Зуенко Александр Анатольевич — канд. техн. наук, ведущий научный сотрудник ИИММ КНЦ РАН.
E-mail: zuenko@iimm.ru
ORCID: https://orcid.org/0000-0002-7165-6651

Фридман Ольга Владимировна — канд. техн. наук, старший научный сотрудник ИИММ КНЦ РАН.
E-mail: ofridman@iimm.ru
ORCID: https://orcid.org/0000-0003-1897-4922

Зуенко Ольга Николаевна — аспирант, стажер-исследователь, ИИММ КНЦ РАН.
E-mail: ozuenko@iimm. ru
ORCID: https://orcid.org/0000-0001-5431-7538

DOI: https://doi.org/10.17308/sait/1995-5499/2022/4/156-179 ISSN 1995-5499
Received 08.07.2022
Accepted 05.12.2022

CONSTRAINT PROGRAMMING PARADIGM IN SOLVING SCHEDULING PROBLEMS: AN ANALYTICAL SURVEY

© 2022 A. A. Zuenko, O. V. Fridman✉, O. N. Zuenko

Institute of Informatics and Mathematical Modeling – a separate subdivision of the Federal Research Center "Kola Science Center Russian Academy of Sciences"
24A, Fersman Street, 184209 Apatity, Murmansk region, Russian Federation

Annotation. The study considers an approach to solving scheduling problems based on the paradigm of constraint programming. Any method of constraint satisfaction includes two mandatory parts: a) the component responsible for the search; b) the component performing reasoning on constraints (constraint propagation). When implementing the former component, a certain type of depth-first search is usually used using heuristics to select the successor of the current node in the search tree. The latter component is implemented using the mechanism of global constraints. Global constraints can be considered as composite constraints, which are a set of simpler same-type constraints. Algorithms of global constraint propagation, as a rule, are supported by appropriate full-fledged theories, which allow organizing high performance computations. The constraint programming technology makes it possible to implement general methods for solving complex combinatorial problems, and also makes it possible to integrate existing methods of operation research theory aimed at solving narrow classes of problems using the mechanism of global constraints. When solving scheduling problems, the main constraints can be classified as follows: constraints on the order of operations, constraints on unary (distributive) resources, constraints on cumulative resources. This study reviews the main classes of scheduling problems and focuses on the constraints types that are used in them. To solve these problems, general-purpose constraint satisfaction methods can be used, but the use of specialized heuristics and global constraints is considered more promising. The study provides an overview of the most popular heuristics used in solving scheduling problems, as well as global constraints such as: global constraint on disjunctive resources, global constraint on cumulative resources, global constraint on the cardinality set of. The study discusses the features of the implementation of various classes of scheduling problems using modern constraint programming environments and libraries.
Keywords: scheduling, unary resources, cumulative resources, scheduling methods, constraint satisfaction problem, constraint propagation, constraint programming.

CONFLICT OF INTEREST

The authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.

REFERENCES

1. *Ghallab Malik, Nau Dana and Traverso Paolo*. (2016) Automated Planning: Theory and Practice. *Cambridge University Press*.

2. *Baptiste Ph., Le Pape C. and Nuijten W.* (2001) Constraint-based scheduling: applying constraint programming to scheduling problems. *Kluwer Academic Publishers*.

3. *Apt K. R.* (2003) Principles of Constraint Programming. *Cambridge University Press*.

4. *Dechter R.* (2003) Constraint Processing. Morgan Kaufmann.

5. *Tsang E.* (1993) Foundation of Constraint Satisfaction. Academic Press.

6. *Rossi F., Van Beek P. and Walsh T.* (2006) Handbook of constraint programming. *Elsevier*.

7. *Bartak R.* (2005) Constraint Satisfaction for Planning and Scheduling. *In Vlahavas, Vrakas (eds.): Intelligent Techniques for Planning*. P. 320–353.

8. *Roman Bartak, Miguel A. Salido and Francesca Rossi* (2011) Constraint Satisfaction Tech-

✉ Fridman Olga V.
e-mail: ofridman@iimm.ru

niques in Planning and Scheduling. *Constraints*. 16(3). P. 223–227. Available from: doi: 10.1007/s10601-011-9109-4.

9. *Lazarev A. A., Lemtyuzhnikova D. V. and Werner F.* (2021) A metric approach for scheduling problems with minimizing the maximum penalty. *Applied Mathematical Modelling*. 89(2). P. 1163–1176. Available from: doi: https://doi.org/10.1016/j.apm.2020.07.048.

10. *Antonova A. S. and Aksyonov K. A.* (2020) Analysis of the methods for accounting the renewable and non-renewable resources in scheduling. *Journal of Physics: Conference Series*. 1694. 1. 012005. Available from: doi: 10.1088/1742-6596/1694/1/012005.

11. *Baptiste P., Laborie P., Le Pape C. and Nuijten W.* (2006) Constraint-Based Scheduling and Planning. Handbook of Constraint Programming. *Amsterdam: Elsevier.* P. 761–799. Available from: doi:10.1016/S1574-6526(06)80026-X

12. *Ruttkay Z.* (1998) Constraint Satisfaction – a Survey. *CWI Quarterly*. 11. P. 123–162.

13. *Lhomme O.* (1993) Consistency techniques for numeric CSPs. *Proceedings of 13th International Joint Conference on Artificial Intelligence*.

14. *Bitner J. R. and Reingold E. M.* (1975) Backtracking Programming Techniques. *Communications of the ACM*. 18. P. 651–655.

15. *Gaschnig J.* (1979) Performance measurement and analysis of certain search algorithms. *Carnegie-Mellon University: Technical Report CMUCS*. P. 79–124.

16. *Prosser P.* (1993) Hybrid Algorithm for the Constraint Satisfaction Problem. *Computational Intelligence*. 9. P. 268–299.

17. *Gaschnig J.* (1977) A general backtrack algorithm that eliminates most redundant tests. *Procceedings of IJCAI*.

18. *Frost D. and Dechter R.* (1994) Deadend driven learning. *Procceedings of the National Conference on Artificial Intelligence*. P. 294–300.

19. *Haralick R. and Elliot G.* (1980) Increasing Tree Efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*. 14. P. 263–314.

20. *Michalewicz Z. and Fogel D. B.* (2000) How to Solve It: Modern Heuristics. *SpringerVerlag*.

21. *Roman Barták* (2008) Artificial Intelligence for Advanced Problem Solving.

22. *Baptiste P., Le Pape C. and Nuijten W.* (1995) Constraint-Based Optimization and Approximation for Job-Shop Scheduling. *Proceedings of the AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, IJCAI-95*.

23. *Smith S. F. and Cheng Ch.-Ch.* (1993) Slack-Based Heuristics For Constraint Satisfaction Scheduling. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. P. 139–144.

24. *Mackworth A. K.* (1977) Consistency in networks of relations. *Artificial Intelligence*. 8(1). P. 99–118.

25. *Baptiste P., Le Pape C. and Nuijten W.* (2001) Constraint-Based Scheduling. *Springer*.

26. *Baptiste P. and Le Pape C.* (1996) Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling. *Proceedings of PLANSIG*.

27. *Vil´ım P., Bartak R. and Cepek O.* (2005) Extension of O(n log n) filtering algorithms for the unary resource constraint to optional activities. *Constraints*. 10(4). P. 403–425. Available from: doi:10.1007/s10601-005-2814-0.

28. *Torres P. and Lopez P.* (2000) On Not-First/Not-Last conditions in disjunctive scheduling. *European Journal of Operational Research*. 127. P. 332–343.

29. *Lazarev A. A. and Gafarov E. R.* (2011) Teoriya raspisanij zadachi i algoritmy. [Problem scheduling theory and algorithms]. *Moscow, Moscow State University named after M.V. Lomonosov*. (In Russian).

30. *Russel S. and Norvig P.* (2010) Artificial Intelligence: A Modern Approach. 3rd edition. *Prentice Hall*.

31. The Job Shop Problem. Available from: https://developers.google.com/optimization/scheduling/job_shop. [Accessed 01.06.2022].

32. IBM CP LEX. Available from: https://www.ibm.com/analytics/cplexoptimizer. [Accessed 01.06.2022].

33. MiniZinc. Available from: https://www.minizinc.org/doc2.5.5/en/index.html [Accessed 01.06.2022].

34. *Arkhipov D. I. and Lazarev A. A. and Tarasov G. V.* (2017) Estimating Maximum Resource Load for Resource-Constrained Project Scheduling Problem. *Proceedings of the 8th In-*

ternational Conference on Optimization Methods and Applications "OPTIMIZATION AND APPLICATINS" (OPTIMA-2017). Moscow: CC RAS. 1987. P. 356–363.

35. *Gimadi E. Kh., Goncharov E. N. and Shtepa A. A.* (2021) A fast algorithm for finding a lower bound of the solution of the Resource-Constrained Project Scheduling Problem tested on PSPLIB instances. *Proceedings of the Institute of Mathematics and Mechanics UrO RAN*. 27. 1. P. 22–36. Available from: doi: 10.21538/0134-4889-2021-27-1-22-36.

36. *Nattaf M., Artigues C. and Lopez P.* (2017) Cumulative scheduling with variable task profiles and concave piecewise linear processing rate functions. *Constraints*. 22. P. 530–547. Available from: doi: https://doi.org/10.1007/s10601-017-9271-4.

37. *Kreter S., Schutt A. and Stuckey P. J.* (2017) Using constraint programming for solving RCPSP/max-cal. *Constraints*. 22. P. 432–462. Available from: doi: https://doi.org/10.1007/s10601-016-9266-6.

38. *Pleskunov M. A.* (2014) Zadachi setevogo planirovaniya: uchebnoe posobie. [Tasks of network planning: a tutorial]. *Yekaterinburg: Ural Publishing House university*. (In Russian).

39. *Henry Amankwah*. (2011) Mathematical Optimization Models and Methods for Open-Pit Mining. *LiU-Tryck, Linköping, Sweden*.

40. *Lerchs H. and Grossmann I. F.* (1965) Optimum Design of Open-Pit Mines, Transactions. *Canadian Institute of Mining and Metallurgy. LXVIII*. P. 17–24.

41. *Picard J.-C. and Smith B. T.* (2004) Parametric Maximum Flows and the Calculation of Optimal Intermediate Contours in Open Pit Mine Design. *INFOR Journal*. 42. 2. P. 143–153. Available from: doi: 10.1080/03155986.2004.11732697.

42. *Pikies T., Kubale M. and Turowski K.* (2022) Scheduling with complete multipartite incompatibility graph on parallel machines: complexity and algorithms. Artificial intelligence. 309. 103711. Available from: doi: 10.1016/j.artint.2022.103711.

43. *Margaux Nattaf, Stéphane Dauzère-Pérès, Claude Yugma, Cheng-Hung Wu.* (2019) Parallel Machine Scheduling with Time Constraints on Machine Qualifications. *Computers and Operations Research*, *Elsevier*. 107. P. 61–76. Available from: doi:10.1016/j.cor.2019.03.004. hal-02067750.

44. *Hua Gong, Yuyan Zhang and Puyu Yuan*. (2020) Scheduling on a Single Machine and Parallel Machines with Batch Deliveries and Potential Disruption. Complexity in Economics and Business, Article ID 6840471, Available from: doi: https://doi.org/10.1155/2020/6840471.

45. *Ek A., Garcia de la Banda M., Schutt A., Stuckey P. J. and Tack G.* (2020) Modelling and Solving Online Optimisation Problems. *Proceedings of the AAAI Conference on Artificial Intelligence*. 34(02). P. 1477–1485. Available from: doi: https://doi.org/10.1609/aaai.v34i02.5506.

46. 1S: Avtomatizirovannoe sostavlenie raspisaniya. Vozmozhnosti produkta. [1C: Automated scheduling. Product features]. Available from: https://solutions.1c.ru/asp_spo/features. [Accessed 01.06.2022]. (In Russian).

47. *R´egin J.* (1996) Generalized arc consistency for global cardinality constraint. *Proceedings of the Thirteenth National Conference on Artificial Intelligence. Portland*. P. 209–215.

48. *Likhtarnikov L. M.* (1996) Zadachi mudreczov.[Tasks of the Wise]. *Moscow, Enlightenment, JSC. Study. lit.* (In Russian).

49. Employee Scheduling. Available from:https://developers.google.com/optimization/scheduling/employee_scheduling. [Accessed 01.06.2022].

50. *Mushfiqur Rahman M., Binte Noor S. and Hasan Siddiqui F.* (2020) Automated Large-scale Class Scheduling in MiniZinc. *Proceedings of the 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*. P. 1–6. Available from: doi: 10.1109/STI50764.2020.9350485.

51. *Demirović E. and Stuckey P. J.* (2018) Constraint Programming for High School Timetabling: A Scheduling-Based Model with Hot Starts. In: van Hoeve, WJ. (eds) Integration of Constraint Programming, Artificial Intelligence and Operations Research. CPAIOR 2018. *Lecture Notes in Computer Science, Springer, Cham*. 10848. Available from: doi: https://doi.org/10.1007/978-3-319-93031-2_10.

52. Traveling Salesperson Problemhttps. Available from: https://developers.google.com/optimization/routing/tsp [Accessed 01.06.2022].

53. *Joshi C. K., Cappart Q., Rousseau L. M. [et al.]* (2022) Learning the travelling salesperson problem requires rethinking generalization. *Constraints*. 27. P. 70–98. Available from: doi: https://doi.org/10.1007/s10601-022-09327-y.

54. *Van Cauwelaert S. and Schaus P.* (2017) Efficient filtering for the Resource-Cost AllDifferent constraint. *Constraints*. 22. P. 493–511. Available from: doi: https://doi.org/10.1007/s10601-017-9269-y.

55. *Koehler J., Bürgler J., Fontana U. [et al.]* (2021) Cable tree wiring-benchmarking solvers on a real-world scheduling problem with a variety of precedence constraints. *Constraints*. 26. P. 56–106. Available from: doi: https://doi.org/10.1007/s10601-021-09321-w.

56. *Fages J. G., Lorca X. and Rousseau L. M.* (2016) The salesman and the tree: the importance of search in CP. *Constraints*. 21. P. 145–162. Available from: doi: https://doi.org/10.1007/s10601-014-9178-2.

57. *Nicolas Isoart*. (2021) The traveling salesman problem in constraint programming. Data Structures and Algorithms [cs.DS]. *Université Côte d'Azur*.

58. Resource Constraints. Available from: https://developers.google.com/optimization/routing/cvrptw_resources. [Accessed 01.06.2022].

Zuenko Alexander A. — PhD, leading researcher, IIMM KSC RAS.
E-mail: zuenko@iimm.ru
ORCID iD is https://orcid.org/0000-0002-7165-6651

Fridman Olga V. — PhD, senior researcher, IIMM KSC RAS.
E-mail: ofridman@iimm.ru
ORCID iD is https://orcid.org/0000-0003-1897-4922

Zuenko Olga N. — postgraduate, trainee researcher, IIMM KSC RAS.
E-mail: ozuenko@iimm. ru
ORCID iD is https://orcid.org/0000-0001-5431-7538