УДК 004.021

Иванов Ю. А.
Аспирант Донецкого национального технического университета

АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ ОПТИМИЗАЦИИ ДЛЯ ЦИКЛИЧЕСКИХ РАСПИСАНИЙ С ПРЕРЫВАНИЯМИ

Рассмотрена задача оптимизации циклической организации вычислительного процесса системы жесткого реального времени. Определены особенности целевой функции. Обоснован выбор целочисленного метода оптимизации задачи построения расписания. Предложен алгоритм решения адаптированным методом ветвей и границ.

Ключевые слова: целочисленная оптимизация, метод ветвей и границ, алгоритм.

ВВЕДЕНИЕ

Повышение эффективности решения различных производственных задач в значительной мере достигается внедрением средств их модельной поддержки. Для эффективного управления объектами необходимо решать задачи оптимизации характеристик процесса. Одними из основных методов являются методы математического программирования. В зависимости от характера целевого множества переменных рассматриваемой модели это могут быть методы линейного, нелинейного или дискретного программирования.

Теория и практика эффективных методов и алгоритмов решения задач математического программирования достаточно полно разработаны [1–4]. У вновь возникающих задач есть собственные особенности, определяющие специфику применяемых к ним методов. При этом прямое решение известным алгоритмом часто может быть затруднено или не является эффективным, для чего необходимо преобразовать или адаптировать саму задачу.

ПОСТРОЕНИЕ МОДЕЛИ ЗАДАЧИ ОПТИМИЗАЦИИ

Рассмотрим модель задачи определения характеристик циклического вычислительного процесса системы жесткого реального времени [5]:

$$\arg\min_{L\in N, L\le T_1} F(L);$$

$$F=\sum_{i=1}^{n}\left(\frac{\tau_i}{T_i'}-\frac{\tau_i}{T_i}\right)+\frac{np}{L};$$

$$F\ge 0; \qquad (1)$$

$$\sum_{i=1}^{n}\frac{\tau_i}{T_i'}\le 1;$$

$$T_i'=\left\rceil\frac{T_i}{L}\right\lceil L;$$

$$i=\overline{1,n},$$

где τ_i, T_i, n, p – параметры, характеризующие вычислительный процесс, L – переменная. Целевая функция является нелинейной. Элементами нелинейности являются операции выделения целой части для определения T_i' и выражение $\frac{1}{L}$. Существенно, что данная задача является целочисленной, так как на переменную наложено ограничение $L\in N, L\le T_1$.

Рассмотрим слагаемые, составляющие целевую функцию F:

$$F^1=\sum_{i=1}^{n}\left(\frac{\tau_i}{T_i'}-\frac{\tau_i}{T_i}\right), \qquad F^2=\frac{np}{L}; \qquad (2)$$

$$F=F^1+F^2.$$

Оба слагаемых содержат в себе нелинейности. График первой части суммы F^1 представляет собой набор отсчетов для целочисленных значений аргумента $L\in[1,T_1]$. График второй части суммы – F^2 является положительной частью гиперболы.

Минимально допустимое значение F^1 определяется значением функции при $T_i'=T_i$, что произойдет, когда в выражении $\left\rceil\frac{T_i}{L}\right\lceil L$ при выделении целой части не будет выполняться округлений. Тогда выражение

$$\min(F^1)=\sum_{i=1}^{n}\left(\frac{\tau_i}{T_i'}-\frac{\tau_i}{T_i}\right)=0 \qquad (3)$$

будет определять минимальную границу целевой функции. При этом ее максимальное значение не может превышать $UL=1-\sum_{i=1}^{n}\frac{\tau_i}{T_i}$, поскольку в этом случае расписание вычислительного процесса нельзя построить. Таким образом, целевая функция ограничена сверху и снизу.

ОБОСНОВАНИЕ И АДАПТАЦИЯ АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗАДАЧИ ОПТИМИЗАЦИИ

Анализ описания (1) приводит к выводу, что речь идет о задаче дискретного программирования с нелинейной целевой функцией. Такая классификация задачи обоснована тем, что для выбора метода ее решения требование дискретности является определяющим.

Особенность целевой функции (2) состоит в том, что она имеет две части. Обе части изменяются нелинейно. При этом характер изменения первой части не может

быть никак коррелирован со второй частью функции. Исходя из этого, можно произвести разбиение множеств допустимых планов поиска решений на подмножества с последовательным вычислением целевых функций-оценок каждого подмножества и получать промежуточные значения целевой функции, уточняемые впоследствии. Кроме того, оговоренные ограничения нижнего и верхнего значения целевой функции дают возможность отсекать ее значения, которые в ходе решения уже не соответствуют ограничениям задачи и поэтому в дальнейшем могут не рассматриваться. Такой подход к решению соответствует комбинаторным алгоритмам метода динамического программирования. Учитывая изложенное, для решения поставленной задачи может быть применен уточненный метод ветвей и границ, в результате применения которого удастся получить переменную L.

Пусть существует некоторое множество H, которое содержит все подмножества элементов суммы целевой функции $\sum_{i=1}^{n}(\frac{\tau_i}{T_i'}-\frac{\tau_i}{T_i})$ для всех допустимых планов изменения переменной $L\in[1,T_1]$, то есть $H=\{H_{g1},H_{g2},...,H_{gT_1}\}$. Каждое подмножество имеет представление $H_{gj}=\{\frac{\tau_1}{T_1'}-\frac{\tau_1}{T_1},\frac{\tau_2}{T_2'}-\frac{\tau_2}{T_2},...,\frac{\tau_n}{T_n'}-\frac{\tau_n}{T_n}\}$. Множество V содержит элементы вида $\frac{np}{L}$ для всех планов изменения переменной L. Первый элемент множества, для $L=T_1$ заносится в исходном виде $\frac{np}{L}$. Все остальные элементы представляются модулем своего относительного отношения к предшествующему элементу множества V. Таким образом, множество планов $V=\{\frac{np}{T_1},V_\Delta^{T_1,T_1-1},...,V_\Delta^{2,1}\}$. Отношение $V_\Delta^{i,i-1}=\left|\frac{np}{L}-\frac{np}{L-1}\right|=np\frac{1}{L(L-1)}$ для переменной $L=[2,T_1-1]$. Графически схема получения значений целевой функции может быть интерпретирована следующим образом (рис. 1).

Структура схемы представляет собой дерево с корнем в точке $F=0$. Ветви дерева – элементы множеств H и V. Из точки $F=0$, путем суммирования всех элементов подмножеств по пути следования направляющих стрелок, определяются значения целевой функции $F(L=j)$. Для определения результата целевой функции присутствуют пересечения при выборе элементов множеств. Алгоритм ветвления в этом случае будет определять набор правил по формированию множества текущего шага C. Вычисление границ выполняется по результатам анализа значений оценки функции F на множестве текущего шага. Изначально множество C является пустым C=0. Далее в C заносится элемент V_1 множества V, $C=\{V_1\}$. Это объясняется тем, что он всегда должен быть учтен для всех возможных планов изменения переменной L, но при этом ни один план еще не

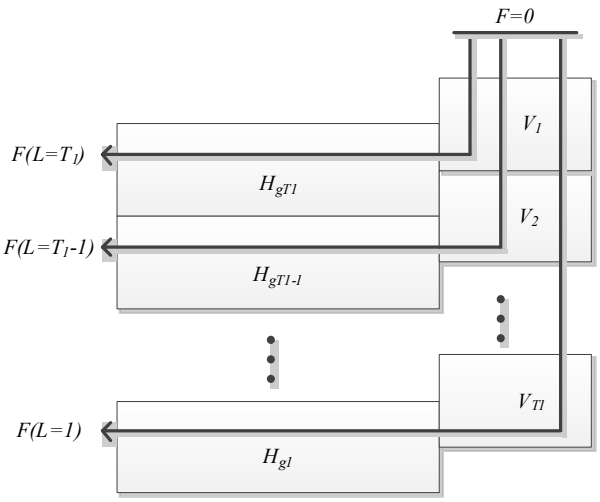

Рис. 1. Схема определения целевой функции F

выбран. Соответственно при этом оценка функции F будет включать только элемент V_1, то есть $F(V_1)$. На следующем этапе выполняется первое ветвление, которое соответствует разделению решений на два варианта:

1) $L=T_1$, этому варианту соответствует подмножество H_{gT_1};

2) $L=[1,T_1-1]$, то есть все остальные варианты, соответствующие оставшимся элементам множеств H и V, а также элемент V_2 как корень поддерева решений.

Таким образом, содержимое множества C будет изменено, $C=\{H_{gT_1},V_2\}$. После этого необходимо выполнить оценку текущего шага, т.е. сравнить значения целевой функции для $F(V_1,V_2)$ и $F(V_1,H_{gT_1}[n])$ (элиминирующий тест). Для получения расчетного значения из подмножества H_{gT_1} всегда выбирается последний элемент. При этом возможны два варианта:

1) Если $F(V_1,H_{gT_1}[n])\le F(V_1,V_2)$, то вариант выбора подмножества H_{gT_1}, которое определяет план выбора переменной $L=T_1$, является оптимальным. В этом случае продолжается рассмотрение множества H_{gT_1}. Выполняется одностороннее ветвление, когда множество, подлежащее разбиению, выбирается только из тех концевых множеств, которые получены на предыдущем шаге ветвления, то есть значение множества C не меняется: $C=\{H_{gT_1},V_2\}$. На следующем шаге будет рассмотрен следующий элемент подмножества $H_{gT_1}[n-1]$, то есть $F(V_1,V_2)$ и $F(V_1,H_{gT_1}[n-1])$. При этом соответственно продолжают рассматриваться 2 плана решений $L=T_1$ и $L=[1,T_1-1]$.

2) Если $F(V_1,V_2)<F(V_1,H_{gT_1}[n])$, значит, по текущему содержимому множества C можно определить, что на данном шаге выбор плана решения $L=T_1$ не является оптимальным. Поэтому необходимо рассмотреть набор вариантов $L=[1,T_1-1]$. В этом случае выполняется ветвление и множество C примет вид $C=\{H_{gT_1},H_{gT_1-1},V_3\}$, так как

вместо элемента V_2, заносятся его ветви – H_{gT_1-1} и V_3. Таким образом, в рассмотрение, помимо добавленного ранее варианта $L = T_1$, добавляются $L = T_1 - 1$ и $L = [1, T_1 - 2]$.

Последовательное выполнение таких действий приведет к получению оптимального значения целевой функции, которое равно значению оценки текущего шага в случае его выбора для первого элемента любого подмножества H_{gi}. [1]. При этом возможна ситуация, когда достигнуто значение V_{T_1}, но решение еще не выбрано. Это означает, что в рассмотрение включены всевозможные планы изменения величины L, но оптимальное значение переменной определить еще не удалось. Тогда выполнение алгоритма продолжается исключительно односторонним ветвлением для всех подмножеств H_{gi}, пока не будет получено оптимальное значение целевой функции.

На одном из шагов при уточнении значения целевой функции F может быть достигнута максимальная граница целевой функции. В таком случае вычисления по данной ветви прекращаются и она в дальнейшем не рассматривается. Максимальная граница, исходя из (1), есть $F = 1$. То есть, план изменения величины L, для которого $F \geq 1$, является недостижимым, поэтому соответствующая ему ветвь и множество H_{gi} исключаются из дерева просмотров.

Схематически дерево поиска решения иллюстрирует рис. 2.

ПОСТРОЕНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

Проведенный анализ позволяет составить алгоритм нахождения оптимального значения переменной L. Пусть $valF[T_1]$ – вектор, который определяет текущее значение целевой функции для всех рассматриваемых планов, *setC, setMinC, setX* – множества, которые содержат подмножества элементов V и H_g, необходимые для анализа на каждой итерации алгоритма. Множество *setC* определяет множество всех элементов, рассматриваемых на одном шаге алгоритма, причем *setMinC* – минимальная величина из множества *setC,setX* – подмножество элементов, связанных с *setMinC* согласно структуре поиска целевой функции (рис. 1).

Рассмотрим последовательность шагов алгоритма.

Изначально все множества инициализируются как пустые.

На первом шаге выполняется поиск множества *setX*, элементы которого связанны с $setMinC$. Если $setMinC = \{0\}$, то выбирается $setX = \{V_1\}$, которое является корнем. Из множества *setC* исключается *setMinC*, как уже рассмотренное, и включается *setX*, как требующее рассмотрения. Одновременно выполняется проверка, является ли множество *setC* пустым, что возможно в ситуации, когда были пройдены все ветви, но при этом допустимое оптимальное значение так и не было найдено. При этом после выполнения FindSet в *setX* будет занесено значение $\{0\}$, так как не осталось нерассмотренных элементов из подмножеств V и H_g. Далее из множества *setC* исключается элемент, рассмотренный на последнем шаге *setMinC*. После объединения с *setX* множество *setC* будет пустым. В этом случае работа алгоритма прекращается, так как для данного набора входных данных найти решение задачи (1) нельзя.

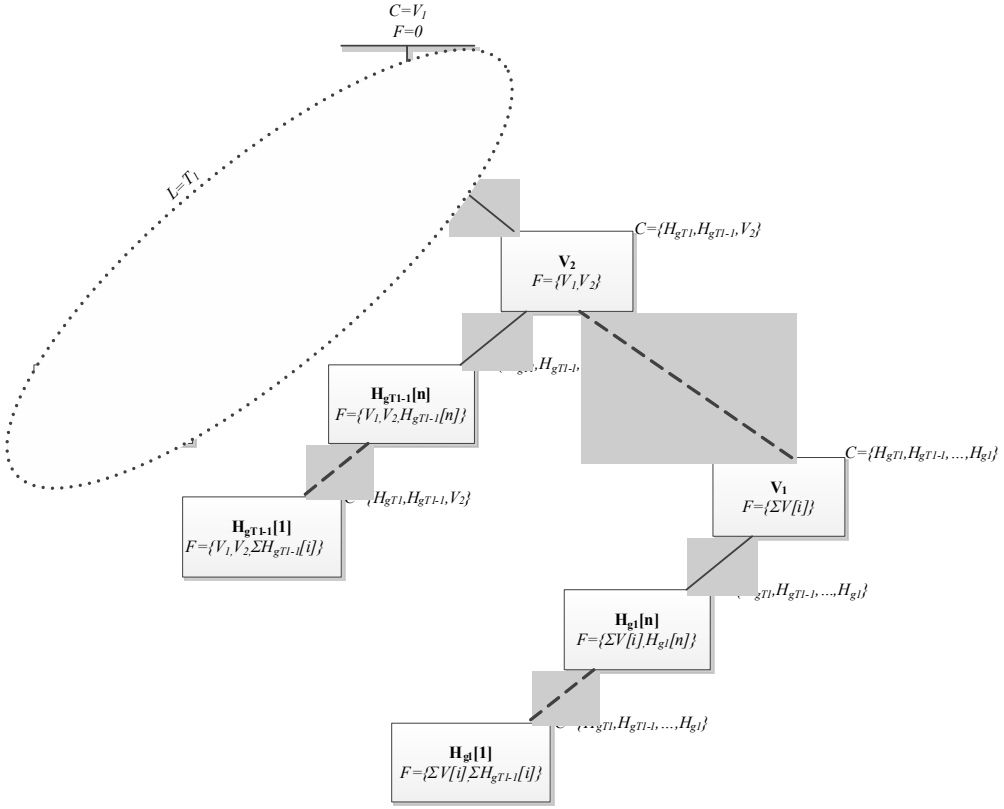

Рис. 2. Дерево поиска решений

На втором шаге выполняется поиск минимального элемента в множестве *setC*. Значение этого элемента заносится в множество *setMinC*. Определение минимума выполняется путем сравнения значений всех элементов множества.

На третьем шаге уточняется текущее значение целевой функции выбранного плана, путем добавления к нему найденного минимума *setMinC*.

На четвертом шаге выполняется элиминирующий тест: превысило ли значение целевой функции максимальную границу, и в этом случае текущая ветка исключается из дальнейшего просмотра.

На пятом шаге выполняется проверка: если минимум, содержащийся в *setMinC,* является первым элементом любого из планов $H_{gi}[1]$, то вычисления прекращаются и план *i* является оптимальным. Отсюда следует, что необходимо выбирать величину $L=i$. В противном случае выполняется переход на шаг 1.

Описанный алгоритм представлен граф-схемой рис. 3.

ТЕСТОВЫЙ ИЛЛЮСТРАТИВНЫЙ ПРИМЕР

Пусть необходимо решить задачу определения характеристик вычислительного процесса системы жесткого реального времени и дан следующий набор входных параметров:

n=4

τ_i	1	3	3	4
T_i	5	16	19	22

p=0,2

Последовательность шагов алгоритма выглядит следующим образом:

1 итерация. После инициализации всех переменных, при первом проходе в множество *setX* будет занесен элемент $V_1 = \frac{np}{T_1} = 0{,}16$. Тогда $F[5] = setMinC = 0{,}16$. Поскольку текущее значение целевой функции меньше 1 и еще есть элементы, которые могут быть рассмотрены, вычисления продолжаются.

2 итерация. Выполняется ветвление $setX = \{H_{gT_1}[4], V_2\} = \{0{,}018; 0{,}04\}$. Выполняется поиск минимального прироста целевой функции, для чего сравниваются между собой значения функций на предыдущих шагах и их возможные приросты. Для L=5 это $F[5]^{-} + H_{gT_1}[4] = 0{,}16 + 0{,}016 = 0{,}178$ и для L=4 – $F[4]^{-} + V_1 + V_2 = 0 + 0{,}16 + 0{,}04 = 0{,}2$. Поскольку 0,178<0,2, выбирается ветвь решения для L=5 и, следовательно, $F[5] = 0{,}178$. Так как значение целевой функции меньше 1 и еще есть элементы, которые могут быть рассмотрены, вычисления продолжаются.

3 итерация. Выполняется ветвление $setX = \{H_{gT_1}[3], V_2\} = \{0{,}042; 0{,}04\}$ (рис. 4).

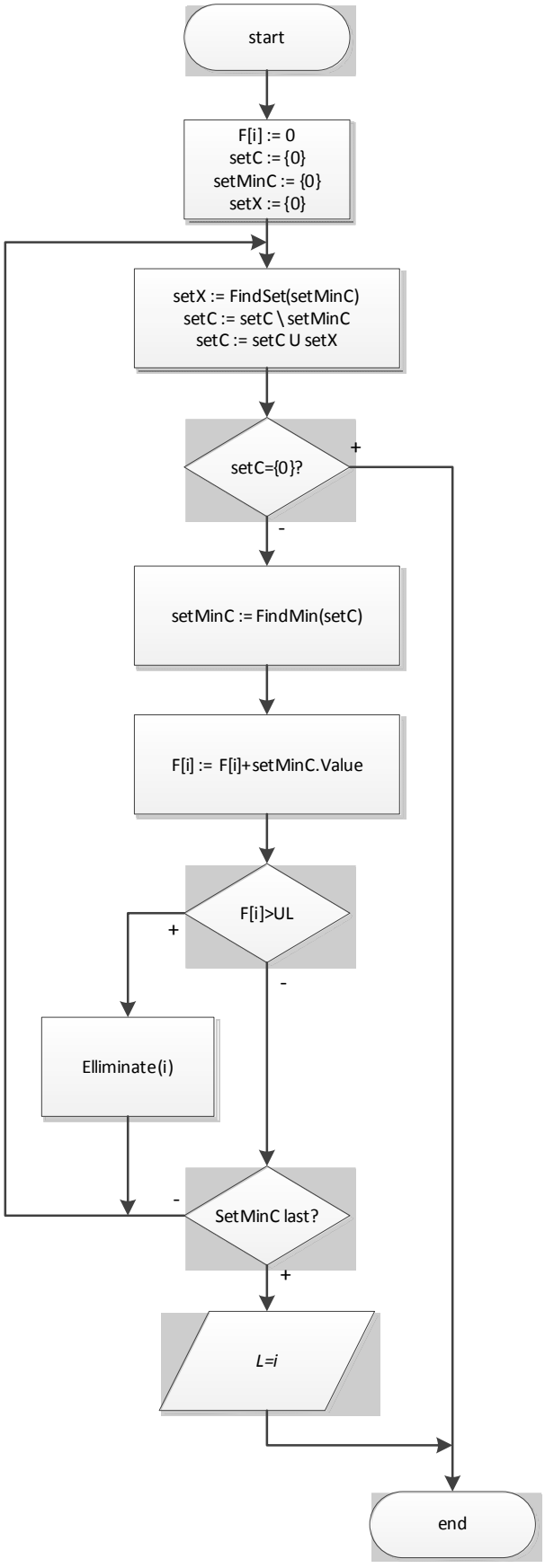

Рис. 3. Граф-схема алгоритма

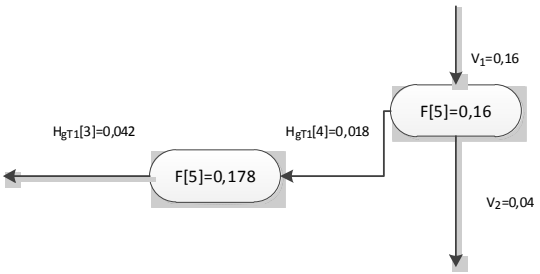

Рис. 4. Построение дерева расчетного примера № 1

Выполняется поиск минимального прироста целевой функции, в данном случае это будет ветвь L=4. Следовательно, уточняется значение $F[4]$ и после выполнения проверок происходит переход на следующую итерацию.

Последовательное выполнение таких итераций обеспечит нахождение решения, которое приведет к вычислению вариантов (рис. 5) и получению решения L=5, со значением целевой функции $F[5] = 0{,}2325$. Работа алгоритма на этом прекращается, поскольку выбран последний элемент из подмножества H_g.

Для подтверждения правильности полученного решения был выполнен расчет значений целевой функции при других значениях L. В результате были получены следующие величины:

L	F
5	0,232
4	0,297
3	0,429
2	0,458
1	0,800

Из расчетов следует правильность нахождения значения целевой функции и определения минимального значения целевой функции. Многочисленные вычислительные эксперименты, как и иллюстрирующий пример, показали полную работоспособность алгоритма.

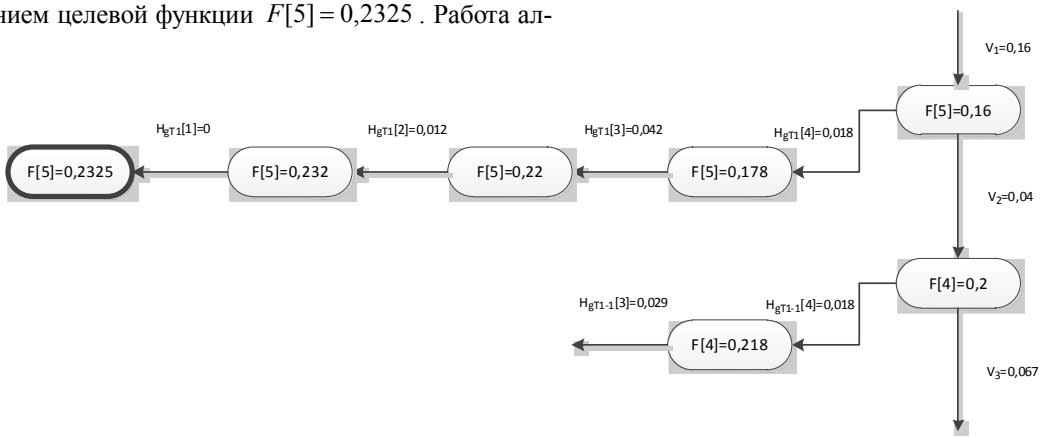

Рис. 5. Построение дерева расчетного примера № 2

ВЫВОДЫ

В данной статье рассмотрена модель задачи определения характеристик циклического вычислительного процесса системы жесткого реального времени. Проанализированы особенности целевой функции и ограничений математической постановки задачи. Рассмотрены возможные подходы к решению задачи оптимизации. На основании анализа особенностей задачи для решения выбран метод ветвей и границ. Базируясь на выбранном методе, составлен алгоритм отыскания переменной искомой целевой функции, учитывающий специфику реальной задачи. Правильность работы алгоритма продемонстрирована на контрольных примерах.

СПИСОК ЛИТЕРАТУРЫ

1. *Корбут, А. А.* Дискретное программирование / А. А. Корбут, Ю. Ю. Финкельштейн. – М. : Наука. Гл.ред.физ.-мат. лит., 1978. – 386 с.
2. *Ху, Т.* Целочисленное программирование и потоки в сетях / Т. Ху. – М. : Мир, 1974. – 520 с.
3. *Пападимитриу, Х.* Комбинаторная оптимизация. Алгоритмы и сложность / Х. Пападимитриу, К. Стайглиц. – М. : Мир, 1984. – 510 с.
4. *Дроздов, Н. Д.* Алгоритмы дискретного программирования / Н. Д. Дроздов // Учебное пособие. Твер. Гос. ун-т. – Тверь : Твер. Гос. ун-т., 2000. – 82 с.
5. *Иванов, Ю.А.* Анализ выполнения программ при моделировании динамических систем / Ю. А. Иванов // Наукові праці Донецького національного технічного університету. Серія «Проблеми моделювання та автоматизації проектування динамічних систем» (МАП-2011). Випуск 10 (197). – Донецьк : ДВНЗ «ДонНТУ», 2011. – С. 234–240.

Стаття надійшла до редакції 13.04.2012.

Іванов Ю. О.

АЛГОРИТМ РІШЕННЯ ЗАДАЧІ ОПТИМІЗАЦІЇ ДЛЯ ЦИКЛІЧНИХ РОЗКЛАДІВ З ПЕРЕРИВАННЯМИ

Розглянута задача оптимізації циклічної організації обчислювального процеса системи жорсткого реального часу. Визначені особливості цільової функції. Обґрунтований вибір цілочисельного метода оптимізації задачі побудови розкладу. Запропонований алгоритм рішення адаптованим методом меж і гілок.

Ключові слова: цілочисельна оптимізація, метод меж і гілок, алгоритм.

Ivanov Y. A.

ALGORITHM FOR SOLVING OPTIMIZATION PROBLEMS OF PREEMPTIVE SCHEDULER

In this paper, we consider a model of the problem to determine the characteristics of the time-step computation process in hard real-time systems. We proposed a formal description of the time-step computation with preemption and its existence conditions.

A necessary condition for the feasibility of the schedule was determined. We took into account the behavior of the function graphs to optimize the schedule. We analyzed the mathematical model features including the objective function and the constraints. In the paper we determined the upper and lower limits of total function variation. The possible approaches to solving the optimization problem were introduced. We proposed to solve that task by the branch and bound method. In the proposed approach the objective function is represented as the set or branches included in the method tree.

We developed the algorithm to determine characteristics of the time-step computation process and give a detailed description. The performance of the algorithm was shown by test case.

Key words: discrete optimization, branch and bound, an algorithm.

REFERENCES

1. Korbut A.A., Finkelshteyn Y.Y. Diskretnoe programmirovanie. Moscow, Nauka, Gl.red.fiz.-mat.lit., 1978, 386 p.
2. Hu T. Tselochislennoe programmirovanie i potoki v setyah. Moscow, Mir, 1974, 520 p.
3. Papadimitriu H., Stayglits K. Kombinatornaya optimizatsiya. Algoritmy i slozhnost. Moscow, Mir, 1984, 510 p.
4. Drozdov N.D. Algoritmy diskretnogo programmirovaniya. Uchebnoe posobie TGU. Tver, 2000, 82 p.
5. Ivanov Y.A. Analiz vypolneniya programm pri modelirovanii dinamicheskih sistem, *Naukovi pratsi Donetskogo natsionalnogo tehnichnogo universitetu, Seriya «Problemi modelyuvannya ta avtomatizatsiyi proektuvannya dinamichnih sistem» (MAP-2011)*, No 10 (197), Donetsk, DVNZ «DonNTU», 2011, pp. 234–240.

УДК 621. 391:519.2:519.7

Лисицкая И. В.[1], Настенко А. А.[2]

[1]Канд. техн. наук, доцент Харьковского национального университета радиоэлектроники
[2]Аспирант Харьковского национального университета радиоэлектроники

ДИФФЕРЕНЦИАЛЬНЫЕ СВОЙСТВА БЛОЧНЫХ СИММЕТРИЧНЫХ ШИФРОВ С МОДУЛЬНЫМИ ОПЕРАЦИЯМИ ВВЕДЕНИЯ ЦИКЛОВЫХ ПОДКЛЮЧЕЙ, ОТЛИЧАЮЩИХСЯ ОТ XOR

Рассматриваются дифференциальные свойства блочных симметричных шифров с применением разных модульных операций вычисления парных разностей.

Ключевые слова: дифференциальные разности; закон распределения переходов дифференциальной таблицы; поцикловые значения максимумов дифференциалов.

ВВЕДЕНИЕ

Современные блочные симметричные шифры строятся с использованием различных операций введения цикловых подключей. По-видимому, авторы считают, что это помогает усилить криптографическую стойкость алгоритмов шифрования [1, 2]. Хотелось бы найти веские аргументы этому, в частности, представляет интерес в свете большого числа публикаций по оценке максимальных значений дифференциальных вероятностей, выполненных, главным образом, по отношению к шифру Rijndael [3–6 и мн. др.], в котором используется операция введения цикловых подключей с помощью побитного XOR, найти соответствующие оценки для шифров с другими операциями введения цикловых подключей и оценить перспективность их использования.

Будет целесообразным здесь напомнить новый подход к оценке стойкости блочных симметричных шифров к атакам дифференциального и линейного криптоанализа, разработанный на кафедре БИТ ХНУРЭ [7], который состоит в оценке соответствующих показателей уменьшенных моделей шифров и определении на основе данных, полученных для уменьшенных моделей, ожидаемых показателей доказуемой стойкости больших прототипов. В ходе реализации этой методики был установлен факт, заключающийся в том, что все итеративные шифры после небольшого начального числа циклов шифрования приобретают дифференциальные свойства случайных подстановок соответствующей степени. Это означает, что интересующие исследователей показатели стойкости могут быть определены расчетным путем из формул, полученных для законов распределения XOR таблиц, выведенных (доказанных) для случайных подстановок [8, 9].

Необходимо отметить, что законы распределения переходов таблиц XOR разностей, построенные для случайных подстановок в известных работах, рассчитывались исходя из предположения, что операцией вычисления разностей пар текстов являлся побитовый XOR «исключающее ИЛИ». Эта операция используется при построении дифференциальных характеристик шифра (дифференциалов) для того, чтобы устранить влияние на построение характеристик цикловых подключей, которые в большинстве блочных симметричных шифров вводятся именно с помощью побитового XOR. Однако, есть немало шифров, где ключ вводится с помощью других модульных операций. Таким образом, встает вопрос об оценке показателей случайности подстановки с отличным от XOR способом вычисления разностей. Это поможет установить ожидаемые значения максимумов дифференциальных вероятностей и для шифров, которые и в этом случае, как ожидается, асимптотически тоже будут повторять свойства случайных подстановок соот-