встает вопрос вычисления кратчайших путей от склада, либо от места производства до точек реализации, путем решения задачи оптимизации маршрута для автотранспорта предприятия. Для решения этой задачи необходимо рассмотреть различные задачи маршрутизации и проанализировать методы их решения, выбрав наиболее приемлемый.

Класс задач, оптимизирующий маршруты для автотранспорта предприятия относится к многокритериальным задачам маршрутизации, поскольку часто нужно указывать среди критериев оптимизации не только время доставки, но и километраж, класс дороги. В современном мире данный класс задач имеет широкое прикладное значение, в связи с чем постоянно разрабатываются и совершенствуются методы их решения.

Цель работы состоит в том, чтобы разработать автоматизированную систему оптимизации маршрутов доставки продукции.

В качестве примера создана программа в среде Delphi, позволяющая решать однокритериальные задачи коммивояжера. Данная программа позволяет заметно ускорить, в сравнении с поиском вручную, процесс нахождения оптимального маршрута, а также сократить процент ошибок при расчетах.

В работе основное внимание уделяется решению задач методом динамического программирования. Дальнейшим развитием тематики могло быть более подробное изучение других подходов к решению, в частности, метода ветвей и границ для многокритериальных задач коммивояжера, оценка его вычислительной сложности и программная реализация алгоритма. Также особого внимания заслуживает рассмотрение различных операторов выбора, как способ увеличения эффективности решающих алгоритмов.

ПОДСИСТЕМА АВТОМАТИЗАЦИИ РЕШЕНИЯ ФИНАНСОВЫХ ЗАДАЧ

Хамов М.В.
Воронежский институт высоких технологий, Воронеж,
e-mail: hamovmax@yandex.ru

Главной задачей современных информационных технологий финансового управления является своевременное предоставление достоверной, в необходимом количестве информации специалистам и руководителям для принятия обоснованных управленческих решений.

Целью данной работы является автоматизация решения финансовых задач путем разработки автоматизированной подсистемы принятия управленческих решений при формировании инвестиционного портфеля.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Анализ современного состояния проблемы принятия эффективных решений в области инвестиционного проектирования;
2. Построение математической модели принятия управленческих решений в области инвестиционного проектирования на основе кластерного анализа;
3. Разработка алгоритма оптимизации инвестиционного портфеля на основе кластерного анализа;
4. Разработка программной подсистемы на основе предложенных алгоритмов.

В результате проведенной работы над дипломным проектом была изучена различная научная литература, связанная с управлением и автоматизацией инвестиционных проектов в условиях экономического риска.

Построенная в результате программа, по сравнению со своими аналогами, отличается ценовой категорией, простотой работы и малыми требованиями к системным ресурсам, что позволяет быстрее производить оптимизацию. Так же было определено, что при использовании нового проекта существенно повышается производительность работ, и снижаются годовые эксплуатационные издержки. Следовательно, существует высокая вероятность успешной реализации проекта.

Секция «Использование информационных технологий для повышения эффективности производства и управления», научный руководитель – Кочеткова О.В., д-р техн. наук, профессор

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗИРОВАННЫХ СИСТЕМ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ С ИСПОЛЬЗОВАНИЕМ VISUAL STUDIO И БАЗЫ ДАННЫХ ORACLE

Варкентин В.В., Барбасова Т.А.
ФГБОУ ВПО «Южно-Уральский государственный университет» (НИУ), Челябинск,
e-mail: tatyana_barbasova@mail.ru

В данной работе рассматривается организация связи программного обеспечения автоматизированных систем управления технологическими процессами [1], разрабатываемого в объектно-ориентированной среде Visual Studio на языке C# с объектно-реляционной СУБД Oracle Database (или Oracle RDBMS).

Для разработки подобных программ первым шагом должно быть подключение пространства имен *System.Data.OracleClient* [2-4]. Данное пространство имен больше не включается в состав поставщика данных *.Net Framework*, поэтому его необходимо подключать отдельно (например, загрузить с официального сайта Oracle).

Для подключения необходимо в окне «Обозреватель решений» («Solution Explorer») кликнуть правой кнопкой мыши по разделу «*References*» (см. рис. 1а) и в появившемся меню выбрать «Добавить ссылку» («*Add Reference*»). В появившемся окне (см. рис. 1б) нажать кнопку *«Обзор» («Browse»)* и указать путь к месту расположения пространства имен.

После этого, в коде программы необходимо добавить использование этого пространства. Для этого, нужно ввести следующее:

using namespace System.Data.OracleClient;

После этого действия можно начинать писать код программы, выполняющий взаимодействие уже непосредственно с базами данных. Первым делом необходимо реализовать подключение к СУБД.

Подключение к базе данных производится с использованием класса OracleConnection(*connectionString*). В переменной connectionString должна содержаться информация о сервере, на котором расположена база данных. Ее должно значение приведено ниже:

string connectionString = "Data Source=(DESCRIPTION=(ADDRESS= (PROTOCOL= TCP)(HOST=[HOST_NAME])(PORT=1521))(CONNECT_DATA=(SERVER=DEDICATED) (SERVICE_NAME=[SERVICE_NAME]))); User ID= [USER_NAME];Password= [USER_PASSWORD]"

Вместо *[HOST_NAME]*, *[SERVICE_NAME]*, *[USER_NAME]* и *[USER_PASSWORD]* необходимо указать параметры подключаемой базы данных, расположенной на удаленном (в частном случае локальном) сервере. Данные параметры описаны в файле «tnsnames.ora», расположенном по следующему пути:

«...\Oracle\product\[version]\server\NETWORK\ADMIN».

а *б*

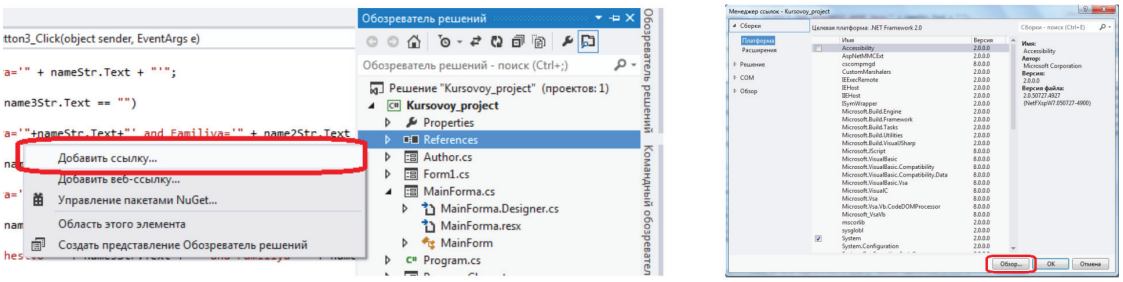

Рис. 1. Окна подключения пространства имен

Подключение к базе данных реализуется следующим кодом:

OracleConnection Oracle_Connect = new OracleConnection(connectionString);
Oracle_Connect.Open();
Oracle_Connection.Close();

Как видно из приведенного фрагмента – создается экземпляр класса OracleConnection названный *Oracle_Connect*, обладающий свойством connectionString и методами Open() и Close(), которые соответственно позволяют создать и завершить подключение к СУБД.

После того как подключение к СУБД было рассмотрено, целесообразно показать, каким образом оправляется запрос к СУБД.

Взаимодействие с СУБД производится при помощи SQL-запросов. Для использования таких команд SQL как *CREATE TABLE*, *DROP TABLE*, *INSERT*, *DELETE*, *UPDATE* и подобных им, то есть тем, которые не получают в ответ никакой информации, можно использовать следующий код:

string MyQuery = "[SQL-запрос]";
using (OracleConnection conn =
new OracleConnection(connectionString))
{ OracleCommand MySQLCommand =
new OracleCommand(MyQuery, conn);
MySQLCommand.Connection.Open();
MySQLCommand.ExecuteNonQuery(); }

OracleCommand – это класс, позволяющий отправить команду к СУБД. Как видно из фрагмента кода, этот класс обладает свойствами *MyQuery* и *conn*. Соответственно необходимо заранее задать значения этих свойств, что позволит конструктору правильно выполнить свои функции.

MyQuery – строковая переменная, в которой хранится текст SQL-запроса. Для правильного функционирования отправки запроса, в конце запроса, хранимого в этой переменной нельзя ставить знак «;».

Conn – строковая переменная, содержащая строку подключения (аналогичную Oracle_Connect).

ExecuteNonQuery – метод, применяющий SQL-команду применительно к свойству *conn*.

Поскольку язык SQL-запросов предназначен не только для создания и редактирования таблиц, но и для отображения результатов этих операций, то необходимо рассмотреть эту возможность средствами языка C# и среды Visual Studio.

Рассмотрим пример, в котором используются возможности Visual Studio по созданию форм (окон с различным наполнением – например, кнопками, полями для ввода текста, выпадающими меню и т.п.).

Добавьте на компонент Form вашей программы (решение «Приложение Windows Form») компонент *dataGridView*. Управление *DataGridView* предоставляет ориентированную настраиваемую таблицу для отображения данных. Класс *DataGridView* поддерживает настройку ячеек, строк, столбцов и платформы с помощью свойств *DefaultCellStyle, ColumnHeadersDefaultCellStyle, CellBorderStyle и GridColor*.

Можно использовать элемент управления DataGridView для отображения данных как с базовым источником данных, так и без него. Без определения источника данных можно создать столбцы и строки, которые содержат данные и добавить их непосредственно в *DataGridView* с помощью свойства *Rows* и *Columns*. Можно также использовать коллекцию *Rows* для доступа к объектам DataGridViewRow, а для чтения или записи значения ячеек напрямую – свойство *DataGridViewRow.Cells* . Индексатор Item также предоставляет прямой доступ к ячейкам.

В качестве альтернативы заполнения элемента управления вручную можно задать свойства *DataSource и DataMember* для привязки *DataGridView* к источнику данных и автоматически заполнить ее данными.

При работе с очень большими объемами данных можно задать свойство «*VirtualMode = true*», чтобы отобразить подмножество доступных данных. Виртуальный режим требует реализации кэша данных, из которого элемент управления *DataGridView* заполнен.

Ниже приведен код, позволяющий реализовать отображение данных вслед SQL-запросу SELECT.

OracleConnection conn = new OracleConnection(connectionString); conn.Open();
OracleCommand command =
conn.CreateCommand(); command.CommandText = MyQuery;
DataTable myTable = new DataTable();
OracleDataAdapter adapt =
new OracleDataAdapter(command);
adapt.Fill(myTable); BindingSource bind =
new BindingSource();
bind.DataSource = myTable; dataGridView1.DataSource = bind.DataSource;

Command – экземпляр класса OracleCommand хранит в своем свойстве *CommandText* текст команды из переменной *MyQuery*. *MyTable* – особая переменная, в которой будет храниться таблица значений.

OracleDataAdapter выполняет роль моста между свойством *DataSet* (расположенный в памяти кэш данных) и базой данных при извлечении и сохранении данных. OracleDataAdapter обеспечивает такой мост с помощью метода *Fill* для того, чтобы загрузить данные из базы данных в *DataSet*, и с помощью метода *Update* для того, чтобы отправить изменения, произведенные в *DataSet*, обратно в источник данных.

OracleDataAdapter заполняет *DataSet* и одновременно создает необходимые таблицы и столбцы для возвращенных данных, если таких таблиц и столбцов еще нет. Однако сведения о первичном ключе не включаются в схему, созданную в явном виде, если для свойства MissingSchemaAction не задан объект AddWithKey. С помощью OracleDataAdap-ter также можно создать схему *DataSet*, в которую будут включены сведения о первичном ключе, прежде чем она будет заполнена данными с помощью метода *FillSchema*.

Код «adapt.Fill(myTable);» заполняет элемент *myTable* данными, полученными в ответ на SQL-запрос «SELECT …».

Компонент BindingSource упрощает привязку элементов управления к форме представления данных. Свойство *DataSource* у компонента BindingSource, предварительно заполнив данными из таблицы *myTable*, можно использовать для заполнения таблицы на форме (компонент dataGridView.DataSource).

Применяя приведенные выше конструкции легко можно писать программное обеспечение автоматизированных систем управления технологическими процессами при использовании СУБД Oracle Database.

Список литературы

1. Автоматизированные системы управления энергоэффективным освещением: монография / под ред. Л.С. Казаринова / Л.С. Казаринов, Д.А. Шнайдер, Т.А. Барбасова и др. – Челябинск: Издательский центр ЮУрГУ; издатель Т. Лурье, 2011. – 208 с., ил.
2. http://msdn.microsoft.com/ru-ru/library/system.data.oracleclient.aspx
3. http://sql-language.ru.
4. http://ru.wikipedia.org/wiki/SQL.

АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИХ ОБЪЕКТОВ НА ОСНОВЕ ТЕХНОЛОГИИ БЕСПРОВОДНОЙ ПЕРЕДАЧИ ДАННЫХ ZIGBEE

Ташкинов А.Ю., Новоселов А.О., Барбасова Т.А.

ФГБОУ ВПО «Южно-Уральский государственный университет» (НИУ), Челябинск, e-mail: tatyana_barbasova@mail.ru

В работе рассмотрены вопросы автоматизации управления технологических объектов на основе технологии беспроводной передачи данных на базе стенда «Сенсорные сети ZigBee в системах автоматического управления», произведённого фирмой УчТехПрофи, г. Челябинск.

Данный стенд представляет собой макет участка железной дороги, на котором организуется движение поездов. Задача автоматизации данного объекта заключается в управлении режимами движения поездов и оптимизации режимов работы непосредственно задействованного в этом движении оборудования. Для решения данной задачи необходимо создать систему автоматического управления, которая могла бы собирать информацию о движении и состоянии на участке дороги и соответствующим образом контролировать узлы железной дороги.

Управление и передача информации в данном макете реализуются на основе технологии Zigbee – беспроводных сетей передачи данных [1]. В комплект стенда входят: макет путей, макет поезда, а также а также некоторые элементы сопутствующего оборудования железной дороги.

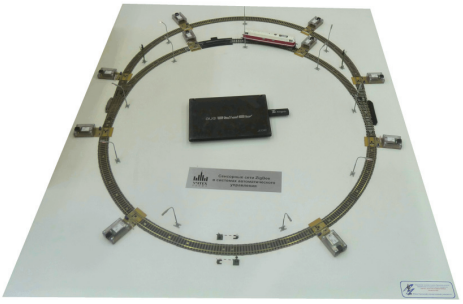

Рис. 1

В состав макета стенда также входят: система управления двигателем, которая позволяет запускать локомотив в разных направлениях и с различной скоростью. Управляется двумя линиями ввода-вывода (для задания режима работы двигателя) и одним ШИМ-выходом (для управления интенсивностью работы); датчик освещенности, смонтированный на верхней части локомотива. Выдает информацию о текущем уровне освещенности на вход АЦП модуля ETRX-2; система освещения (фара), установленная в передней части локомотива. Фара включается с помощью линии ввода-вывода модуля ETRX-2.

Движение макетов поездов осуществляется по путям, проложенным в виде двух сливающихся колец – внешнему и внутреннему, переход между которыми регулируется стрелочными переводами, а для световой индикации возможности взреза стрелки используется двухцветные семафоры. Железнодорожный переезд имитируется с помощью шлагбаумов. Также предусмотрено освещение дороги – вдоль путей расположены фонари с возможностью плавного регулирования яркости свечения. Для управления перечисленными устройствами и сбора информации о движении поезда в стенде используются рельсовые модули, которые равномерно расположены по ходу железнодорожного полотна. Рельсовый модуль (схематично показан на рисунке 2) состоит из: приемопередатчика ZigBee Telegesis ETRX-2 (4); датчика температуры, подключенный ко входу (3); двух датчиков Холла (1 и 2), установленных по краям железнодорожного полотна и позволяющих обнаруживать проезжающие над ними поезда (на поездах прикреплены магниты), зеленого светодиода (6), горящего при наличии питания модуля, красного светодиода (7), подключенного к порту ввода/вывода модуля, интерфейсного разъема (5), при помощи которого через специальный переходник можно подключать модуль к компьютеру. Интерфейс основан на переходнике USB->UART и имеет гальваническую развязку.

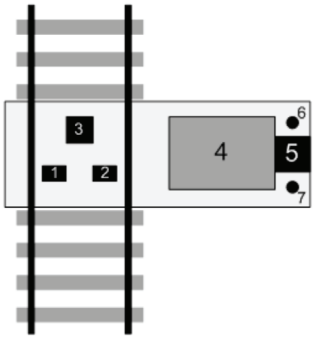

Рис. 2. Схематичное изображение рельсового модуля