3. Fomin B. F., Kachanova T. L. Physics of Systems is a postcybernetic paradigm of systemology // Intern. Sympos. Science 2.0 and Expansion of Science «S2ES» / the context of The 14th World-Multi-Conference «WMSCI 2010». June 29th–July 2nd. Orlando, Florida, USA, 2010. Orlando, 2010. P. 41–48.

4. Фреге Г. Смысл и денотат // Семиотика и информатика. 1977. Вып. 8. С. 181–210.

5. Fomin B. F., Kachanova T. L. // J. of Systemics, Cybernetics and Informatics. 2013. Vol. 11, № 2. P. 73–82.

6. Качанова Т. Л., Фомин Б. Ф. Основания системологии феноменального. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 1999.

7. Качанова Т. Л., Фомин Б. Ф. Метатехнология системных реконструкций. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2002.

8. Качанова Т. Л., Фомин Б. Ф. Введение в язык систем. СПб.: Наука, 2009.

9. Качанова Т. Л., Фомин Б. Ф. Методы и технологии генерации системного знания. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2012.

T. L. Kachanova, B. F. Fomin
Saint Petersburg state electrotechnical university «LETI»

SYSTEM ONTOLOGY OF CLASSES

In the paper, the constructive approach to build a classification system under the paradigm of physics of open systems is proposed. The problem of how to obtain scientifically proven knowledge about classes' ontology is solved. The general method of natural classification in complex subject domains is developed.

Physics of open systems, natural classification, ontological knowledge, intension, extension, archetype, taxon

УДК 681.322

В. В. Цехановский, В. Д. Чертовской
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Особенности компьютерной реализации задач статического линейного программирования

Сформулированы требования, предъявляемые к программным продуктам при компьютерной реализации задач статического линейного программирования большой размерности. Показано, что универсальный программный продукт отсутствует. Предложено использовать интеграцию СУБД InterBase и среды Delphi или C++ Builder. Приведена программная реализация в среде Delphi. Отмечена предпочтительность использования среды C++ Builder, реализация в которой аналогична приведенной в данной работе.

Реализация, задача, статическое линейное программирование, интеграция, программный продукт

До недавних пор основными задачами в АСУП были задачи «прямого счета». Именно на их решение были рассчитаны стандартные программные продукты, и прежде всего СУБД. С переходом России к рыночным отношениям и повышением динамичности и качества хозяйственных процессов стало перспективным использование в корабле-, приборо- и машиностроении адаптивных автоматизированных систем управления производством [1], [2]. В связи с этим появилась необходимость компьютерного решения высокоразмерных оптимизационных задач, в качестве которых все шире используются задачи статического линейного программирования (СЛП). Особенностям компьютерного решения таких задач посвящена настоящая работа, служащая продолжением и развитием публикаций [2], [3].

Постановка задачи. Адаптивная автоматизированная система управления производством, как показано в работе [1], имеет трехуровневую структуру. Для иллюстрации сути однородного метода математического описания, базирующегося

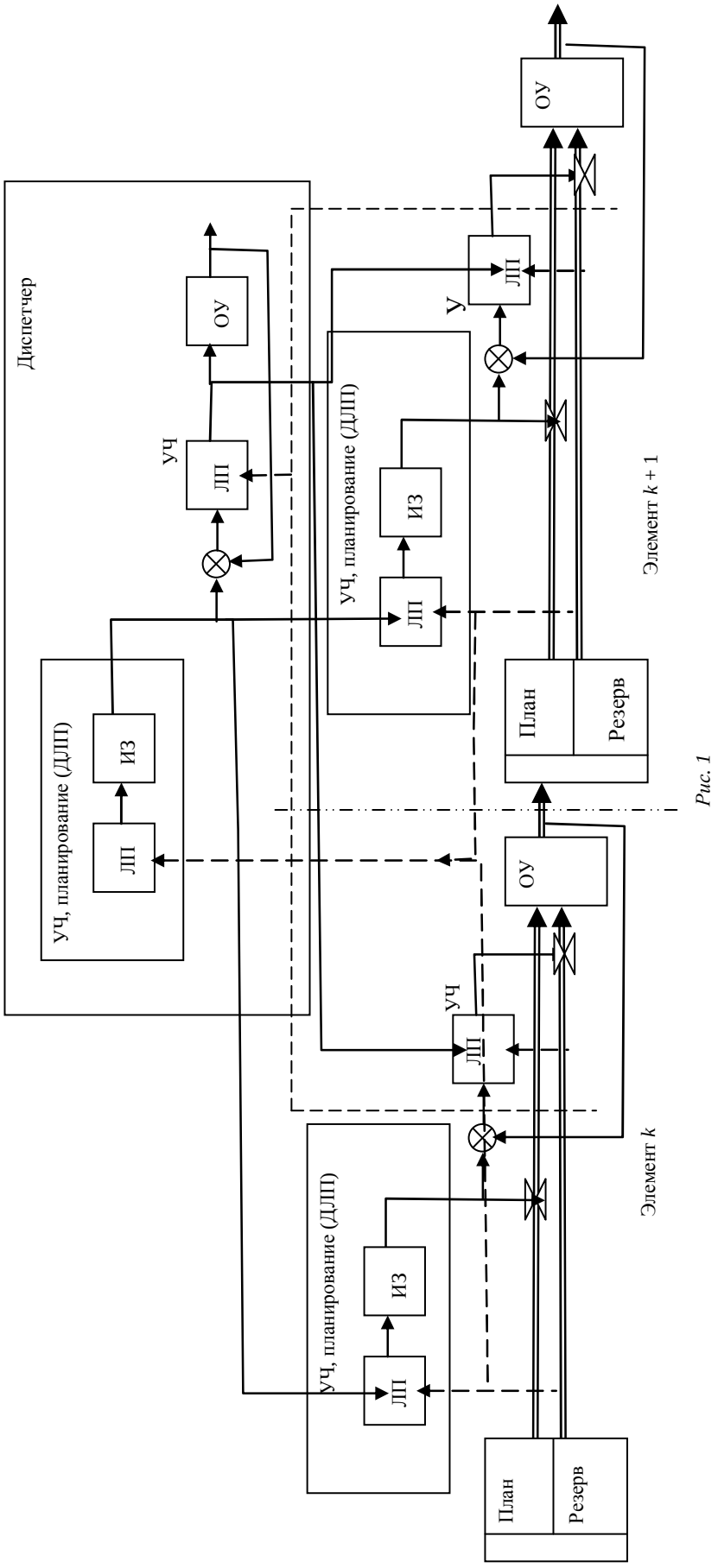
Диспетчер
УЧ, планирование (ДЛП)
ЛП
ИЗ
уч
ЛП
ОУ
Элемент k
План
Резерв
у
ЛП
ОУ
Элемент k + 1

Рис. 1

на задаче СЛП, достаточно показать два уровня (рис. 1 – схема управления «диспетчер–цехи»: ЛП – (статическое) линейное программирование; ДЛП – динамическое линейное программирование; УЧ – управляющая часть; ОУ – объект управления; ИЗ – инерционное звено).

Из рис. 1 видно, что основой описания как процесса планирования, так и процесса управления является задача СЛП.

Математически она может быть представлена следующим образом:

$$\left.\begin{array}{ll} \mathbf{FP} \to \max & (1) \\ \mathbf{b}_* \le \mathbf{DP} \le \mathbf{b}^* & (2) \\ \mathbf{R}_* \le \mathbf{P} \le \mathbf{R}^* & (3) \end{array}\right\}.$$

Здесь векторы прибыли **F**, плана **P**, верхней и нижней границ спроса $\mathbf{R}_*$, $\mathbf{R}^*$ имеют размерность *J*, а векторы наличного количества ресурсов $\mathbf{b}_*$, $\mathbf{b}^*$ – размерность *I*. Выражение (1) называется целевой функцией, а выражения (2), (3) – ограничениями.

Иллюстрируем последующие выкладки на таком числовом примере:

$$f = 60p1 + 70p2 + 120p3 + 130p4 \to \max,$$
$$p1 + p2 + p3 + p4 \le 16;$$
$$6p1 + 5p2 + 4p3 + 3p4 \le 110;$$
$$4p1 + 6p2 + 10p3 + 13p4 \le 100;$$
$$p1 \ge 0,\ p2 \ge 0,\ p3 \ge 0,\ p4 \ge 0.$$

Составляющими задачи СЛП являются данные и оптимизационный алгоритм линейного программирования. В компьютерном решении подобных задач можно выделить два случая:

А. Размерность задачи невелика.

Б. Размерность задачи значительна (например, 1000×3000).

Случай А характерен для проверки работоспособности предлагаемого компьютерного решения. В силу небольшой размерности данные могут быть размещены в самой программе.

При использовании пакета MatLab представленная числовая задача может быть решена с помощью следующей программы:

```
function x= lp13(f);
A=[1 1 1 1; 6 5 4 3; 4 6 10 13];
b=[16; 110; 100];
f=[–60; –70; –120; –130];
lb=zeros(4,1);
[x]=linprog(f,A,b,[],[],lb);
```

Решение этой задачи – **P** = (10, 0, 6, 0).

Нетрудно видеть, что такая технология решения при большой размерности (случай Б) доставит пользователю массу неприятностей. Случай Б характерен для реальной системы, в которой данные упорядочиваются не в виде отдельных несвязанных таблиц, а в форме баз данных. В реализации задачи СЛП возникает взаимодействие базы данных и алгоритмической части.

К такой реализации предъявляются следующие основные требования.

1. Высокая размерность таблиц с исходными данными.

2. Количество выходных таблиц значительно.

3. Распространенность (стандартность) используемого программного пакета алгоритмической части.

4. Возможность оптимизации процессов планирования и управления.

5. Предпочтительно использование только одного вида программного продукта.

6. Удобный ввод (таблицы) и вывод (таблицы) данных в обычной табличной форме с простым интерфейсом доступа.

7. Режим клиент–сервер при наличии многих пользователей.

Наблюдается противоречивость требований 1 и 2, с одной стороны, и требований 3 и 4, с другой стороны.

Действительно, первые два требования, и даже требования 6, 7 легко реализуются с помощью СУБД InterBase в средах Delphi или C++ Builder. Однако бедность языка SQL не дает возможности строить достойную алгоритмическую часть.

В свою очередь, пакет MatLab обладает проверенной программой linprog и, как отмечено, слабыми возможностями хранения данных.

Сказанное ставит под сомнение удовлетворение требования 5, поскольку требуется интеграция программных продуктов.

Решение задачи. Одним из возможных методов решения задач служит интеграция InterBase и MatLab.

Здесь возможны два варианта:

1) передача данных из InterBase в пакет MatLab, в котором далее решается задача, и возврат результатов в InterBase;

2) трансформация программы linprog в среду Delphi или C++ Builder.

Вариант 1. Для этого варианта между основными продуктами InterBase и MatLab следует разместить еще один программный продукт –

Open DataBase Connection (ODBC), вызываемый из Интернета [4]. В свою очередь, в пакете MatLab нужно дополнительно задействовать опции DataBase ToolBox Visual Query Builder. Возможны ручная и программная процедуры связи основных продуктов.

Из-за множества программ ручной режим получается достаточно сложным. При этом таблицы в MatLab передаются по одной, что утомительно для пользователя при значительном количестве таблиц БД. Результат передачи показан на рис. 2.

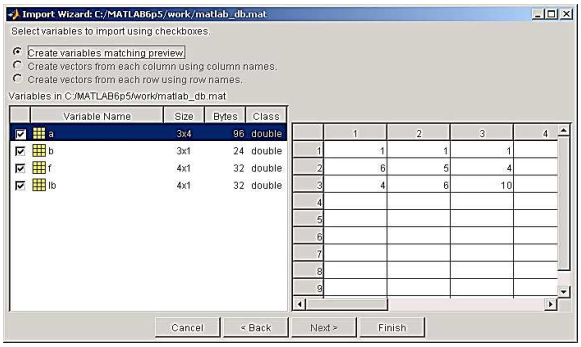

Рис. 2

Не менее сложна и процедура передачи результата.

Программная процедура передачи упрощается незначительно. Заметим при этом, что при изменении данных в БД передачи данных и результата следует повторить.

Рассмотренные особенности не позволяют рекомендовать данный вариант для широкого использования.

Вариант 2. В нем (рис. 3) используется программа lp121.m пакета MatLab:

```
function x= lp121(f,A,b,[],[],lb);
[x]=linprog(f,A,b,[],[],lb);
```

Она преобразуется во внешнюю процедуру lpa.dll [5]–[7] и передается в Delphi (рис. 3).

Приведем фрагмент программы Delphi для данного варианта:

```
unit Un_DLL_MX;
interface
type
  pDouble = ^Double;
  mxComplexty = (mxREAL, mxCOMPLEX);
  pmxArray=^mxArray;
  mxArray=record
  end;
                // Из библиотеки libmx.dll
 function mxCreateDoubleMatrix(m,n:Integer;
    mxData:mxComplexty):pmxArray; cdecl; external 'libmx.dll';
    procedure mxSetName(arr_ptr:pmxArray; const name:String);
      cdecl; external 'libmx.dll';
    function mxGetPr(arr_ptr:pmxArray):pDouble;cdecl; external 'libmx.dll';
    function mxGetPi(arr_ptr:pmxArray):pDouble;cdecl; external 'libmx.dll';
    function mxGetM(arr_ptr:pmxArray):Integer; cdecl; external 'libmx.dll';
    function mxGetN(arr_ptr:pmxArray):Integer; cdecl; external 'libmx.dll';
    procedure mxDestroyArray(arr_ptr:pmxArray); cdecl; external 'libmx.dll';
    procedure mxFree(var ram:THandle); cdecl; external 'libmx.dll';
    function mxTranspose(x:pmxArray):pmxArray; cdecl; external 'libmx.dll';
                // Из библиотеки mclmcr.dll
    function mclInitializeApplication (A:THandle; B:Integer):Boolean;
```

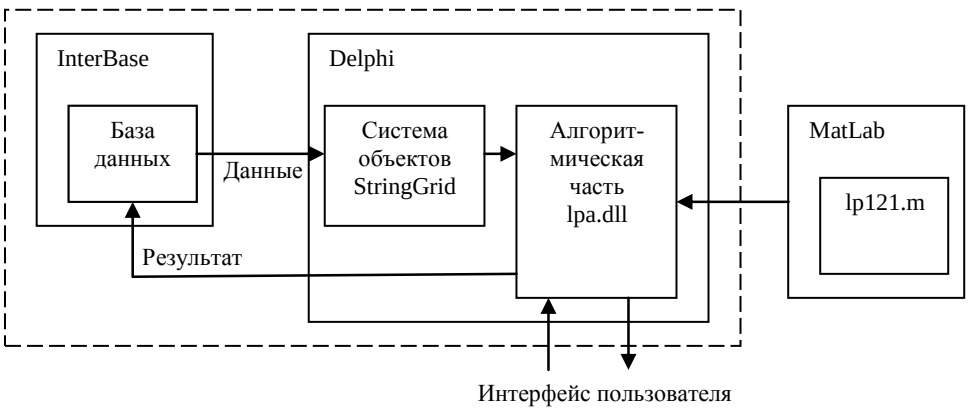

Рис. 3

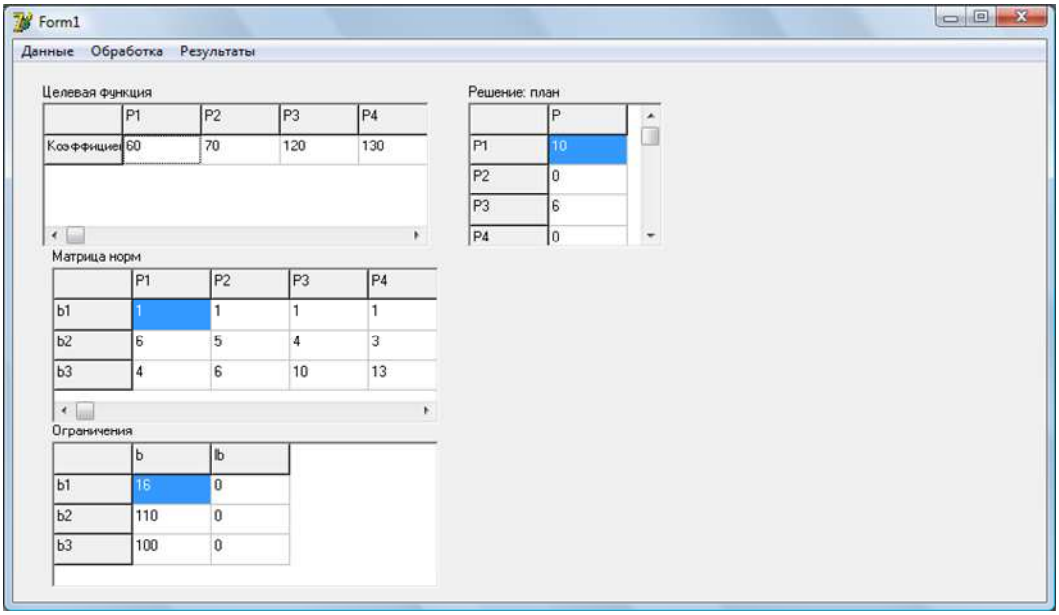

Рис. 4

```
cdecl; external 'mclmcr.dll';
function mclTerminateApplication:Boolean; cdecl; external 'mclmcr.dll';
//Из библиотеки lpa_dll
function _lpa_dllInitialize:Boolean; cdecl; external 'lpa_dll.dll';{инициализация}
function _lpa_dllTerminate :Boolean; cdecl; external 'lpa_dll.dll';
procedure _mlflp121(i:Integer; var pout: pmxArray; f,A,b,lb: pmxArray);
cdecl; external 'lpa_dll.dll'; {i – число выходных параметров; var – выходной параметр; f,A,b,lb – входные параметры}

implementation

end.
```

Интерфейс пользователя прикладной задачи показан на рис. 4.

Задача реализовывалась с применением среды Delphi. Особо отметим, что все сказанное справедливо и для среды C++ Builder, поскольку обе среды разрабатывались одной фирмой.

Реализация задачи СЛП требует интеграции программных продуктов, чтобы удовлетворять сформулированным требованиям. При интеграции InterBase и MatLab в большей мере учитывает требования трансформация программы MatLab в среду Delphi или C++ Builder. Реализация проведена для среды Delphi. Однако предпочтительнее применение среды C++ Builder, поскольку язык C++ более распространен и обладает большими возможностями по сравнению с языком Object Pascal.

СПИСОК ЛИТЕРАТУРЫ

1. Чертовской В. Д. Интеллектуализация автоматизированного управления производством. СПб.: Изд-во С.-Петерб. ун-та, 2007. 164 с.

2. Советов Б. Я., Цехановский В. В., Чертовской В. Д. Адаптивные автоматизированные системы управления производством. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2013. 186 с.

3. Чертовской В. Д. Системный анализ в проектировании адаптивных автоматизированных систем управления производством // Тр. XVIII Междунар. науч.-практ. конф. «Системный анализ в проектировании и управлении» / СПбГПУ. СПб., 2014. С. 41–47.

4. Советов Б. Я., Цехановский В. В., Чертовской В. Д. Теория адаптивного автоматизированного управления. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2009. 256 с.

5. Подкур М. Л., Подкур П. Н., Смоленцев Н. К. Программирование в среде Borland C++ Builder с математическими библиотеками MatLab C/C++. М.: ДМК Пресс, 2006. 496 с.

6. Смоленцев Н. К. MatLab и программировании на Visual C++, Borland JBuilder VBA. М.: ДМК Пресс, СПб.: Питер, 2009. 464 с.

7. Бей И. Взаимодействие разноязыковых программ. М.: Вильямс, 2005. 880 с.

V. V. Tsehanovsky, V. D. Chertovskoy
Saint Petersburg state electrotechnical university «LETI»

SPECIAL FEATURES OF COMPUTER REALIZATION
FOR STATIC LINEAR PROGRAMMING TASKS

Requirements to program products for computer realization of static linear programming tasks are formed. It is shown universal product is absent. It is suggested to use database InterBase in environment Delphi or C++ Builder. The possibility of such realization is confirmed by numerical example in environment Delphi. It is marked that application in environment C++ Builder is analogous but is preferred.

Realization, task, static linear programming, integration, program product

УДК 004.896, 004.414.2

С. В. Лебедев, М. Г. Пантелеев
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)

Концептуальная модель подсистемы оценки обстановки интеллектуального агента реального времени

Анализируется роль оценки обстановки в рамках интеллектуальных агентов реального времени. Определяется множество требований и общих принципов построения соответствующих подсистем. На основе анализа подходов к оценке ситуации, а также с учетом предложенных требований и принципов формулируется концептуальная модель подсистемы. Предложен обобщенный процесс проектирования подсистемы с использованием сформулированной модели.

Интеллектуальный агент реального времени, оценка обстановки, концептуальная модель

Концепция интеллектуальных агентов (ИА) в последние годы находит все более широкое практическое применение, обусловливая актуальность разработки моделей, методов и средств проектирования и разработки программного обеспечения данного класса систем. Под ИА в рамках настоящей статьи будем понимать программную или программно-аппаратную систему, способную осуществлять автономное целенаправленное поведение в некоторой предзаданной среде (как, например, физический мир).

Внешние интерфейсы ИА включают множество сенсоров и исполнительных механизмов. По внутренней организации ИА могут подразделяться на реактивные и на основе модели [1]. К последним можно отнести так называемые делиберативные агенты [2].

Отличительной особенностью делиберативных архитектур служит использование явной внутренней символьной модели мира (ММ), конструируемой на основе восприятий и используемой для формирования собственных действий. Важно отметить, что ММ семантически богаче непосредственного восприятия среды. Именно такого рода описания требуются при решении сложных задач [3]. Недостатком таких систем является, в общем случае, сложность построения и большая временная емкость по сравнению с реактивными архитектурами.

На рис. 1 представлена одна из возможных архитектур делиберативного ИА, в которой за построение ММ отвечает подсистема оценки обстановки (ПОО). ПОО преобразует низкоуровневые данные, получаемые подсистемой восприятия, в высокоуровневое представление – ММ, используемая подсистемой планирования для формирования действий.

Настоящая статья посвящена основаниям процесса проектирования и разработки ПОО. Для этого определяется совокупность функциональных