**Инструкция по оформлению статей на конференцию ИТиС-2017**

1. Статья объемом от 3 до 10 страниц формата А4 в редакторе Microsoft Word.
2. Размеры полей: левое – 2 см, правое – 2 см, верхнее – 2 см, нижнее – 2см. Шрифт TimesNewRoman, размер – 14, интервал – одинарный (для всей статьи).
3. Порядок размещения материала:
   * сведения об авторах (не более трех), которые состоят из инициалов и фамилий в именительном падеже (через запятую, заглавными буквами, по центру, без точки);
   * название статьи (заглавными буквами, без абзаца, по центру, без заключительной точки);
   * аннотация на русском языке (не более 500 знаков);
   * текст статьи (абзац 1,25, ссылки на литературу в квадратных скобках);
   * иллюстрации в статье могут быть двух типов – «плавающие» – прижатые к верхнему или нижнему краю области текста, и «встроенные» – размещенные между абзацами внутри колонок текста;
   * при использовании графических средств иллюстрацию выполнять как единый объект «рисунок».
   * сложные формулы набирать с помощью редактора MathType. Настройки редактора формул: Full – 12pt; Subscript/Superscript – 7pt; Sub-subscript – 5pt; Symbol – 18pt; Sub-symbol – 12pt;
   * литература указывается в конце основного текста с заголовком «Библиографический список», без заключительной точки, нумерация источников цифрами с точкой;
   * контактная информация об авторах статьи: название организации/института/ университета (без абзаца, по центру, без точки), должность, ученое звание, электронная почта;
   * фамилии и инициалы авторов, название статьи и аннотация на английском языке.

ЮУрГУ

доктор физико-математических наук, профессор,

**А. В. Панюков, Е. А. Загирова**

**ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МАКСИМИННОЙ СТРАТЕГИИ УПРАВЛЕНИЯ ИНВЕСТИЦИОННОЙ ДЕЯТЕЛЬНОСТЬЮ ПРЕДПРИЯТИЯ**

Предложена техника программной реализации алгоритма для задачи выбора оптимального пакета инвестиционных проектов развития предприятия. Критерием формирования оптимальной инвестиционной программы предприятия, является чистый дисконтированный доход инвестиционной программы в целом.

## Постановка задачи

Инвестиционная программа представляет собой совокупность реализу­емых инвестиционных проектов предприятия, состоящих из перечня объек­тов инвестиций, их характеристик и объемов финансирования [1, 2].

В качестве основного критерия для формирования оптимальной инвес­тиционной программы предприятия, выбираем чистый дисконтированный доход (ЧДД) инвестиционной программы в целом, т.е. сумму чистых дискон­тированных доходов проектов, которые были включены в инвестиционную программу.

Введем необходимые обозначения [1, 2]:

*  – множество из н инвестиционных проектов, которые могут быть включены в состав инвестиционной программы;
*  – множество продолжительностей реализации инвестиционных проектов (в расчетных периодах);
* – горизонт планирования (число расчетных периодов);
*  – объемы финансирования инвестиционной программы предприятия по расчетным периодам.

Каждый из инвестиционных проектов  охарактеризован двумя показателями:

* – величина ЧДД, приведенного к моменту начала реализации проекта, если он был начат в период .
*  – потребность в финансировании инвестиционного проекта  в расчетный период  от начала реализации инвестиционной программы, при условии, что он будет начат в период .

Показатели доходов и расходов являются прогнозируемыми величи­нами и зависят от ряда факторов. Поэтому целесообразно считать  и случайными величинами, а для каждого проекта и всех расчетных периодов получены интервальные оценки ЧДД , потребностей  и финансовых ресурсов предприятия .

Для построения модели введем булевы переменные



Поскольку реализация инвестиционного проекта  может начаться не позже чем в период , то должно выполняться следующее условие:



Условие реализуемости инвестиционной программы имеет вид



ЧДД для всего портфеля проектов равен



Максиминная (осторожная) стратегия направлена на получение макси­маль­ного гарантированного результата. Применяя нижние оценки ЧДД и объемов финансирования по расчетным периодам и верхние оценки потреб­ностей финансирования, гарантируется оптимальность значения следующей задачи [1, 2]:



 (1)

 (2)

 (3)

## Алгоритм решения

Решение задачи поиска оптимального пакета проектов по максиминной стратегии проводится методом динамического программирования через полный перебор всех возможных комбинаций активных проектов (состояний системы) в каждый момент времени. Комбинация проектов с максимальным ЧДД и является оптимальной. Алгоритм решения задачи состоит из следующих основных шагов.

* Шаг 0. Инициализация. Составить список всех состояний на текущий момент,. Это состояния, в которых проекты либо не начались, либо активировались в момент. Исключить из списка все состояния, которые не удовлетворяют условиям, т.е. комбинация требует инвестиций больше, чем предприятие может выделить (2) или какой-либо из проектов комбинации будет активен и за пределами горизонта событий.
* Шаг 1. Динамика системы во времени. Начать цикл по времени. На каждой итерации переменная времени увеличивается на 1. Если данная переменная сравняется по значению с горизонтом событий, то цикл прекращается и происходит переход к шагу 3, иначе к шагу 2.
* Шаг 2. Тело цикла. Составить новый список состояний. Просмотреть все состояния системы из списка состояний. Построить все возможные состояния системы исходя из предыдущего состояния, т.е. добавить в комбинацию активных проектов другие возможные комбинации проектов. Если не удалось активировать еще проекты, оставить текущее состояние системы. Удалить старый список состояний.
* Шаг 3. Максимальной ЧДД. Найти максимальное по ЧДД состояние системы. Это и будет решением задачи.

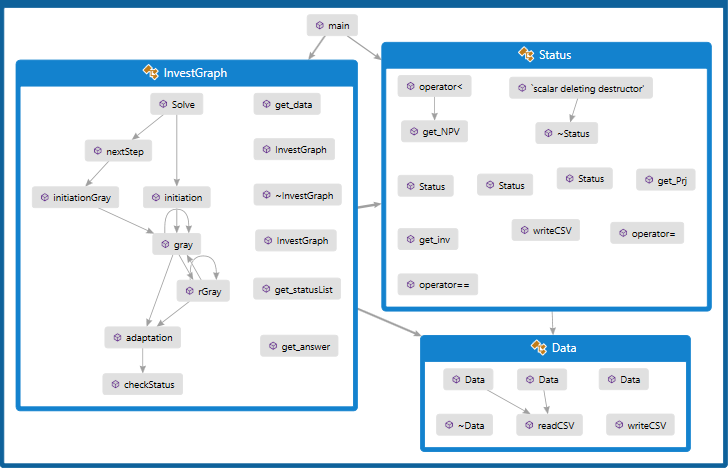
Т.к. в данном алгоритме перебираются все возможные варианты состояния системы, то его результативность и конечность очевидны и в дальнейшем не обсуждаются.

## Описание реализации классов

Вся задача и ее решение представляется как динамическая система, изменяющаяся с течением времени. Для реализации использовался язык C++. Было получено несколько классов, которые представлены на диаграмме зависимостей вместе со своими методами (рис. 1).

Описание класса InvestGraph, на базе которого и строится все решение, представлено на рис. 1 и 2. Далее будут описаны переменные данного класса, расположенные в закрытой части.

Переменная data хранит все входные данные задачи, в answer записывается состояние системы с максимальным ЧДД, statusList хранит в себе список всех состояний системы на текущей итерации цикла, в statusListNextStep записываются все новые состояния, полученные на данной итерации.



*Рис. 1 – Диаграмма зависимостей*

|  |
| --- |
| class InvestGraph  {  Data data;  Status answer;  list <Status> statusList;  list <Status> statusListNextStep;  void gray(int time, vector <int>& g, const vector <int>& baseVec, int k = 0);  void rGray(int time, vector <int>& g, const vector <int>& baseVec, int k = 0);  bool checkStatus (Status& status);  void adaptation (const vector <int>& baseVec, const vector <int>& g);  void initiationGray(const int time, const int k, const vector <int>& baseVec);  void initiation(); //инициализация списка  void nextStep( const int time);  public:  InvestGraph(void){};  InvestGraph(string filename):data(filename){};  ~InvestGraph(void){};  Data get\_data() {return data;}  Status get\_answer() {return answer;}  list <Status> get\_statusList() {return statusList;}  void Solve()  }; |

*Рис. 2 – Описание класса* InvestGraph

Задача решается в две строчки:

InvestGraph system("input.csv");

system.Solve();

Т.е. первоначально создается объект класса, где на вход подается файл формата \*.csv, из которого читаются данные к задаче (по умолчанию файл «input.csv») и записываются в data. Затем вызывается главный метод Solve()(рис. 6),который и запускает алгоритм решения.

Описание класса Data представлено на рис. 1 и 3. Все переменный подробно описаны в комментариях.

|  |
| --- |
| class Data  {  public:  Data(void){ readCSV("input.csv");}  Data(string filename){ readCSV(filename);}  ~Data(void){};  int nPrj;//количество проектов  int m; // горизонт планирования  vector <int> lenght; //длительность проектов, вектор размера nPrj  vector <double> resource; //финансовые ресурсы компании, вектор размера m  vector <vector <double> > NPV; //net present value чистый дисконтированный доход  vector <vector <double> > inv; //ежегодные инвестиции(вложения) в проект  void readCSV ( string filename);  void writeCSV ();  }; |

*Рис. 3 – Описание класса* Data

Стоит отметить, что в данной задаче не используются интервальность величин ЧДД , инвестиций  и ресурсов предприятия , а только нижние (, ) и верхние границы (), потому нет нужды хранить интервалы. Данные в документе записаны в том же порядке, что и в описании класса. Далее, на примере, будет показано, как именно. Для описания состояния системы используется класс Status (рис. 1, 4). В векторе Prj хранятся периоды начала реализации  проектов  или число , как идентификатор того, что проект не начался.

В переменной NPV хранится суммарный ЧДД по всем активным проектам данного набора. В векторе inv хранятся требуемые суммарные инвестиции в проекты в каждый период времени [0, *m*].

|  |
| --- |
| class Status  {  vector <int> Prj;//активные, не активные проекты  double NPV;//чистый дисконтированный доход пакета проектов  vector <double> inv;//затраты на проеткы в момент i, inv[i]  public:  Status(void){};  Status(const vector <int>& \_Prj, const Data& data);  ~Status(void){};  vector <int> get\_Prj(){ return Prj;}  vector <double> get\_inv(){ return inv;}  double get\_NPV(){ return NPV;}  vector <int> get\_Prj() const { return Prj;}  vector <double> get\_inv() const { return inv;}  double get\_NPV() const { return NPV;}    void writeCSV( string filename = "output\_status.csv");  bool const operator <(Status const &right) const;  bool operator ==(Status const &right) const;  void writeStatus();  }; |

*Рис. 4 – Описание класса* Status

|  |
| --- |
| Status(const vector <int>& \_Prj, const Data& data){  Prj = \_Prj;  NPV = 0;  //ресурсы, требуемые данным составом проектов в каждый момент времени  for (int i = 0; i < data.m; i++) inv.push\_back(0);  for (int i = 0; i < \_Prj.size(); i++){  //если проект был начат  if (Prj[i] != -1){  int j = \_Prj[i] + data.lenght[i];  //во время выполнения проекта требуются ресурсы  for (int k = \_Prj[i]; k < j; k++){  inv[k] += data.inv[i][k - \_Prj[i]];  }  NPV += data.NPV[i][\_Prj[i]];//находим общий ЧДД  }  }  } |

*Рис.5 – Конструктор класса* Status

|  |
| --- |
| void InvestGraph:: Solve(){  initiation();  for (int i = 0; i < data.m; i++ ){  cout << "NEXT STEP - " << i<< '\n';  nextStep(i+1);  }  double npv = statusList.begin()->get\_NPV();  answer = (\*(statusList.begin()));  for (list <Status>::iterator i = statusList.begin(); i != statusList.end(); i++)  if ((i->get\_NPV()) > npv){  npv = (\*i).get\_NPV();  answer = (\*i);  }  }; |

*Рис. 6 – Метод* Solve() *класса* InvestGraph

|  |
| --- |
| void InvestGraph:: initiation(){  vector <int> \_prj(data.nPrj, -1);  gray(0, \_prj, \_prj);  statusList = statusListNextStep;  statusListNextStep.clear();  } |

*Рис. 7 – Метод* initiation() *класса* InvestGraph

|  |
| --- |
| void InvestGraph:: nextStep(const int time){  //Для каждого элемента списка состояний  for (list <Status>::iterator i = statusList.begin(); i != statusList.end(); i++){  vector <int> baseVec = (\*i).get\_Prj();  //количество проектов, которые еще не активировались  int k = 0; for (int i = 0; i < baseVec.size(); i++) if (baseVec[i] == -1) k++;  //просмотреть варианты на основе текущего (текущий автоматически попадает в варианты)  //проверка состояний проводится в функциях adaptation и checkStatus  initiationGray(time, k, baseVec);  }  statusList = statusListNextStep;  statusListNextStep.clear();  } |

*Рис.8 – Метод* NextStep() *класса* InvestGraph

Следует обратить внимание на конструктор класса, где на основе вектора проектов и данных задачи определяется состояние системы (рис. 5).

После введения данных, решается посредством запуска метода Solve() (рис. 6). Он вызывает метод initiation() (рис. 7), который реализует Шаг 0 алгоритма. Также Solve() осуществляет временной цикл Шага 1, где телом цикла является метод NextStep() (рис.8). По завершении цикла получаем конечный список состояний, в котором ищется максимальное по ЧДД методом перебора и записывается в переменную Answer.

В методах initiation() и NextStep() описывается подготовка данных к вызову методов gray() и rGray() (рис. 9) бинарного перебора всех вариантов состояний системы (бинарный, т.к. проекты имеют только два варианта состояния: начаться в текущий период времени или не начаться). Перед вызовом метода gray() исходя из вектора проектов рассматриваемого состояния системы (базового вектора) создается вектор не активных проектов данного состояния (каждый элемент равен ). Далее методы gray() и rGray(), в основе которых лежит рекурсивный метод Грея [3, 4], осуществляют бинарный перебор элементов нового вектора. Параметры методов следующие: период времени, ссылка на перебираемый вектор, базовый вектор и идентификатор  (при первом вызове ).

|  |
| --- |
| void InvestGraph::gray (int time, vector <int>& g, const vector <int>& baseVec, int k){  if (k == g.size() ){//получили новый вектор по Грею  adaptation(baseVec, g);  }  else{  g[k] = -1;  gray(time, g, baseVec, k + 1);  g[k] = time;  rGray(time, g, baseVec, k + 1);  }  } |
| void InvestGraph::rGray(int time, vector <int>& g, const vector <int>& baseVec, int k){  if (k == g.size() ){//получили новый вектор по Грею  adaptation(baseVec, g);  }  else{  g[k] = time;  gray(time, g, baseVec, k + 1);  g[k] = -1;  rGray(time, g, baseVec, k + 1);  }  } |

*Рис. 9 – Методы* gray() и rGray() *класса* InvestGraph

В методах gray() и rGray() при получении новой комбинации вызывается метод adaptation()(рис. 10), который проводит окончательную обработку вектора проектов, формирует новое состояние, проверяет его на потребность в инвестициях (метод checkStatus()(рис. 10)) и добавляет в список новых состояний при удовлетворении состояния всем условиям. Параметрами метода adaptation() являются вектор проектов, сгенерированный по алгоритму Грея, и базовый вектор. Создается вектор идентичный базовому и значения, соответствующие не активным проектам, заменяются значениями сгенерированного вектора, тем самым создавая новую комбинацию проектов состояния системы. При этом проводится проверка на возможность активации каждого проекта, т.е. возможна ситуация с выходом окончания проекта за пределы горизонта планирования, что является недопустимым. В таком случае новое состояние не генерируется и метод завершается.

После генерации всех новых состояний системы метод NextStep() записывает их в список текущих состояний statusList и очищает список новых состояний statusListNextStep.

Таким образом, в методе Solve() просчитываются все возможные состояния системы. Далее среди них находится состояние с максимальным чистым дисконтированным доходом по комбинации проектов и записывается в answer. Осталось вывести ответ в файл командой:

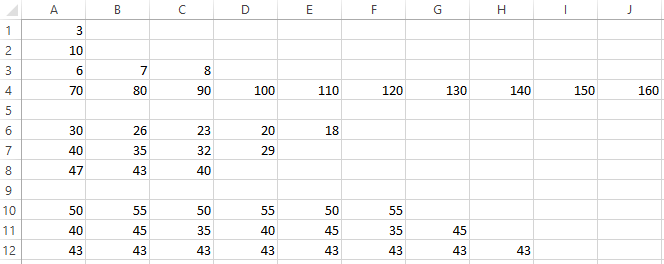
system.get\_answer().writeCSV("output.csv");

|  |
| --- |
| bool InvestGraph::checkStatus (Status& status){  bool result = true;  vector <double> Res = status.get\_inv();  for (int i = 0; i < Res.size(); i++){  if ( Res[i] > data.resource[i]) {  result = false;  break;  }  }  return result;  } |
| void InvestGraph::adaptation (const vector <int>& baseVec, const vector <int>& g){  vector <int> vec;  bool endPrj = true;  //составляем новый вектор состояния проектов  for (int i = 0, j = 0; i < baseVec.size(); i++){  vec.push\_back(baseVec[i]);  if (baseVec[i] == -1){  //если проект умещается в сроки  if (data.m - g[j] > data.lenght[i]){  vec[i] = g[j];  j++;  }  else{  endPrj = false;//проект вышел за горизонт планирования  break;  }  }  }  if (endPrj){  Status status(vec, data);  //если инвестиции в проект меньше ресурсов, выделяемых компанией  if (checkStatus(status)){  //добавляем в список состояний  statusListNextStep.push\_back(status);  }  }  } |

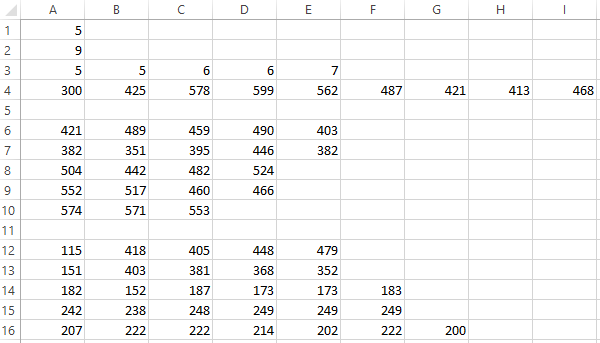
*Рис. 10 – Методы* checkStatus() и adaptation() *класса* InvestGraph

**Примеры решения**

Рассмотрим применение данного кода к решению задачи. На рис. 11-13 представлены условия к задачам, то, какой вид имеют данные. Данные разделены пустой строкой на три блока. В первом блоке четыре строки. В первой строке записано количество проектов, во второй – горизонт планирования (начиная с нулевого периода), в третьей – продолжительность каждого проекта, в четвертой – количество имеющихся у предприятия ресурсов в каждый период времени. Во втором блоке приводятся оценки ЧДД проектов в каждый период времени возможного начала реализации . В каждой строке отображена информация по одному проекту. В третьем блоке приводятся затраты на каждый проект в каждый период времени, начиная с момента активации проекта по тому же принципу, что и предыдущий блок.

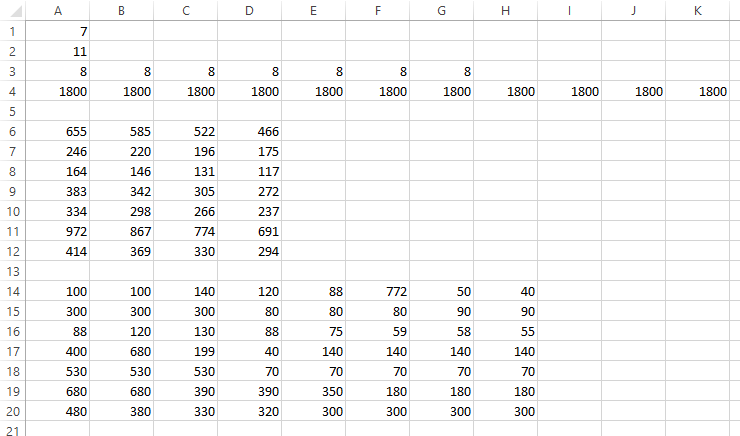


*Рис. 11 – Условие задачи с тремя проектами*

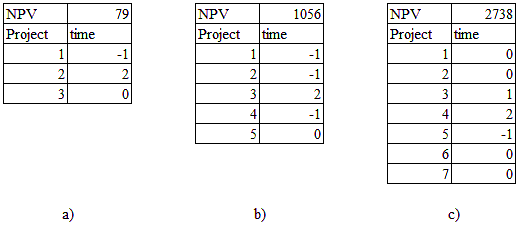


*Рис. 12 – Условие задачи с пятью проектами*

Результаты решения представлены на рис. 14. При формировании инвестиционной программы по максиминной стратегии гарантированный ЧДД программы для задачи о трех проектах равен 79, о пяти проектах равен 1056, о семи проектах равен 2738, как это показано в первых строках таблиц. Далее, в первом столбце показан номер проекта, во втором – период, в который следует начать проект.  означает то, что проект не должен быть выполнен.



*Рис. 13 – Условие задачи с семью проектами*

**

*Рис. 14 – Решение задачи*

*a) с тремя проектами, b) с пятью проектами, c) с семью проектами*

**Заключение**

Предложенная реализация позволяет найти оптимальный пакет инвестиционных проектов при условии гарантированности результата. Данная реализация не позволяет решать задачи большой размерности: если задачи на пять проектов и горизонт планирования около десяти периодов решаются достаточно быстро (меньше секунды), то решение задачи на семь проектов занимает намного больше времени (несколько минут). Направление дальнейшего исследования – построение программного обеспечения, позволяющего решать задачи большой размерности, а также реализовать методы поиска оптимального портфеля проектов с учетом рисков.

**Библиографический список**

1. Panyukov A. V., Kozina E. N. Forming of the Competitive Investment Programs for Enterprises / A. V. Panyukov, E. N. Kozina // Workshop on Computer Modelling in Decision Making (CMDM 2016). CEUR Workshop Proceedings. Vol. 1726. P. 89-99. URL <http://ceur-ws.org/Vol-1726/paper-09.pdf>
2. Панюков А.В. Применение математического моделирования для выбора инвестиционной программы предприятия / А.В. Панюков, Е.Н. Козина // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. Том 5, № 4, 2016. С. 19-31. DOI: <http://dx.doi.org/10.14529/cmse160402>
3. Аникушин М. Коды Грея и задачи перебора // [Эл. ресурс]/ Режим доступа: <https://habrahabr.ru/post/200806/> , свободный. – Загл. с экрана.
4. Рейнгольд Э. Комбинаторные алгоритмы. Теория и практика / Э.Рейнгольд, Ю.Нивергельт, Н.Део // М.: Мир – 1980. C. 476

**Сведения об авторах**

**Панюков Анатолий Васильевич** – доктор физико-математических наук, профессор Южно-Уральского государственного университета, *a\_panyukov@mail.ru.*

**Загирова Екатерина Альбертовна** – студент магистратуры Южно-Уральского государственного университета, Челябинск, *zagirovaea@mail.ru.*

1. **V. Panyukov, E. A. Zagirova**

**SOFTWARE IMPLEMENTATION OF MAXIMIN STRATEGY FOR COMPANIES INVESTMENT ACTIVITIES MANAGEMENT**

The paper offer a technique of the algorithm software implementation for the problem the optimal set of investment projects company development choosing. The criterion of the optimal investment program of the enterprise, is the net present value of the investment program as a whole.