***А. П. Димитриев**, канд. техн. наук, доцент Чувашского государственного университета им. И. Н. Ульянова, г. Чебоксары, dimitrie1@yandex.ru*

Модели и алгоритмы в системах автоматизированного перевода текста

Основной проблемой данного исследования является моделирование оптимального подбора слов при автоматизированном переводе текста. С этой целью предложено использовать ряд алгоритмов составления расписаний с оптимизацией целевой функции, производится их сравнение между собой и с известными алгоритмами. Используется упрощенная математическая модель, где учебные занятия представлены объектами. Рассмотрена работа алгоритмов на качественно различных наборах исходных данных.

Ключевые слова: алгоритм, сеть Петри, оптимизация, автоматизированный перевод текста.

Введение

Проблема автоматизированного перевода текстов с языков народностей Российской Федерации относится к одной из актуальных. Языки малочисленных народов вытесняются русским языком [7]. Одна из мер для их сохранения — комплексная компьютеризация, в том числе создание машинных фондов языков и текстовых корпусов. Находящиеся в них тексты будут представлять культурную ценность, будучи переведены на русский язык.

По данной теме проводились работы по гранту Российского фонда фундаментальных исследований[1]. Для качественного перевода текстов на естественных языках необходимо, помимо прочего, оптимизировать выбор переводных основ из нескольких вариантов для каждого слова текста-оригинала. Оптимизация состоит в получении такого набора переводных основ, в котором сумма численных выражений связи между каждой парой основ была бы максимальной:

$$\max C = \sum_{\substack{i=1,\, m_i \\ j=1}}^{n} \frac{l_{i,j}}{|i-j|+1},$$

где n — число слов в предложении, m_i — число вариантов перевода i-го слова, $l_{i,j}$ — количественное выражение частоты встречаемости словосочетаний с участием основ i-го и j-го слова, которое берется из базы данных [7].

Целью работы является построение алгоритма такого выбора. Так как данная задача относится к задачам дискретной оптимизации, за основу взяты разработанные автором алгоритмы составления расписания учебных занятий [9, с. 58–60, с. 66–71] и широтно-импульсной модуляции (ШИМ) мощности нескольких потребителей [8].

Математическая модель

Переводимое предложение можно рассматривать как упорядоченный набор объектов (слов), формирующий результирующее значение в виде переведенного на другой язык текста, качество которого характеризуется значением целевой функции. Вместе с тем набором объектов можно также описать и расписание учебных занятий, представляющее собой последовательность учебных пар и свободного времени, а также описать потребителей электроэнергии в случае решения задачи оптимального

[1] РФФИ, № проекта 08-07-97003 — р_поволжье_а.

управления групповой нагрузкой при ШИМ мощности [8].

Задача составления оптимального расписания является *NP*-полной задачей, т. е. такой, для которой не найдено алгоритмов ее решения за время, пропорциональное полиному от количества входных данных (однако не доказано отсутствие таких алгоритмов). В работе [9, с. 22–51] составляется близкое к оптимальному расписание на основе математической модели, основанной на сетях Петри. На кафедре компьютерных технологий функционирует научно-педагогическая школа «Математическое моделирование» по моделированию на сетях Петри, результаты работы которой в области автоматизированной обработки текстов опубликованы также в [6, 10] и ряде других публикаций.

Сети Петри имеют переходы, моделирующие события (*transitions*), и позиции, моделирующие ресурсы (*places*), обозначаемые символами t и p, соответственно (с индексами). На рисунке 1 представлена раскрашенная сеть Петри, моделирующая оптимизацию перевода текста, имеющая сходство с *J*-сетями, рассматриваемыми в [10].

В отличие от классических сетей Петри, имеются дуги с двумя стрелками, означающие, что имеется дуга в прямом и дуга в обратном направлении и отрезки без стрелок, только хранящие информацию о связях маркеров или их частей с другими маркерами или их частями. Прямые (вертикальные или горизонтальные) линии — переходы. Индексы в обозначениях переходов аналогичны индексам в обозначениях связанных с ними крайних (листовых) позиций, например, для $p_{1,2}$ это $t_{1,2}$, поэтому на рис. 1 не приводятся. Каждая часть маркера означает один из вариантов перевода соответствующей основы. Количество связей между $p_{0,i}$ и $p_{0,k}$ вначале равно $m_i \times m_k$, все они представляются

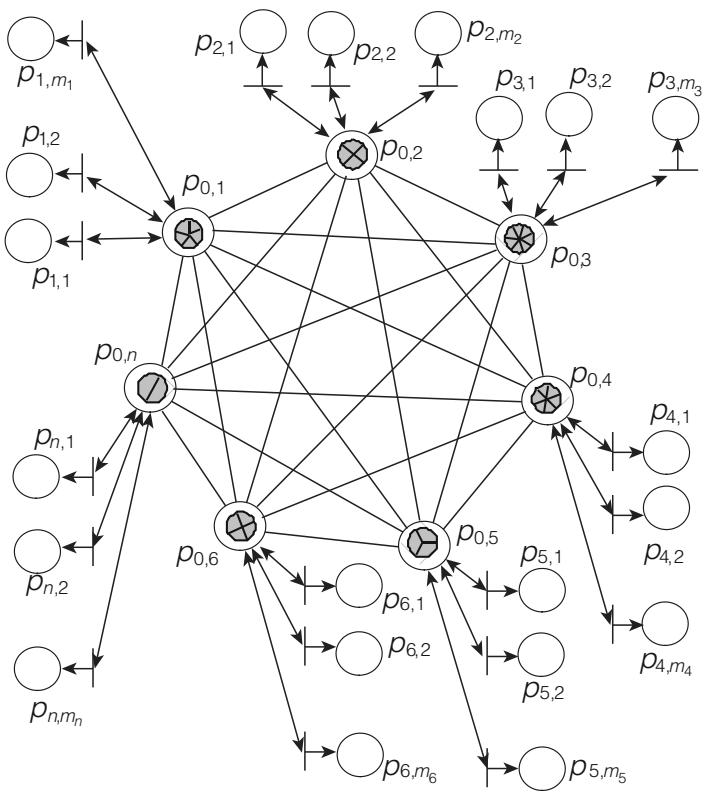

Рис. 1. Сеть Петри, моделирующая оптимизацию перевода

одним отрезком, где m_i — это количество возможных переводов i-й основы.

При срабатывании какого-либо перехода $t_{i,j}$ сначала целый маркер извлекается из $p_{0,i}$, затем он помещается в $p_{i,j}$, и маркер, цвет которого становится равным j и не состоящий из частей, помещается в $p_{0,i}$. Хранимая информация теперь не включает данные о связях каких-либо частей маркера в $p_{0,i}$ с другими маркерами или частями, а взамен только об их связях с выбранным вариантом (j). То есть количество связей между $p_{0,i}$ и $p_{0,k}$ теперь равно m_k, при условии, что $p_{0,k}$ пока содержит маркер, состоящий из m_k частей. Если же там целый маркер, связь остается только одна.

В конце работы сети Петри все наклонные отрезки будут означать только одну связь. Задачей является максимизация полученной суммы этих связей.

Далее символы t и p используются в другом контексте. Для упрощения модели учебные занятия представим непрерывными импульсами, характеризующимися шириной p, силой тока (значимостью) t и дискретными моментами возникновения внутри рассматриваемого временно́го интервала длиной m (например, рис. 2, где в строках единицы означают присутствие импульса, а нули — отсутствие). Далее импульс для терминологической унификации будем называть объектом, свойства которого задаются шириной (длительностью импульса), значимостью (в соответствии с применяемым критерием) и начальной точкой его расположения на временно́й оси (моментом появления импульса). В зависимости от изменения электрической нагрузки может понадобиться перенос по временно́й оси начальных точек. Занятия по расписанию проводятся с периодичностью в одну неделю, аналогично чему мощности коммутируются с периодичностью m. Такая модель соответствует и ШИМ мощности [8].

Каждый объект занимает ряд соседних временны́х квантов на оси времени. Если начальная позиция объекта расположена близко к конечной точке временно́го интервала, то временны́е кванты занимаются до конца интервала, а остаток объекта переносится на начало. Задачу составления расписания можно формально представить как задачу нахождения вектора начальных точек, минимизирующего нагрузку на электрическую сеть, величина которой равна произведению сопротивления проводки на выражение

$$C = \sum_{h=1}^{m} \sum_{j,k=1}^{n} t_{hj} t_{hk}, \qquad (1)$$

где n — число объектов, $t_{n,j}$ — величина j-го тока в h-й момент. Здесь вторая сумма характеризует нагрузку в h-й момент. От начальной позиции объекта и до момента, определяемого его шириной, величина соответствующего тока равна значимости объекта, а в остальное время она нулевая.

Хаотичное расположение объектов на оси времени негативно сказывается на величине C. В этом случае значение C для $n = 40$ может быть на 30% больше минимального. Она представляет собой целевую функцию задачи, ее размерность — A^2, из которой при учете сопротивления проводки получаем ватты.

В задаче составления расписания также имеется минимизируемая целевая функция, учитывающая более десяти факторов с различными весовыми коэффициентами и не имеющая размерности [9]. Модель адекватна расписанию учебных занятий, но упрощает его, так как сводит учет психолого-педагогических требований [9, с. 10–21] к учету значений параметров объектов и не учитывает ограничения, такие как занятость преподавателя или принадлежность аудитории определенной кафедре.

Обозначения значимостей t и ширин p, или процентов мощности, использованы по аналогии с раскрашенными и временны́ми сетями Петри. Срабатывание переходов приводит к определенным событиям, характеризующимся каким-либо параметром (например, временем протекания начавшегося в результате события процесса), а число позиций p соответствует числу вре-

менны́х квантов, отводимых объекту на временно́й оси.

Эти величины интерпретированы в [4] для задачи перевода текстов, где указано, что разработан алгоритм, первоначально предназначенный для управления групповой нагрузкой сталеплавильных электропечей при ШИМ мощности. Для составления предложения текста на естественном языке проведена частичная аналогия. Например, каждая печь соответствует слову предложения, а вектор начальных точек — выбранной совокупности вариантов перевода. Интегральная оценка силы тока и ширины импульса для отдельной печи отчасти соответствует частоте совместного употребления основ — кандидатур на места в предложении. Целью теперь является, в отличие от составления расписания подачи тока в печи с минимальной нагрузкой на электросеть, выбор самого благозвучного переводимого предложения из всех возможных.

Решение методом перебора

В рассмотренной выше модели имеется m^n вариантов размещения объектов, полностью перебрать которые практически невозможно. В гранте РФФИ, № проекта 98-01-03287 р-98 волга автором разработана программа, выполняющая полный перебор вариантов размещения для $n = 7, m = 10$ и неполный для $n = 12, m = 50$ на компьютере с процессором 80286, 12 МГц за несколько минут. Использована стратегия слепого поиска [3, с. 138], состоящая, в отличие от направленного, в полном переборе всех возможных вариантов размещения. Слепой поиск может производиться в ширину, когда вершины некоторого уровня будут просматриваться только после просмотра всех вершин предыдущего уровня либо в глубину (примененный в программе), когда вершины просматриваются вниз и слева направо, начиная от корневой вершины [3, с. 142]. Под вершинами понимаются варианты размещения. В нашем случае корневая вершина моделирует ситуацию, когда все объекты расположены начальными позициями на первом моменте. Практически это означает наихудшее размещение, так как все они присутствуют одновременно и насколько хватает их ширин.

Для ускорения перебора применялись методы оптимизации кода, впоследствии преподаваемые студентам. К ним можно отнести, например, следующие методы:

1) вынесение инвариантных значений из циклов [1, с. 645]. Например, вместо того чтобы для каждой комбинации параметров заново вычислять произведение значений двух объектов, использовался заранее подготовленный двумерный массив этих произведений;

2) подстановка кода функций в вызывающий объект [1, с. 644]. Вначале программа имела рекурсивный вызов функций. Затем был сформирован массив, хранящий точки возврата, и необходимость в использовании функций отпала. Действия, ранее выполняемые в теле функции, перенесены в вызывающую программу;

3) машинно-зависимые методы оптимизации [1, с. 648]. Вместо организации поэлементного копирования массива внутри цикла использована функция *move* языка *Turbo Pascal* 7.0, быстро копирующая непрерывные участки памяти на основе использования ассемблерных команд.

Данные методы применялись в сочетании с некоторыми характерными для решения задачи эвристиками и методом ветвей и границ [16], идея которого состоит в исключении из рассмотрения тех комбинаций, которые заведомо не могут доставлять минимум целевой функции. В результате время перебора сократилось в шесть раз.

Метод ветвей и границ в данном случае использует минимальное найденное к указанному шагу значение целевой функции, уменьшенное на величину, соответствующую наихудшему размещению на последующих шагах. Если текущая оценка неполного размещения больше полученной разности, дальнейшие действия по размещению не имеют смысла и пропускаются, а теку-

щим становится следующее неполное размещение. В связи с этим можно организовать вычисление минимального значения на нескольких процессорах и даже компьютерах, которое они бы передавали друг другу[2].

Составление расписания на основе генетических алгоритмов

Для оптимизации расписаний большей размерности необходимо применять какие-либо другие алгоритмы. Например, близкое к оптимальному решение для аналитически неразрешимых проблем можно найти с помощью метода биологической эволюции [11]. Основные принципы работы генетических алгоритмов сводятся к представлению параметров системы в виде хромосомы, скрещиванию хромосом (кроссовер), мутации и выбору наиболее приспособленных особей для скрещивания.

Генетические алгоритмы для составления расписаний учебных занятий применены в [2, 11]. В классическом генетическом алгоритме [11, с. 12] два потомка, полученные в результате одноточечного кроссовера, после мутации замещают родителей. Одноточечный кроссовер [11, с. 9] подразумевает случайный выбор одной из точек разрыва при скрещивании, левее которой гены берутся от первого родителя, а правее — от второго.

В алгоритме *CHC* (*Cross-population selection, Heterogeneous recombination and Cataclysmic mutation*) [11, с. 14–15] отбор особей в следующее поколение ведется и между родительскими особями, и между их потомками, так как используются популяции небольших размеров. Поэтому после нахождения решения алгоритм перезапускается и лучшая особь копируется в новую популяцию, а оставшиеся являются сильной мутацией существующих особей и поиск повторяется. При скрещивании все особи разбиваются на пары, и скрещиваются только те пары, в которых хромосомы различны.

Островная модель [11, с. 13–14] заключается в разбиении всей популяции на некоторое количество развивающихся независимо популяций одного вида и расселенных по нескольким островам. Время от времени может происходить миграция хороших особей между островами.

В гибридном алгоритме генетический алгоритм сужает пространство поиска, а затем производится оптимизация каким-либо «классическим» методом [11, с. 13].

Прежде чем применять сложные генетические алгоритмы, необходимо выяснить, как простой генетический алгоритм *Genitor* [11, с. 13] с равномерным кроссовером работает с вышеописанной математической моделью, соответствующей ШИМ мощности. Его особенностью является отсутствие ограничений на тип кроссовера и мутации, потомок замещает особь, обладающую наименьшей приспособленностью. В равномерном кроссовере каждый бит потомка выбирается случайным образом либо от первого, либо от второго предка [11, с. 9].

Для сравнения используем метод, разработанный в [8], на который далее будем ссылаться как на алгоритм 2 (табл. 1). При $n = 12$ разработанная программа (рис. 2) показала, что применение алгоритма 2 дает лучшие результаты (значение C меньше). На следующем шаге задавались значения параметров $n = 40$, $m = 100$ с целью обеспечить соответствие параметрам расписания учебных занятий. Для удобства пользователя входные данные программы размещаются в одном из текстовых файлов и представляют собой списки значимостей и ширин объектов с указанием количества объектов. В указанном случае использован файл с именем *f*1, данные которого мож-

[2] Это, в частности, послужило фактором включения раздела «Обмен данными между приложениями» в дисциплину «Операционные системы» в рамках педагогической деятельности автора статьи. Например, студентам в лабораторной работе предлагается передавать числовой вектор с одного компьютера на другой по протоколам *TCP* или *UDP*, чтобы можно было проверить, что по переданным числам действительно вычисляется передаваемый результат.

но увидеть на рис. 3 в виде треугольников. На рисунке 2 единицы означают размещение объекта в данный момент времени, а нули — отсутствие. Каждая строка характеризует один объект из 40 существующих. В последней строке указано значение *C*, полученное в результате выполнения алгоритма 2. В самом низу экранной формы мелким шрифтом указано значение *C* в результате применения генетического алгоритма.

Размер популяции выберем равным 800. Для исследования использован компьютер *Pentium* IV, 1,5 ГГц. Алгоритм 2 практически мгновенно находит решение, близкое к оптимальному. Генетический алгоритм за несколько секунд находит худшее по качеству решение. Если мутации сделать целенаправленными с использованием алгоритма 2, то это еще более замедляет работу, но приводит к лучшему решению: отношение значения *C* в результате применения генетического алгоритма к значению *C* при использовании алгоритма 2 составляет примерно 0,99990.

Для получения лучших решений количество мутаций (параметр настройки «*Mutation*» на рис. 2) надо увеличивать. К этому количеству прибавляется псевдослучайное число от 0 до 99, и если сумма более 100, производится мутация гена, если сумма больше 49, ген берется от первого родителя, иначе — от второго. Практически требуется доводить количество мутаций до 60 и более, что увеличивает время работы до 4 мин. для 1000 скрещиваний. Поскольку перевод текстов необходимо проводить быстро, генетический алгоритм оказывается непригоден. Временна́я сложность комбинированного алгоритма выражается в виде $O(qn^3m^3)$, где q — общее число скрещиваний.

Рассмотрим работу алгоритма *Genitor* при переводе предложения «***Тăвансемпе пĕрлешнĕ чух чăваш тĕнчи çĕкленнĕ чух чун савăнать чĕре сикет татах та хастар пулас килет***» (отрывок из гимна Чувашской Республики, слова И. Тукташ), для этого разработана специальная программа на основе представленной в работе [7]. Размер популяции выберем равным 100 особям. Максимального найденного всевозможными алгоритмами (см. ниже) значения (2209,38) генетический алгоритм достигает сравнительно медленно, приблизительно за 10–15 с на современном компьютере, совершая при этом около 1–1,5 млн скрещиваний.

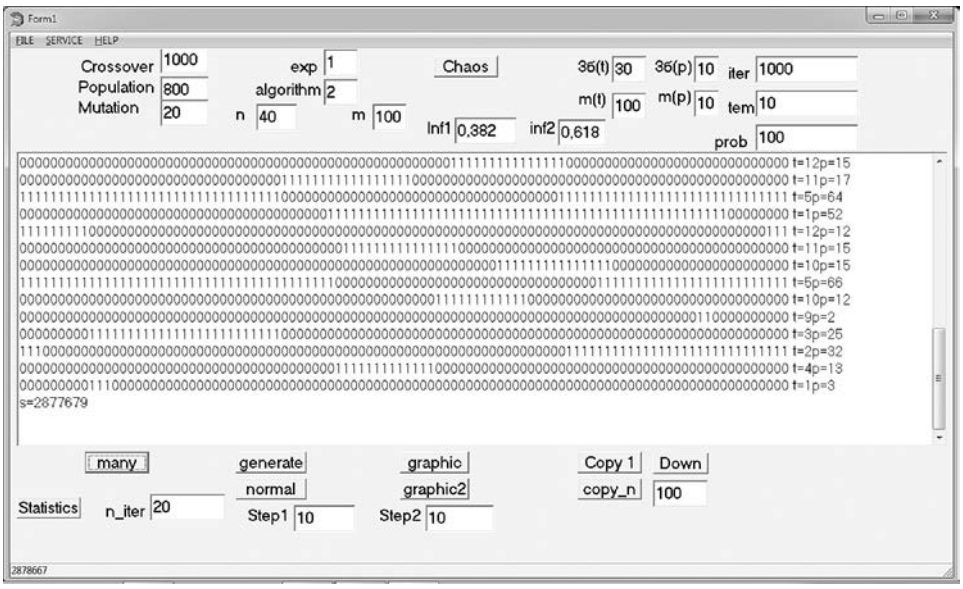

Рис. 2. Пользовательский интерфейс программы на основе алгоритма 2

Усовершенствованные алгоритмы на основе сортировки

Предложенный в [8] метод положен в основу создания набора усовершенствованных алгоритмов. Для более полного их исследования взяты различные наборы входных данных. Большинство файлов с входными данными получено с помощью генератора псевдослучайных чисел с равномерным распределением. Программа позволяет создавать такие файлы для указанного числа и диапазона изменения параметров. С целью выявить параметр (p или t), оказывающий наибольшее влияние на C, взяты различные диапазоны их изменения: $t \approx p$, $t > p$, $t >> p$, $t << p$.

В файле $f4$ взяты значения $t \in [40,44]$, $p \in [40,59]$, т. е. нет слишком больших или слишком малых значений, $m = 100$. В файле $f3$ взяты значения $t \in [100,129]$, $p \in [30,39]$, $m = 100$, т. е., в отличие от предыдущего файла, заведомо $t > p$. В файле $f6$ значения $t \in [0,299]$, $p \in [0,19]$, $m = 20$, т. е. имеются всевозможные значения и $t >> p$. В файле $f2$ значения $t \in [0,99]$, $p \in [0,99]$, $m = 100$. В файле $f5$ значения $t \in [0,19]$, $p \in [0,99]$, $m = 100$, т. е. $t << p$.

Входные данные в виде диаграммы в координатах $3t - p$ (в логарифмическом масштабе, нулевые p не отображены) представлены на рис. 3. Файл $f1$ был сформирован вручную. Данные файла на диаграмме имеют вид спирали, тогда как другие равномерно распределены по области значений.

Первая группа алгоритмов, включающая алгоритмы с номерами 0, 1, 2, 3, 4, 5 (табл. 1), а также алгоритм 7, предполагает первоначальную сортировку объектов по соответствующему признаку, затем расположение объектов по очереди в наилучших моментах. Временна́я сложность $O(n^2m^2)$. В таблице 1 информативность $I(x)$ представляет собой меру близости ширины объекта к доле x временно́го интервала: $I_j(x) = -\left|p_j - x\right| / m$. Термин «информативность» используется по аналогии с информативностью монохромного изображения, а именно меры, основанной на энтропии К. Шеннона и зависящей от наличия на изображении мелких деталей [13, с. 70–85]. Ее суть в том, что для представления в памяти ЭВМ сжатого изображения с мелкими деталями требуется больше единиц информации, чем без таковых. Соответственно, информация о расположении объектов нулевой либо максимальной ширины не нужна, так как их расположение никак не отражается на C. В противовес этому информация о расположении остальных объектов важна.

Рассмотрим сначала случай $x = 1/2$. При увеличении ширины объекта в диапазоне

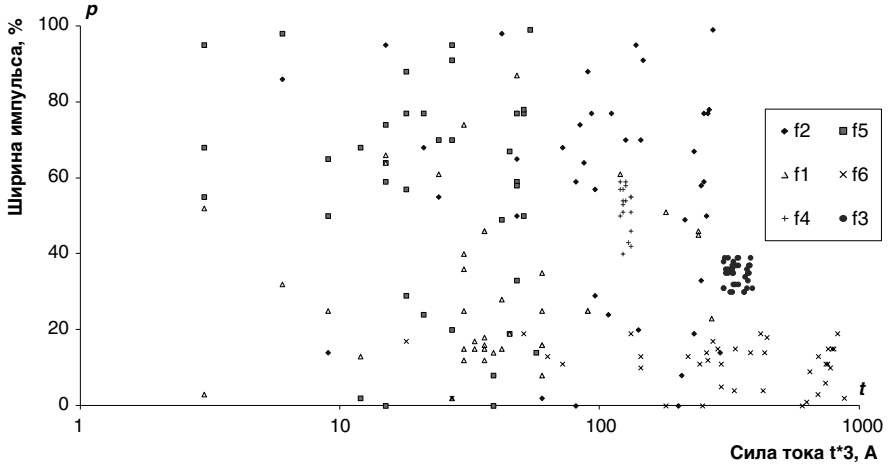

Рис. 3. Входные данные (одинаковые фигуры соответствуют одному файлу)

$[0, m/2]$ информативность увеличивается, поскольку уменьшается возможность выбора у других объектов, чтобы избежать присутствия одновременно с данным объектом (которое приводит к увеличению C).

Увеличение ширины после $m/2$ приводит к уменьшению информативности, что связано с инверсией наличия объекта. Влияние инверсии заметно в табл. 1, где характеристики работы алгоритма 7 приводятся для сравнения с алгоритмом 2. Алгоритм 7 основан на начальной сортировке по *pt* и представляет собой вырожденный случай алгоритма 2 до $I(1)$, т. е. в нем инверсия не учитывается. В пяти случаях из шести лучшими оказываются результаты, получаемые с помощью алгоритма 2.

Рассмотрим вопрос, нельзя ли использовать для решения задачи теорию методов одномерной оптимизации. Например, среди них известен «метод золотого сечения» [12, с. 166–170], основанный на том, чтобы для сужения интервала изменения исследуемого параметра выбирать каждый раз точку, в которой вычисляется значение функции, на определенной доле длины отрезка, соединяющего две ранее выбранные точки (она равна $(\sqrt{5}-1)/2 \approx 0{,}618$). В случае если в данной точке значение функции более оптимально, она берется в качестве одной

Таблица 1

Краткое описание алгоритмов

№ алгоритма	Комментарии
0	Без сортировки, для сравнения
1	Сортировка по $I(1/2)t^2$
2	Сортировка по $I((p\,div\,2)+1)t$
3	Сортировка по $\max\{I(0{,}618), I(0{,}382)\}tM(t)$
4	Сортировка по $tM(t)$
5	Сортировка по $I(1/2)tM(t)$
11	1-й алгоритм с модификацией — выбором размещения по двум координатам
13	Сортировка по $\max\{I(0{,}618), I(0{,}382)\}tM(t)$
14	Модифицированный 4-й алгоритм
16	Сортировка по t, двухкоординатный
18	Сортировка по $t^{3/2}M(t)/p$
19	Сортировка по $(t^a)M(t)/(1/2))$, a задается
23	Выбор по $\max(tM(t)\ \max\{I(0.618), I(0.382)\}))$
24	Выбор по $\max(tM(t))$
25	Выбор по $\max(tM(tp))$
29	Выбор по $\max(t^aM(t)/(1/2))$, a задается вручную
33	Выбор на каждом шаге по $\max(tM(t))$, критерий размещения — минимум от среднеквадратичного отклонения (см. далее) сумм t по моментам, умноженных на p
34	То же, что и 33, но вместо математического ожидания взят ноль

Таблица 2

Результаты вычислительного эксперимента: найденное значение *C*

№ алгоритма	Области значения входных данных					
	t =>> [1,90], *p* =>> [1,87], *m* =>> 100	*t* =>> [0,99], *p* =>> [0,99], *m* =>> 100	*t* =>> [0,19], *p* =>> [0,99], *m* =>> 100	*t* =>> [0,299], *p* =>> [0,19], *m* =>> 20	*t* =>> [100,129], *p* =>> [30,39], *m* =>> 100	*t* =>> [40,44], *p* =>> [40,59], *m* =>> 100
0	2896973	44329614	1602430	73739134	116249869	33985348
1	2877557	44253074	1599880	73661021	116188398	33986846
2	2877679	44255828	1599702	73660003	116250352	33985752
3	2878231	44257894	1599773	73660772	116173281	33985057
4	2877793	44255778	**1599552**	73655950	116201663	33986463
5	2877981	44254710	1599590	73658771	116210932	33987676
6	2877620	**44250810**	1599532	73655950	116260119	33986600
7	2877926	44262881	1599949	73683475	116170927	33986926
11	2877874	44250927	**1599519**	73655949	116202606	33990667
13	2877940	44259000	1599720	73659965	116197626	33983926
14	2877673	44250973	**1599492**	**73655734**	116205575	33986307
16	**2877508**	**44250376**	**1599493**	**73654746**	116233667	33985820
18	2877797	44252058	1599695	73657218	116217727	**33983240**
19	2877740	44252431	**1599548**	73656500	116256766	33985461
23	2878501	44254390	1599724	73663721	116169192	**33983686**
24	**2877487**	**44250810**	**1599532**	73656893	**116156423**	33986979
25	**2877511**	44252704	**1599570**	73656893	**116147524**	33985557
29	2877773	44255100	1599625	73657701	116206329	**33983393**
33	**2877487**	44252668	**1599549**	73656893	116210334	33985221
34	**2877487**	**44250810**	**1599532**	73656893	**116156423**	33986979

из двух базовых точек для следующего шага, что не требует вычисления функции для базовой точки. Если просто делить отрезок пополам, как в методе деления отрезка пополам [12, с. 165], то оптимальная точка чаще остается вне получаемого отрезка, поэтому число вычислений метода «золотого сечения» меньше.

В связи с этим в некоторых алгоритмах исследовалась информативность «золотого сечения» $\max\{I(0.618), I(0.382)\}$. Выявлено, что при прочих равных условиях примерно в 61% случаев это приводит к лучшему *C*, чем при $x = 1/2$, но в то же время среди предложенных, как правило, есть лучшие алгоритмы. Для выявления степени влияния *p* относительно *t* в алгоритме 29 *t* бралось в произвольной степени. Установлено, что оптимальной является степень в диапазоне $[1.4,1.6]$. В результате проведения более

3000 экспериментов выявлено, что оптимально значение $x \in [0.3,0.5]$.

Несмотря на логичность рассуждений, получаемые вышеперечисленными алгоритмами размещения редко идеальны. Например, размещение двух объектов за шаг вместо одного при использовании тех же результатов сортировки примерно в 69% случаев улучшает результаты на 0,001–0,022%, хотя и замедляет работу в m раз. В таблице 1 это алгоритмы 11–19. Они работают аналогично первой группе алгоритмов, но происходит размещение двух объектов (данный и предыдущий) за шаг (две координаты). Временна́я сложность $O(n^2m^3)$.

Поэтому предложена вторая группа алгоритмов (с 23 по 29), основанная на выборе объекта каждый раз того, со значением t_i которого получится наибольшая сумма произведений с t для оставшихся невыбранными:

$$i : \sum_{j=1}^{n} t_i t_j = \max_{i \in S} \sum_{j=1}^{n} t_i t_j \Big|_{j \notin S},$$

где S — множество выбранных объектов, причем можно задавать учет их ширин, при этом во время сложения t_i умножаются на $|p_i - n/2|$.

Во второй группе алгоритмов программа перестает использовать некоторые данные о выбранных объектах, что приводит к уменьшению размерности системы в ракурсе этих данных, хотя по-прежнему размерность относительно других данных неизменна. Временна́я сложность в данном случае составляет $O(n^2m^2)$.

В таблице 1 через $M(t)$ обозначена ожидаемая сумма множимых на t (при делении на их число это математическое ожидание). Например, в 24-м алгоритме это сумма значимостей объектов, пока не внесенных в расписание. В алгоритме 25, дающем иногда наилучший результат, математическое ожидание берется от произведения значимостей и ширин остальных объектов, не внесенных в расписание.

Альтернативные алгоритмы

Первые три группы алгоритмов основаны на первоначальной перестановке объектов, будь то сортировка или учет математического ожидания. Затем они работают однообразно — очередной объект (или два) ставятся в наилучшую позицию. В итоге какова бы ни была перестановка, она сама по себе не гарантирует, что расписание станет идеальным, так как впоследствии может оказаться, что объект надо было размещать не туда, где он уже расположен. Очевидно, что какая-либо перестановка обязательно должна быть, так как необходимо в некой последовательности выбирать объекты для размещения. Выбор объектов тоже должен быть, так как иначе ни один объект не будет размещен. Таким образом, необходимо изменить поведение алгоритма после перестановки.

Как альтернатива, в [6] предложен метод, основанный на объединении объектов в кластеры. Это имело целью уменьшить сложность системы объектов и, соответственно, временну́ю сложность алгоритма ее оптимизации. Кластеры составлялись из объектов так, чтобы заполнить временной интервал приблизительно целое число раз и иметь примерно равную сумму значимостей этих объектов на протяжении временного интервала (чтобы манипулировать с ними как с единым целым). Однако они при этом получали неравномерно распределенное по временно́му интервалу слагаемое, и в итоге значение функции становилось примерно на 3% больше, чем при использовании алгоритма 2.

Если алгоритм 7 применить к расписанию учебных занятий, а затем выполнить стохастическую оптимизацию [9, с. 69–71], аналогичную алгоритму имитации отжига [15], то его целевая функция улучшается.

В основе метода [15] лежит имитация отжига металла, подразумевающая медленное снижение температуры. Как известно, раскаленная сталь может закаляться (очень быстро охлаждаясь), а также отжигаться (очень медленно, при этом снимаются внут-

ренние напряжения). В случае отжига происходит минимизация энергии атомов, состоящая в их перемещении из одной ячейки кристаллической решетки в другую. Вероятность перемещения уменьшается с понижением температуры, особенно это касается новых положений с большей энергией.

Интерпретируя метод для набора параметров, отметим, что случайный переход к новому набору параметров возможен не только если он в итоге лучше, но и если хуже предыдущего. Допустимое отклонение в сторону ухудшения уменьшается со временем, так как «температура» постепенно снижается, так же, как и вероятность самого факта такого отклонения.

Рассматриваемая программа реализует и этот метод. В ней можно регулировать число итераций при каждой «температуре», число шагов ее изменения и отклонение в сторону худшего расписания. Однако при условиях [8] здесь меньше ограничений и слагаемых целевой функции, чем в расписании учебных занятий, поэтому результаты все же отстают от наилучших результатов, достижимых алгоритмом 1. Например, для файла *f*1 частное значений *C*, получаемых, соответственно, имитацией отжига и алгоритмом 1, составляет от 1,0017 до 1,0019, а время выполнения становится недопустимо большим. Так, при $n = 40$, $m = 100$ при числе итераций 10^4 и пяти шагах изменения «температуры» время вычисления на современном компьютере достигает 22 с.

Для разработки другого подхода рассмотрим пример, проиллюстрированный на рис. 4. Рассматривается изложенная выше задача ШИМ мощности, которая возникает при управлении группой сталеплавильных электропечей, а также группой электродвигателей в современной электронной технике (во втором случае токи маломощные, порядка 10 А).

Здесь для наилучшей особи, полученной в программе, определялось значение влияния объектов на возможный последующий объект по моментам времени, т. е.

$$S = \sum_{i=1}^{n} t_i(T)\Big|_{F_i(T)=True},$$

где F_i — функция, принимающая значение *True*, если согласно размещению в данный момент времени T производится коммутация мощности *i*-му объекту, иначе *False*. При этом вариация составляет около 10%.

Из рисунка 4 видно, что у решения, относящегося к оптимальным (согласно определению оптимальности [3] в отличие от ми-

Рис. 4. График зависимости $S(T)$

нимальности), т. е. близкого по условному критерию близости к минимальному расписанию, график представляет собой почти горизонтальную линию. Результаты работы моделирующей программы показывают, что для расписаний, более отдаленных по значению C от оптимального, соответствующие графики имеют больший разброс значений. Используя терминологию теории случайных процессов, можно сказать, что, чем меньше среднеквадратичное отклонение $S(T)$, тем C меньше, т. е. можно при оптимизации стремиться к «гладкости» графика в значении «близости к среднему значению» (обычно в аналитической геометрии под гладкостью подразумевается отсутствие изломов).

В [9, с. 58–61] алгоритм 2 так и называется — «сглаживание», при этом имелось в виду выравнивание мелкими объектами неровностей, оставляемых объектами значительного размера, однако тогда это не ставилось самоцелью.

Данный фактор учитывают два последних алгоритма, аналогичные второй группе алгоритмов, но их цель — минимизировать не C, а среднеквадратичное отклонение $S(T)$. Временна́я сложность $O(n^2m^2)$. Алгоритм 33 работает со среднеквадратичным отклонением относительно среднего значения S, 34 — относительно нуля. Для разных входных данных лучше то один, то другой алгоритм (в табл. 2 выделены наилучшие результаты).

В программе также реализован метод мультистарта [14], комбинирующий метод Монте-Карло, называемый методом статистических испытаний [12, с. 104–106], с методом покоординатного спуска [12, с. 174–176]), примененным в [7]. Метод Монте-Карло состоит в многократной генерации псевдослучайных наборов входных параметров и выборе из этих наборов наилучшего. Для каждого набора вначале производится однокоординатный спуск. Последний состоит в последовательном изменении каждой координаты до значения, оптимального при остальных неизменных координатах, итеративно, пока ни одно из изменений не приведет к улучшению.

Работает метод мультистарта долго, его результаты значительно хуже, чем в других предложенных алгоритмах. Для файла $f1$, например, отличие обычно не менее 1%: один раз получено число 2 890 266, а чаще еще больше (тогда как минимум, полученный алгоритмом 33, — 2 877 487). Временна́я сложность $O(q_0n^2m^2)$, где q_0 — число исходных точек, создаваемых методом Монте-Карло.

Для решения задачи предлагается новый алгоритм, оптимизирующий рассмотренным выше методом имитации отжига [15] результат сортировки исходных данных, т. е. он как бы сортирует их для дальнейшей обработки по сотням и тысячам критериев, тогда как в табл. 1 их всего около 20.

Исходные данные для алгоритма: начальное число итераций It, заданная температура T, текущая температура t, увеличение числа итераций d, разброс R.

1. Выполнить сортировку объектов по какому-либо критерию для алгоритмов 1-й группы. Последовательность номеров отсортированного списка обозначить как отправную точку s в n-мерном пространстве. Вычислить $C(s)$ по формуле (1), завершив для этого выбранный алгоритм.

2.0. Число выполненных итераций $j := 0$.

2.1. В окрестности s, определяемой разбросом R, выбрать точку s_1 псевдослучайной перестановкой двух или более номеров s.

2.2. Разместить объекты в последовательности, задаваемой s_1 в относительно оптимальные позиции (расстановка их поочередно в наилучших на момент расположения позициях, выполняя для этого обзор каждой из m позиций для вновь располагаемого объекта).

2.3. Вычислить $C(s_1)$ в соответствии с полученным размещением. Если $C(s) > C(s_1)$, то $s := s_1$.

2.4. Иначе, с вероятностью t / T, если $|C(s) - C(s_1)|$ меньше некоторого постоянного порога, $s := s_1$.

2.5. $j := j + 1$.

2.6. Если $j > It$, то $\{t := t - 1; It := It + d\}$. Иначе перейти на 2.1.

2.7. Если $t > 0$, то перейти на 2.0.

3. Конец.

При этом часто удается получить лучшее значение C, чем для любого из ранее реализованных алгоритмов. Так, для файла *f*1 производился обмен трех номеров в цепочке при $lt = 1000$, $T = 10$, $t = 5$, $d = 100$, $R = \pm 7$, для получения отправной точки использован алгоритм 1 и достигнуто $C = 2877339$, что значительно лучше всех остальных результатов. Время вычислений составило 40 с, но для ускорения можно изменить параметры за счет увеличения C.

В задаче перевода текста, сравнивая данный алгоритм с генетическим, следует снова отметить чрезмерно большие временны́е затраты, но получаемые результаты могут использоваться для определения минимума целевой функции, который недостижим генетическим алгоритмом. Однако время значительно снижается при отказе от использования имитации отжига, если вместо этого использовать случайный поиск (случайная перестановка нескольких элементов цепочки). Платой за это является увеличение целевой функции примерно до значений, достигаемых генетическим алгоритмом.

Заключение

Таким образом, предложена новая методика получения квазиоптимального размещения, моделирующего оптимизацию перевода предложения на естественном языке.

Вычислительные эксперименты показали, что для модели в большинстве случаев наилучшие результаты получаются при использовании алгоритмов 16, 24 и 34. Если $t >> p$, предпочтение следует отдать алгоритму 16, а если $t \approx p$ — алгоритму 18.

Данная программа зарегистрирована в Реестре программ для ЭВМ Федеральной службы по интеллектуальной собственности [5][3].

При машинном переводе график зависимости целевой функции от времени недоступен, поэтому алгоритмы четвертой группы, основанные на использовании среднеквадратичного отклонения, оказываются непригодны.

При автоматизированном переводе текстов целесообразно применять алгоритм 25, позволяющий проводить адаптивный выбор, когда при решении о предпочтительном варианте перевода очередного слова необходимо отдельно учитывать ожидаемую в дальнейшем его смысловую связь со всеми еще не выбранными вариантами перевода слов предложения и отдельно — с уже выбранными. Проведенные исследования в области перевода текстов показали также эффективность метода мультистарта.

Для применения рассмотренных алгоритмов к задаче перевода программа [7] модифицирована, и выявлено, что наибольшее время занимает работа генетического алгоритма. Использование метода имитации отжига для минимизации непосредственно целевой функции происходит значительно быстрее, но для минимизации результатов сортировки также медленно. Наибольшей скоростью работы отличается алгоритм 25. Все алгоритмы подбирают переводные слова из близких по смыслу областей деятельности.

Указанные приложения рассматривались в рамках дисциплин, преподаваемых автором. Исходная постановка задачи [8] была использована как олимпиадное задание на факультете дизайна и компьютерных технологий.

Список литературы

1. *Гордеев А. В.*, *Молчанов А. Ю.* Системное программное обеспечение: учебник для вузов. СПб.: Питер, 2003. — 736 с.
2. *Григорьев А. В.*, *Желтов В. П.* Генетический алгоритм на основе многоуровневых субпопуляций // Проблемы повышения качества образования: Деп. в НИИ ВО. Тез. докл. I межвузовской науч.-практ. конф. Чебоксары: Изд-во Чуваш. ун-та, 2002. С. 29–30.

[3] Версия без применения метода имитации отжига, мультистарта и вывода статистики в файл.

3. *Девятков В. В.* Системы искусственного интеллекта: учеб. пособие для вузов. М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. — 352 с.
4. *Димитриев А. П.* Алгоритм кластеризации слов в тексте // Динамика нелинейных дискретных электротехнических и электронных систем: материалы 10-й Всерос. науч.-техн. конф. Чебоксары: Изд-во Чуваш. ун-та, 2013. С. 115–116.
5. *Димитриев А. П.* Программа для исследования применимости алгоритмов решения задачи размещения к исходным данным с задаваемыми статистическими характеристиками. Свидетельство № 2013614618 от 16 мая 2013 г.
6. *Димитриев А. П.* Раскрашенная сеть Петри и задача размещения // Материали за 9-а международна научна практична конференция // «Бъдещите изследвания». 2013. Т. 27. Математика. София: «Бял ГРАД-БГ» ООД. С. 55–58.
7. *Димитриев А. П.* Чувашско-русский переводчик: программная реализация // Прикладная информатика. 2011. № 6 (36). Ноябрь — декабрь. С. 43–46.
8. *Димитриев А. П., Александров В. Н.* Один из методов оптимального управления групповой нагрузкой с широтно-импульсным регулированием мощности // Наука, образование, культура: Материалы XXX студ. науч. конф. Чебоксары: Чуваш. ун-т. 1996. С. 108.
9. *Желтов В. П., Димитриев А. П.* Стохастическая оптимизация расписания на сетях Петри. Чебоксары: Изд-во Чуваш. ун-та, 2001. — 213 с.
10. *Желтов П. В.* Модели поиска и копирования символьных данных на J-сетях // Прикладная информатика. 2012. № 4 (40). С. 81–83.
11. *Каширина И. Л.* Введение в эволюционное моделирование: учеб. пособие. Воронеж: ВГУ, 2007. — 40 с. URL: http://window.edu.ru/resource/519/59519/files/may07048.pdf [Дата обращения 14.05.2013 г.].
12. *Турчак Л. И., Плотников П. В.* Основы численных методов: учеб. пособие. М.: Физматлит, 2003. — 304 с.
13. *Чэн Ш.-К.* Принципы проектирования систем визуальной информации. М.: Мир, 1994. — 408 с.
14. *Чипига А. Ф., Колков Д. А.* Анализ методов случайного поиска глобальных экстремумов многомерных функций // Фундаментальные исследования. 2006. № 2. С. 24–26. URL: http://fr.rae.ru/pdf/2006/02/Chipiga_2.pdf [Дата обращения 30.07.2013].
15. *Kirkpatrick S., C. D. Gelatt Jr., M. P. Vecchi.* Optimization by Simulated Annealing // Science. 1983. № 220 (4598). P. 671–680.
16. *Land A. H., Doig A. G.* An autmatic method of solving discrete programming problems // Econometrica. V. 28 (1960). P. 497–520.

A. Dimitriev, *Ph. D. (Eng.), Assistant Professor, Chuvash State University n. a. I. N. Ulyanov, the city of Cheboksary, dimitrie1@yandex.ru*

Model and algorithms for automated text translation

The model for optimal words selection for he automated text translation is proposed. A number of scheduling algorithms optimising target function were developed and comparative analysis was done. A simplified mathematical timetable modelis used. Computational experiments were made and the results obtained for various input data are considered

Keywords: algorithm, Petri net, optimization, automated text translation.