На правах рукописи

Ворожцов Артём Викторович

Построение алгоритмов ортогонального представления графа с указанными портами рёбер

Специальность 05.13.18 –
Математическое моделирование,
численные методы и комплексы программ

Автореферат диссертации
на соискание учёной степени
кандидата физико-математических наук

МОСКВА – 2005

Работа выполнена на кафедре информатики Московского физико-технического института (государственного университета)

Научный руководитель:

доктор физико-математических наук, профессор Малинецкий Г.Г.

Официальные оппоненты:

доктор физико-математических наук —
кандидат физико-математических наук —

Ведущая организация:

Кафедра информатики Московского физико-технического института (государственного университета)

Защита состоится _____ в _____ на заседании диссертационного совета К 212.156.02 при Московском физико-техническом институте (государственном университете) по адресу: 141700, Московская обл., г. Долгопрудный, Институтский пер., д. 9.

С диссертацией можно ознакомиться в библиотеке МФТИ.

Автореферат разослан _____.

Учёный секретарь
диссертационного совета Федько О. С.
кандидат физико-математических наук

Общая характеристика работы

Актуальность темы

Системы автоматической визуализации данных сегодня востребованы и активно развиваются. Это связано с увеличением объёмов информации и сложности структур. Требуются инструменты для наглядного представления и анализа информации.

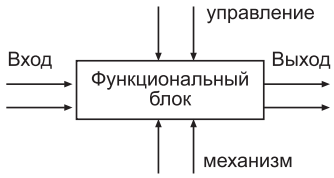


Рис. 1

Актуальной задачей является автоматическое построение организационных IDEF-диаграмм. IDEF (ICAM Definition) — это стандарт моделирования технологических операций, компьютерных систем и бизнес-процессов. В частности стандарт IDEF0 касается моделирования управляющих бизнес-процессов предприятия. Этот стандарт описывает правила представления схемы управления предприятия в виде набора функциональных блоков, соединённых ломанными линиями. Блоки изображаются прямоугольниками, стороны которых называются *портами*. В верхнюю часть функционального блока (северный порт) идут стрелки от функциональных блоков, которые им управляют, в западный порт поступает вход, из восточного порта идёт выход, в южный порт выходят стрелки от блоков, определяющих механизм работы данного блока (см. рис. 1).

Задача представления графов, для ребер которых указаны порты, возникает при построении не только IDEF-диаграмм, но и ряда других схем, например UML-диаграмм, dragon-схем и др.

Цели работы

Целью работы была разработка системы автоматического построения IDEF-диаграмм, то есть системы, решающей задачу оптимального ортогонального представления графов с указанными портами рёбер (Optimal Orthogonal Graph Layout with Ports, OOGLP), где оптимальность определяется функцией штрафа, зависящей от числа пересечений и изгибов представления рёбер, а также от площади представления.

Первой задачей стала разработка алгоритма решения задачи OOGLP, основанного на математическом моделировании физической системы с потенциальной энергией, соответствующей функции

штрафа за представление графа.

Следующей задачей стало исследование возможности применения известных методов отжига, масштабирования и метода последовательных локальных улучшений и сравнение результативности их применения по отдельности и при комбинировании.

Третьей важной задачей работы стала классификация общих методов (метаэвристик) решения $\mathcal{NP}$-сложных и трудно формализуемых сложных задач, исследование различных возможностей применения этих методов при решении задачи OOGLP, в частности, исследование базовых методов, основанных на сведении оптимизационных задач к численному моделированию физических систем.

Методы исследования

Для решения поставленных задач использовался метод сведения оптимизационной $\mathcal{NP}$-сложной задачи к построению физической модели и математическому моделированию. Также использовались методы последовательных локальных улучшений, жадные алгоритмы, метод масштабирования, и спектральные методы поиска оптимального порядка вершин графа.

Для проведения численных исследований, визуализации данных и сравнительного анализа методов были использованы системы `graphviz` и Mathematica.

При построении программного комплекса использовались языки программирования C и Perl.

Научная новизна

Только некоторые этапы решения задачи OOGLP можно свести к ряду известных $\mathcal{NP}$-сложных оптимизационных задач, в частности, к задаче поиска оптимального линейного порядка вершин графа (Optimal Linear Arrangement, OLA) и задаче поиска максимального ациклического подграфа ориентированного графа (задача MAXACYCL). Можно рассматривать задачу OOGLP как обобщение задачи OLA на двумерный случай. Задачи OLA и MAXACYCL подробно исследованы, но разработанные методы их решения не имеют хороших оценок качества работы. Задача OOGLP ранее в научных работах не рассматривалась[1].

[1] Поиск работ по темам связанным с задачей OOGLP осуществлялся по аннотированным библиографическим спискам “Теория представления графов”, в архиве препринтов `http://arxiv.org/`, статьях журнала “Journal of Graph Algorithms and Applications”, и системе `http://citeseer.ist.psu.edu`.

Известный точный метод решения задачи поиска оптимального ортогонального представления графа степени 4 для случая, когда порты рёбер не фиксированы, оказался не применимым к задаче OOGLP.

Поэтому, решение задачи OOGLP требовало разработки новых алгоритмов.

Для задачи поиска оптимального представления графа впервые получены результаты относительно качества различных алгоритмов для различных типов графов и исследована эффективность "сложения алгоритмов" (алгоритм, являющийся сложением двух приближённых оптимизационных алгоритмов, по определению есть алгоритм, выбирающий лучший из результатов, полученных двумя данными алгоритмами).

В работе исследованы базовые методы (метаэвристики) $\mathcal{NP}$-программирования, которые можно использовать при конструировании приближённых алгоритмов решения $\mathcal{NP}$-сложных и плохо формализуемых алгоритмических задач. Обзор таких мета-методов (metaheuristics) можно найти в книгах серии "Operations Research / Computer Science Interfaces"[2]. В работе описан новый метод **macro** — метод введения макро-сущностей: макро-объектов, макро-действий и макро-языка. Описаны общие принципы осуществления метасистемных переходов на уровне конструирования алгоритма, в частности новый подход решения сложных задач, в котором задача выделения макро-сущностей решается алгоритмом, а не человеком.

Разработан новый метод сравнительного анализа эвристических алгоритмов. Введены важные понятия надёжности и полезности эвристических алгоритмов и описаны методы экспериментальной оценки этих величин. Разработанный метод сравнительного анализа применён к реализованным эвристическим алгоритмам, приближённо решающих задачу OOGLP.

Практическое значение работы

Результаты работы могут быть использованы на предприятиях в системах моделирования и контроля бизнес процессов.

[2]Metaheuristics, Progress as Real Problem Solvers Series: Operations Research /Computer Science Interfaces Series, Vol. 32 Ibaraki, Toshihide; Nonobe, Koji; Yagiura, Mutsunori (Eds.) 2005, XII, 414 p. 106 illus., Hardcover.

Essays and Surveys in Metaheuristics Series: Operations Research/Computer Science Interfaces Series, Vol. 15 Ribeiro, Celso C.; Hansen, Pierre (Eds.) 2001, 664 p., Hardcover.

Разработанная система (программный комплекс) позволяет менее чем за минуту получать IDEF-диаграммы, содержащие менее 200 узлов.

Кроме того, разработанные алгоритмы могут быть применены для визуализации произвольных графов, где сторона выхода в узел и сторона выхода из узла зафиксированы для всех рёбер.

Разработанные алгоритмы будут включены в систему моделирования бизнес процессов Verball[3].

Разработанные методы экспериментальной оценки надёжности и полезности эвристических алгоритмов могут быть успешно применены для анализа других известных приближенных алгоритмов решения оптимизационных задач.

Публикации

По теме диссертации опубликовано 9 работ, из них 4 препринта, 2 статьи в реферируемых журналах, 3 тезизов докладов.

Структура диссертации

Диссертация состоит из введения, семи глав и заключения, изложенных на 120 страницах и иллюстрирована 26 рисунками. Библиографический список включает 65 наименований.

Содержание работы

Введение

В введении описана структура диссертации и сформулированы основные положения диссертации. Рассматриваемая в диссертации задача ортогонального представления графов с указанными портами рёбер (Optimal Orthogonal Graph Layout with Ports, OOGLP) описывается следующими условиями:

а) граф произвольный;

б) ребра представляются “ортогональными” ломанными, то есть ломанными, звенья которых параллельны осям координат;

в) вершины (узлы) представляются прямоугольниками заданных размеров;

[3]Verball — инструмент построения информационных систем предприятий, `http://orgway.org`.

г) для каждого ребра указана сторона узла, откуда оно должно выходить, и сторона узла, куда оно должно входить;

д) необходимо минимизировать следующую целевую функцию:

$$\begin{aligned}\text{Cost} = \quad & \alpha_1 \cdot (\text{длина рёбер}) + \\ & \alpha_2 \cdot (\text{число изгибов рёбер}) + \\ & \alpha_3 \cdot (\text{число пересечений рёбер}) + \\ & \alpha_4 \cdot \sqrt{(\text{площадь ограничивающего прямоугольника})}.\end{aligned}$$

Пункт (г) здесь наиболее важен. Ранее представления с таким ограничением не рассматривались. Эта задача более общая нежели задача построения IDEF-схем.

В общем случае эта задача является $\mathcal{NP}$-сложной, поэтому полиномиальные алгоритмы могут решать эту задачу только приближённо.

Глава 1. Обзор задач и алгоритмов теории представления графов

Первая глава содержит обзор задач и алгоритмов, связанных с поиском оптимальных представлений графов на плоскости.

Рассмотрены известные компьютерные приложения и библиотеки алгоритмов, позволяющие решать задачи визуализации и анализа графов. Особое внимание уделено системе `graphviz`.

Ключевые работы и библиографические списки работ по теме визуализации графов приведены в аннотированной библиографии Р. Тамассия и др.[1]

Ключевым моментом решения многих задач представления графов оказывается поиск оптимального линейного порядка вершин ориентированного графа с взвешенными рёбрами (Optimal Linear Arrangement, OLA). Это $\mathcal{NP}$-сложная задача, на которой опробован практически весь "арсенал" эвристических алгоритмов ("жадные" алгоритмы, метод отжига, генетические алгоритмы и другие).

Известные приближённые алгоритмы решения задачи OLA послужили источниками базовых идей, использованных при разработке алгоритмов решения задачи OOGLP, которую можно рассматривать как обобщение задачи OLA на двумерный случай.

[1] Battista G. D., Eades P., Tamassia R., Tollis I. G., Algorithms for Drawing Graphs: an Annotated Bibliography, 1994,
`ftp://ftp.cs.brown.edu/pub/papers/compgeo/`.

Глава 2. Общие методы $\mathcal{NP}$-программирования

В этой главе разобраны методы $\mathcal{NP}$-программирования [9], которые использовались при построении алгоритмов решения задачи OOGLP.

Под $\mathcal{NP}$-программированием имеется в виду множество методов, позволяющих решать сложные алгоритмические задачи. Очень часто — это $\mathcal{NP}$-сложные задачи, но в общем случае это могут быть просто сложные, плохо-формализуемые задачи, которые трудно отнести к какому-либо сложностному классу. Таковы, к примеру, задача кластеризации точек в метрическом пространстве, задача перевода с одного языка на другой и рассматриваемая задача OOGLP.

Важным представляемым в этой главе результатом является разбор общих базовых методов $\mathcal{NP}$-программирования, из композиции которых получаются известные подходы решения оптимизационных задач: генетические алгоритмы, метод отжига, метод масштабирования и другие.

Разобраны следующие базовые методы:

model: Сведение задачи к моделированию физической системы

divide: Деление на подзадачи (“разделяй и властвуй”)

scale: Огрубление (упрощение) задачи

reduction: Сведение задачи к другой

local: Метод локальных улучшений

parallel: Распараллеливание

competition: Естественный отбор лучших

macro: Выделение макро-объектов и проецирование задачи на макро-объекты

meta: Метасистемный переход на уровне алгоритма (различные мета-алгоритмы)

В главе показано, как известные подходы к решению оптимизационных задач представляются в виде композиции базовых методов. В частности, показано, что генетические алгоритмы являются синтезом нескольких компонент:

$$\text{генетические алгоритмы} = \text{parallel} + \text{competition} + \text{local}.$$

Последовательные локальные улучшения (local) в генетических алгоритмах проявляются в том, что мутация и рекомбинация генов — это малые локальные изменения параметров особи-решения. Идея

параллельности parallel проявляется в том, что особи-решения живут параллельно, независимо друг от друга. Метод competition есть отбор лучших с последующим их поощрением. Если к указанной сумме добавить еще составляющую meta, то получится террариум алгоритмов. Естественному отбору уже будут подвергаться не сами приближенные решения, а алгоритмы, которые их ищут.

Приведено также разложение на базовые компоненты *метода масштабирования*:

$$\text{метод масштабирования} = \text{scale} + \text{macro} + \text{local} + \text{meta}.$$

Методы scale и macro во многом схожи. Оба связаны с уменьшением уровня детализации рассматриваемых данных.

Метод macro, в определённом смысле, обратен операции разбиения на части divide и заключается в том, что несколько объектов объединяются в группу и с группой связывается *макро-объект*.

Затем формулируется новая задача в терминах этих макро-объектов, решение которой даёт приближённое решение исходной задачи.

Способность эффективно выделять макро-объекты и переформулировать задачу в терминах макро-объектов является, пожалуй, самой важной способностью человеческого разума. Разработка методов программирования этой способности является важнейшим шагом к решению многих современных актуальных задач. Программирование на уровне макро-объектов подразумевает целенаправленый поиск макро-объектов и введение соответствующих им программных сущностей.

Отметим, что важнейшим шагом на пути создания интеллектуальных систем является перенос этих задач на “плечи” компьютеров и мета-алгоритмов.

Глава 3. Задача OOGLP

Определение 2.1. **Ортогональное представление графа с указанными портами рёбер** — это плоская схема графа, в которой вершинам соответствуют прямоугольники указанного размера (**узлы**), а рёбрам — ломанные линии, соединяющие соответствующие узлы. При этом для каждого конца ребра указана сторона света узла (**порт** = восток, запад, север или юг), к которой он должен крепиться. Узлы не должны перекрываться друг с другом и пересекаться с рёбрами (исключая концы). Стороны узлов и звенья ломанных параллельны осям координат. Пример описания портов

ребра: “ребро e_1 выходит из *севера* узла a_1 и входит в *восток* узла a_2”.

Ключевым шагом решения задачи поиска оптимального ортогонального представления графа с указанными портами рёбер является построение физической модели взаимодействующих материальных точек, в которой материальные точки соответствуют вершинам графа, а парные взаимодействия между материальными точками определяются рёбрами графа (наличием рёбер между соответствующей парой вершин и портами этих рёбер). На основании этой физической модели строится несколько приближённых алгоритмов решения задачи OOGLP.

Второй важный шаг решения задачи OOGLP— это разделение алгоритма решения на два последовательных этапа — ранжирование вершин (расположение их на плоскости) и поиск оптимальных представлений рёбер (представлений с минимальным числом пересечений и изгибов).

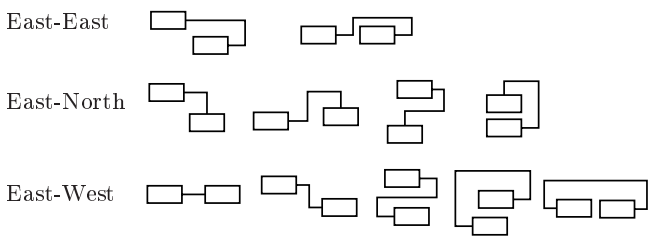


Рис. 1: Простые пути для рёбер типа восток-восток, восток-север и восток-запад.

Ключевой метод решения оптимизационных задач со сложной целевой функцией заключается в поиске альтернативной целевой функции Cost′. которая близка настоящей целевой функции Cost и при этом допускает эффективный алгоритм поиска минимума. Понятие близость целевых функций можно грубо пояснить так: из неравенства $Cost(a) > Cost(b)$ должно часто следовать неравенство $Cost'(a) > Cost'(b)$.

Замены настоящей целевой функции на искусственную соответствует метаэвристике reduction — сведению исходной задаче к другой, более простой.

Замена сложной целевой функции на другую — практически единственный конструктивный подход к задаче OOGLP. Решение

задачи можно разбить на **два этапа**:

- поиск оптимального положения вершин на плоскости,
- оптимальное проведение рёбер для заданного положения вершин.

Представление графа — это описание того, где находятся узлы, и как между ними проходят рёбра. Целевая функция является сложной функцией от представления в целом. Но для упрощения задачи целевая функция разбивается на две составляющие, среди которых одна зависит только от положения узлов, вторая — от представления рёбер. В результате получаем две более простые задачи вместо одной сложной.

На рисунке 1 показаны примеры простых путей рёбер с заданными портами. Простыми мы называем пути, которые имеют минимальное число изломов при заданных положениях узлов. Так для ребра типа "восток-восток" есть только два возможных простых пути (с точностью до симметрии относительно горизонтали). Слева на рисунке указаны наиболее оптимальные варианты расположения узлов для каждого типа ребра.

Рассмотрение простых путей позволяет нам определить получить функцию оценки качества расположения узлов. Алгоритмы, предлагаемые в диссертации, основаны на сопоставлении каждому типу простого пути определённого закона взаимодействия соответствующих вершин. Каждому типу простого пути соответствует некоторый набор пружин (не обязательно гармонических), которые устанавливаются между вершинами графа.

Глава 4. Приближенные алгоритмы решения задачи OOGLP

В главе 4 показано, как методы $\mathcal{NP}$-программирования, описанные в главе 2, "работают" при решении задачи OOGLP.

Представлены результаты сравнительного анализа алгоритмов, касающиеся качества и времени работы, а также эффективности их комбинирования.

Решение задачи OOGLP разбито на два этапа:

- поиск оптимального положения вершин на плоскости (ранжирование вершин)
- оптимальное проведение рёбер для заданного положения вершин.

Разработано четыре различных эвристических алгоритма, решающие первую из этих подзадач — алгоритмы `model`, `mover`, `spectr` и `stem` ранжирования вершин графа:

- `model` — метод, основанный на математическом моделировании физической системы взаимодействующих узлов графа, в которой взаимодействия определяются рёбрами графа.
- `mover` — метод последовательных локальных улучшений, в котором выбирается некоторая начальная конфигурация (представление графа), и на каждом шаге выбирается такое локальное изменение конфигурации, которое как можно сильнее уменьшает значение штрафа.
- `spectr` — метод поиска оптимальной конфигурации, основанный вложении множества конфигураций в линейное пространство и приближении функции штрафа квадратичной формой. Метод сводится к поиску доминантного вектора положительной матрицы.
- `stem` — метод выделения как можно большего подграфа, который может быть построен без решения сложной оптимизационной задачи, а достройка остальной части графа проводится исходя из минимизации штрафа на каждом шаге. Указанный подграф можно назвать каркасом, метод — методом выделения каркаса.

В алгоритмах `model` и `mover` присутствует большой набор взаимодействий вершин. В частности, есть взаимодействия со сложной зависимостью от расстояния и наличия поблизости других вершин.

В методах `stem` и `spectr` также рассматриваются взаимодействия вершин графа. А именно, строятся матрицы доминирования узлов друг над другом для горизонтального и вертикального направлений. Элемент a_{ij} этих матриц отображает "желание" вершины i быль правее (выше) вершины j. А именно, выполнено $a_{ij} > 1$, если расположение i-й вершины правее (выше) j-й вершины приводит к меньшему значению штрафа за представления рёбер, соединяющих эти вершины.

При построении матриц могут быть учтены лишь простые и только парные взаимодействия. Недостаток этого метода заключается в том, что он сильно огрубляет настоящую функцию штрафа. То же самое касается метода `stem`.

Задача оптимального проведения рёбер

Вторая подзадача – оптимальное проведение рёбер для заданного положения узлов – решается общим методом, основанным на алгоритме Дейкстры поиска кратчайших путей в графе.

Задача заключается в том, чтобы для фиксированного положения вершин графа на плоскости провести рёбра вдоль линий ортогональной решетки, стараясь минимизировать штраф за пересечения, изгибы и длины рёбер.

Известно, что задача определения возможности проведения рёбер с менее, чем k пересечениями, при заданном положении вершин является $\mathcal{NP}$-полной. Поэтому и эту подзадачу мы также вынуждены решать не точным, а приближённым алгоритмом.

Алгоритм основан на методе "жадности" (greedy и local), и заключается последовательном проведении рёбер с минимизацией штрафа на каждом шаге.

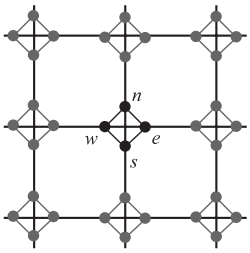


Рис. 2: Каждый **узел целочисленной решетки** имеет с каждой стороны своего представителя. Вес рёбер (w, n), (n, e), (e, s), (s, w) равен штрафу за поворот.

Рассмотрим как осуществляется один такой шаг. Будем считать, что установлены штрафы за пересечение других рёбер, за изгибы и за длину ребра. Оптимальный путь ребра ищется как кратчайший путь в графе-решетке, который модифицируется по мере проведения рёбер. Граф решётки $G_g = (V_g, E_g)$ мы будем называть **решёткой**, его вершины — **узлами**, а рёбра решётки — **звеньями**. **Рёбрами** мы будем называть рёбра исходного графа $G = (V, E)$, а также последовательность звеньев решетки, по которым они они проведены.

Для учёта штрафов за изгибы предлагается немного модифицировать граф решётки, а именно, *учетверить каждый узел решетки* v, то есть заменить узел v на четыре узла v_e, v_w, v_s, v_n, которые являются восточным, западным, южным и северным представителями узла v исходной решетки (см. рис. 2).

Учетверённая решётка позволяет последовательно нарисовать все ребра и минимизировать штраф за суммарную длину рёбер и изгибы. Осталось учесть пересечения. Для этого после проведения каждого ребра проводится расщепление звеньев и узлов решётки,

по которым прошел путь. Когда путь проходит через узел решетки, он расщепляет его на два, которые становятся вплотную справа и слева от пути, а сам путь оказывается уже проходящим не через узлы решетки, а по каналу между этими узлами (рис. 3). Также происходит расщепление звеньев связанных с узлами, по которым прошёл путь. При этом вес внутренних звеньев, которые пересекают путь, должен увеличится на величину штрафа за пересечение. Внешние звенья, пересекающие путь, удаляются.

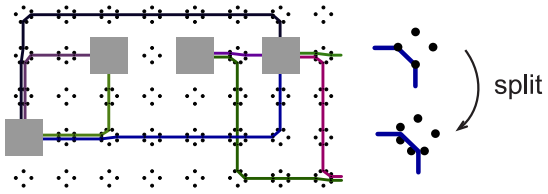


Рис. 3: Пример решетки и путей, которые расщепляют **узлы решетки**. Каждая **точка целочисленной решетки** представлена с каждой стороны света несколькими узлами решетки.

В этом методе конечный результат начинает зависеть от последовательности, в которой проводились рёбра. Для выбора наиболее подходящего порядка используется ряд простых эвристик.

Время работы алгоритма определяется размером графа решетки $G_g = (V_g, E_g)$. По сути, алгоритм сводится решению задачи поиска кратчайшего пути $|E|$ раз. Кратчайший путь ищется алгоритмом Дейкстры, оптимизированном для разреженных графов. Время работы этого алгоритма есть $O(E_g \log |E_g|)$, где $|E_g|$ — число ребер в графе решетки. Изначально $|E_g| \approx 16 \cdot X \cdot Y$, где $X \times Y$ — размер решетки. Каждое проведённое ребро исходного графа увеличивает число ребер решетки.

Важно отметить, что именно на этап проведения рёбер при типичных размерах графа (число узлов $|V| < 100$) тратится большая часть времени поиска оптимального представления. Если предположить, что узлы расположены в представлении более менее равномерно, так что размер решетки $X \times Y$ пропорционален числу узлов $|V|$, то можно получить следующую ассимптотику времени работы алгоритма проведения рёбер: $O(|V| \log |V|)$. Но у этой ассимптотики довольно большой численный коэффициент.

Метод `model`

Метод `model` оказался наиболее успешным методом ранжирования вершин. Он основан на математическом моделировании физической модели взаимодействующих вершин графа. Оптимальное положение вершин графа в этом методе находится как результат эволюции физической системы, состоящей из материальных точек (вершин графа), взаимодействие между которыми определяется рёбрами графа и портами этих рёбер. То есть в линейке

задача → мат. модель → алгоритм → программа

добавлен ещё один шаг — сведение исходной задачи к физической, и приближение функции штрафа функцией потенциальной энергии взаимодействующих материальных точек:

задача → физ. модель → мат. модель → алгоритм → программа

При реализации метода `model` была использована идея масштабирования. А именно, при моделировании законы взаимодействия между вершинами меняются со временем — вначале эти взаимодействия простые и быстро вычисляемые, и шаг времени в модели велик. “Огрублённые” законы позволяют быстро получить грубое приближение оптимального положения вершин графа. Со временем законы взаимодействия становятся более сложными, моделирование эволюции системы происходит с меньшим временным шагом и требует больших вычислительных затрат.

Каждый тип ребра задаёт приоритетное относительное положение узлов на плоскости. В соответствии с этим была введена функция взаимодействия между вершинами, которая определяется типами рёбер, их соединяющих. Минимум потенциальной энергии, связанной с отдельным ребром, достигается в ситуации, когда вершины расположены относительно друг друга так, как того “хочет” ребро. Ребро типа “восток-восток” не указывает, с какой стороны по вертикали должен располагаться конец ребра относительно начала, но ясно, что, этому ребру будет “неудобно”, если его конец и начало имеют одинаковую координату по вертикали. В соответствии с этим для такого типа ребра естественно ввести небольшое отталкивание по вертикали, которое действует лишь в случае, когда разница в координатах менее единицы. В случае, если расстояние сильно больше единицы, это ребро, напротив, должно работать как притягивающая пружина.

В разработанной программе `ortho` реализовано несколько различных типов взаимодействия. Все функции взаимодействия зави-

сят от обеих координат взаимодействующих вершин. В частности, если вершины находятся довольно далеко друг от друга вдоль одной из осей, то взаимодействия, порождаемые рёбрами, которые их соединяют, ослабляются. Эта эвристика основана на том, что, скорее всего, рёбра между далёкими вершинами пойдут "окольными путями", огибая препятствия на своём пути, и не так важно относительное расположение этих вершин. Другой пример: если ребро имеет тип "восток-запад" и вершины находятся на одной горизонтали, то сила их притяжения вдоль горизонтального направления сильнее, чем обычно. Есть несколько тройных взаимодействий.

Метод сведения поиска оптимального представления к математическому моделированию физической системы позволяет реализовать много важных метаэвристик. В частности, метод macro можно воплотить следующим образом. Найти висячие вершины и прикрепить и к тем вершинам, с которыми они соединены ребром. Моделировать их динамику не нужно, так как ясно, что лучшее их положение — это положение рядом со своим родителем с той стороны, которая удобна для типа ребра, исходящего из висячей вершины. Аналогично можно рассматривать висячие деревья.

Кроме того, если граф большой, но разбивается на кусочки, которые слабо соединены друг с другом, то естественно смоделировать эволюцию в каждом из этих кусочков по отдельности, и лишь потом включать "межкомпонентные" взаимодействия.

В метод `model` предоставляет возможность влиять на площадь представления. А именно, можно ввести силу притяжения узлов к некоторому центру. Для получения представлений с маленькой площадью можно также устанавливать на плоскости границы, от которых вершины отталкиваются.

Разработанный метод `model` в среднем проявляет себя лучше всех методов, а именно среднее значение штрафа за представление на тестовых графах (как плотных, так и разреженных) меньше, чем остальных методов. Лишь для 43% случаев для тестовой базы графов этот метод проигрывает одному из других методов.

Важно отметить, что координаты вершин, получаемые в этом методе являются действительными числами и необходимо провести операцию их округления — вычислить целочисленные узлы решетки, где следует поместить узел. При этом необходимо разрешать возникающие конфликты, когда несколько вершин имеют близкие координаты и "хотят" оказаться в одном и том же узле целочисленной решётки. Был разработан простой и надёжный алгоритм,

решающий данные конфликты. Основным параметром операции округления координат является параметр ε, который соответствует расстоянию на котором вершины координаты начинают "слипаться" друг с другом. Алгоритм, реализованный в программе `ortho`, пробует различные значения этого параметра и в конце выбирает лучшее из получившихся представлении. Кроме того, обнаружено, что после операции округления координат и фиксирования вершин в узлах решётки полезно применять алгоритм `mover` последовательных локальных улучшений.

Таким образом, алгоритм `model`, реализованный в программе `ortho`, является комбинацией нескольких алгоритмов, отличающихся выбранным параметром округления ε и наличием завершающего этапа последовательных локальных улучшений.

Время работы алгоритма определяется числом взаимодействий и числом шагов эволюции. Число взаимодействий пропорционально числу рёбер графа, а число шагов зависит от активности динамики. Максимальное значение числа шагов определяется эвристической функцией, линейно зависящей от числа вершин графа. Для такой зависимости есть целый ряд причин. Но процесс эволюции может завершится раньше (и для большинства тестовых графов завершается раньше) при условии, когда вершины графа практически перестают менять своё положение. Поэтому время работы алгоритма `model` имеет асимптотику $O(|E|\cdot|V|)$, где $|V|$ – число вершин графа, $|E|$ – число рёбер графа.

Глава 5. Анализ приближённых алгоритмов

В пятой главе предложено несколько новых методов сравнительного анализа алгоритмов, основанных на численном эксперименте, а именно, на построении турнирной таблицы алгоритмов и вычисления величин *надёжности* и *полезности* алгоритмов.

Пусть $X = \{x_1, x_2, \ldots, x_n\}$ — множество различных входных данных, а $A = \{A_1, A_2, \ldots, A_m\}$ — множество имеющихся эвристических алгоритмов решения оптимизационной задачи.

Обозначим величину значения целевой функции на результате работы алгоритма A_i для входных данных x_j как $\mathcal{E}_i(x_j)$.

Турнирная таблица. Для каждой пары алгоритмов (A_i, A_j) подсчитаем долю ε_{ij} задач, на которых алгоритм A_i обыграл алгоритм A_j (в случае ничьи алгоритмы делят очко попалам):

$$\varepsilon_{ij} = (\text{число задач } x, \text{ для которых } \mathcal{E}_i(x) > \mathcal{E}_j(x))$$
$$+0.5 \cdot (\text{число задач } x, \text{ для которых } \mathcal{E}_i(x) = \mathcal{E}_j(x)).$$

Попытки естественным образом определить меры сложности задач, эффективности алгоритмов, а также меры специфичности алгоритмов и задач приводят нас к циклическим определениям.

Введём меру полезности алгоритма A_i при решении задачи j:

$$W_{ij}(T) = \frac{e^{-\frac{\mathcal{E}_i(x_j)}{T}}}{\sum_k e^{-\frac{\mathcal{E}_k(x_j)}{T}}}.$$

Величина T соответствует некоторой эталонной сложности задачи или фиксированной единице измерения штрафа.

Если проводить аналогию с статистической физикой, то задачи X играют роль физических систем, а алгоритмы отображаются в уровни энергии, запараметризованные задачами. Величина W_{ij} полезности алгоритма A_i при решении задачи x_j равна вероятности пребывания на энергетическом уровне $\mathcal{E}_i(x_j)$ в физической системе, соответствующей задаче x_j. Введём агрегатную меру полезности алгоритма A_i:

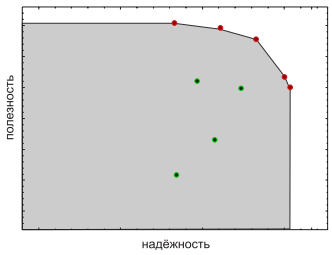
$$U_i = \sum_j W_{ij}(T). \tag{2.1}$$

Это сумма вероятностей пребывания в уровне энергии, задаваемом алгоритмом A_i, по всем задачам (физическим системам) x_j.

При $T \to 0$ величина U_i стремится к числу задач, в которых алгоритм A_i оказался лучше всех остальных алгоритмов (в случае, когда несколько алгоритмов получили конфигурации с одним и тем же значением функции штрафа, очко поровну делится между ними). При $T \to \infty$ распределение по алгоритмам становится равномерным и величина $T(n \cdot W_{ij} - 1)$ стремится к разности среднего штрафа за задачу x_j по всем алгоритмам и штрафа алгоритма A_j: $T(n \cdot W_{ij} - 1) \to (\sum_{i=1}^{n} \mathcal{E}_i(x_j)/n - \mathcal{E}_i(x_j))$.

Таким образом, при маленьких значениях T величина U_i отображает полезность алгоритма – то, насколько часто данный алгоритм оказался лучше других, а при больших значениях T — надёжность — то, насколько хорошо данный алгоритм работает в среднем.

Если надёжность алгоритма есть мера абсолютная и не зависит от набора алгоритмов, то мера полезности алгоритма сильно зависит от множества рассматриваемых алгоритмов A. То есть о полезности алгоритма можно говорить лишь в контексте соревнования между алгоритмами из фиксированного набора.



Подсчитаем для каждого алгоритма его полезность и надежность и поставим соответсвующие точки на плоскости. Найдем выпуклую оболочку этих точек и проведем из точек из точек с максимальной надёжностью и максимальной полезностью вертикальную и горизонтальную касательные к соответствующим осям. Получим прямоугольник, верхний правый угол которого скруглён и представлен ломаной из звеньев выпуклой оболочки.

Эта фигура и определяет достигнутый данным множеством алгоритмов результат — достигнутые значения пар надёжности и полезности. Если новый разработанный алгоритм попадёт внутрь этой выпуклой оболочки, то это плохо — он оказался бесполезным и его можно не включать в библиотеку. Если же он отобразился в точку, которая стоит снаружи фигуры, то это новый результат и его стоит рассматривать как полезный или надёжный алгоритм (или алгоритм, имеющий высокие показатели как надёжности, так и полезности).

Предлагаемые методы во многом основаны на работах Ю.И. Журавлева по теме “Корректные алгебры над множествами некорректных (эвристических) алгоритмов”.

Результаты сравнительного анализа

Для разреженных графов (40 вершин, средняя степень вершин равна 3) получается следующая турнирная таблица алгоритмов:

```
                0     1     2     3
 model( 0): 0.500 0.650 0.875 0.775
  stem( 1): 0.350 0.500 0.500 0.675
spectr( 2): 0.125 0.500 0.500 0.625
 mover( 3): 0.225 0.325 0.375 0.500
```

Для более плотных графов (40 вершин, средняя степень вершин равна 7) метод `spectr` обходит метод `stem`.Это связано с тем, что для метода `spectr` характерны представления, занимающие маленькую площадь. Кроме того, метод `spectr` ищет глобальный минимум некоторой функции, приближающей функцию штрафа, а остальные методы осуществляют пошаговые локальные улучшения.

Как для плотных, так и для разрежённых графов метод `model` оказывается в среднем лучшим, то есть суммарная величина штрафа для него минимальна. Введём алгоритм `best`,

соответствующий “сумме” этих четырёх алгоритмов: `best` = `model` + `spectr` + `stem` + `mover`. Он по определению означает алгоритм, который запускает все четыре алгоритма и из полученных представлений графа выбирает наилучшее. Обозначим величину среднего штрафа алгоритма `best` за единицу.

Для тестовой базы графов типичных для IDEF диаграмм со средней степенью вершин 5.5 получается следующие относительные величины средних штрафов:

	штраф	очки
best	1.000	86.000
model	1.053	29.105
spectr	1.060	28.565
local	1.090	13.029
stem	1.096	15.301

Второй столбец (очки) отображает меру полезности алгоритма, которая вычислялась по формуле 2.1 при малых значениях параметра T (1/10 от величины дисперсии штрафа).

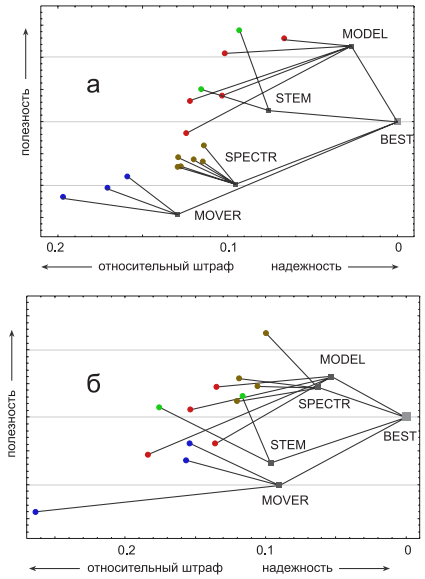


Рис. 4

На рисунке 4 приведена более подробная диаграмма сложения алгоритмов для двух различных наборов графов: (а) – разреженные графы (средняя степень вершины равна 3) и (б) – плотные графы (средняя степень вершины равна 6). Там показано, что каждый из алгоритмов `model`, `spectr`, `stem` и `mover` является в свою очередь “суммой” нескольких сходных алгоритмов, отличающихся лишь каким-то набором параметров. Все алгоритмы в сумме дают алгоритм `best`. Алгоритм `spectr` в случае (б) обгоняет алгоритм `stem` и оказывается по качеству очень близок к `model`. Интересно также отметить, что в случае (б) алгоритм `stem` оказался полезнее `mover`, хотя в среднем работает хуже. Видно, что добавление к методу `model` трёх других алгоритмов (приводящее к учетверению времени работы) приводит к уменьшению среднего штрафа лишь на 5%. Для тестовой базы графов, в которой есть как разреженные так и плотные графа, выигрыш от сложения достигает 10%.

Примеры ортогональных представлений

Разработанный программный комплекс `ortho` предоставляет возможность задавать параметры алгоритмов, комбинировать различные алгоритмы, выбирать тип представления, а также балансировать между качеством результата и временем работы программы. Кроме того, в программе можно задавать параметры целевой функции (функции штрафа): штрафы за пересечения и изгибы рёбер, их суммарную длину, а также штраф за общую площадь представления.

Далее приведены примеры представлений, построенных с помощью разработанной программы `ortho`.

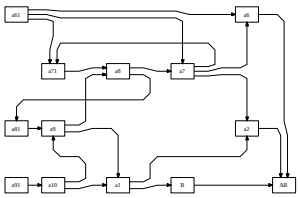


Рис. 5

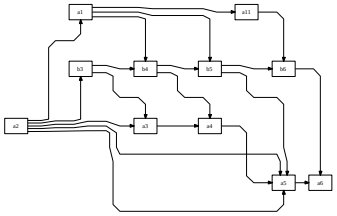


Рис. 6

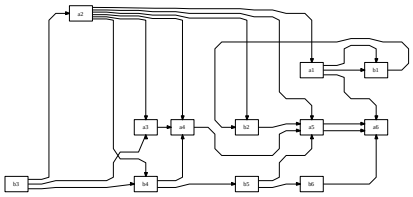


Рис. 7

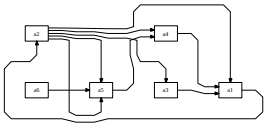


Рис. 8

Глава 6. Задача OLA и упорядочивание по вычислительной сложности

В главе 6 диссертации исследуется проблема определения понятия сложности вычислений и связь этой задачи с задачей OLA (задачей поиска линейного порядка вершин графа, на котором достигается минимум некоторого функционала). Показано, что на множестве вычислимых функций можно построить ориентированный граф, взвешеннные рёбра которого отображают меру сложностного превосходства одной функции над другой или “способность” одной функции вычислять (эмулировать) другую функцию. В работе дан абстрактный подход к определению этой меры. Описанный подход имеет довольно большую область применения: сравнение сложно-

сти функций, сравнение пропускных способностей информационных каналов, ёмкость Шеннона графов и др. Один из важнейших результатов состоит в том, что неоднозначность понятия сложности вычислений (зависимость этой величины от выбранного вычислителя) можно преодолеть, сведя её к проблеме ранжирования по матрице парных взаимодействий, то есть к задаче, решаемой в диссертации [1].

Основные результаты

1. Разработано несколько приближённых алгоритмов, решающих задачу оптимального ортогонального представления графа на плоскости с указанными портами рёбер (задачу OOGLP). В качестве основного метода использовался метод математического моделирования физической системы взаимодействующих вершин графа;

2. Проведён сравнительный анализ разработанных алгоритмов, указаны достоинства и недостатки каждого из них. Исследована эффективность комбинирования этих алгоритмов и показано, что метод, основанный на моделировании физической системы взаимодействующих вершин графа, является наиболее гибким и надежным алгоритмом для широкого класса графов, а алгоритм, основанный на выделении каркаса графа, часто оказывается лучшим для разрежённых графов, поэтому на практике полезно комбинировать эти алгоритмы. Построена диаграмма сложения алгоритмов на плоскости {полезность, надёжность}.

3. Разработан комплекс программ, в котором реализованы указанные алгоритмы, и который с помощью задаваемых параметров позволяет балансировать между временем, качеством и компактностью представления, а также позволяет получать представления графов в различных графических форматах и различных стилях.

4. Исследованы базовые методы $\mathcal{NP}$-программирования, позволяющих эффективно приближенно решать $\mathcal{NP}$-сложные и трудно формализуемые задачи: метод огрубления задачи, метод масштабирования, жадные алгоритмы, метод локальных улучшений, метод введения макро-сущностей, метод отбора

лучших и общая идея метасистемного перехода, используемая при построении сложных систем и алгоритмов. Предложен новый метод конструирования алгоритмов, основанный на введении макро-сущностей (макро-объектов и макродействий с ними). Указаны варианты использования базовых методов для построения приближённых алгоритмов решения задачи OOGLP.

Список публикаций по теме диссертации

1. *Ворожцов А. В.* Алгебраические методы определения сложности: Препринт / ИПМ РАН. – М., 2001. – №17. – 23 с.

2. *Ворожцов А. В.* Прогнозирование и теория сложности: Препринт / ИПМ РАН. – М., 2001. – №18. – 17 с.

3. *Ворожцов А.В.* Индустрия знаний.// Информационные технологии и вычислительные системы. – 2003. – том. 1-2. – С. 145–148.

4. *Ворожцов А.В.* Задача о лидере и упорядочивание по вычислительной сложности.// Моделирование процессов управления. – М.:МФТИ, 2004. – С. 218–228.

5. *Ворожцов А.В.* Эвристические алгоритмы кластеризации для маломерных пространств.// Современные проблемы фундаментальных и прикладных наук: Труды XLV Научной Конференции МФТИ. Часть VIII. – Долгопрудный, 2003 – С. 54.

6. *Ворожцов А.В., Винокуров Н.А.* Два приближенных алгоритма решения задачи коммивояжера.// Моделирование процессов управления. – М.:МФТИ, 2004. – С. 211–217.

7. *Ворожцов А.В.* Критерии интеллектуальности искусственных систем: Препринт / ИПМ РАН. – М., 2004. – №60. – 26 с.

8. *Ворожцов А.В.* Критерии интеллектуальности искусственных систем.// Рефлексивные процессы и управление. – 2004 – Том 4, №2. – С.18-29.

9. *Ворожцов А.В.* Мета-методы $\mathcal{NP}$-программирования: Препринт / ИПМ РАН. – М., 2005. – №43. – 18 с.

Ворожцов Артём Викторович

Построение алгоритмов ортогонального представления графа с указанными портами рёбер

Изд.лиц. ИД № ?????? от ??.??.2005
Формат 60×84 1/16. Печать офсетная. Усл. печ.л. 1,25.
Подписано в печать ??.??.????. Тираж 70 экз. Заказ № ???.

Московский физико-технический институт
(государственный университет)
Отдел автоматизированных издательских систем
«ФИЗТЕХ-ПОЛИГРАФ»

141700, Московская обл., г. Долгопрудный,
Институтский пер., 9