ЗАО ЛЕДАС

А. Ершов, И. Иванов, В. Корниенко, С. Прейс, А. Рассказов, И. Рыков

LSE: НОВОЕ ВЫЧИСЛИТЕЛЬНОЕ ЯДРО СИСТЕМЫ ПЛАНИРОВАНИЯ LEDAS SCHEDULER 3.0 И ПЕРСПЕКТИВЫ ЕГО ИСПОЛЬЗОВАНИЯ

Препринт
15

Новосибирск, 2005

Данный препринт открывает серию публикаций о проекте LEDAS Scheduler 3.0 — новой версии интегрированной среды, предназначенной для формулирования, редактирования и решения задач планирования. Версия 3.0 основана на новом вычислительном ядре LSE, которое позволяет повысить эффективность решения задач планирования на порядки, особенно на самых распространённых классах задач, за счет разработанных и реализованных компанией высокопроизводительных методов. Препринт предоставляет необходимую информацию о составляющих проекта: архитектуре, наборе алгоритмов и интерфейсе нового ядра LSE, текущих результатах тестирования. Кроме того, всесторонне рассматриваются аспекты интеграции LEDAS Scheduler 3.0 в другие продукты, в частности, в разрабатываемые компанией ЛЕДАС клиент-серверную систему автоматического планирования собраний MORE+ для традиционных и мобильных платформ и аналогичную по функциональности Microsoft Project систему проектного планирования для платформы Eclipse.

Серия препринтов издается российской фирмой ЛЕДАС — разработчиком и производителем интеллектуальных программных решений.

пр. Лаврентьева, 6, Новосибирск, 630090.
http://ledas.com
email: info@ledas.com

Содержание

ВВЕДЕНИЕ

Задачи планирования имеют отношение практически ко всем сферам человеческой деятельности. Эти задачи могут достаточно сильно различаться в зависимости от прикладной области, в которой они возникают, однако все они имеют много общего. Сам термин “задача планирования” подразумевает необходимость расставить во времени определённые действия с участием определённых людей, машин, оборудования, возможно, с учётом их места расположения и с учётом требуемых для этого материальных ресурсов — денег, материалов и т.п. Такие задачи обычно естественным образом включают разнообразные ограничения, которые надо принять во внимание — это и доступность тех или иных участников, ограничения на время, на материальные ресурсы, на последовательность действий и многие другие.

Разработка программного обеспечения, позволяющего автоматически решать задачи в ограничениях, — это одно из базовых направлений деятельности компании ЛЕДАС. Компетенция сотрудников компании в этой сфере была заложена еще в академической среде 80-х годов, а технологии работы с ограничениями в задачах планирования были реализованы нашей компанией в проекте LEDAS Scheduler в 1999–2001 годах. LEDAS Scheduler представляет собой расширяемую среду, предназначенную для формулирования, редактирования и решения задач планирования. По сравнению с Microsoft Project, LEDAS Scheduler решает значительно более широкий класс задач за счет поддержки максимально широкого набора ограничений и целевых функций. По сравнению с ILOG Scheduler, LEDAS Scheduler представляет собой более интегрированный программный продукт, позволяющий эффективно осуществлять хранение, редактирование и обмен данными с другими приложениями с помощью расширяемого набора развитых инструментальных средств.

В связи с нарастающей рыночной востребованностью высокоразвитых систем планирования, компания ЛЕДАС поставила задачу радикальной индустриализации своего решения и осенью 2004 года начала разработку LSE — вычислительного ядра новой версии LEDAS Scheduler 3.0.

Основные черты LSE:

- повышение эффективности вычислений на порядки, особенно на самых распространённых классах задач, за счет разработанных и реализованных компанией высокопроизводительных методов;

- продуманная архитектура и API, способствующая эффективной интеграции LSE в другие программные продукты.

Первые результаты тестирования LSE показали, что на базе из нескольких тысяч тестов, включающей задачи планирования 15 различных классов, новый решатель добивается в среднем 5–10% отклонения получаемого субоптимального решения от точного оптимального. LSE является прекрасно масштабируемым и высокопроизводительным инструментом: для задач с числом работ до 1000 он находит решение менее чем за одну секунду, время вычислений растет квадратично относительно размера решаемой задачи.

Специалисты компании делают вывод: решатель LSE создает базу для программного продукта принципиально нового уровня, который будет доступен пользователям для интеграции, начиная с весны 2005 года — в начале в форме программы раннего доступа LEDAS Scheduler EAP, апробированной ранее при разработке двухмерного и трехмерного (см. `http://lgs3d.ledas.com/download`) геометрических решателей.

Основные крупные классы приложений, на которые ориентируется LEDAS Scheduler 3.0:

- планирование и управление проектами;
- системы поддержки принятия решений;
- ресурсно-календарное планирование.

Наряду с разработкой нового ядра среды LEDAS Scheduler, компания ЛЕДАС, с учетом своих собственных рыночных интересов, разрабатывает следующие программные компоненты, способные легко интегрироваться с версией 3.0:

- система автоматического планирования собраний MORE+ с архитектурой клиент-сервер, включающая клиенты для различных (в том числе мобильных) платформ;
- аналогичная Microsoft Project система проектного планирования для платформы Eclipse (описание платформы см. на `http://www.eclipse.org`).

Данный препринт открывает серию публикаций о проекте LEDAS Scheduler 3.0, предоставляя информацию о его составляющих: архитектуре, наборе алгоритмов и интерфейсе нового ядра LSE, текущих результатов тестирования, а также описывая возможности интеграции LSE и LEDAS Scheduler.

Начнем с терминологии. Традиционно в задачах планирования выделяют следующий набор сущностей:

- *Работа или задача (Task/Job/Activity)* — некая деятельность во времени, характеризуемая временами начала/конца и длительностью.
- *Ресурс (Resource)* — человек, машина, механизм, оборудование, помещение и т.п., которые могут использоваться для выполнения работы и быть соответственно занятыми (частично или полностью) во время выполнения работы. Ресурсы, которые могут быть заняты частично (или которые могут быть множеством мелких неразличимых частей), называются *дискретными;* ресурсы, существующие только как единое целое, называются *унарными*. Оба этих типа ресурсов относятся к классу возобновляемых (*renewable*).
- *Расходуемые ресурсы (Consumable/Costs)* — ресурсы, которые могут расходоваться в процессе выполнения той или иной деятельности. Часто отдельно выделяют стоимость — расходование денег.
- *Календарь (Calendar)* — временной график доступности ресурсов или возможности выполнения работы.
- *Использование ресурса (Resource assignment)* — связь между работой и ресурсом, говорящая о том, что ресурс нужен для данной работы. Для дискретных и расходуемых ресурсов характеризуется необходимым количеством единиц ресурса. Использование ресурсов — один из основных источников ограничений, возникающих в задачах планирования. Ограничение возникает из невозможности использовать один и тот же унарный ресурс одновременно для выполнения двух работ; для дискретных ресурсов аналогичное ограничение возникает из-за того, что потребности в ресурсе в какой-то момент времени превосходят его фактическое наличие.
- *Пул ресурсов (Resource pool)* — множество возобновляемых ресурсов, каждый из которых может быть использован работой или несколькими работами. Эта сущность расширяет понятие ресурса, позволяя задавать связь работы не с конкретным ресурсом, а с какими-то ресурсами из заданного множества. Различные пулы могут произвольным образом пересекаться друг с другом, что делает этот инструмент существенно более гибким, чем унарные или дискретные ресурсы.
- *Порядок работ (Task order, precedence)* — ещё одно типичное ограничение в задачах планирования, задающее порядок следования: одна работа выполняется раньше (*predecessor*) или позже (*successor*) другой работы. Ограничение может связывать в любой

комбинации начала и концы двух работ, а также характеризоваться положительным или отрицательным лагом между работами.

- *Ограничения на время выполнения (Release and due dates/Task constraints)* — ограничения, привязывающие работы к реальному времени. Эти ограничения формулируются в виде “начало не раньше, чем” или “конец не позже чем”. Далее в тексте под ограничениями будут пониматься не только такие ограничения, а все возможные ограничения, характерные для задач планирования — как ресурсные, так и ограничения следования.

Все эти сущности уже поддерживались версией LEDAS Scheduler 2.5, и будут поддерживаться в версии 3.0. Прежнее вычислительное ядро использовало мощные универсальные механизмы решения задач в ограничениях (которые предоставляет математический решатель от компании ЛЕДАС) и развитые средства представления задач, позволяло формулировать практически любые задачи планирования и решать их. Помимо собственно среды LEDAS Scheduler, на основе этого ядра было реализовано несколько разноплановых продуктов: основанная на web-технологиях система планирования собраний MORE, система планирования управления персоналом LETOM и др.

Ядро системы LEDAS Scheduler 2.5 имело два существенных недостатка. Во-первых, будучи универсальным, оно не обладало достаточной производительностью для решения специальных классов задач, часто возникающих в реальных приложениях. Такие задачи могут иметь достаточно большой размер — сотни, тысячи и даже десятки тысяч работ, но структура ограничений и связей между этими работами обычно носит специальный характер, что позволяет использовать специализированные и, вследствие этого, более эффективные алгоритмы удовлетворения ограничений. Во-вторых, LEDAS Scheduler 2.5 был ориентирован на поиск точного оптимального решения, что является, в соответствии с классическими математическими теориями, неэффективно разрешимой задачей. Именно поэтому все промышленные программные продукты, предназначенные для решения задач планирования, вместо точного оптимального решения ищут так называемое субоптимальное (близкое к оптимальному) решение. Такой прагматический подход используется и в новом вычислительном ядре LSE, которое будет применено в LEDAS Scheduler 3.0.

Детали архитектуры и алгоримической части LSE будут приведены в разделе 1; в разделе 2 читатель может познакомиться с архитектурой среды LEDAS Scheduler и интеграционными возможностями этой системы; далее в разделе 3 последует описание разрабатываемых компанией ЛЕДАС при-

ложений, использующих для своей работы вычислительные способности нового ядра, после чего приводится короткое заключение.

1. ВЫЧИСЛИТЕЛЬНОЕ ЯДРО

1.1. Общая архитектура LSE

LSE представляет собой платформонезависимый C++ модуль-библиотеку. Программный интерфейс этого модуля (LSE API) состоит из абстрактных классов. Такая организация позволяет максимально отделить интерфейс LSE от его реализации, гарантируя, что изменение интерфейса будет происходить только при появлении новых сущностей или новой функциональности, но не при изменении в коде реализации. Собственно LSE состоит из классов, реализующих интерфейсные абстрактные классы, и дополнительных классов, обеспечивающих вычислительную часть (препроцессор, валидатор, методы, эвристики).

Итак, LSE API включает в себя несколько абстрактных классов, каждый из которых представляет различные сущности — работы, ресурсы, ограничения, задачу в целом и решатель. Создание и удаление объектов класса ILSEProblem (задача планирования) и ILSEEngine (решатель) происходит с помощью статических методов, вся остальная работа с этими классами и классами, представляющими элементы задачи, происходит с помощью виртуальных методов. С их помощью можно создавать работы, ресурсы и ограничения, добавлять их в ILSEProblem, указывать ILSEEngine проблему, решение которой надо найти, запускать и получать результаты вычислений. Имеется возможность использовать во время вычислений указываемую приложением функцию-коллбэк, с помощью которой можно прерывать вычисления, производить отрисовку графического пользовательского интерфейса и т.п.

Основной класс, через который происходит управление работой LSE, — это LSEEngine. Его главная функция LSEEngine::Run() запускает препроцессор, который производит первоначальную обработку задачи планирования, например, переносит ограничения, наложенные на групповые работы, на уровень индивидуальных работ. Далее определяется тип решаемой задачи, и запускается наиболее подходящий для ее решения вычислительный метод или набор методов. Каждый такой метод представлен отдельным классом, поддерживающим, кроме необходимых функций задания входных и получения выходных данных, механизм коллбэков, позволяющих переда-

вать управление приложению или прерывать процесс вычислений. Большинство методов в своей работе используют эвристики, оформленные как отдельные функции. После того как метод получил решение, LSEEngine::Run() осуществляет его валидацию с помощью внешнего по отношению к методу валидатора; в случае запуска нескольких методов, из полученных решений выбирается наиболее оптимальное, удовлетворяющее всем ограничениям задачи.

1.2. Задачи планирования, решаемые LSE

На данный момент LSE эффективно решает задачи в базовой постановке, которая расширяет классическую задачу календарно-ресурсного планирования, известную как RCPSP (resource constrained project scheduling problem, см. [1]).

Задача планирования в базовой постановке — это четверка (***J, R, P, K***), в которой

1. Множество ***J*** из *N* работ, для которых определены их длительности $d_j \in Z^+$. Предполагается, что задача планирования решается в дискретной шкале времени: *t* = 0, 1, … . *Расписанием* называется множество $S = \{s_j \in Z^+ \cup \{0\} \mid j \in J\}$ моментов стартов работ. При этом работа считается активной (выполняющейся) в моменты $t = s_j, s_j + 1, \ldots s_j + d_j - 1$.
2. Множество ***R*** из *M* возобновимых (renewable) дискретных ресурсов, для которых определены их количества $K_r \in Z^+$.
3. Задано множество ***P*** ограничений предшествования (job order, precedence constraints). Каждое ограничение задается четвёркой (j_1, j_2, *t*, *l*), где *t* ∈ {‘start-finish’, ‘start-start’, ‘finish-finish’, ‘finish-start’}, $l \in Z^+$ — временной промежуток (лаг) между работами j_1 и j_2. Определяют технологический порядок следования работ: если, например, *t* = ’finish-start’, то работа j_2 не может начаться раньше, чем через *l* единиц времени после окончания j_1. ($s_{job2} \geq s_{job1} + d_{job1} + l$). Смысл остальных типов аналогичен.

   Замечание. Эти ограничения могут быть представлены в виде взвешенных дуг в графе с вершинами-работами. Необходимое и достаточное условие существования расписания, удовлетворяющего всем таким ограничениям — отсутствие в графе контуров положительного веса. В LSE поддерживаются только бесконтурные

графы, что, однако, является типичной ситуацией в задачах календарно-ресурсного планирования.

4. Задано множество ***K*** назначений ресурсов работам (assignments) $k_{jr}, j \in J, r \in R$. Определяют следующее ограничение: в каждый момент времени и для каждого ресурса сумма потребностей всех работ, выполняемых в этот момент времени не должна превышать количество этого ресурса: $\sum_{j \in A(t)} k_{jr} \le K_r, \forall t \ge 0, \forall r \in R$, где $A(t) = \{j \in J \mid s_j \le t < s_j + d_j\}$ — множество активных в момент t работ. В случае, когда матрица (k_{jr}) разреженная (то есть не очень много ненулевых назначений), экономнее эти назначения задавать в виде списка троек (Job, Resource, Capacity).

В LSE дополнительно рассматриваются следующие ограничения:

5. Для некоторых работ могут быть заданы сроки их появления $v_j \in Z^+$ (release dates). Работа не может начаться раньше своего release date ($s_j \ge v_j$). Моделируются при помощи добавления фиктивной работы длительности 0, являющейся 'finish-start'-предшественником этих работ с лагами равными v_j.
6. Для некоторых работ могут быть заданы директивные сроки их исполнения $u_j \in Z^+$ (due dates). Работа, для которой задано ограничение due date, должна закончиться не позднее этого времени ($s_j + d_j \le u_j$).

   Замечание. Реализованные в LSE алгоритмы не гарантируют построения расписания, удовлетворяющего ограничениям этого типа. Это является естественным, поскольку задача составления любого расписания, удовлетворяющая ограничениям с due date является столь же сложной, как и задача составления оптимального расписания, то есть не является эффективно разрешимой. Для задач такого типа производятся эвристические попытки соблюсти директивные сроки, а также производится валидация их соблюдения.
7. Несовместность работ (unsimultaneous/incompatible jobs). Задаётся множеством работ и определяет, что в любой момент времени выполняться может не более одной работы из этого списка. Моделируется при помощи добавления фиктивного ресурса в количестве 1, и назначений его каждой из этих работ в количестве 1.

8. Иерархии работ. На множестве работ может быть задано отношение иерархии, являющееся частичным порядком, граф которого представляет собой лес. Число уровней иерархии работ (высота леса) неограниченно. Задание ресурсных или порядковых ограничений для включающей работы эквивалентно заданию соответствующих ограничений для всех включенных в нее работ.

Допустимым (или корректным) расписанием называется расписание S, удовлетворяющее всем ограничениям 3, 4, 5, 7. Задача состоит в нахождении такого из допустимых расписаний, для которого время окончания проекта (величина $C_{\max} = \max\{c_j = s_j + d_j \mid j \in J\}$, то есть время окончания последней работы) минимально. Если безусловным приоритетом является именно строгая минимальность времени окончания проекта, то мы имеем дело с задачей поиска оптимального решения, если же необходимо минимизировать как время окончания проекта, так и время поиска решения задачи, то говорят о поиске субоптимального решения.

Кроме базовой постановки задачи, нами будет рассматриваться далее расширенная постановка, которая включает работы двух типов — обычные, с фиксированными длительностями, и работы, для которых длительность зависит от состава и количества используемых ресурсов.

В этом случае определение расписания помимо стартов работ также включает выбор состава и количества используемых каждой работой ресурсов. Длительность исполнения назначения (assignment) определяется как задаваемый на входе объём этого назначения (capacity), делённый на суммарное количество всех выделенных согласно расписанию ресурсов; такие величины и определяют длительности нефиксированных работ. Работы с нефиксированными длительностями делятся на два типа: для первого типа (параллельное использование ресурсов) длительность работы определяется как максимум из длительностей исполнения назначений, в которых она фигурирует; для второго типа (последовательное использование ресурсов) — как сумма длительностей исполнений этих назначений.

Вернемся к базовой постановке. Как уже было сказано, ограничения 5, 7 моделируются в терминах ограничений 3, 4. Перевод ограничений 5, 7 в 3, 4 и развёртывание иерархической структуры 8 задачи осуществляется в начале работы решателя LSE с помощью препроцессора.

Таким образом, базовая постановка задачи для LSE сводится к задаче RCPSP и для её решения можно использовать известные методы решения RCPSP. Кроме того, считаем, что все ограничения из 3 имеют тип ‘finish-

start' (остальные типы преобразуются к этому в препроцессоре), и вес соответствующей дуги в графе равен лагу.

Рассмотрим применяемые в LSE алгоритмы решения задач в базовой постановке.

1.3. Метод критического пути

В случае полного отсутствия ресурсов (а, следовательно, и ограничений типа 4) все ограничения задачи содержатся в ориентированном графе предшествования $G = (J, P)$. В этом случае длина оптимального расписания равна длине критического, то есть самого длинного, пути в G. Под длиной пути подразумевается сумма весов всех вершин (длительностей соответствующих работ) и весов всех дуг, входящих в этот путь. В этом расписании время старта любой работы определяется как длина критического пути до этой работы. Существует общеизвестный квадратичный алгоритм для отыскания такого расписания; точнее, алгоритм имеет трудоемкость $O(N^2 + CN)$, где C — число ограничений вида 3, а N — число работ.

Изложим вкратце этот алгоритм. Для каждой работы через P_j будем обозначать множество всех её непосредственных предшественников, то есть $P_j = \{i \in J \mid (i, j) \in E(G)\}$, где $E(G)$ — множество дуг графа G. Через $w(i, j)$ обозначим вес дуги $(i, j) \in E(G)$.

1. Положим $\forall j, s'_j = 0$
2. Выполнять {
   a) для $j = 1$ до N $\quad \{ s''_j = s'_j \}$
   b) для $j = 1$ до N $\quad \{ s'_j = \max\{s''_j, \max\{s''_i + d_i + w(i, j) \mid i \in P_j\}\}\}$
   } пока ($\exists j : s''_j \neq s'_j$)
3. $\forall j , s_j = s'_j$

Рангом вершины называется число дуг в наибольшем по этому показателю пути до данной вершины. Очевидно, что для работ, не имеющих предшественников (ранга 0), значение s'_j не изменится во время выполнения цикла. По индукции легко проверить, что для вершины ранга k окончательное значение s'_j установится не более чем после k итераций цикла. Ввиду того, что в n-вершинном графе максимальный ранг вершины не превышает $n - 1$, получаем, что выполниться может не более N итераций. Тру-

доёмкость же одной итерации равна $O(N+C)$: для вычисления всех максимумов нам надо просмотреть для каждой вершины все дуги, входящие в неё, то есть каждая дуга графа просматривается ровно один раз.

Полученное расписание обладает одним интересным свойством: ни одну работу нельзя сдвинуть на более ранний срок, то есть уменьшить время её старта. Расписание с таким свойством называется ранним. Безресурсное раннее расписание единственно и является оптимальным, при этом могут существовать другие оптимальные расписания, не являющиеся ранними.

В случае отсутствия ресурсов LSEEngine использует алгоритм критического пути и строит раннее расписание.

Наряду с ранним расписанием полезно также понятие *T*-позднего расписания. Если взять $T \geq T_{opt}$, где T_{opt} — длина оптимального расписания, то можно построить допустимое расписание, время окончания которого равно *T*, и в котором ни одну работу нельзя сдвинуть на более поздний срок. Естественно, T_{opt}-позднее расписание также оптимально. В нём времена стартов работ, принадлежащих критическому пути равны стартам этих работ в раннем расписании, для остальных s_j строго больше соответствующих стартов в раннем расписании.

1.4. Серийный метод

1.4.1. Описание серийного метода

Перейдём к рассмотрению общего случая, когда помимо ограничений предшествования имеются ресурсные ограничения. В этом случае задача (RCPSP) является NP-трудной, она содержит в себе, в частности, такой известный NP-трудный класс задач, как Job Shop (см. [2]). Поэтому не существует эффективных алгоритмов получения её оптимального решения.

Однако в реальных приложениях часто не требуется нахождения именно наикратчайшего расписания проекта. Для сложных проектов хорошим результатом является нахождение допустимого расписания приемлемой длины (например, на 10–30% длиннее оптимального).

В настоящее время в этом направлении существует множество алгоритмов. Большинство из них относятся к классу эвристических, то есть не гарантирующих никаких оценок отклонения для получаемого решения от оптимального. Однако на практике получающаяся погрешность в среднем не превышает 10%, что является очень хорошим результатом.

В LSE реализована одна из таких эвристических схем, известная как serial scheduling scheme (см.[2]). Она отличается своей простотой и общностью, и вместе с тем показывает прекрасные результаты. По своей сути данная схема является так называемым последовательным жадным алгоритмом.

Работа алгоритма состоит из *N* этапов, на каждом из которых мы фиксируем время выполнения одной из работ (то есть, после *k* этапов имеется частичное расписание для *k* работ, удовлетворяющее всем ограничениям).

Обозначим через S_n (scheduled set) множество работ, уже фиксированных к моменту начала этапа *n* (таким образом, $|S_n| = n-1$). Через D_n (decision set) обозначим множество ещё не фиксированных работ, у которых фиксированы все их предшественники (то есть, $D_n = \{j \notin S_n \mid P_j \subseteq S_n\}$). Именно из него выбирается работа, фиксируемая на этапе n. Этот выбор производится в соответствии с эвристическим правилом, которое может быть определено тем или иным образом в соответствии с некоторыми соображениями, отвечающими здравому смыслу.

Примером такого эвристического правила является LST-правило (latest start time): из всех работ D_n выбирать ту, для которой время старта в $C_{\max}$-позднем расписании для соответствующей безресурсной задачи наименьшее. Это означает, что эта работа, скорее всего, имеет самую длинную цепочку последователей, и ее выгодно поставить в расписании как можно раньше. Выбрав $h \in D_n$, мы фиксируем её в расписании, выбирая самое раннее время, на которое можно поставить эту работу, удовлетворив все ограничения предшествования и ресурсные ограничения. Понятно, что такое время найдётся: например, время $\max\{c_j + w(j,h) = s_j + d_j + w(j,h) \mid j \in S_n\}$ является допустимым временем старта новой работы. Также ясно, что новое s_h будет найдено в интервале от $\max\{c_j + w(j,h) \mid j \in P_h\}$ до $\max\{c_j + w(j,h) \mid j \in S_n\}$.

Эвристическое правило выбора можно формализовать при помощи функции $\nu(j), j \in D_n$. Мы выбираем работу с наибольшим значением $\nu(j)$, если таких работ несколько, то с наименьшим номером. Через $A(t)$ попрежнему обозначаем множество активных работ частичного расписания: $A(t) = \{j \in S_n \mid s_j \le t < s_j + d_j\}$. Итак, мы получаем следующий формальный алгоритм для серийного метода:

1. $n = 1;\ S_n = \emptyset$;
2. Пока $|S_n| < N$, делать
   a) $D_n = \{j \notin S_n \mid P_j \subseteq S_n\}$;
   b) $h = \min\{j \mid j = \arg\max\{\nu(i) \mid i \in D_n\}\}$;
   c) $t_{\min} = \max\{c_j + w(j,h) \mid j \in P_h\};\ t_{\max} = \max\{c_j + w(j,h) \mid j \in S_n\}$;
   d) $t^* = \min\{t \in [t_{\min}, t_{\max}] \mid k_{rh} + \sum_{j \in A(\tau)} k_{jr} \le K_r, \forall \tau = t, t+1, \ldots t + d_h - 1, \forall r \in R\}$;
   e) $s_h = t^*;\ S_{n+1} = S_n \cup \{h\}$;
   f) $n = n + 1$.

Этот алгоритм реализован в LSE с квадратичной трудоёмкостью и показывает очень высокую скорость — для задач с тысячами работ он строит субоптимальное расписание менее чем за одну секунду.

В рамках этого метода реализовано несколько эвристик выбора очередной расписываемой работы из D_n. Наряду с LST-правилом это LFT (latest finish time, работа с наименьшим временем окончания в позднем безресурсном расписании), MTS (most total successors, работа с наибольшим количеством всех последователей) и другие. Ввиду высокой скорости работы алгоритма задача решается с использованием всех этих эвристик, из полученных решений выбирается наилучшее.

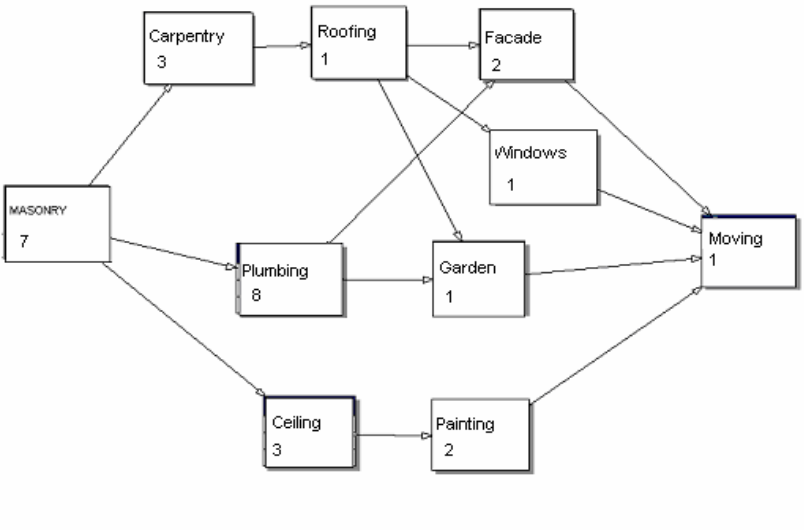
Получаемое алгоритмом расписание также является ранним: ни одну из работ нельзя сдвинуть на более ранний срок, не сдвигая другие работы. Легко заметить, что среди ранних расписаний задачи RCPSP есть оптимальное решение. Действительно, если взять какое-либо оптимальное не раннее расписание, то из него можно простым поочередным сдвигом работ на более ранний срок получить раннее расписание, длина которого будет не больше, чем длина исходного расписания; следовательно, оно также будет оптимальным. Все ранние расписания можно получить, если на каждом этапе перебирать все варианты выбора $h \in D_n$. Таким образом, существует схема полного перебора, позволяющая найти оптимальное решение задачи. На этой схеме основан реализованный в LSE оптимальный метод ветвей и границ. Естественно, этот метод не является вычислительно эффективным (напомним, решаемая задача поиска оптимального расписания NP-трудна). Серийный метод просматривает лишь одну ветвь из этого дерева перебора, находя одно допустимое решение.

1.4.2. Пример работы серийного метода

Для примера рассмотрим проект строительства дома: Имеется 9 работ, выполняемых в определённом порядке и один дискретный ресурс — двое рабочих(эта задача описана в [3]). Вот таблица, определяющая ограничения предшествования и ресурсные ограничения:

Название	Длительность	Предшественники	Рабочие
Masonry	7		1
Carpentry	3	Masonry	1
Roofing	1	Carpentry	1
Plumbing	8	Masonry	1
Façade	2	Roofing, Plumbing	1
Windows	1	Roofing	1
Garden	1	Roofing, Plumbing	1
Ceiling	3	Masonry	1
Painting	2	Ceiling	1
Moving	1	Windows, Garden, Façade, Painting	1

Граф предшествований имеет при этом следующий вид (вес всех дуг равен нулю, что соответствует нулевому лагу):



Для безресурсной задачи можно легко вычислить ранги, ранние и поздние старты (жирным выделены работы из критического пути):

Название	Ранг	Ранний старт	Поздний старт
Masonry(1)	**0**	**0**	**0**
Carpentry(2)	1	7	11
Roofing(3)	2	10	14
Plumbing(4)	**1**	**7**	**7**
Façade(5)	**3**	**15**	**15**
Windows(6)	3	11	16
Garden(7)	4	15	16
Ceiling(8)	1	7	12
Painting(9)	2	10	15
Moving(10)	**5**	**17**	**17**

Это расписание оптимально для безресурсной задачи. Однако для исходной задачи оно не является допустимым: в момент времени $t = 7$ выполняются 3 работы, требующие в сумме 3 единицы ресурса, больше, чем имеется в наличии. Нетрудно понять, что оптимальное расписание исходной задачи не может иметь длину равную 18. Действительно, первая и последняя работы выполняются неодновременно с другими работами, на остальные же работы остается 10 единиц времени, в то время как их суммарная длина равна 21. Но одновременно может выполняться не более двух работ, значит, их выполнение не может занимать менее 11 единиц времени.

С другой стороны, просто построить допустимое расписание длины 19, которое и будет оптимальным. Рассмотрим работу серийного алгоритма с применением эвристики LST: на первом этапе $D_1 = \{1\}$, поэтому ставим работу 1 на самое раннее возможное время (0). Затем $D_2 = \{2,4,8\}$. Из этих работ будет выбрана 4, так как она имеет самый ранний поздний старт и поставлена на время $s_4 = 7$. Далее:

$D_3 = \{2,8\}$ (ни одна новая работа ещё не стала доступной), $h = 2, s_h = 7$;

$D_4 = \{3,8\}, h = 8, s_h = 10$ (как только закончится одна из двух выполняющихся работ)

$D_5 = \{3,9\}, h = 3, s_h = 13$

$D_6 = \{5,6,7,9\}, h = 5, s_h = 15$ (работа 5 — последователь 4)

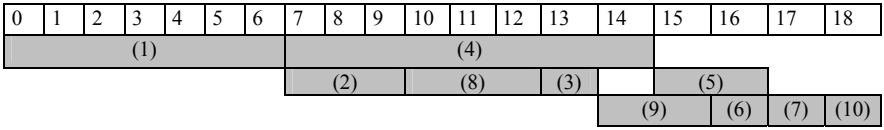
$D_7 = \{6,7,9\}, h = 9, s_h = 15$

$D_8 = \{6, 7\}, h = 6, s_h = 16$

$D_9 = \{7\}, h = 7, s_h = 17$

$D_8 = \{8\}, h = 10, s_h = 18$

Как и положено, s_h каждый раз выбиралось как самое раннее время, на которое можно поставить работу, удовлетворив все ограничения. В результате получено оптимальное расписание. Вот графическая иллюстрация построенного расписания (порядок построения — сверху вниз):



1.5. Метод стохастического поиска

Еще одним методом, используемым в LSE, является метод стохастического поиска (точнее, multi-pass Serial Method, см. [2]), основанный на серийном методе. Он осуществляет запуск серийного метода фиксированное число раз, при каждом запуске по-разному осуществляя выбор $h \in D_n$ на каких-то этапах и, тем самым, просматривая различные ветви дерева ранних расписаний. Из полученных расписаний выбирается лучшее. Простейший пример варьирования выбора — это использование в каждом запуске определенной эвристики, однако эвристик не так много. Другой способ — случайный выбор $h \in D_n$, при этом для достижения более высокого эффекта можно пытаться выбирать наиболее подходящее распределение случайной величины. Каждое такое распределение называется рандомизированной эвристикой стохастичесого метода. В LSE реализовано две таких рандомизированных эвристики: использующая равномерное распределение — на каждом этапе любая работа выбирается из D_n с одинаковой вероятностью:

$$prob(j) = \frac{1}{|D_n|}, \forall j \in D_n,$$

и неравномерная эвристика, в которой работы с большим значением $\nu(j)$ выбираются с большей вероятностью, например, таким образом:

$$\rho_j = v(j) - \min\{v(i) \mid i \in D_n\}, \forall j \in D_n,$$

$$prob(j) = \frac{(\rho_j + 1)^\alpha}{\sum_{i \in D_n} (\rho_j + 1)^\alpha}, \alpha \in [0, \infty).$$

При $\alpha = 0$ мы получаем равновероятное распределение, при $\alpha = \infty$ — детерминированный эвристический выбор, $\alpha = 1$ представляет некий промежуточный вариант. Тестирование этих вариантов на различных задачах показало, что наилучший рекорд при одинаковом количестве запусков серийного метода даёт равновероятный подход. Показательно, что при этом средняя длина получаемых расписаний — наихудшая. Это объясняется большей «дисперсией» получаемых расписаний. Если же выбор опирается на неравномерную эвристику, то мы просматриваем только «хорошие» расписания, однако лучшее из этих хороших проигрывает лучшему, выбранному из такого же числа абсолютно случайных расписаний.

Очевидно, время работы этого алгоритма в фиксированное число раз (количество запусков задается извне через интерфейс метода) больше времени работы серийного метода. Метод стохастического поиска показывает высокие результаты в смысле оптимальности решения. По умолчанию осуществляется 500 запусков с равновероятным выбором и 6 запусков с различными детерминированными эвристиками, при этом средняя погрешность на базе из примерно тысячи тестов составляет 5–10%.

1.6. Метод ветвей и границ

Как уже было упомянуто, в LSE реализован также алгоритм, позволяющий находить точное оптимальное решение задачи в базовой постановке. Он основан на методе ветвей и границ, относящемся к классу методов неявного перебора (см. [4]). По оценкам специалистов, в настоящее время это один из самых быстрых точных алгоритмов решения данной задачи (существуют также алгоритмы, основанные на методе динамического программирования и методе 0-1 программирования, см. [5] и [6]).

Две составляющие любого метода типа ветвей и границ это переборная схема (дерево) и отсекающие правила (границы). Как уже было сказано, переборная схема основана на переборе всех ранних расписаний, строящихся посредством серийного метода путём рассмотрения на каждом этапе всех вершин множества D_n. Количество ветвей в этом дереве равно количеству упорядочений множества работ $\{1, 2, ...N\}$, согласованных с ограничениями предшествования: если дуга $(i, j) \in E(G)$, то работа j должна сто-

ять правее *i*. Таким образом, чем больше ограничений предшествования, тем меньше ветвей имеет дерево перебора. Самый сложный случай для метода ветвей и границ — в отсутствии ограничений предшествования. В этом случае дерево перебора содержит $N!$ ветвей.

Отсекающее правило позволяет пропускать большое число бесперспективных с точки зрения оптимальности вариантов, основываясь на информации об имеющемся рекорде. Рекордом называется наилучшее из построенных к данному моменту времени решений, а также длина этого решения.

Пусть длина текущего рекорда равна *T*. Будем обозначать через s_j^{late} моменты стартов работ в *T*-позднем расписании соответствующей безресурсной задачи (в которой убраны все ресурсные ограничения и оставлены только ограничения предшествования).

Отсекающее правило довольно просто: если для выбранной работы $h \in D_n$ вычисленное раннее время старта $s_h \geq s_h^{late}$, то текущее частичное расписание ($S_n \cup \{h\}$) нет необходимости достраивать, так как длина любого полученного из него раннего расписания будет больше либо равна рекорду. Это достаточно очевидный факт ввиду того, что s_h^{late} — это по определению самое позднее время, когда мы можем начать работу h, чтобы получить расписание длины не более *T* (рекорда), удовлетворяющее всем ограничениям предшествования. Следовательно, все полученные на этой ветке ранние расписания будут иметь длину не меньшую, чем рекордная длина, и не дадут нам лучшего результата.

Менее тривиальным и более мощным является следующее отсекающее правило: если $s_h \geq s_h^{late}$, то нет необходимости достраивать частичное расписание, полученное после предыдущего этапа (S_n), то есть рассматривать другие варианты выбора из текущего D_n. Это можно понять из следующего наблюдения: если на этом шаге мы распишем какую-либо другую работу, то *h* придётся расписывать позднее и время её старта при этом может только увеличиться. То есть на любой из веток мы придём к первому случаю. Использование этого правила позволяет ускорить работу алгоритма примерно в два раза (по сравнению с первым).

К сожалению, в любом случае трудоёмкость этого алгоритма растёт экспоненциально, и он практически неприемлем для решения задач более чем с 10–30 работами.

1.7. Решение задач в расширенной постановке

1.7.1. Поддержка директивных сроков

Любой алгоритм решения задачи с ограничениями 3, 4, не являющийся точным, то есть находящий допустимое, но не оптимальное расписание, не может гарантировать даже нахождения допустимого расписания, в случае наличия ограничений типа 6 — директивных сроков. Действительно, для задачи, в которой директивный срок каждой из работ равен длине оптимального расписания, любое допустимое расписание является оптимальным.

Вследствие этого удовлетворение подобного рода ограничений может являться лишь желаемым, но не обязательным условием (как и, например, минимизация длины расписания) и должно достигаться эвристически с помощью субоптимальных алгоритмов. Например, работы, для которых задан директивный срок, а также для их предшественников, можно выбирать из D_n в первую очередь.

LSE в процессе решения задачи особо выделяет работы, для которых выполняется соотношение $r_j + d_j = u_j$. Для таких работ, которые мы будем называть фиксированными, в любом допустимом расписании $s_j = r_j$; из практических соображений бывает важно ни в коем случае не сдвигать такие работы с их места. В случае, если у фиксированной работы нет предшественников, а это типичный случай для многих классов реальных задач, она сразу ставится на своё место. Если же у неё есть предшественники, то они ставятся в расписание как только они попадают в множество допустимых работ. Таким образом, гарантируется, что фиксированные работы всегда будут стоять на своих исходных местах, если только это прямо не противоречит ограничениям предшествования.

Возможна ситуация, в которой работы, для которых задан директивный срок, или их предшественники конфликтуют в смысле первоочередности постановки в расписание с фиксированными работами и их предшественниками. Такая проблема может быть решена различными способами. С одной стороны, мы можем уравнять эти два класса работ в правах на первоочередность постановки в расписание; с другой стороны, можно всегда выбирать из множества допустимых работ в первую очередь фиксированные работы и их предшественников. По умолчанию в LSE приоритет в таком случае отдается фиксированным работам и их предшественникам.

1.7.2. Поддержка пулов ресурсов

Работа с пулами ресурсов — одна из отличительных особенностей LSE. Напомним, что пул представляет собой несколько однотипных ресурсов объединённых между собой (например, переносные компьютеры и настольные, являясь ресурсами различного типа, могут в то же самое время быть объединены в пул всех компьютеров). Пул наравне с обыкновенными ресурсами может быть назначен работам в некотором количестве. Это значит, что работа может использовать любые из ресурсов, входящих в пул, в суммарном количестве равном заданному в назначении. Таким образом, простой ресурс может быть назначен работе — как в качестве самостоятельного ресурса, так и в качестве элемента в составе пула.

Добавление пулов (равно как и других расширений) ещё более усложняет задачу, и без того NP-трудную, где нахождение точного решения практически возможно лишь для малых размерностей. Поэтому реализация точных алгоритмов для расширенных постановок не имеет смысла. Любые дальнейшие расширения постановки задачи рассматриваются в LSE лишь в контексте эвристических алгоритмов.

Итак, при фиксировании работы, которой назначены пулы ресурсов, мы можем использовать тот же жадный алгоритм, что и в серийном методе для базовой постановки. Однако помимо определения времени начала её выполнения (s_h) мы также должны выбрать, какие же именно ресурсы будут назначены этой работе. Логичной эвристикой в этом случае является назначение тех ресурсов, которые наименее загружены участием в выполнении ещё не расписанных работ. То есть, для каждого ресурса из пула мы вычисляем величину $z_r = \sum_{j \notin S_n} k_{jr} d_j$ и используем в первую очередь те ресурсы, для которых z_r минимальна. Легко также, используя это расширенное толкование серийного метода, обобщается и метод стохастического поиска.

1.7.3. Решение задач с работами нефиксированной длительности

В случае наличия работ с нефиксированными длительностями при установке работы помимо старта s_j и выбора ресурсов из пула мы должны также определить количество используемых в назначении ресурсов, из которого определяется длительность выполнения работы. При этом, однако, ничто не мешает использовать тот же алгоритм последовательной установки работ. В эвристику добавляется следующая часть, относящаяся к выбору

величины использования ресурса: устанавливаемая работа использует ресурсы так, чтобы минимизировать свою длительность (то есть использует все имеющиеся в наличии ресурсы в случае последовательного использования — чтобы минимизировать длины всех назначений; в случае параллельного — использует ресурсы так, чтобы минимизировать длину максимального назначения). Выбор ресурсов из пулов производится описанным выше способом.

2. СРЕДА РЕШЕНИЯ ЗАДАЧ ПЛАНИРОВАНИЯ LEDAS SCHEDULER

2.1. Особенности функциональности LEDAS Scheduler

Деятельность компании ЛЕДАС в области разработки средств задания, расчета и оптимизации расписаний с самого начала была направлена на создание решения, принципиально отличающегося от существующих на рынке приложений. Основными задачами, которые ставились перед коллективом разработчиков компании ЛЕДАС, были создание алгоритмической базы, способной работать с недоопределенными данными, задача оптимизации планов по произвольным критериям и спецификация требований к конструируемому плану в виде произвольных ограничений. Данные условия не только выгодно отличают приложение, созданное на базе разработанных методов решения задач с произвольными ограничениями, но и делают его чуть ли не единственным инструментом, способным решать столь общие задачи.

LEDAS Scheduler 3.0 поддерживает следующие виды объектов:

- Работы
- Ресурсы
  - – унарные
  - – расходуемые
  - – дискретные
- Пулы унарных ресурсов
- Ограничения
  - – предшествования, директивные сроки, ресурсные (в том числе параметризованные)
  - – пользовательские (написанные на языке Scheduler)

Связи между работами и ресурсами (пулами) формируются заданием назначений, которые могут быть как обязательными для выполнения (жесткими), так и лишь желательными (мягкими). С помощью назначений задается и объем потребления ресурсов.

Все вышеперечисленные объекты, кроме пулов и ограничений, могут содержать внутренние, локальные слоты и ограничения. Общие черты в структуре таких объектов объединяются понятием «класса». Каждый объект по умолчанию является экземпляром некого стандартного класса, однако пользователь имеет возможность определять собственные классы, расширяя стандартный набор слотов и ограничений. При модификации класса соответствующим изменениям подвергаются все объекты, принадлежащие данному классу.

Целиком задача планирования представляется в LEDAS Scheduler в виде специальной сущности — проекта. Наборы наиболее часто используемых классов можно сгруппировать в библиотеки классов, которые могут быть импортированы разными конкретными проектами. Параметризованные пользовательские ограничения также могут быть оформлены в виде классов ограничений и включены в библиотеку.

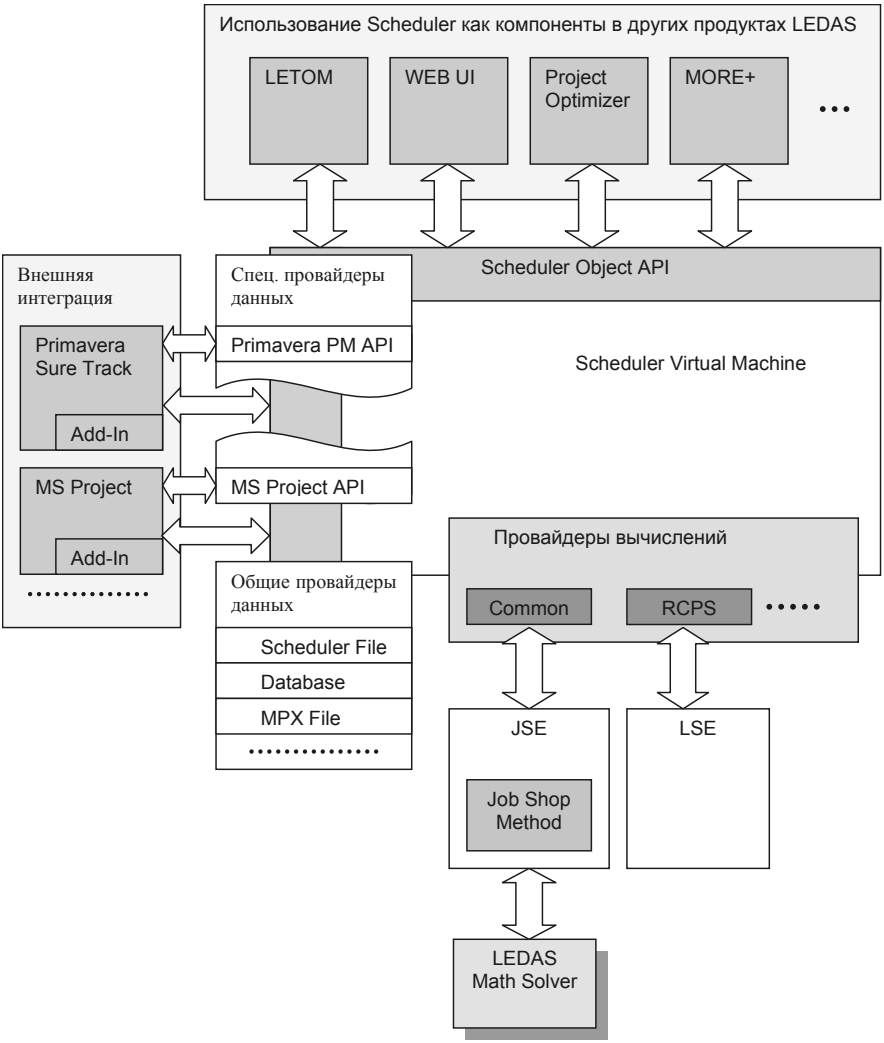
Критерий оптимизации задачи указывается пользователем путем задания в модели соответствующей цели. Типичными целями являются минимизация общей длительности проекта и минимизация суммарного расхода ресурса.

2.2. Архитектура LEDAS Scheduler

Ниже на рисунке изображена архитектура LEDAS Scheduler с точки зрения возможности развития системы для решения практических задач.

Архитектура LEDAS Scheduler проектировалась с учетом следующих целей:

- максимальная переносимость на разные платформы;
- поддержка различных форматов/источников данных;
- потенциальная возможность встраивания вычислительной части LEDAS Scheduler в другие системы;
- расширяемость вычислительной части LEDAS Scheduler дополнительными/специализированными вычислительными методами.



В состав среды LEDAS Scheduler 2.5 входило шесть модулей:

- JSVM — виртуальная машина LEDAS Scheduler;
- JSE — вычислительный модуль;

- LEDAS Math Solver — внешний универсальный вычислитель;
- JSDataProvider — набор модулей поддержки различных форматов/источников данных;
- JSCalcProvider — набор модулей подключения специализированных вычислительных модулей;
- JSUI — графический пользовательский интерфейс.

В версии 3.0 к ним добавилось новое вычислительное ядро LSE.

JSVM

Виртуальная машина LEDAS Scheduler осуществляет все операции по редактированию расписаний. JSVM предоставляет высокоуровневый объектно-ориентированный программный интерфейс (API) — набор абстрактных C++ классов. В нем имеются специализированные классы для объектов задач планирования (IJSJob, IJSResource и другие), а также дополнительные классы для работы с проектами, сессиями, провайдерами (IJSProject, IJSVM и другие). Этот API позволяет создавать несколько сессий одновременно, загружать данные из разных источников, модифицировать данные, получать решение с использованием разных вычислителей, сохранять изменённую модель и найденные решения, получать уведомления об изменениях в проекте, объединять проекты. Для сохранения/чтения данных JSVM обращается к JSDataProvider, а для перевычисления расписаний обращается к JSCalcProvider.

JSDataProvider

Данное семейство модулей поддерживает общий протокол обмена данными с модулем JSVM. Каждый модуль из семейства JSDataProvider обеспечивает доступ к определенному формату/источнику данных. В настоящее время реализованы три модуля из этого семейства:

- **JSFileProvider** — сохраняет данные в файлах специализированного формата LEDAS Scheduler;
- **JSODBCProvider —** использует ODBC-совместимые источники данных (например, СУБД MS Access);
- **JSMPXProvider —** осуществляет чтение/запись файлов в формате MPX, что позволяет обмениваться данными с другими приложениями класса Project Management.

JSCalcProvider

Как и JSDataProvider, это семейство модулей, поддерживающих протокол взаимодействия виртуальной машины LEDAS Scheduler и подключенных к нему вычислительных модулей. Примером может служить модуль JSGenericProvider, осуществляющий взаимодействие с универсальным вычислителем JSE (который, в свою очередь, использует LEDAS Math Solver), а также модуль JSFastProvider для подключения специализированного решателя задач планирования LSE.

JSE

Универсальный вычислитель для решения широкого класса задач планирования, JSE предоставляет объектно-ориентированный программный интерфейс в виде набора абстрактных C++ классов, который позволяет создавать модель, модифицировать её, находить решения, используя разные методы и эвристики.

Вычислительный модуль JSE интегрируется в LEDAS Scheduler при помощи вычислительного провайдера для задач общей постановки JSGenericProvider. Так как JSE полностью поддерживает всю объектную модель Scheduler (включая ограничения, задаваемые пользователем), провайдер для него оптимальным образом реализует передачу всей модели, поддержку счёта, получение найденного решения, а также поддержку разбора и компиляции пользовательских ограничений. JSE содержит в себе специальный эффективный метод решения задач класса Job Shop.

LEDAS Math Solver

Это универсальный внешний математический решатель, обладающий средствами для моделирования широкого спектра математических задач с недоопределенными данными и набором методов поиска решений. Структура задачи, решаемой LEDAS Math Solver, может быть произвольной, содержащей различного рода ограничения над целочисленными, вещественными, логическими и множественными переменными.

JSUI

Это графический пользовательский интерфейс, который обеспечивает взаимодействие пользователя с системой посредством таблиц, графиков, и т.п. Всю информацию об объектах плана получает от JSVM и туда же передает команды пользователя.

LSE

Вычислительный модуль LSE интегрируется в Scheduler при помощи вычислительного провайдера JSFastProvider. Сейчас специализированный вычислительный модуль LSE поддерживает решение ограниченного класса задач, поэтому его вычислительный провайдер проверяет принадлежность задачи этому классу. При положительном результате проверки провайдер формирует задачу в терминах вычислителя LSE, передаёт её на счёт, получает найденное решение.

LSE поддерживает унарные и дискретные ресурсы, ограничение предшествования и неодновременности, пулы унарных ресурсов, иерархии работ, директивные сроки. Некоторые сущности на данный момент поддерживаются ограниченно, так, например длительность работ может быть только точной, в то время как среда LEDAS Scheduler поддерживает работы нефиксированной длительности. Некоторые сущности не поддерживаются совсем: пользовательские классы работ, ресурсов и ограничений, потребляемые ресурсы, переменные и пользовательские ограничения. Эти сущности не нужны для формулирования большинства практически важных задач, тем не менее, дальнейшее совершенствование LSE предполагает в будущем и их поддержку. Есть также некоторые возможности, которые поддерживаются только этим вычислительным модулем, например, прерывание и продолжение поиска решения, пулы дискретных ресурсов.

2.3. Сравнение LEDAS Scheduler с аналогами

LEDAS Scheduler можно сравнивать, с одной стороны, с приложениями класса project management, лидером среди которых является MS Project (см. `http://www.microsoft.com/office/project`), с другой стороны, со специализированными процессорами предназначенными для решения задач планирования, такими, как ILOG Scheduler
(см. `http://www.ilog.com/products/scheduler`).

2.3.1. Сравнение с MS Project

MS Project — это удобное средство для «ручного» составления и сопровождения планов работ небольших проектов. В нем широко представлены средства для анализа структуры и сопровождения проектов — различные виды диаграмм, отчетов, сортировки, группировки, табличные функции и т.д. Также есть возможность организовать распределенную работу с проектами — имеются средства для одновременного доступа к данным через

сеть, организована рассылка сообщений и т.д. Автоматизация расчета планов представлена в MS Project на сравнительно низком уровне, и ограничивается обнаружением противоречивых данных на этапе их редактирования, а также расчетом допустимых расписаний, который могут быть достаточно далеки от оптимальных. Развитие LEDAS Scheduler, напротив, было в основном направлено на повышение эффективности вычислительной части. Поэтому сравнивать вычислительные возможности LEDAS Scheduler и MS project просто нельзя — LEDAS Scheduler предоставляет средства на порядок мощнее, чем MS Project.

Пользовательский интерфейс LEDAS Scheduler содержит минимальный набор средств отображения/редактирования данных (табличные представления, диаграмма Гантта, PERT, диаграмма ресурсов). Однако этот набор может быть расширен как в рамках существующего модуля JSUI, так и в рамках нового интерфейсного модуля, подключенного к JSVM.

Средств распределенной работы с проектами в LEDAS Scheduler пока нет, но продуманная архитектура системы позволит подключить их без разрушения целостности всей системы.

2.3.2. Сравнение с ILOG Scheduler

ILOG Scheduler представляет собой надстройку над ILOG Solver — библиотеку C++ классов, предназначенных для решения задач планирования. Библиотека содержит классы, соответствующие сущностям таких задач (ресурсы, работы, ограничения), а также набор мощнейших современных алгоритмов, каждый из которых является если не лучшим, то одним из лучших среди известных на сегодняшний день алгоритмов для решения задач определенного класса.

Кроме самих алгоритмов, ILOG Scheduler также предоставляет довольно гибкие схемы для управления ими. Все вместе образует набор средств для построения приложений, эффективно решающих разные классы задач планирования. Библиотека является расширяемой, т.е. в случае необходимости пользователь может добавить к системе новый инструмент (тип ограничений или алгоритм), написав соответствующий класс на C++.

В LEDAS Scheduler аналогичные функции выполняет один из модулей системы — JSE. Он, аналогично ILOG Scheduler, состоит из двух компонент — базового решателя (LEDAS Math Solver) и объектно-ориентированной библиотеки работ/ресурсов/ограничений/алгоритмов. В отличие от ILOG Scheduler, LEDAS Scheduler поддерживает объектно-ориентированную модель задачи, не только на уровне разработчика приложения, но и на уровне конечного пользователя, т.е. пользователь может

самостоятельно вводить новые классы работ/ресурсов и определять ограничения, присущие лишь объектам этих классов.

JSE позволяет формулировать на высоком уровне достаточно широкий круг задач (не меньший, чем у ILOG Scheduler, и не вполне с ним совпадающий). Некоторые классы задач, такие как задачи класса Job Shop, поддержаны в JSE специальными алгоритмами поиска решений, и его производительность при решении этих задач сравнима с производительностью ILOG Scheduler. Вычислительная мощность при решении общих задач, которые можно сформулировать в LEDAS Scheduler, в версии 2.5 оставляет желать лучшего, так как при решении задач используется один универсальный метод. Эта проблема решается, во-первых, выделением новых важных подклассов задач, и реализацией для них специализированных алгоритмов, во-вторых, подключением внешних решателей, таких как LSE, через механизм JSCalcProvider.

Если сравнивать LEDAS Scheduler и ILOG Scheduler с точки зрения построения специализированных приложений, то LEDAS Scheduler обладает рядом преимуществ. ILOG Scheduler оставляет за скобками все, что не относится непосредственно к решению задачи. Редактирование данных, их хранение и обмен данными с другими приложениями — все эти вопросы должен решать разработчик приложения, использующего ILOG Scheduler

LEDAS Scheduler предоставляет интегрированное решение, облегчающее построение специализированных приложений на его основе:

- Виртуальная машина LEDAS Scheduler (модуль JSVM) представляет собой API, при помощи которого происходит редактирование данных, при этом, насколько возможно, автоматически поддерживается целостность редактируемых данных.
- Семейство модулей JSDataProvider поддерживает несколько форматов и источников данных для сохранения рассчитываемых проектов. Один из модулей этого семейства, JSMPXProvider, реализует импорт/экспорт данных в файлы формата MPX, являющегося стандартом обмена данными между приложениями класса Project Management.
- Продвинутые разработчики прикладных программ имеют возможность создать свой модуль семейства JSDataProvider, для чего предоставляется соответствующий API.

3. ПРИКЛАДНЫЕ ПРОЕКТЫ

Специалисты компании ЛЕДАС видят широкую сферу приложения для ядра LSE и LEDAS Scheduler в целом. Планирование является неотъемлемой частью разнообразных бизнес-процессов: это проектное планирование в сфере IT и планирование загрузки оборудования на промышленных предприятиях, планирование вылетов самолётов в авиакомпаниях и планирование собраний сотрудников компаний.

Планирование также является важной составной частью в процессах управления жизненным циклом изделий (PLM) и потому разработка LEDAS Scheduler укладывается в стратегию iPLM, провозглашенную компанией ЛЕДАС (см. [7], а также http://ledas.com/iplm.php). В жизненном цикле изделия планирование возникает практически на всех этапах от проектирования и прототипирования изделия до его производства и постпродажного сопровождения. Поэтому мы считаем, что ядро для решения задач планирования может быть естественным образом интегрировано в комплексные PLM-системы. Компания ЛЕДАС не занимается разработкой такого рода систем самостоятельно и приглашает к сотрудничеству заинтересованные компании.

Проект LEDAS Scheduler и лежащее в его основе решение представляют собой достаточно общие для задач удовлетворения и оптимизации ограничений подходы, сформулированные в общих же терминах. Естественным образом возникает желание специфицировать задачу под конкретные условия и бизнес-логику определенной сферы применения.

На данный момент компания ЛЕДАС участвует в двух проектах по интеграции ядра LEDAS Scheduler в конечно-пользовательские системы — это система планирования собраний и встреч MORE+ и система проектного планирования для платформы Eclipse. Оба проекта используют современные программные и аппаратные платформы. Ожидается, что они будут востребованы на рынке подобных систем и смогут конкурировать с существующими системами подобных классов. Остановимся на обоих проектах чуть более подробно.

3.1. Система планирования собраний и встреч MORE+

3.1.1. Описание функциональности системы MORE+

Идея создания системы автоматизации составления собраний возникла в компании ЛЕДАС естественным образом, как только проектная база при-

близилась к порогу, за которым участие одних и тех же специалистов в различных проектах стало необходимой и неотъемлемой частью производственного процесса компании. Правильная организация работ осложнялась, кроме того, и тем, что некоторые из проектов требовали привлечения внешних исполнителей для узкоспециализированных задач.

Тем самым возникла задача составления оптимального расписания, удовлетворяющего условиям индивидуальных планов работ каждого из разработчиков в каждом из проектов компании. Решение должно быть основано на наличии эффективного доступа к системе с возможностью инициации и сопровождения собраний с мобильных устройств, таких как карманные компьютеры, персональные ассистенты, мобильные и smart-телефоны, а также — на способности системы самостоятельно и автоматически разрешать все возникающие конфликты входных данных с минимальным участием участников собраний. Условие простоты работы со стороны пользователей и администраторов системы мы оставляем за рамками условий задачи, так как это требование присутствует во всех проектах, не предполагающих обширной унификации и покрытия широкого числа специальных областей применения.

Предполагается широкое использование мобильных терминалов (мобильных телефонов и PDA) для работы с системой. Само планирование будет вестись на сервере, к которому будет возможно подключиться по специальному протоколу с мобильных устройств (с использованием Wi-Fi или GPRS), а также через web-интерфейс. В будущем предполагается также разработка плагинов к самым распространённым системам персонального планирования, таким как Microsoft Outlook и Lotus Notes. С любого терминала можно будет инициировать собрания, и получать запросы о персональной доступности и уведомления о запланированных собраниях.

Система может быть востребована в компаниях, деятельность которых предполагает общение между сотрудниками и/или с участием клиентов и партнёров. Это может быть и динамичная IT-компания, в которой проводится много семинаров для обсуждения текущих проектов, и крупная торговая компания, в которой сотрудники учатся и обмениваются опытом, и сервисная компания, в которой регулярные встречи с партнёрами и клиентами являются основой успешного бизнеса. Принимая во внимание узкую направленность вышеприведенных условий по сравнению с алгоритмической базой проекта LEDAS Scheduler, очевидным образом возникает задача кастомизации решений из LEDAS Scheduler и, соответственно, упрощение приложения для его реализации. Далее мы кратко опишем возможности приложения MORE+.

MORE+ предназначена для автоматизации согласования времён собраний и встреч. Она будет автоматически учитывать уже запланированные мероприятия для участников очередного собрания, а также персональные ограничения на возможности участия в планируемом собрании. При этом система не только будет находить максимально удобное для участников время, но и, при необходимости, перепланировать другие мероприятия, в случае если без этого обойтись нельзя.

В отличие от большинства подобных систем, MORE+ будет сама сводить индивидуальные расписания участников, графики доступности аудиторий для встреч и выдавать построенное расписание, не требуя от организатора поиска подходящего всем времени и согласования этого времени со всеми участниками.

3.1.2. Математическая постановка задачи MORE+

В задаче имеются следующие сущности:

1. Множество собраний. Каждое собрание — работа фиксированной длительности, связанная несколькими ограничениями использования ресурсов — сотрудников, и, опционально, аудиторий, оборудования, и прочих ресурсов.
2. Множество личных дел. Каждое личное дело есть работа фиксированной длительности и фиксированного положения в абсолютном времени, использующая ровно один ресурс — некоторого сотрудника.
3. Множество личных ограничений на собрания. Каждое такое ограничение выражает невозможность конкретного сотрудника участвовать в конкретном собрании в определенное время (например, вследствие его неготовности к данному собранию). Моделируется работой фиксированной длительности и фиксированного положения в абсолютном времени, имеющей ровно одно ограничение неодновременности с некоторым собранием. Ограничение неодновременности моделируется введением дополнительного унарного ресурса, который более нигде не используется.
4. Множество сотрудников. Каждый сотрудник есть унарный ресурс, который может быть использован в нескольких работах — собраниях и личных делах.
5. Множество аудиторий. Каждая аудитория есть унарный ресурс, который может быть использован в нескольких работах-собраниях, причем каждая работа использует не более одного такого ресурса.
6. Множество других ресурсов (оборудование и прочее). Это могут быть произвольные унарные и множественные дискретные ресурсы.

7. Множество ограничений использования ресурса (assignments). Это классические ограничения, указывающие, какие ресурсы какие работы используют (а если ресурс множественный, то — и в каком количестве).
8. Есть ограничения привязки к абсолютному времени — для личных дел и личных ограничений на собрания они жестко фиксируют положение работы, для собрания — позволяют ему изменяться в некотором интервале.
9. Есть набор статусов собраний, для каждого указывается, является ли он уже установленным или нет, и стартовое расписание — в нем проставлены времена проведения установленных собраний и предполагаемые пользователем времена проведения неустановленных собраний. Предполагается, что при решении в первую очередь будут смещаться неустановленные собрания, и уже во вторую очередь — установленные собрания.

В отличие от типичных задач LSE, в задачах MORE+ нет ограничений предшествования (кроме привязок к абсолютному времени — release and due dates), работ нефиксированной длительности, недискретных и потребляемых ресурсов, ресурсных пулов. Следовательно, задача MORE+ является частным случаем задач LSE в базовой постановке, которая уже сейчас эффективно решается LSE.

Тем не менее, у нее есть своя специфика:

1. Большое количество фиксированных работ. Однако специальный способ обработки задач с фиксированными работами в LSE позволяет разумным образом решать такие задачи.

2. Высокое внимание к натуральности решения. Под этим термином подразумевается сохранение, насколько это возможно, порядка работ и времен их стартов, причем в соответствии с заданными приоритетами (установленное/неустановленное собрание). Во многих практических задачах натуральность решения не менее важна, чем его оптимальность. В данный момент в рамках проекта LSE производится разработка методов, обеспечивающих не только (суб)оптимальность, но и натуральность решения задачи планирования.

3.2. Система проектного планирования для платформы Eclipse

Системы проектного планирования широко используются менеджерами всех уровней. Безусловный лидер среди подобных систем — Microsoft Pro-

ject, проданы миллионы копий этой системы и продажи ежегодно растут. К плюсам этой системы, безусловно, стоит отнести тщательно продуманный и очень удобный пользовательский интерфейс, большой набор представлений данных как для отображения на экране, так и в форме отчётов, а также интеграцию с приложениями из состава Microsoft Office (напр. Microsoft Outlook). Минусами являются ограниченные возможности моделирования реальных проектных задач (отсутствуют пулы ресурсов, поддержка расходуемых ресурсов имеет существенные ограничения и т.д.), вычислительные возможности для оптимального планирования и анализа рисков практически отсутствуют, работа только в среде Microsoft Windows — с распространением Linux этот фактор приобретает существенное значение.

Компании ЛЕДАС и Xored Software совместно разрабатывают систему проектного планирования на платформе Eclipse. Система разрабатывается как улучшенный аналог Microsoft Project. Он будет работать на всех ОС, поддерживаемых платформой Eclipse, в том числе Linux. Ядро LSE в системе обеспечит вычислительные возможности гораздо более широкие, чем в продукте от Microsoft, в том числе и оптимизацию расписаний работ с учётом ограничений на ресурсы. Кроме того, система будет поддерживать более широкий класс задач планирования — ядро LSE обеспечит возможности планирования с пулами альтернативных ресурсов и с различными схемами использования расходуемых ресурсов (например, расходования ресурсов простыми ресурсами).

С пользовательской точки зрения система будет близка к Microsoft Project. Она будет интегрирована со средствами разработки на платформе Eclipse, что сделает её особенно привлекательной для IT-компаний, использующих другие средства на платформе Eclipse.

4. ЗАКЛЮЧЕНИЕ

Новый решатель для ограничений в задачах планирования от компании ЛЕДАС находится сейчас в стадии активной разработки. Весной-летом 2005 года его коммерческая версия будет доступна для лицензирования. Примерно в это же время появятся первые приложения на его основе. Мы приглашаем все заинтересованные компании к сотрудничеству в разработке, развитии, продвижении LEDAS Scheduler 3.0 и интеграции в свои приложения.

Уже сейчас новое ядро LSE проекта LEDAS Scheduler показывает отличное качество получаемых результатов. На базе из нескольких тысяч тестов, включающей задачи из 15 различных классов, новый решатель добивается хорошего качества субоптимальных решений, показывает исключительную производительность и масштабируемость. Интеграция нового ядра LSE в среду LEDAS Scheduler позволила значительно повысить эффективность последней.

Говоря о проекте LEDAS Scheduler хочется подчеркнуть важный на наш взгляд факт. Во-первых, созданное решение расширяет линейку разрабатываемых в компании ЛЕДАС решателей. Так, при упоминании ядра проекта LEDAS Scheduler естественным образом возникает серия решателей от компании ЛЕДАС: геометрический решатель для средств автоматизированного проектирования в продуктах LGS 2D и LGS 3D (http://lgs.ledas.com и http://lgs3d.ledas.com); используемый в LEDAS Scheduler 2.5 математический решатель для разрешения и оптимизации задач с ограничениями LEDAS Math Solver (http://ledas.com/solver.php); решатель для работы с задачами удовлетворения ограничений в условиях совместного распределенных сред LEDAS Collaborative Solver (http://ledas.com/lcs.php). Нам хочется подчеркнуть преемственность проекта LEDAS Scheduler 3.0 в продуктовой линейке, связанной со средствами создания и сопровождения планов работ.

В планах компании дальнейшее развитие LSE как в сторону повышения эффективности решения, так и в сторону расширения класса решаемых задач. Одновременно с развитием LSE будут развиваться сама система LEDAS Scheduler, как и продукты на ее основе. Так, система проектного планирования получит средства анализа рисков, возможных сценариев развития проекта и стоимостного анализа. Система планирования встреч не только пополнится новыми клиентами для различных платформ и приложений, но и получит более развитые средства управления планированием, такие как приоритеты собраний и участников. Маркетинг решений на основе LSE также поможет определить направления дальнейшего развития как собственно приложений, так и самого ядра.

СПИСОК ЛИТЕРАТУРЫ

1. Brucker, P. et al: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research, 112 (1999) 3–41
2. Kolish, R.: Serial and Parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research, 90 (1996) 320–333
3. Le Pape, C.: Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems. Intelligent Systems Engineering, 3(2) (1994)
4. Hartmann, S., Drexl, A.: Project scheduling with multiple modes: a comparison of exact algorithms. Networks, 32 (1998) 283–297
5. Carruthers, J.A., Buttersby, A.: Advances in critical path methods. Operational Research Quarterly, 17 (1966) 359-380
6. Patterson, J.H., Roth, G.W.: Scheduling a project under multiple resource constraints: A zero-one programming approach. AIIE Transactions, 8 (1976) 449-455
7. Ushakov, D.: Adding intelligence to software solutions for PLM: New LEDAS strategy. LEDAS preprint, 12 (2004)

А. Ершов, И. Иванов, В. Корниенко, С. Прейс, А. Рассказов, И. Рыков

LSE: НОВОЕ ВЫЧИСЛИТЕЛЬНОЕ ЯДРО СИСТЕМЫ ПЛАНИРОВАНИЯ LEDAS SCHEDULER 3.0 И ПЕРСПЕКТИВЫ ЕГО ИСПОЛЬЗОВАНИЯ

Препринт
15

Рукопись поступила в редакцию 11.01.2005
Рецензент Д. Ушаков
Редактор О. Дробышевич

Подписано в печать 26.01.2005
Формат бумаги 60 × 84 1/16
Тираж 50 экз.

ЗАО РИЦ «Прайс-курьер»
630090, г. Новосибирск, пр. Акад. Лаврентьева, 6, тел. (383-2) 30-72-02