8. З а п е ч н и к о в С. В. Повышение стойкости средств криптографической защиты информации на основе применения схем управления ключевым материалом // Материалы XII Международной конференции "Комплексная защита информации", Ярославль, 2008 г. – М.: РФК-Имидж Лаб, 2008. – С. 85–87.
9. З а п е ч н и к о в С. В. Модель "активной безопасности" и возможности ее реализации в системах криптографической защиты информации // Безопасность информационных технологий. – 1998. – № 4. – С. 52–54.
10. З а п е ч н и к о в С. В. Контроль целостности информационных ресурсов при распределенном хранении данных // Безопасность информационных технологий. – 2008. – № 2. – С. 86–91.

Статья поступила в редакцию 20.11.2008

Сергей Владимирович Запечников родился в 1974 г., окончил в 1997 г. Московский государственный инженерно-физический институт (технический университет). Канд. техн. наук, доцент кафедры "Информационная безопасность банковских систем" национального исследовательского ядерного университета "МИФИ". Автор 88 научных работ в области информационной безопасности и защиты информации.

S.V. Zapechnikov (b. 1974) graduated from the Moscow Engineering and Physics Institute (technical university) in 1997. Ph. D. (Eng.), assoc. professor of "Data Security of Banking Systems" department of the National Research Nuclear University "MIFI". Author of 88 publications in the field of data security and data protection.

УДК 004.78:025.4.036

В. М. В и ш н е в с к и й, Р. В. Ж е л е з о в

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННО-СПРАВОЧНАЯ СИСТЕМА ПОИСКА ОПТИМАЛЬНЫХ ПУТЕЙ ПРОЕЗДА НА ПАССАЖИРСКОМ ТРАНСПОРТЕ

Рассмотрены принципы построения и реализации информационно-справочной системы поиска оптимальных путей проезда на пассажирском транспорте. Описан оригинальный алгоритм поиска кратчайших путей с учетом расписаний пассажирского транспорта. Приведена архитектура программно-аппаратной реализации системы и интернет-сайта для доступа к справочной информации.
E-mail: vishn@iitp.ru

Ключевые слова: *информационная система, расписание, пассажирский транспорт, оптимизация, Интернет.*

Предоставление справочной информации об оптимальных путях проезда на пассажирском транспорте является необходимым условием для качественного обслуживания пассажиров. Полнота предоставленной информации не только помогает пассажиру, но и повышает эффективность пассажирских перевозок, уменьшает нагрузку на транспортные сети вследствие оптимизации пассажиропотока.

Первые электронные справочные системы расписаний транспорта появились в 80-х годах прошлого века. На постсоветском пространстве

функционируют системы ЭКСПРЕСС — на железнодорожном транспорте, СИРЕНА — на воздушном транспорте [1].

К сожалению, существующие системы справочно-информационного обслуживания пассажиров далеки от совершенства. В настоящее время в России даже в рамках отдельных видов транспорта отсутствует возможность поиска маршрута проезда с пересадкой. Исключение составляют автоматизированные информационные системы на воздушном транспорте, которые способны находить маршруты с пересадкой. Но на железнодорожном транспорте (который обслуживает автоматизированная система ЭКСПРЕСС) поиск сложного маршрута с пересадками невозможен. Например, на запрос пассажира о маршруте проезда из Калининграда во Владивосток, система ЭКСПРЕСС выдает ответ: “Прямое сообщение отсутствует. Подготовьте ответ вручную”. Данная задача гораздо более сложна для железнодорожного транспорта, чем для воздушного, так как число железнодорожных станций превосходит число аэропортов в сотни раз.

В Европе наиболее распространена система HAFAS [2], которая используется на железнодорожном транспорте в нескольких странах. Система HAFAS выдает удовлетворительные результаты поиска, хотя они не всегда оказываются оптимальными. Главной причиной этого является то, что алгоритм поиска внутри системы использует эвристические методы, чтобы уменьшить пространство поиска. В прототипе HAFAS имелось два уровня сети: *статический*, состоявший из ребер, стоимость которых определялась расстоянием, и *динамический*, где ребра включали информацию о времени прибытия и отправления поездов. Алгоритм использовал статический уровень, чтобы выявить нужную часть графа, без учета информации о времени. Это может приводить к потере оптимального пути.

Еще более фундаментальная задача — поиск интермодального маршрута (пути проезда с использованием нескольких видов транспорта). Следует отметить, что транспортная сеть в Российской Федерации весьма неоднородна. Есть населенные пункты, куда можно попасть только по воздуху. Имеются регионы, лишенные железнодорожного транспорта; есть населенные пункты, до которых можно добраться только на поезде. Поэтому разработка информационной системы, которая позволит объединить информацию из действующих автоматизированных систем на различных видах транспорта в целях получения наиболее полной справочной информации о возможности проезда с учетом пересадок и наличия мест на разных видах транспорта, является актуальной задачей.

Алгоритмы поиска оптимальных путей проезда на пассажирском транспорте. Существует два основных подхода к моделированию информации о расписаниях как задачи поиска кратчайшего пути:

время–расширенный и время–зависимый [3, 4]. Общее свойство обоих подходов заключается в том, что результат вычисляется с помощью некоторого алгоритма поиска кратчайшего пути к специально построенному графу. Время–расширенный граф конструируется таким образом, что каждый узел соответствует определенному времени (прибытия или отправления) на станцию, а ребра между узлами представляют либо элементарные соединения между двумя событиями, либо время ожидания на станции. На практике это приводит к созданию очень большого (хотя и разреженного) графа. Что касается время-зависимого подхода, то идея состоит в том, чтобы избежать создания узлов графа для каждого события. Вместо этого время-зависимый граф строится так, что каждый узел представляет станцию и два узла считаются связанными ребром, если соответствующие станции связаны элементарным соединением. Вес ребра присваивается "на лету" и зависит от времени, когда конкретное ребро будет использовано алгоритмом поиска кратчайшего пути.

Задача поиска формулируется следующим образом. Расписание состоит из следующих данных: *станции* (либо остановки, порты и др.), *маршруты* (поезда, автобусы, курсирующие между станциями), *время* отправления и прибытия маршрутов на станции, и *дни* курсирования. В математических терминах это можно записать так: имеется набор маршрутов Z, набор станций B и набор элементарных соединений C, элементы которого c — кортеж пяти величин в форме $c = (Z, S_1, S_2, t_d, t_a)$. Каждое элементарное соединение понимается как маршрут Z, отправляющийся со станции S_1 во время t_d, следующая станция маршрута S_2, время прибытия на которую t_a. Длина элементарного соединения c обозначается length(c). Это время между прибытием и отправлением c. Расписание действует для числа N дней курсирования. Каждому маршруту соответствует битовое поле из N битов, определяющих, в какие дни маршрут курсирует. На станции $S \in B$ возможна пересадка с одного маршрута на другой только в случае, если время между отправлением и прибытием на станцию S больше либо равно минимальному времени пересадки для этой станции (S). Для станций, расположенных близко друг к другу, возможно прохождение пешком. Это можно описать введя так называемые *пешие ребера* между станциями. Формально, мы рассматриваем пешее ребро как элементарное соединение c, где маршрут Z, времена отправления и прибытия t_d и t_a не определены, но length(c) определено и равно времени пешего перехода.

Пусть $P = (c_1, \ldots, c_k)$ — последовательность элементарных соединений (включая пешие ребра), с заданными отправлениями $dep_i(P)$ и прибытиями $arr_i(P)$ для каждого элементарного соединения c_i, $1 \leqslant i \leqslant k$. Полагаем, что время прибытия и отправления включают

даты прибытия и отправления и учитывается также время, прошедшее с первого дня поездки. Время $t = a \cdot 1440 + b$, где $a \in [0, 364]$ и $b \in [0, 1439]$. Такая последовательность P называется *постоянным соединением* станций $A = S_1(c_1)$ и $A = S_2(c_k)$. Формально постоянное соединение удовлетворяет следующим условиям:

c_i действует 1 день $[dep_i(P)/1440]$,

$S_2(c_i) = S_1(c_{i+1})$,

$dep_i(P) \equiv t_d(c_i) (\mathrm{mod}\ 1440)$,

$arr_i(P) = dep_i(P) + \mathrm{length}(c_i)$,

$$dep_{i+1}(P) - arr_i(P) \le \begin{cases} 0, & \text{если } Z(c_{i+1}) = Z(c_i) \text{ или } c_i - \text{пешее ребро;} \\ transfer(S_2(c_i)), & \text{иначе.} \end{cases}$$

Наиболее известная частная задача о расписаниях называется также *задачей наискорейшего прибытия.* Запрос (A, B, t_0) определяет станцию отправления A, станцию прибытия B и время отправления t_0. Соединение допустимо, если отправление из пункта A не происходит раньше заданного времени t_0. Критерий оптимизации — минимизировать разницу между временем прибытия и заданным временем отправления. В другой задаче — *задаче минимального числа пересадок* необходимо отыскать путь с наименьшим числом пересадок для станций A и B, вообще не принимая во внимание время прибытия и отправления.

Представленные подходы могут быть использованы для моделирования реальной задачи, например, учитывающей время пересадок. Чтобы учесть время пересадки во время–расширенной модели, на графе создается копия всех узлов прибытия и отправления со станции, которые назовем *узлами пересадки*. Для каждого узла прибытия существует два исходящих ребра: одно ребро — к отправлению того же маршрута, другое ребро — к узлу пересадки со временем, бóльшим или равным сумме времени прибытия в узел и минимального времени, требуемого для пересадки на данной станции.

Часто пассажиры бывают заинтересованы в том, чтобы найти самый дешевый путь из пункта A в пункт B в заданном интервале времени. К сожалению, тарифная система в большинстве стран настолько сложна, что практически невозможно решить такую задачу поиска и точно и быстро. Даже в стандартных тарифах стоимость проезда обычно неаддитивна. Еще хуже обстоит дело, если требуется учесть специальные тарифы.

Многокритериальная оптимизация по нескольким критериям необходима, когда пассажиру требуется найти путь с минимальным числом пересадок, который начинается после заданного момента времени и не заканчивается слишком поздно. Вычисление оптимального

пути по нескольким критериям сводится к решению задачи поиска кратчайшего пути по нескольким критериям (MOSP, multi-objective shortest path) — основной задачи в области многоцелевой оптимизации [5]. Задача многокритериальной оптимизации обычно NP-полная (действительно, в случае MOSP), так как размер набора Парето обычно экспоненциально возрастает. Даже для очень простых графов (цепь узлов, соединенных ребрами) в задаче с двумя критериями число решений в наборе Парето растет экспоненциально. Поэтому на практике для нахождения решения многокритериальной задачи используются приближенные подходы. Можно применить такой подход, как относительная важность критериев оптимизации по весу и последующая оптимизация весовой суммы. Этот подход сводит многокритериальную задачу к задаче с одним параметром, которая может быть решена стандартным алгоритмом Дейкстры (используется модель, где каждый критерий неотрицателен и аддитивен). Недостаток такого подхода — проблемы выбора подходящих весовых параметров. Каждый пассажир имеет свои предпочтения, и система может установить параметры неверно.

Еще один способ выбрать одно Парето-оптимальное решение — это *лексикографический порядок*. В упрощенной версии время–расширенного подхода лексикографически первое решение может быть рассчитано для любого d значений как вес ребер. Например, если $d = 2$ и первый элемент — время в пути, а второй — число пересадок, то среди всех быстрых путей находится один с минимальным числом пересадок. В практической версии время–расширенного подхода могут быть использованы только те критерии, где первый критерий — время в пути.

Исследования известных алгоритмов поиска по расписаниям показали, что без применения специальной ускоряющей техники (модернизации алгоритмов) производительность обычно оказывается недостаточной. Средняя производительность особенно важна в системах с центральным сервером, который должен обрабатывать сотни запросов одновременно, например в Интернете или на транспортных терминалах (вокзалах, аэропортах). Подходы, описанные для решения упрощенной задачи наискорейшего прибытия, были хорошо изучены для время–расширенной и время–зависимой моделей. Нахождение всех подходящих путей с учетом времени пути, стоимости, числа пересадок, естественно, гораздо сложнее, чем просто поиск наискорейшего пути.

Алгоритм, который гарантирует точное решение (оптимальный путь) для любого запроса, называют *сохраняющим расстояние* алгоритмом. На практике такие алгоритмы используются не очень часто, так как время их работы обычно оказывается недопустимо большим.

Перечислим некоторые ускоряющие подходы, которые позволяют повысить производительность алгоритма поиска. При применении техники *ограничение угла* используют информацию о пространственном расположении географических объектов. На этапе *препроцессинга* (предварительной обработки данных) применяют алгоритм Дейкстры, чтобы вычислить кратчайшие пути из узла до всех других станций [6]. Результаты вычислений не сохраняются, так как это потребовало бы слишком много места, но сохраняются два значения углов $a(v,w)$ и $b(v,w)$, которые сопоставляются ребру графа. Эти значения вычисляются и хранятся в системе для каждого ребра. Углы рассчитываются следующим образом. Пусть (v,w) — ребро, соединяющее событие v с другим событием w на время–расширенном графе, и пусть S_v и S_w — станции, к которым принадлежат эти события. Нарисуем угловой сектор между углами $a(v,w)$ и $b(v,w)$ с вершиной в точке S_v. Углы a и b определяются таким образом, что конечные точки кратчайших путей, проходящих через ребро, будут внутри сектора. Позднее, во время работы алгоритма ребра (v,w) могут быть проигнорированы поиском, если целевой узел не входит в угловой сектор. Алгоритм является точным, но требует предварительной обработки данных.

Основная идея другой техники *выбора станций* применяется при планировании маршрута на транспорте [7]. В результате число рассматриваемых узлов может уменьшиться в несколько раз. Пусть $G = (V,E)$ означает маршрутный граф; $V^{\sim}$ — набор выбранных узлов, из которых строится вспомогательный граф $G^{\sim}$. Вспомогательный граф G' и веса ребер создаются и задаются только один раз на этапе препроцессинга. После этого любой запрос может быть обработан на вспомогательном графе G', и путь будет соответствовать кратчайшему пути на графе G. Техника *поиск в направлении цели* использует потенциальную функцию набора узлов. Веса ребер в очереди изменяются так, чтобы направить поиск по графу в сторону цели. Пусть λ — эта потенциальная функция. Новый вес ребра (v,w) определяется как $l(v,w) = l(v,w) - \lambda(v) + \lambda(w)$.

Двунаправленный поиск, ускоряющий алгоритм Дейкстры, может быть успешно применен и для графа транспортных маршрутов. Прямой алгоритм Дейкстры ищет из узла отправления, а обратный — из узла прибытия.

Иерархический поиск требует препроцессинга, во время которого исходный граф $G = (V,E)$ разбивается на несколько уровней $(l+1)$ и дополняется дополнительными кратчайшими путями между определенными узлами [8]. Чтобы найти кратчайший путь между узлами с помощью алгоритма Дейкстры достаточно рассмотреть меньший подграф многоуровневого графа. Еще одна интересная техника использует понятие *радиуса достижения*. Если v — вершина на кратчайшем пути

P из s в t, то радиус достижения по отношению к $P, r(v, P)$ есть минимум от длины префикса P (часть пути от s к v) и длины суффикса (часть пути от v к t). Радиус достижения узла $v, r(v)$ — это максимум $r(v, P)$ по всем кратчайшим путям, которые содержат v. Вычислив предварительно $r(v)$ для всех узлов можно использовать эту информацию во время работы алгоритма Дейкстры. Техника *прямоугольника кратчайших узлов* похожа на метод *ограничения угла*. На этапе препроцессинга обрабатываются все деревья кратчайших путей. Для каждого ребра $e \in E$ вычисляется набор узлов $S(e)$, кратчайшие пути к которым проходят через ребро e, и для каждого $e \in E$ сохраняется ограничивающий прямоугольник $S(e)$. Во время работы алгоритма можно быстро узнать, по какому ребру необходимо двигаться дальше, исключая "на лету" те ребра, которые не ведут к цели.

Оригинальный алгоритм поиска оптимального пути на пассажирском транспорте. В разработанной информационно-справочной системе используется оригинальный алгоритм поиска, который строго решает двухкритериальную задачу поиска пути с минимальным числом пересадок и с наискорейшим временем прибытия [9, 10].

На первом этапе проводится оптимистический поиск по виртуальному графу. В результате находятся все пути с одинаковым минимальным числом пересадок. Найденные пути анализируются на следующем шаге алгоритма, где проверяется допустимость каждого с учетом дней курсирования и решается задача наискорейшего прибытия. Затем среди допустимых путей выбирается оптимальный (наискорейший). Если все пути оказались недопустимы, то повторяется первый этап поиска по графу. Важно, что поиск по графу должен быть оптимистическим (находить все возможные пути с минимальным числом пересадок) — обеспечивается точностью алгоритма.

Есть два широко распространенных способа представления графа $G = (V, E)$: как набор смежных вершин или как матрицу смежности. Но оригинальный алгоритм реализован на встроенном языке базы данных и адаптирован к работе с таблицами, и граф в явном виде вообще не хранится в системе. Оригинальный алгоритм оперирует со специально преобразованными исходными данными. Это позволяет решать задачу точно, не сводя ее к эвристическим методам. Также преимуществом является то, что в оригинальной реализации не требуется полного препроцессинга, при изменениях в исходных данных.

Алгоритм Дейкстры, адаптированный к работе с преобразованными данными, также может быть применен для поиска. Но он не является достаточно быстрым для практической реализации в информационно-справочной системе. Для ускорения работы оригинального алгоритма применяются специальные техники. Известные

методики, например методика “ограничения угла”, не могут быть применены. Поскольку в оригинальной реализации информация о ребрах графа вообще не хранится в системе.

В качестве альтернативы разработаны несколько техник ускорения, используемых в системе. Метод *пересадочного подграфа* заключается в выделении потенциальных узлов пересадки. В процессе загрузки исходных данных о расписаниях можно анализировать указанные в расписании маршрута станции и определять потенциальные узлы пересадки. Разработаны несколько критериев, на основе которых станция добавляется в пересадочный подграф. Техника *пересадочный подграф* схожа с техникой *иерархического поиска*, но в отличие от последней пересчет пересадочного подграфа проводится очень быстро. Еще один очень эффективный метод *балансировка двунаправленным поиском* заключается в особом выборе прямого или обратного поиска на каждом шаге итерации. Разработан также метод *ускорения последней итерации*. Для каждой станции вычисляется ограничивающий прямоугольник станций, достижимых без пересадок. На последнем шаге итерации используется информация об ограничивающем прямоугольнике каждой станции для выяснения, может ли станция быть пересадочным пунктом или ее заведомо можно исключить. Еще один эффективный метод ускорения, хотя и не являющийся точным, — это метод *ускорения* A^*, основанный на эвристическом алгоритме A^*. Хотя алгоритм A^* в данной задаче не является методом, сохраняющим расстояния, но на практике почти всегда оказывается точным.

Данные о наличии свободных мест — наиболее быстро обновляющаяся часть транспортной информации. Если пропускная способность канала связи с автоматизированными системами ограничена, а объем данных велик, то физически невозможно импортировать актуальную информацию о наличии свободных мест из автоматизированных систем. Поэтому проверка наличия мест проводится на последнем шаге работы алгоритма, в тот момент когда уже найден оптимальный путь проезда по остальным критериям.

Разработанный алгоритм положен в основу информационно-справочной системы.

Архитектура и принципы построения автоматизированной информационно-справочной системы. Разработанная информационно-справочная система — это целый комплекс программ и программных модулей, которые обеспечивают ее функционирование [11]. Программный комплекс включает следующие программные компоненты.

1. Специализированная база данных геоинформации и расписаний.

2. Средства подготовки данных для информационно-справочной системы.

3. Средства импорта данных из промежуточного формата XML и других внешних форматов.

4. Средства редактирования данных в системе.

5. Модуль ядра. Реализация алгоритмов поиска маршрута на транспорте.

6. Службы интеграции с другими информационными системами: *а*) интеграционный сервис, функционирующий в режиме on-line; *б*) эмулятор терминала системы ЭКСПРЕСС.

7. Справочный интернет-портал. Интерфейс доступа к справочной системе

Специализированная база данных содержит несколько десятков таблиц и реализацию алгоритмов работы с данными. Алгоритмы запрограммированы в виде хранимых процедур на языке T-SQL. Специфика информационно-справочной системы потребовала разработки специальных *средств подготовки данных* — среды визуального описания транспортного графа. Программа представляет собой Windows-приложение, написанное на языке C++. С помощью этой программы можно описать фрагменты единого графа. В программе можно добавлять и удалять узлы, менять их названия и расположение. Программа сохраняет результат работы в файлах формата XML. Для каждого региона может быть создан отдельный XML-файл, где указано расположение местных транспортных узлов. Таким способом возможно по частям описать все фрагменты железнодорожной и автобусной сети. С помощью разработанной программы для системы были построены фрагменты графа с общей численностью около двадцати тысяч узлов.

После того как географические данные подготовлены в промежуточном XML-формате они загружаются в систему с помощью *программы импорта данных*. Программа обрабатывает данные XML-формата, затем подключается к базе данных системы и вносит в нее изменения. Текущая база данных загружена из нескольких десятков файлов XML и образует граф примерно из 20 000 вершин. В это число вошло большинство железнодорожных станций, вокзалов, платформ на территории РФ, ближнего и дальнего зарубежья. Также загружены основные автобусные станции нескольких регионов России. Процесс загрузки состоит из следующих шагов: считывание данных из файла, анализ данных, устранение неоднозначностей, формирование объектной структуры, сохранение данных. Программа импорта состоит из двух функциональных модулей: (i) загрузчик географических данных, (ii) загрузчик расписаний.

Алгоритм импорта географических объектов содержит следующие этапы расчета.

1. Распознавание названия и типа объекта.

2. Поиск в базе данных объектов со схожим названием.
3. Добавление объекта в базу данных.
4. Обновление топологических связей.

Алгоритм импорта расписаний имеет следующую схему:

1. Определение структуры данных.
2. Считывание данных в промежуточную таблицу.
3. Объектное структурирование данных.
4. Сохранение структур в базу данных.
5. Корректировка вспомогательных данных.

Если в маршрут следования поезда входит станция, принадлежащая гиперузлу, то в базу данных добавляются дополнительные записи о расписании. В процессе импорта расписания маршрута специальным образом обновляется пересадочный подграф.

Создание справочного запроса к системе начинается с указания параметров (пункты отправления и назначения, виды транспорта, дата поездки и др.). После того как пользователь настроил параметры запроса и нажал кнопку “Искать”, программный объект, содержащий все настройки, передается в модуль поиска пути. Используя идентификаторы пунктов назначения и отправления, модуль получает необходимую информацию о географических объектах. После этого алгоритм пытается отыскать беспересадочный маршрут между заданными пунктами. Если прямой маршрут не найден, то используется оригинальный алгоритм поиска пути с пересадками. Хранимая процедура возвращает временну́ю таблицу, в которой содержится информация о найденных путях проезда. Сначала эти пути сортируются по длине пути. Затем алгоритм рассматривает по порядку все участки пути. Для каждого участка определяется время отправления и прибытия наискорейшего маршрута. В результате такого анализа путей вычисляется наискорейшее время прибытия для каждого из них. После этого список путей сортируется по времени наискорейшего прибытия. На последнем шаге наискорейший из найденных путей обрабатывается модулем взаимодействия с внешними системами.

Подсистема интеграции с внешними системами используется для получения информации о наличии свободных мест. В подсистему входят следующие компоненты: модуль интеграции в составе модуля поиска пути; сервис интеграции с внешними системами; эмулятор терминала системы ЭКСПРЕСС; программа контроля взаимодействия с внешними системами.

Когда пользователь делает запрос, то поиск осуществляется в следующей последовательности: (i) находятся возможные пути проезда, (ii) проверяются даты курсирования маршрутов, (iii) проверяется наличие свободных мест. В подсистеме интеграции сначала проводится

поиск данных в кэше выполненных запросов. В кэш попадают результаты предыдущих запросов к внешним системам, которые сохраняются там в течение установленного времени. Если данные не найдены в кэше, подсистема отправляет запрос во внешние автоматизированные системы. Способ отправки запроса зависит от особенностей внешней системы. Запрос может быть синхронным или асинхронным. Если запрос *синхронный*, то обслуживающий процесс выполняет его немедленно. Если используется *асинхронный* способ взаимодействия, например с системой ЭКСПРЕСС, то запрос ставится в очередь. Когда в очереди появляются запросы на обслуживание, к работе подключается программа-сервис взаимодействия. Запрос из очереди трансформируется сервисом в запрос по необходимому протоколу, понятному внешней системе. Одновременно могут работать несколько сервисов взаимодействия, и они могут быть запущены на нескольких вычислительных машинах для просмотра единой очереди запросов. Сервисы могут специализироваться на обслуживании запросов к определенным внешним системам.

Взаимодействие с системой ЭКСПРЕСС осуществляется через сервис-эмулятор терминала. При получении запроса, для которого необходимо получить информацию из системы ЭКСПРЕСС, эмулятор терминала преобразует запросы системы в пакеты BSC-3 сети ЭКСПРЕСС, инкапсулирует их в пакеты TCP/IP и передает на шлюз доступа к системе ЭКСПРЕСС. Далее шлюз обменивается информацией с ХОСТ-ЭВМ и возвращает ответ эмулятору. Процесс протекает асинхронно: один запрос системы может отображаться в серию запросов-ответов между эмулятором и ХОСТ-ЭВМ.

Для получения справочной информации о возможности проезда через сеть Интернет разработан сайт доступа к информационно-справочной системе. Сайт предоставляет пользователям информацию о пути проезда на железнодорожном и автобусном транспорте РФ, ближнего и дальнего зарубежья с учетом пересадок, обеспечивая поиск:

- расписания транспорта для прямых маршрутов проезда между двумя пунктами;
- пунктов пересадки, когда прямого пути между пунктами нет;
- маршрутов только на отдельном интересующем виде транспорта (автобусы, поезда);
- пути проезда между двумя пунктами с пересадкой в третьем явно указанном пункте;
- пути проезда с ограничением на максимальное число пересадок;
- интермодального пути проезда (разные виды транспорта);
- пути проезда на заданную дату. Маршруты транспорта, которые не удовлетворяют указанной дате, не отображаются в результатах;

- пути проезда со всех возможных вокзалов города или явное указание с какого вокзала искать путь.

Кроме того, сайт предоставляет услуги по отображению

- информации о расписании движения по станции;
- найденных путей проезда на интерактивной карте;
- интерактивной схемы беспересадочных маршрутов для указанной станции.

Внедрение разработанной системы позволит резко повысить ка-

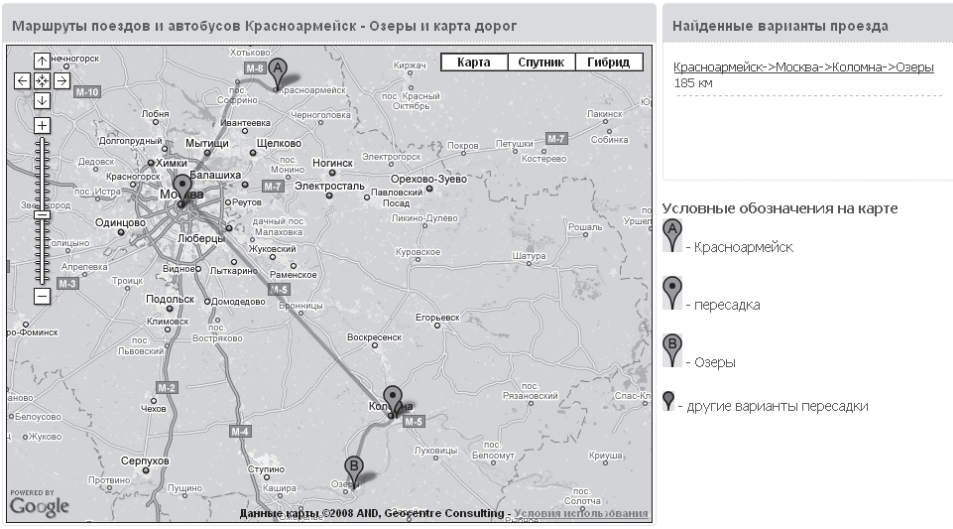

Рис. 1. Результат поиска пути проезда по Московской области из г. Красноармейск в г. Озеры

Рис. 2. Результат поиска пути проезда из г. Бобруйск (Белоруссия) в г. Дальнегорск (Приморский край) с использованием железнодорожного и автомобильного транспорта

Рис. 3. Интерактивная схема беспересадочных маршрутов по железнодорожной станции Калининград

чество информационного обслуживания пассажиров, оптимизировать пассажиропотоки и загрузку транспортных средств.

Примеры, демонстрирующие работу автоматизированной системы, приведены на рис. 1–3.

СПИСОК ЛИТЕРАТУРЫ

1. Вишневский В. М. Теоретические основы проектирования компьютерных сетей. – М.: Техносфера, 2003. – 512 с.
2. HAFAS. A timetable information system by HaCon Ingenieurgesellschaft mbH, Hannover, Germany. http://www.hacon.de/hafas/
3. Matthias Muller-Hannemann, Frank Schulz, Dorothea Wagner, Christos Zaroliagis. Timetable information: Models and algorithms in algorithmic methods for railway optimization, Springer Berlin / Heidelberg, 2007.
4. Brodal G. S. and Jacob R. Time-dependent networks as models to achieve fast exact time-table queries. Technical Report ALCOMFT-TR-01-176, BRICS, University of Aarhus, Denmark, 2001.
5. Muller-Hannemann M. and Schnee M. Finding all attractive train connections by multicriteria Pareto search // Proc. of the 4th Workshop in Algorithmic Methods and Models for Optimization of Railways (ATMOS 2004).
4. Brodal G. S. and Jacob R. Time-dependent networks as models to achieve fast exact time-table queries // Proc. 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003), vol. 92 of Electronic Notes in Theoretical Computer Science. Elsevier, 2004.

5. Cooke K. L. and Halsey E. The shortest route through a network with time-dependent internodal transit times // Journal of Mathematical Analysis and Applications, 14:493–498, 1966.
6. Wagner D. and Willhalm T. Geometric speed-up techniques for finding shortest paths in large sparse graphs // Proc. 11th European Symposium on Algorithms (ESA 2003). Vol. 2832 of LNCS. P. 776–787. Springer, 2003.
7. Wagner D. and Willhalm T. Speed-up techniques for shortest path computations. In Algorithmic Methods for Railway Optimization, LNCS. Springer.
8. Schulz F., Wagner D. and Zaroliagis C. Using multi-level graphs for timetable information in railway systems // Proc. 4th Workshop on Algorithm Engineering and Experiments (ALENEX). Vol. 2409 of LNCS. P. 43–59. Springer, 2002.
9. Вишневский В. М., Железов Р. В., Атанасова Т. Н. Единая справочная система на пассажирском транспорте Российской Федерации // Distributed Computer and Communication Networks, Техносфера, 2005. – С. 165–172.
10. Железов Р. В. Особенности алгоритма поиска маршрута на транспорте с учетом расписаний движения // Труды междунар. сем. Распределенные компьютерные и телекоммуникационные сети (DCCN'2007), М., ИППИ РАН, 2007. – С. 28–32.
11. Железов Р. В. Реализация единой справочной системы пассажирского транспорта на базе технологий Microsoft // Сб. трудов конф. Технологии Microsoft в теории и практике программирования. – М.: МГТУ им. Н.Э. Баумана, 2006. – С. 10–13.

Статья поступила в редакцию 6.11.08

Владимир Миронович Вишневский родился в 1944 г., в 1971 г. окончил МИЭМ. Д-р техн. наук, профессор, заведующий отделом Института проблем передачи информации им. А.А. Харкевича РАН. Автор 170 научных работ в области теории и практики создания информационно-вычислительных сетей и систем.

V.M. Vishnevskii (b. 1944) graduated from the Moscow Institute of Electronic Engineering in 1971. D. Sc. (Eng.), professor, head of department of the Institute for Problems of Information Transfer n.a. A.A. Kharkevich, RAS. Author of 170 publications in the field of theory and practice of creating information and computation networks and systems.

Роман Владимирович Железов родился в 1981 г., в 2005 г. окончил МФТИ. Канд. техн. наук, автор 8 научных работ в области теории и практики создания информационно-вычислительных сетей и систем

R.V. Zhelezov (b. 1981) graduated from the Moscow Physics and Technology Institute in 2005. Ph. D. (Eng.), assoc. professor of the Moscow Physics and Technology Institute. Author of 8 publications in the field of theory and practice of creating information and computation networks and systems.