МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Санкт-Петербургский
государственный университет аэрокосмического приборостроения

О. Л. Смирнов

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ

Учебное пособие

Санкт-Петербург
2001

УДК 658.51.012.011.056(075)
ББК 32.965
С50

Смирнов О. Л.

С50 Автоматизация технологического проектирования: Учеб. пособие / СПбГУАП. СПб., 2001. 66 с.: ил.

Рассматривается системный подход к проектированию рабочих оптимальных сборочно-монтажных процессов производства электронной аппаратуры, создаваемых на основе стандартных типовых технологических процессов подготовки, установки и пайки навесных и электрических элементов, описанных в отраслевых стандартах радиоэлектронной промышленности. Определяются параметры состояния и критерии качества процессов и операций, устанавливаются зависимости критериев от параметров, формулируются постановки задач оптимизации параметров процессов, выбираются методы оптимизации и устанавливается их взаимосвязь в процессе решения задач проектирования, описываются используемые методы оптимизации и применение ЭВМ для их реализации.

Предназначено для студентов специальности "Проектирование и технология радиоэлектронных средств" дневного и вечернего обучения.

Рецензенты:
кафедра конструирования и производства радиоппаратуры СЗПУ;
кандидат технических наук доцент *Д. Н. Сербинов*

Утверждено
редакционно-издательским советом университета
в качестве учебного пособия

Предисловие

Для повышения конкурентоспособности отечественных изделий в условиях быстрой смены ассортимента необходимо использовать САПР на всех этапах подготовки производства. Создание эффективных САПР невозможно без разработки методов проектирования оптимальных технологических процессов и, в частности, процессов сборки и монтажа. В пособии описывается применение результатов, полученных в теории исследования операций и принятия решений для постановки и разработки методов решения задач проектирования оптимальных процессов сборки и монтажа электронной аппаратуры (ЭА).

Задачу проектирования сборочно-монтажных процессов можно сформулировать следующим образом. На сборочном участке собираются изделия нескольких наименований. Рабочие места должны обеспечивать полный цикл сборки. Известны сборочные чертежи, спецификации, схемы сборочного состава, объем и сроки выпуска изделий, каталоги с характеристиками оборудования и технологической оснастки для выполнения операций, размеры сборочного участка. Необходимо определить состав оборудования и технологической оснастки, распределение операций по рабочим местам, порядок их выполнения и режимы операций так, чтобы минимизировать стоимость всех процессов для заданных сроков, точности и надежности. Искомыми оптимизируемыми переменными являются бинарные переменные выбора оборудования и распределения операций по рабочим местам, а также комбинационные переменные порядка запуска операций на рабочих местах. При решении усложненной задачи выбора оптимальных режимов выполнения операций в качестве дополнительной группы переменных выступает зависимость минимальной стоимости выполнения каждой операции от остальных ее технико-экономических критериев: времени, точности и надежности. Искомым результатом является зависимость минимальной стоимости общего сборочно-монтажного процесса от других его технико-экономических критериев.

На основе формулировки задачи разработана система уравнений и неравенств, определяющая зависимость критериев качества общего процесса от указанных выше переменных. Эту систему можно назвать математической моделью общего сборочно-монтажного процесса. На основе модели ставится задача ее оптимизации. Для решения задачи приходится использовать в основном четыре метода оптимизации: линей-

ного целочисленного программирования, динамического программирования, метод ветвей и границ и метод ветвей и частичных решений бинарных переменных. Вначале описывается теория многокритериальной оптимизации многоуровневых иерархических систем, включающая понятия о безусловно лучших и эффективных решениях, условия существования и экстремальные свойства эффективных решений, и методика поиска эффективных систем на основе эффективных компонентов. Далее описываются особенности применения перечисленных выше методов оптимизации. Затем формулируется задача, приводится ее математическое описание. Обсуждаются возможные варианты постановки задачи оптимизации общего процесса, описывается методика применения ранее описанных методов для решения общей задачи оптимизации сборочно-монтажных процессов.

1. Понятие о безусловно лучших и эффективных решениях

Будем называть $X = (x_1, x_2, ..., x_n)$ решением задачи проектирования устройства или процесса. Рассматриваются только допустимые решения X, т. е. такие, которые принадлежат множествам:

– допустимых решений $D_X = (X_{\min} < X < X_{\max})$;

– допустимых функций $F(X) = \{f_1(X), f_2(X), ..., f_m(X)\}$, выполняемых проектируемым устройством или процессом $D_F = \{F_{\min}(X) < F(X) < F_{\max}(X)\}$;

– допустимых критериев эффективности $K(X) = \{k_1(X), k_2(X), ..., k_r(X)\}$ функционирования изделия или процесса $D_K = \{K_{\min}(X) < K(X) < K_{\max}(X)\}$.

Таким образом, допустимыми являются $X \in D = (D_X \cap D_F \cap D_K)$.

Решение X безусловно лучше решения X' ($X < X'$), если для всех критериев эффективности $k_t(X)$, $t = 1, ..., T$ $k_t(X) \le k_t(X')$, кроме хотя бы одного $t = s$, для которого существует строгое неравенство $k_t(X) < k_t(X')$.

Решение X является эффективным, если не существует ни одного другого решения X', безусловно лучшего, чем решение X. Множество эффективных решений будем обозначать через E.

2. Теорема о существовании эффективных решений

Теорема 1. Для существования множества эффективных решений E необходимо и достаточно, чтобы множество D допустимых решений было замкнуто, а критерии эффективности $k_t(X)$, $t = 1, ..., T$ были непрерывными функциями от X.

Лемма 1. Пусть критерий $k_t(X)$ – непрерывная функция от X, а множество D допустимых решений X замкнуто, тогда подмножество $D_C = \{X: k_t(X) = C\}$ также замкнуто.

Доказательство леммы. Множество замкнуто, если предел любой сходящейся последовательности точек этого множества тоже принадлежит этому множеству. Так как функция $k_t(X)$ – непрерывна, то какую бы последовательность X_i, $i = 1, ..., \infty$, сходящуюся к X', ни взять, соответствующая последовательность $k_t(X_i)$, $i = 1, ..., \infty$ сходится к $k_t(X')$. Возьмем последовательность $X_i \in D_C$, сходящуюся к X'. Значения $k_t(X_i) = C$,

$i = 1, \ldots, \infty$. В силу непрерывности $k_t(X)$ ее значение для предельной точки X' тоже равно C. Тогда, по определению, подмножества $D_C X' \in D_C$. Поскольку предельная точка сходящейся последовательности тоже принадлежит тому же подмножеству D_C, значит это подмножество D_C – замкнуто.

Доказательство теоремы. Выделим из D подмножество D_1, состоящее из X, для которых $k_1(X)$ равно минимальному значению $k_{1\min}$:

$$D_1 = \{X : k_1(X) = \min_{X \in D} k_1(X) = k_{1\min}\}.$$

Если D_1 содержит одну точку X, т. е. для всех $X' \in D$ $k_t(X) < k_t(X')$, то X безусловно лучше всех сравнимых с ней X' из D по определению. Другими словами, не существует X' из D безусловно лучше, чем X, т. е. X есть эффективное решение по определению. Значит, $X \in E$, следовательно, теорема доказана.

Если D_1 содержит не одну точку X, то, по лемме, D_1 – замкнуто. Выделим из D_1 подмножество D_2, состоящее из X, для которых $k_2(X)$ равно минимальному значению $k_{2\min}$:

$$D_2 = \{X : k_2(X) = \min_{X \in D_1} k_2(X) = k_{2\min}\}.$$

Если D_2 содержит одну точку X, то она, как было показано выше, является эффективной. Значит, E существует, следовательно, теорема доказана. Если D_2 содержит не одну точку X, то, по лемме, D_2 – замкнуто.

Продолжая выделять подмножество D_3 и другие, можем дойти до T-го, в котором D_T может иметь не одну точку. Тогда любая из них эффективна, так как не существует среди них другая, которая была бы безусловно лучше, чем рассматриваемая. Следовательно, множество эффективных решений E существует. Теорема доказана.

3. Теорема об экстремальных свойствах эффективных решений

Теорема 2. Пусть $X \in E$ и $k_t(X) = A_t$, $t = 1, \ldots, T$. Тогда для $t = s$

$$A_s = \min_{X \in D} k_s(X) \Big|_{k_t(X) = A_t,\, t \neq s}.$$

Доказательство. Пусть нашлось в D решение X', для которого $A_s < k_s(X)$ при $k_t(X) = A_t$, $k \neq s$. Тогда, по определению, X' безусловно лучше, чем X. Значит, решение X, по определению эффективности, не эффективно, что противоречит условию теоремы. Следовательно, $A_s = \min\limits_{X \in D} k_s(X)\Big|_{k_t(X)=A_t, t\neq s}$, что и требовалось доказать.

4. Динамическое программирование

Постановка задачи такова: найти минимум (максимум) $f(x_1, x_2, ..., x_n) = f_1(x_1)+f_2(x_2)+...+f_n(x_n)$ при условии, что $x_1+x_2+...+x_n = A_n$, где $A_n \le A$. Подобного рода задачи часто встречаются в экономике, технологии, организации производства.

Обозначим $F_n(A_n) = \min\limits_{x_1, x_2, ..., x_n} \left(f_1(x_1) + f_2(x_2) + ... + f_n(x_n)\right)$, тогда

$$\min_{x_1, x_2, ..., x_n} \left(f_1(x_1) + f_2(x_2) + ... + f_n(x_n)\right) =$$

$$= \min_{x_n}\left(\min_{x_1, x_2, ..., x_{n-1}} \left(f_1(x_1) + f_2(x_2) + ... + f_n(x_n)\right)\right).$$

Так как при поиске внутреннего минимума последнее слагаемое $f_n(x_n)$ не меняется, то последнее выражение можно переписать как

$$\min_{x_n}\left(f_n(x_n) + \min_{x_1, x_2, ..., x_{n-1}} \left(f_1(x_1) + f_2(x_2) + ... + f_{n-1}(x_{n-1})\right)\right).$$

Второе слагаемое внутри скобок можно обозначить как $F_{n-1}(A_{n-1})$. Тогда можно записать, что $F_n(A_n) = \min\limits_{x_n}\left(f_n(x_n) + F_{n-1}(A_{n-1})\right)$, где $x_1+x_2+...+x_{n-1} = A_{n-1}$.

Для $n = 2$ $F_2(A_2) = \min\limits_{x_2}\left(f_2(x_2) + f_1(x_1)\right)$, где $x_1+x_2 = A_2$.

Решение начинается с определения сначала $F_2(A_2)$ и $x_2^{\text{opt}}(A_2)$, затем $F_3(A_3) = \min\limits_{x_3} f_3\left((x_3) + F_2(A_2)\right)$ и $x_3^{\text{opt}}(A_3)$ и т. д. и, наконец, $F_n(A_n)$ и $x_n^{\text{opt}}(A_n)$.

Для любого A_n' определяется $F_n(A_n')$ и соответствующие $x_n^{\text{opt}}(A_n')$, $x_{n-1}^{\text{opt}}\left(A_{n-1}' = A_n' - x_n^{\text{opt}}(A_n')\right)$, $x_{n-2}^{\text{opt}}\left(A_{n-2}' = A_{n-1}' - x_{n-1}^{\text{opt}}(A_{n-1}')\right)$ и т. д. до $x_1^{\text{opt}}\left(A_1' = A_2' - x_2^{\text{opt}}(A_2')\right)$.

5. Метод оптимизации многокритериальных иерархических систем с аддитивными критериями

В настоящее время резко увеличился ассортимент устройств ЭВМ одного и того же назначения с примерно одинаковыми функциональными свойствами и параметрами, но различающимися ценой и надежностью. Если взять такой ряд устройств и сгруппировать их по возрастанию ненадежности, т. е. интенсивности отказов, и для каждой группы выбрать устройство с минимальной стоимостью, то получится множество эффективных по Парето устройств одного назначения с функциональной зависимостью между рассматриваемыми критериями качества $C_{min}(\lambda)$, где C_{min} – минимальная стоимость, а λ – интенсивность отказов эффективных устройств. График зависимости имеет вогнутый вид и при возрастании значения интенсивностей отказов стремится к пределу, определяемому стоимостью покупных компонентов и поточной сборки. Из опыта известно, что, чем надежнее устройство, тем оно дороже, и что одно и то же увеличение надежности для более надежных устройств достигается с большим трудом, чем для менее надежных. Поэтому зависимость $C_{min}(\lambda)$ можно аппроксимировать экспонентой

$$C_{min}(\lambda) = C_0 - Ae^{-k\lambda}. \quad (1)$$

Пусть имеется компьютерная система типа Entertaitment PC (игровой центр), состоящая из системного блока (СБ), к которому через порты USB присоединены монитор (М) и принтер (П), а к порту IEEE 1394 через размножитель присоединены DV-видеомагнитофон (ВМ) и DV-

видеокамера (ВК). Система состоит из двух подсистем: первой (СБ, М, П) и второй (ВМ, ВК). Для каждого устройства определен ряд эффективных изделий с одними и теми же требуемыми техническими характеристиками, но отличающихся интенсивностью отказов. Зависимость $C_{min}(\lambda)$ для них приведена на рис. 1. Графикам зависимостей *1 – 5* соответствуют устройства СБ, П, М, ВМ и ВК. Для удобства обозначим их минимальные стоимости через C_{11}, C_{12}, C_{13} для первой подсистемы (с минимальной стоимостью C_1) и C_{21} , C_{22} – для второй подсистемы (с минимальной стоимостью C_2).

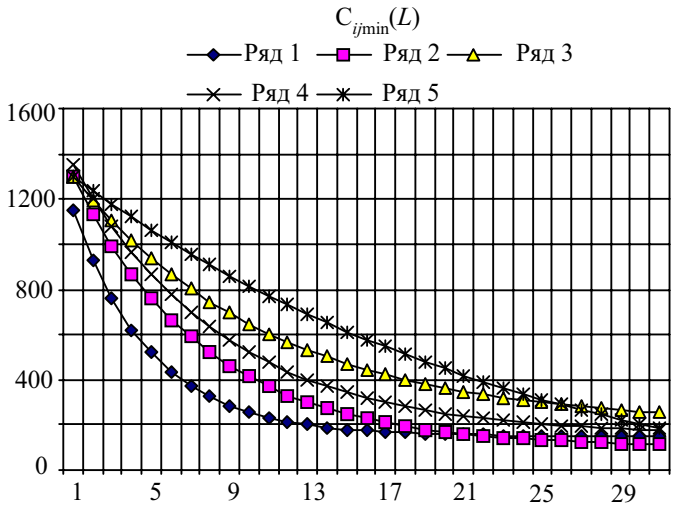

Рис. 1. Графики *1* – 5 минимальной стоимости устройств СБ, П, М, ВМ, ВК в зависимости от λ

Значения коэффициентов в формуле (1) для устройств СБ, М, П, ВМ, ВК приведены в табл. 1.

Таблица 1

Коэффициенты в формуле (1) для системных устройств и стоимости их ремонта

Стоимость	C_0	*A*	*k*
C_{11}	150	1000	0,25
C_{12}	100	1200	0,15
C_{13}	200	1100	0,1
C_{21}	150	1200	0,13
C_{22}	–300	1600	0,04
$C_{рем}$	50	4000	0,24

Требуется выбрать из рядов устройств такие, чтобы общая стоимость $C_{общ}$, равная стоимости их приобретения C и ремонта $C_{рем}$, была минимальной. Поскольку критерии C и λ аддитивны, т. е. и стоимость C, и интенсивность λ системы, соответственно, равны стоимости и интенсивности отказов устройств, то эту задачу можно решить методом динамического программирования.

Вначале определяется зависимость от λ минимальной стоимости C_{112} первых двух устройств СБ и П

$$C_{112}(\lambda_{112}) = \min_{\lambda_{11}} (C_{11}(\lambda_{11}) + C_{12}(\lambda_{112} - \lambda_{11})). \quad (2)$$

Затем определяется зависимость от λ минимальной стоимости C_1 первой подсистемы по зависимости $C_{13}(\lambda_{13})$ для третьего устройства М и только что полученной зависимости (2) по аналогичной формуле, отличающейся от (2) только индексами у переменных:

$$C_1(\lambda_1) = \min_{\lambda_{13}} (C_{13}(\lambda_{13}) + C_{112}(\lambda_1 - \lambda_{112})). \quad (3)$$

Далее аналогично определяется зависимость $C_2(\lambda_2)$ по известным стоимостям $C_{21}(\lambda_{21})$ и $C_{22}(\lambda_{22})$, а также зависимость $C(\lambda)$ всей системы по известным стоимостям $C_1(\lambda_1)$ и $C_2(\lambda_2)$. Для определения оптимального значения стоимости системы C_{opt} воспользуемся зависимостью стоимости ремонта составляющих ее устройств, которая увеличивается экспоненциально с ростом интенсивности их отказов. Эту зависимость можно представить формулой

$$C_{рем} = C_0 + Ae^{-k(30-\lambda)},\ 0 < \lambda < 30. \quad (4)$$

Далее определяется стоимость приобретения устройств и их ремонта

$$C_{общ}(\lambda_{общ}) = C(\lambda) + C_{рем}(\lambda). \quad (5)$$

Поскольку стоимость системы $C(\lambda)$ – убывающая, а стоимость ремонта $C_{рем}(\lambda)$ – возрастающая, то сумма имеет минимум $C_{общmin}$ внутри заданного интервала значений интенсивности λ. Оптимальной системой является та, для которой стоимость устройств и их ремонта – минимальна. На рис. 2 приведены графики зависимости минимальных стоимостей C_1 и C_2 обеих подсистем, всей системы C, стоимости ремонта $C_{рем}$ устройств и общей стоимости $C_{общ}$.

Оптимальной является система с интенсивностью в 19 отказов. Стоимость ее покупки – 4115 у.е., стоимость ремонта – 325 у.е. Самая доро-

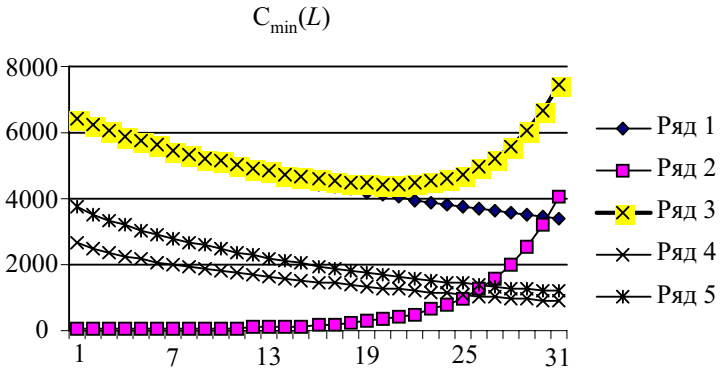

Рис. 2. Графики зависимостей: *1* – С , *2* – $С_{рем}$, *3* – $С_{общ}$, *4* – $С_1$, *5* – $С_2$

гая система с той же интенсивностью отказов стоит 5280 у.е. (соответствующие стоимости подсистем – 2921 и 2369 у.е.). Выбор оптимальных компонентов позволяет сократить расходы на покупку системы на 28 %, а общие расходы на систему и ремонт – на 26 %.

Чтобы найти стоимости оптимальных устройств (это те из эффективных, стоимости которых минимизируют общие расходы), надо определить ту пару подсистем, стоимость которых равна 4115 у.е., а суммарная интенсивность отказов равна 19 отказам за планируемый период. Чтобы узнать соответствующие этим подсистемам устройства, надо поступить так же. Для первой подсистемы определяется тройка устройств, у которых суммарная интенсивность отказов та же, что и для подсистемы, а стоимость устройств равна стоимости подсистемы. Так же поступают со второй подсистемой. В табл. 2 приведены оптимальные стоимости устройств, подсистем и их интенсивности отказов.

Таблица 2

Оптимальные стоимости и интенсивности отказов устройств, подсистем и системы

Критерий	Устройство			Подсистема					Система
	$У_{11}$	$У_{12}$	$У_{13}$	$У_{21}$	$У_{22}$	$У_1$	$У_2$	У	$С_{общ}$
$С_{opt}$, у.е.	437	587	1015	2039	776	1300	2076	115	4450
λ_{opt}	5	6	3	14	5	0	5	9	19

Таким образом, предложена методика выбора оптимальных компонентов системы, минимизирующая стоимость их приобретения и ре-

монта на примере постановки и решения задачи выбора оптимальных компонентов вычислительной системы методом динамического программирования.

6. Содержательная постановка задачи оптимизации сборочно-монтажных процессов производства ЭА

Производство ЭА характеризуется стремительным расширением ассортимента изделий, ускоряющейся сменой их поколений, расширением функциональных возможностей аппаратуры, улучшением качества, надежности и дизайна при одновременном уменьшении веса, габаритов и стоимости.

Для поддержания конкурентоспособности отечественного производства одним из необходимых факторов является оптимизация и автоматизация проектирования технологических процессов. Для этого необходима математическая модель процессов, связывающая параметры процессов с критериями их качества, и разработка методов оптимизации этих параметров. Существующие модели сборочно-монтажных процессов не охватывают весь комплекс параметров и свойств процессов, вследствие чего разработанные к настоящему времени методы и алгоритмы не позволяют проектировать оптимальные процессы.

Поэтому задача разработки моделей, позволяющих охватить все этапы проектирования процессов от выбора оборудования и распределения операций по рабочим местам до определения порядка выполнения операций на каждом рабочем месте, является актуальной.

Предлагаются модели, позволяющие разработать методы оптимизации проектируемых сборочно-монтажных процессов в электронике для многономенклатурного производства с частой сменой изделий.

Проектирование общего процесса должно включать выбор оборудования. Во-первых, этот выбор необходим при организации нового производства. Во-вторых, в действующем производстве изменение ассортимента изделий может повлечь необходимость частичной замены старого оборудования новым. Задача проектирования с частичным выбором оборудования есть частный случай проектирования общего процесса с полным выбором оборудования.

На рис. 3 приведена схема общего процесса сборки и монтажа двух электронных блоков $Б_{21}$ и $Б_{22}$. Общий процесс – граф $\mathbf{P}(P, U)$, где P – множество всех операций $p \in P$, а U – множество отношений $u \in U$ порядка их выполнения. Общий процесс состоит из процессов $\mathbf{p} \in \mathbf{P}(P, U)$ изготовления отдельных блоков ЭА.

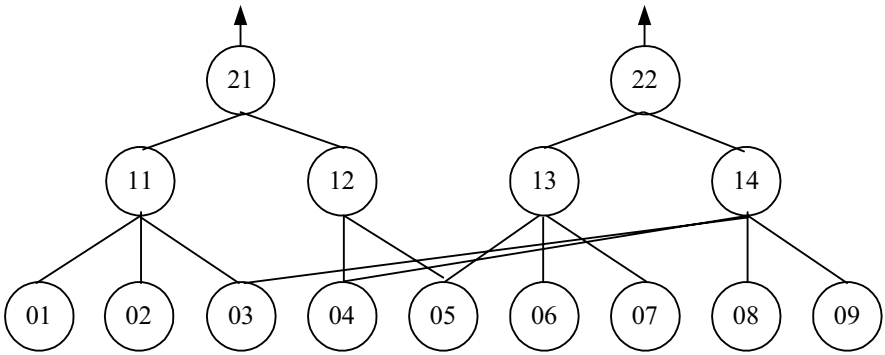

Рис. 3. Структурная схема общего сборочно-монтажного процесса

Исходными данными являются ассортимент, объем и сроки выпуска изделий, конструкторская документация и структурные схемы сборочного состава изделий, номенклатура и характеристики существующего технологического оборудования, величина производственной площади технологического подразделения и необходимая нормативная документация (типовые технологические процессы, нормы времени и т. д.).

Выходными данными являются число рабочих мест, номенклатура и характеристики выбранного технологического оборудования и оснастки, состав и структура процессов сборки и монтажа изделий, распределение всех операций по рабочим местам и порядок их выполнения на каждом рабочем месте. В модель включаются соотношения для определения размера партии, режимов работы оборудования, операционного времени, себестоимости, точности и надежности выполнения операций и процессов. Критериями эффективности являются себестоимость всех процессов, сроки выпуска изделий, неравномерность загрузки рабочих мест, качество и надежность сборки изделий.

Проектирование процесса – выбор одного из допустимых. Назовем его решением. Из теории многокритериальной оптимизации следует, что:

– существуют эффективные решения, если множество значений допустимых параметров замкнуто и критерии качества – непрерывные функции от параметров;

– решение эффективно, если среди сравнимых с ним нет безусловно лучшего, чем оно;

– два решения сравнимы, если все критерии одного не хуже (или не лучше) соответствующих другого;

– из двух сравнимых решений безусловно лучше то, у которого хотя бы один критерий строго лучше соответствующего критерия другого;

– только для эффективных решений между альтернативными критериями имеется такая взаимозависимость, при которой улучшение одного критерия влечет ухудшение хотя бы одного из других;

– для получения такой взаимозависимости необходимо получить предельное улучшение любого критерия при фиксированных значениях остальных, которые, в свою очередь, меняются в заданных пределах;

– какой бы критерий ни выбрать в качестве улучшаемого при заданном значении остальных, результатом оптимизации будет одна и та же взаимозависимость между критериями.

Поэтому в качестве целевого можно выбрать любой критерий, который упрощает решение задачи – определение взаимосвязи между критериями. Решение будет проще, если в качестве целевого взять стоимостной критерий. Если целевым критерием необходимо выбрать время, то решение можно получить, минимизируя стоимость для разных значений времени. Данная зависимость позволяет получить зависимость времени от стоимости.

Выбор оборудования, распределение операций общего процесса по рабочим местам, выбор размера партий и порядка выполнения операций на рабочих местах, а также выполнение каждой операции – многовариантны. Каждый вариант отличается значениями критериев качества: стоимостью, временем выполнения, точностью и надежностью. Целью проектирования общего процесса и операции является поиск их эффективных вариантов и определение взаимозависимости критериев качества проектирования и выполнения операции для этих вариантов. Эти две задачи связаны между собой.

7. Математическая постановка задачи проектирования оптимальных сборочно-монтажных процессов изготовления ЭА с неизменяемыми режимами выполнения операций

Для постановки общей задачи необходимы:

P_0 – множество начальных операций $p_0 \in P_0$ общего процесса **p**;

P_{p_0} – множество всех операций $p \in P_{p_0}$ на пути от начальной p_0 до одной из конечных (p_{21} или p_{22});

B – множество всех существующих типов оборудования $b \in B$, которые могут быть использованы для выполнения операций общего процесса.

Если режим выполнения операции на оборудовании фиксирован, то ее стоимость, время, точность и надежность выполнения – постоянные величины. Тогда критерии качества общего процесса можно представить следующей системой уравнений и неравенств, составляющие которой определены далее:

– стоимость общего процесса

$$C = \sum_{p \in P} \sum_{b \in B} (C_{pb} t_{pb} N_p + C_p \tau_p(\boldsymbol{\Delta}, \boldsymbol{\pi}) k_p) \delta_{pb} \to \min; \tag{5}$$

– сроки выпуска каждого изделия

$$\sum_{p \in P_{p_0}} \sum_{b \in B} \left(t_{pb} N_p + \tau_p(\boldsymbol{\Delta}, \boldsymbol{\pi}) k_p \right) \delta_{pb} \le T_{\mathbf{p}}, \quad p_0 \in P_0, \quad \mathbf{p} \in \mathbf{P}; \tag{6}$$

– неравномерность загрузки каждого рабочего места

$$\sum_{p \in P} t_{pb} N_p \delta_{pb} \le T_b (1 + \delta T_b), \quad b \in B; \tag{7}$$

$$\sum_{p \in P} t_{pb} N_p \delta_{pb} \ge T_b (1 - \delta T_b), \; b \in B; \tag{8}$$

– запрет выполнения операции на нескольких рабочих местах

$$\sum_{b \in B_p} \delta_{pb} = 1, \; p \in P; \tag{9}$$

– запрет выполнения операций на неустановленном оборудовании

$$\sum_{p \in P_b} \delta_{pb} \le \delta_b |P_b|,\ b \in B; \tag{10}$$

– ограничение на общую площадь установленного оборудования

$$\sum_{b \in B} S_b \delta_b \le S. \tag{11}$$

В системе (5)–(11) неизвестными являются:

– вектор $\mathbf{\Delta}$ выбора оборудования δ_b и распределения δ_{pb} операций по рабочим местам

$$\mathbf{\Delta} = \{\delta_b,\ b \in B,\ \delta_{pb},\ p \in P,\ b \in B\}; \tag{12}$$

– число партий для каждой операции

$$k_p,\ p \in P;$$

– вектор очередности выполнения операций на рабочих местах

$$\boldsymbol{\pi} = \{\pi_b,\ b \in B\},$$

где π_b – порядок выполнения операций на рабочем месте b:

$$\pi_b = \{\pi_{b1}, \pi_{b2}, \pi_{b3}, \ldots, \pi_{bi}, \ldots, \pi_{bni}\}, \tag{13}$$

где π_{bi} – операция p, распределенная на рабочее место с оборудованием b и поставленная на i-е место в очередь на выполнение.

Переменные δ_b и δ_{pb} – бинарные. Если $\delta_b = 1$, то это означает, что оборудование b приобретено и установлено в технологическом подразделении, иначе оборудование b не установлено в подразделении. Если $\delta_{pb} = 1$, то на этом оборудовании выполняется операция p. В противном случае операция p выполняется на другом виде выбранного оборудования.

8. Стоимость общего сборочно-монтажного процесса производства ЭА

Первым критерием качества процесса является стоимость С его выполнения (5). Она зависит от:

C_{pb} – стоимости выполнения каждой операции p на оборудовании b в единицу времени;

t_{pb} – штучного времени выполнения этой операции;

N_p – общего числа повторений этой операции;

τ_p – времени ожидания выполнения операции над партией компонентов;

C_p – стоимости ожидания выполнения операции в единицу времени;

k_p – числа партий компонентов, над которыми выполняется рассматриваемая операция.

Стоимость C_{pb} определяется выражением

$$C_{pb} = C_{pbz} + C_{pbe} + C_{pba} + C_{pbr}, \quad (14)$$

где C_{pbz} – стоимость заработной платы в единицу времени, зависящая от тарифной ставки C_z, коэффициентов занятости рабочего K_z, планового выполнения нормы K_p и социального страхования K_s:

$$C_{pbz} = C_z K_z K_p (1 + K_s); \quad (15)$$

C_{pbe} – стоимость электроэнергии и других видов энергии в единицу времени;

C_{pba} – стоимость амортизации оборудования в единицу времени, зависящая от стоимости оборудования $C_о$, коэффициентов его амортизации $K_а$ и ремонта $K_р$, стоимости единицы площади $C_п$, величины ее под установку оборудования $S_п$ и коэффициента ее амортизации $K_п$:

$$C_{pba} = C_о (K_а + K_р) + C_п S_п K_п; \quad (16)$$

C_{pbr} – стоимость реновации оборудования в единицу времени, зависящая от коэффициента эффективности капиталовложений K_r, стоимости оборудования и площади под него:

$$C_{pbr} = K_r (C_о + C_п S_п). \quad (17)$$

9. Штучно-калькуляционное время и ограничение на сроки процессов и загрузку рабочих мест

Штучное время t_{pb} вычисляется по формуле

$$t_{pb} = t_{pb}^{оп}(1 + \delta t_{pb}^{оп}), \quad (18)$$

где $\delta t_{pb}^{оп}$ зависит от подготовительно-заключительного $T_{pb}^{пз}$ и операционного $t_{pb}^{оп}$ времени, а также от размера партии $n_p = N_p/k_p$:

$$\delta t_{pb}^{оп} = T_{pb}^{пз} /(t_{pb}^{оп} n_p). \quad (19)$$

Вторым критерием качества общего процесса является вектор **T** заданных сроков $T_{\mathbf{p}}$ выпуска каждого изделия (процесс $\mathbf{p} \in \mathbf{P}$). Он зависит от суммы времен выполнения операций в цепочке их от каждой начальной операции p_0 до конечной операции технологического процесса сборки одного из соответствующих изделий. Этот критерий описывается системой неравенств (6).

Третьим критерием является неравномерность загрузки рабочих мест δT_b, равная 5–10 % загрузки оборудования b. Неравенства (7) и (8) определяют допустимую загрузку рабочих мест операциями общего процесса.

Система уравнений (9) учитывает требование выполнения операции над партией на одном рабочем месте. Система неравенств (10) запрещает распределение операций на не установленное в подразделении оборудование b. Если оборудование не установлено, то $\delta_b = 0$. Тогда для удовлетворения неравенству (10) δ_{pb} также должны быть равны нулю. Если же $\delta_b = 1$, то по условию (10) на оборудовании b может выполняться любое число операций, для которых разрешено использование этого оборудования. Система неравенств (11) накладывает ограничение на общее число устанавливаемого в подразделении оборудования.

10. Математическая постановка задачи проектирования оптимальных сборочно-монтажных процессов изготовления ЭА с изменяемыми режимами выполнения операций

Если режим выполнения операции на оборудовании не один, то ее стоимость, время, точность и надежность выполнения являются функциями от режима. Тогда для каждой операции необходимо найти множество эффективных режимов ее выполнения и определить для них взаимосвязь между критериями качества этой операции в виде зависимости минимальной стоимости ее выполнения $C_{pb\min}$ от текущих значений времени выполнения t_{pb} операции, ее точности Δ_{pb} и надежности (интенсивности отказов) Λ_{pb}

$$C_{pb\min} = C_{pb_0} t_{pb}^{\alpha t} \Delta_{pb}^{\alpha\Delta} \Lambda_{pb}^{\alpha\Lambda}. \qquad (20)$$

При этом изменится вид целевой функции и добавятся системы неравенств, учитывающие ограничения точности и надежно-

сти общего технологического процесса. Стоимость процесса примет вид

$$C = \sum_{p \in P} \sum_{b \in B} \left(C_{pb_0} t_{pb}^{\alpha t} \Delta_{pb}^{\alpha \Delta} \Lambda_{pb}^{\alpha \Lambda} N_p + C_p \tau_p(\boldsymbol{\Delta}, \boldsymbol{\pi}) k_p \right) \delta_{pb} \to \min. \qquad (21)$$

Ограничение по точности общего процесса примет вид

$$\sum_{p \in P_{p_0}} \sum_{b \in B} K_{pb}^2 \Delta_{pb}^2 \delta_{pb} \le \Delta_{\mathbf{p}}^2, \quad p_0 \in P_0, \mathbf{p} \in \mathbf{P}, \qquad (22)$$

где K_{pb}^2 – коэффициент влияния погрешности текущей операции на погрешность изделия, создаваемого процессом $\mathbf{p} \in \mathbf{P}$.

Ограничение по надежности общего процесса примет вид

$$\sum_{p \in P} \sum_{b \in B} N_p \Lambda_{pb} \delta_{pb} \le \Lambda. \qquad (23)$$

Таким образом, получены модели выбора оборудования, распределения операций по рабочим местам, определения числа партий и режима выполнения каждой операции, порядка их выполнения на рабочих местах, т. е. полного перечня работ по разработке общего процесса технологического подразделения. Эти модели позволяют сосредоточить усилия на поиск методов решения задачи минимизации целевой функции для произвольных значений остальных критериев качества общего процесса и получения зависимости $C_{\min}$ (**T**, **N**, **δT**, **Δ**, **Λ**), где **T** – вектор сроков выпуска изделий, **N** – вектор объемов выпусков изделий заданной номенклатуры, , **δT** – вектор неравномерности загрузки рабочих мест, **D** – вектор точности параметров изделий, **Λ** – заданная интенсивность отказа общего процесса.

11. Нецелочисленное решение задач линейного программирования симплекс-методом

Постановка задачи линейного программирования в общем виде такова. Дана линейная целевая функция от n переменных

$$a_1 x_1 + a_2 x_2 + \ldots + a_n x_n.$$

Найти максимум (минимум) ее $\max\limits_{x_1, x_2, ..., x_n} (a_1x_1 + a_2x_2 + ... + a_nx_n)$ при условии, что удовлетворяются следующие m неравенств: $b_{i1}x_1 + b_{i2}x_2 + ... + b_{in}x_n \{<, \le, =, \ge, >\} b_i$, $i = 1, ..., m$.

Решение заключается в поиске оптимальных переменных, удовлетворяющих решению задачи. Неравенства определяют в пространстве переменных выпуклый многогранник, называемый симплексом, а допустимые значения переменных – точку внутри или на краю симплекса. Оптимальная точка находится в одной из вершин симплекса.

Симплекс-метод заключается в последовательном обходе вершин симплекса до тех пор, пока не попадется оптимальная вершина. Для сокращения пути обхода в текущей вершине делается пробный шаг движения точки по каждому исходящему из этой вершины ребру и выбирается ребро, максимально увеличивающее значение целевой функции при перемещении точки на единицу длины этого ребра. По этому ребру перемещаются в следующую вершину. Проверяется ее оптимальность и, если она не оптимальна, повторяется переход в следующую вершину. Вершина оптимальна, если при движении точки по любому исходящему из нее ребру значение целевой функции уменьшается (при поиске максимума) или, наоборот, увеличивается, если ищется минимум целевой функции.

Опишем методику решения на следующем примере.

Дано:

$$y = x_1 + 2x_2 . \tag{24}$$

Найти $y_{\max}$ при условии, что

$$\begin{cases} x_1 \le 7; \\ x_1 + x_2 \le 9; \\ x_1 + 3x_2 \le 15; \\ x_2 \le 4. \end{cases} \tag{25}$$

На рис. 4 показан симплекс, соответствующий условиям (25), и прямая GH, соответствующая уравнению целевой функции (24), если ее значение равно двум. Оптимальной будет вершина D, так как она является точкой касания прямой GH к симплексу при наибольшем удалении

ее от начала координат. В ней достигается наибольшее значение целевой функции при соблюдении условий (25).

Для обхода вершин симплекса необходимо преобразовать неравенства в равенства введением дополнительных переменных. Целевая функция y переименовывается в x_0 и записывается

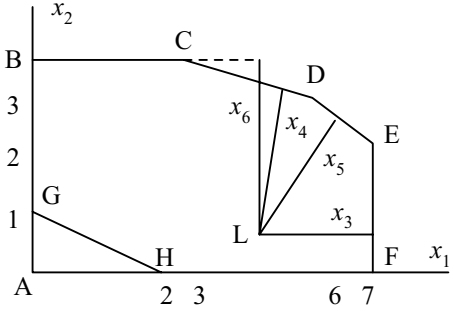

Рис. 4. Графическое решение задачи линейного программирования

$$\begin{cases} x_0 - x_1 - 2x_2 = 0; \\ x_1 + x_3 = 7; \\ x_1 + 3x_2 + x_4 = 15; \\ x_1 + x_2 + x_5 = 9; \\ x_2 + x_6 = 4 \end{cases} \qquad (26)$$

в виде равенства в системе (26) вместе с неравенствами (25), преобразованными в равенства введением в каждое по одной положительной дополнительной переменной. Геометрический смысл дополнительных переменных x_3, ..., x_6 становится понятным при взгляде на точку L на рис. 4. Значения x_3, ..., x_6 есть расстояния от точки L до прямых BC, CD, DE и EF. Если одно из них равно нулю, то точка L лежит на соответствующей прямой, если два смежных расстояния равны нулю, то точка L лежит в вершине симплекса, на пересечении прямых, которым принадлежит точка L. Отсюда следует важный вывод: нулевые значения пары дополнительных переменных, соответствующих смежным ребрам, определяют вершину симплекса, из которой эти ребра исходят. Назовем такую пару переменных с нулевыми значениями базисом, хотя общепринято называть базисом совокупность остальных переменных с ненулевыми значениями.

Из рис. 4 видно, что вершинам A, B, C, D, E, F соответствуют базисы (x_1, x_2), (x_1, x_6), (x_6, x_4), (x_4, x_5), (x_5, x_3), (x_3, x_2). В 3-мерном пространстве базисом будет совокупность нулевых значений трех переменных, а в N-мерном – N переменных.

Следующим важным понятием является каноническая форма системы (26). Каноническая форма системы равенств для одного базиса не может быть канонической для другого базиса. Это следует из ее определения. Форма системы (26) – каноническая, если в целевом уравнении системы (26) кроме x_0 имеются только базисные переменные, а в остальных равенствах кроме базисных – только по одной небазисной переменной. Из определения следует, что система (26) – каноническая для базиса (x_1, x_2). Каноническая форма нужна для определения того, являются ли текущий базис и соответствующая ему вершина симплекса оптимальной или нет, а также для определения ребра, по которому нужно перейти к следующей вершине в случае неоптимальности текущей вершины.

Вершина оптимальна (при поиске максимума целевой функции), если в целевом уравнении канонической системы перед коэффициентами всех базисных переменных стоит знак плюс. Это означает, что в оптимальной вершине по какому бы исходящему из нее ребру мы ни сделали пробный шаг, значение целевой функции будет уменьшаться, что говорит о ее максимальном значении в оптимальной вершине. Если вершина неоптимальна, то ребру с максимальным увеличением целевой функции при переходе по нему в следующую вершину соответствует базисная переменная с максимальным абсолютным значением ее отрицательного коэффициента в целевом уравнении. Сам переход точки по ребру в следующую вершину соответствует удалению этой базисной переменной из базиса и включению в него другой переменной, значение которой в новой вершине становится равным нулю.

Для определения такой переменной для каждого равенства определяется отношение его правой части к коэффициенту удаляемой из базиса переменной в левой части равенства. Равенство с минимальным значением этого отношения называется опорным, а его дополнительная переменная принимает нулевое значение в новой вершине и входит в состав базисных переменных. Опорное равенство переходит в каноническую форму системы (26) для новой вершины без изменения, а целевая функция и остальные равенства должны быть умножены на такой коэффициент, что при сложении с опорным равенством в них исчезла прежняя базисная переменная, которая в новой вершине была удалена из базиса.

Таким образом получают каноническую форму системы (26) для новой вершины и по ее виду делают вывод об оптимальности или неопти-

мальности этой вершины, и в случае неоптимальности принимают решение об удалении из базиса очередной переменной с максимальным значением отрицательного коэффициента, и цикл повторяется, пока не встретится оптимальная вершина.

Анализ системы (26) для приведенного примера показывает, что базис (x_1, x_2) и соответствующая вершина А – неоптимальны, значение целевой функции $x_0 = 0$ в силу равенства нулю базисных переменных x_2, а удаляемой из базиса переменной является x_2, так как при ней наибольший по модулю коэффициент – со знаком минус. Значение, с которым x_2 удаляется из базиса, равно четырем. Оно определяется в соответствии с описанным выше правилом

$$x_2 = \min (7/0,\ 15/3,\ 9/1,\ 4/1) = 4. \qquad (27)$$

При этом опорным равенством становится пятое, а базисной переменной – x_6. С его помощью система (26) алгебраически преобразуется в каноническую систему для следующего базиса (x_1, x_6) и соответствующей ему вершины В. Для преобразования системы (26) в новую каноническую форму нужно с его помощью удалить из остальных уравнений и целевого уравнения исключаемую из базиса (x_1, x_2) переменную x_2, а само опорное уравнение оставить без изменения. Для этого уравнение с удаляемой переменной и опорное умножаются на такие коэффициенты, чтобы при их сложении члены с удаляемой переменной взаимно исключались. В данном случае для исключения x_2 из целевого уравнения надо сложить с ним удвоенное опорное. Для исключения его из третьего уравнения надо из него вычесть утроенное опорное, а для четвертого надо из него вычесть одно опорное. В результате этого получим каноническую форму системы для вершины В и соответствующего ей базиса (x_1, x_6)

$$\begin{cases} x_0 - x_1 - 2x_6 = 8; \\ x_1 + x_3 = 7; \\ x_1 + x_4 - 3x_6 = 3; \\ x_1 + x_5 - x_6 = 5; \\ x_2 + x_6 = 4. \end{cases} \qquad (28)$$

Из анализа системы (28) следует, что вершина В неоптимальная, значение целевой функции в ней равно 8, удалять из базиса нужно переменную x_1 со значением

$$x_1 = \min (7/1, 3/1, 5/1, 4/0) = 3,$$

а опорным является третье уравнение. Для удаления x_1 из целевого уравнения оно складывается с опорным, а для удаления x_1 из второго и четвертого уравнений опорное вычитается из них. Получаем каноническую систему для базиса (x_6, x_4) и вершины C следующего вида:

$$\begin{cases} x_0 + x_4 - x_6 = 11; \\ x_3 - x_4 + 3x_6 = 4; \\ x_1 + x_4 - 3x_6 = 3; \\ x_4 + x_5 + 2x_6 = 2; \\ x_2 + x_6 = 4. \end{cases} \qquad (29)$$

Из анализа системы (29) следует, что вершина C – неоптимальная, значение целевой функции в ней равно 11, удалять из базиса нужно переменную x_6 со значением

$$x_6 = \min (4/3, -, 2/2, 4/1) = 1,$$

а опорным является четвертое уравнение. Для удаления x_6 из целевого уравнения оно складывается с половиной опорного, для удаления его из второго и третьего уравнений три опорных, соответственно, вычитаются из двух вторых и складываются с двумя третьими, для удаления x_6 из пятого уравнения опорное вычитается из двух пятых уравнений. Получаем каноническую систему для базиса (x_4, x_5) и вершины D следующего вида:

$$\begin{cases} x_0 + 0,5x_4 + 0,5x_5 = 12; \\ x_3 + x_4 - 3x_5 = 2; \\ 2x_1 - x_4 + 3x_5 = 12; \\ x_4 + x_5 + 2x_6 = 2; \\ x_2 + x_4 - x_5 = 6. \end{cases} \qquad (30)$$

Из анализа системы (30) следует, что вершина D – оптимальная, значение целевой функции в ней равно 12, удалять из базиса ничего не нужно. Из третьего и пятого следует, что координаты x_1 и x_2 оптимальной вершины D равны, соответственно, 6 и 3. Решение завершено.

12. Решение целочисленных задач линейного программирования методом отсечений (теоретическое обоснование)

Пусть решена задача линейного программирования симплекс-методом и получена каноническая форма целевой функции и ограничительных равенств для оптимального базиса, т. е. оптимальной вершины симплекса, и некоторые небазисные переменные x_{k_j} из исходной постановки задачи оказались нецелочисленными:

$$x_0 + c_{i_1} x_{i_1} + c_{i_2} x_{i_2} + \ldots + c_{i_n} x_{i_n} = c; \tag{31}$$

$$x_{k_j} + a_{ji_1} x_{i_1} + a_{ji_2} x_{i_2} + \ldots + a_{ji_n} x_{i_n} = b_{k_j}, \quad j = 1, \ldots, m; \tag{32}$$

$$x_{k_j} \geq 0 \text{ и должны быть целыми.} \tag{33}$$

Обоснование метода.

Выбираем нецелочисленное x_{k_s} ,

$$x_{k_s} + a_{si_1} x_{i_1} + a_{si_2} x_{i_2} + \ldots + a_{si_n} x_{i_n} = b_{k_s}, \tag{34}$$

в котором некоторые a_{si_t} и b_{k_s} – дробные.

Обозначим через [D] – целую, а через [f] – дробную части рационального числа.

Из условий $x_{k_s} \geq 0$ и $[a_{si_t}] \leq a_{si_t}$ следует

$$x_{k_s} + [a_{si_1}] x_{i_1} + [a_{si_2}] x_{i_2} + \ldots + [a_{si_n}] x_{i_n} \leq b_{k_s}. \tag{35}$$

Поскольку $x_{i_1}, \ldots, x_{i_n}$ и x_{k_s} – целые, то и $x_{k_s} + [a_{si_1}] x_{i_1} + [a_{si_2}] x_{i_2} + \ldots + [a_{si_n}] x_{i_n}$ – целое.

Тогда

$$x_{k_s} + [a_{si_1}] x_{i_1} + [a_{si_2}] x_{i_2} + \ldots + [a_{si_n}] x_{i_n} \leq [b_{k_s}]. \tag{36}$$

Добавим в левую часть (36) дополнительную переменную *x*, тогда

$$x_{k_s} + [a_{si_1}] x_{i_1} + [a_{si_2}] x_{i_2} + \ldots + [a_{si_n}] x_{i_n} + x = [b_{k_s}]. \tag{37}$$

Уравнение (37) получено алгебраическими операциями из уравнений (32), поэтому любое решение для системы (32) удовлетворяет и (37). Следовательно, уравнение (37) надо добавить в систему (32) и решать ее симплекс-методом для получения нового решения, учитывающего ограничение на целочисленность переменных и определяемого уравнением (37).

Для упрощения решения преобразуем (37) в уравнение, выраженное через дробные части входящих в него коэффициентов a_{si_t} и свободного члена b_{k_s}, вычитанием (11) из (8). Получим

$$(-f_{si_1})x_{i_1} + (-f_{si_2})x_{i_2} + ... + (-f_{si_n})x_{i_n} = -f_{k_s} \ , \qquad (38)$$

где

$$f_{si_t} = a_{si_t} - [a_{si_t}], \ \ f_{k_s} = b_{k_s} - [b_{k_s}]. \qquad (39)$$

Алгоритм отсечения

1. Найти оптимальное решение задачи с целочисленной функцией (31) и линейными ограничениями (32), не учитывая (33), но требуя положительности x_{k_j}.

2. Прекратить вычисления, если текущее решение – целочисленное. В противном случае выбрать дробную небазисную переменную, составить ограничение (38) из уравнения, содержащего эту небазисную переменную в текущем оптимальном решении. Добавить к исходной задаче линейного программирования 2 это новое ограничение и вернуться к шагу 1.

13. Решение целочисленных задач линейного программирования методом отсечений (пример решения)

Рассмотрим задачу: максимизировать $21x_1 + 11x_2$ при ограничении

$$7x_1 + 4x_2 + x_3 = 13, \qquad (40)$$

где x_1, x_2, x_3 – неотрицательные целые. Переменная x_3 – дополняющая, поскольку не входит в целевую функцию. Выразим целевую в соответствии с принятой схемой

$$x_0 - 21x_1 - 11x_2 = 0. \qquad (41)$$

Можно показать, что, выполнив шаг 1, получаем решение задачи в виде

$$x_0 + x_2 + 3x_3 = 39;$$

$$x_1 + 4/7x_2 + 1/7x_3 = 1 + 6/7. \quad (42)$$

Поскольку значение x_1 – дробное, нужно добавить отсекающее ограничение с коэффициентами, равными дробным частям коэффициентов из равенства (3), взятых с отрицательным знаком:

$$-4/7x_2 - 1/7x_3 + x_4 = -6/7, \quad (43)$$

где x_4 – вторая дополняющая переменная.

Выполняя шаг 2, решаем задачу линейного программирования (42), дополнив ее ограничением (43). Новое оптимальное решение выражается уравнениями

$$\begin{aligned} x_0 &+ (2+3/4)x_3 + (1+3/4)\ x_4 = (37+1/2); \\ x_1 &+ x_4 = 1; \\ x_2 &+ 1/4x_3 - (1 + 3/4)x_4 = -1/2. \end{aligned} \quad (44)$$

Поскольку x_2 – дробное, составим новое отсекающее уравнение, пользуясь дробными коэффициентами последней строки из решения (44):

$$-1/4x_3 - 1/4x_4 + x_5 = -1/2. \quad (45)$$

Выполняя шаг 3, решаем задачу линейного программирования (42), дополнив ее ограничением (45). Новое оптимальное решение выражается уравнениями

$$\begin{aligned} x_0 + x_1 &+ 11x_5 = 33; \\ -x_1 + x_3 &+ x_5 = 1; \\ 2x_1 + x_2 &+ x_5 = 3; \\ x_1 + x_4 &= 1. \end{aligned} \quad (46)$$

Поскольку полученное решение $x_1 = 0$, $x_2 = 3$ и $x_3 = 1$ – целочисленное, поставленная задача является решенной.

14. Метод ветвей и границ

Это мощный метод нелинейной оптимизации, дающий точное решение. Используется для алгоритмических целевых функций $f(x)$, где $x \in A$.

Для определения $\min\limits_{x \in A} f(x)$ находят нижнюю границу $f(x)$, которую обозначим через $G(f(x), A)$. Она должна обладать следующими свойствами:

$$G\left(f(x), A\right) \le \min_{x \in A} f(x); \tag{47}$$

$$G\left(f(x), A_1\right) \ge G\left(f(x), A\right),\ A_1 \subset A; \tag{48}$$

$$G\left(f(x), A_1\right) = \min_{x = A_1} f(x). \tag{49}$$

Алгоритм решения задачи следующий.

1. Множество A разбивают на непересекающиеся два A_1 и A_2, находят нижние границы $G(A_1)$ и $G(A_2)$ и выбирают наименьшую из них.

2. Если соответствующее подмножество состоит из одного элемента (решения), то решение заканчивается. В противном случае переходят к п. 1, в котором разбиению на два непересекающиеся подмножества подвергается выбранное с наименьшей нижней границей.

На рис. 5 показана функция $f(x)$ с областью определения (a, b). Требуется найти ее минимум $f_{\min}(x)$ и соответствующее значение аргумента x_{opt}. Прежде всего, на основании анализа выражения, определяющего функцию, устанавливается выражение для границы $G(a, b)$, зависящей от области определения (a, b) и имеющей значение, не большее, чем все значения функции $f(x) \le G(a, b)$ в области ее определения. Далее вся область (a, b) делится на несколько, чаще две, равные и непересекающиеся области (a, c) и (c, b). Для этих областей определяются две границы $G(a, c)$ и $G(c, b)$. Так как $G(c, b) < G(a, c)$, а минимальные значения функции в этих областях находятся рядом с границами и несколько выше их, то, вероятно, минимум функции в области (c, b) $f_{\min(a, b)}(x)$ также меньше минимума функции в области (a, c) $f_{\min(a, c)}(x)$:

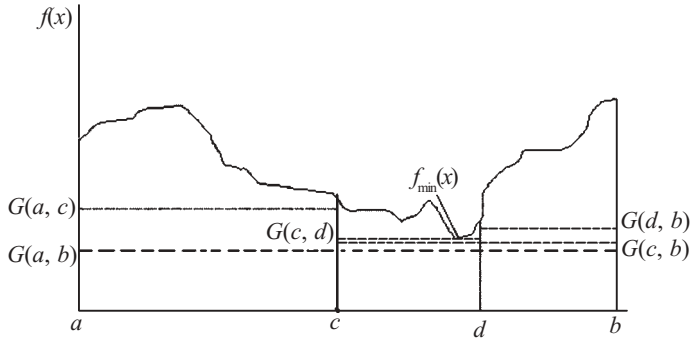

Рис. 5. Поиск $f_{\min}(x)$ методом ветвей и границ

$$f_{\min(c,\ b)}(x) \le f_{\min(a,\ c)}(x). \qquad (50)$$

Чем более точно граница отслеживает минимум функции в ее области определения, тем вероятнее соотношение (50). На основании этой информации делается вывод о расположении глобального минимума в области (c, b). Следовательно, для дальнейшего поиска расположения этого минимума в области (c, b) выполняется ее разделение на равные и непересекающиеся области (c, d) и (d, b) и определение в них нижних границ $G(c, d)$ и $G(d, b)$. На основании сравнения уже трех границ $G(a, c)$, $G(c, d)$ и $G(d, b)$ в еще нерасчлененных областях делается вывод, что глобальный минимум, вероятно, расположен в области (c, d). Далее производится ее разделение и вычисление границ в полученных подобластях и т. д. Тонкость заключается в том, что при выборе подобласти с вероятным местонахождением глобального минимума функции нужно сравнивать нижние границы для всех еще нерасчлененных подобластей. Все это повторяется до тех пор, пока не определится подобласть с двумя свойствами: она не может быть расчленена на подобласти (имеет только одно значение аргумента x') и значение границы в ней $G(x')$, равное $f(x')$, будет меньше или равно значению границ $G(A)$ во всех еще нерасчлененных подобластях определения A функции $f(x)$. Поскольку локальные минимумы функций $f_{\min(A)}(x) \ge G(A)$, то, по определению минимума, значение $f(x')$ является глобальным минимумом $f_{\min}(x)$, а значение x' является оптимальным значением x_{opt}.

15. Целочисленное решение задач линейного программирования методом ветвей и границ

В данном случае границей максимального значения целевой функции с целочисленным решением служит максимальное значение этой функции с нецелочисленным решением, получаемое симплекс-методом. Далее из всего множества допустимых решений исключается подмножество решений с дробной частью оптимального нецелочисленного значения одной из переменных. При этом происходит деление всего множества решений на два непересекающихся, в которых вычисляется граница повторением нецелочисленного решения симплекс-методом. Далее сравниваются границы и продолжается поиск целочисленного решения повторением деления выбранного подмножества решений и вычислением в образующихся подмножествах вер-

хних границ. Для пояснения сказанного рассмотрим систему (51). Найти $x_1 \geq 0$ и $x_2 \geq 0$ такие, что

$$x_0 = x_1 + x_2 = x_{0\max};$$

$$\begin{aligned} x_1 + 9/16\, x_2 &\leq 51/16; \\ -2\, x_1 + x_2 &\leq 1/3; \\ x_1 \text{ и } x_2 &- \text{ целые.} \end{aligned} \qquad (51)$$

Симплекс, удовлетворяющий условиям (51), приведен на рис. 6. Решения, удовлетворяющие четвертому условию, показаны точками. Без этого условия решением задачи является точка A с координатами (3/2, 10/3), в которой достигается максимум x_0, равный 29/6. Ближайшими целочисленными значениями для x_1 являются 1 и 2. Рассмотрим решение задачи (51) сначала при условии $x_1 <= 1$, а потом при условии $x_1 >= 2$. Тем самым симплекс S разбивается на два непересекающихся симплекса S_1 и S_2 (рис. 7). В обоих симплексах без учета целочисленности решениями задачи будут соответственно точки C и B с координатами (1, 7/3) и (2, 23/9) и максимальными значениями x_0, равными 10/3 и 41/9.

Значение $x_{0\max}$ в симплексе S_1 с учетом целочисленности x_1 и x_2 не может быть больше 10/3, а в симплексе S_2 – может, хотя и не превысит значение 41/9. Поэтому дальнейший поиск $x_{0\max}$ осуществляется в S_2.

Ближайшими к оптимальному значению $x_2 = 23/9$ являются целочисленные точки со значениями $x_2 = 2$ и $x_2 = 3$. Поэтому рассматривается решение задачи (51) с учетом (помимо дополнительного условия $x_1 >= 2$) сначала условия $x_2 <= 2$, а потом $x_2 >= 3$. Тем самым симплекс S_2 разбивается на симплекс S_{21} и несуществующий симплекс S_{22} (рис. 8). В первом оптимальной является вершина D с координатами $x_1 = 33/16$, $x_2 = 2$

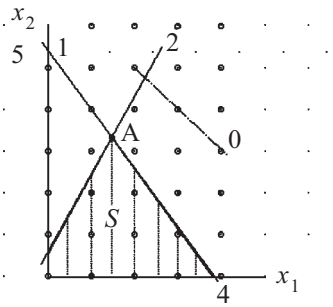

Рис. 6. Постановка задачи

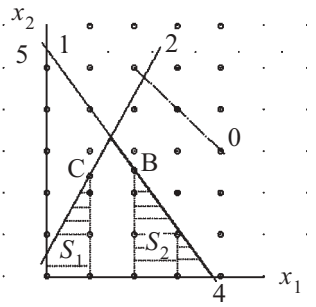

Рис. 7. Разбиение симплекса S

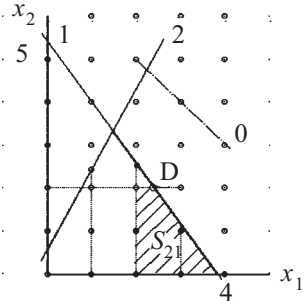

Рис. 8. Разбиение симплекса S_2

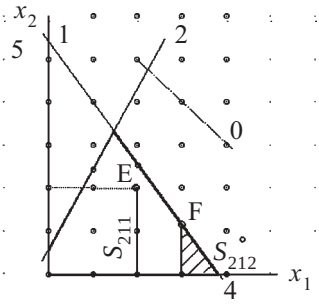

Рис. 9. Разбиение симплекса S_{21}

и значением целевой функции $x_{0max} = 61/16$. Переменная x_2 является целочисленной. Ближайшими к оптимальному $x_1 = 33/16$ являются целочисленные значения $x_1 = 1$ и $x_2 = 2$. Поэтому задача (51) решается с учетом сначала $x_1 <= 2$, а потом $x_1 >= 3$. Симплекс S_{21} разбивается при этом на S_{211} и S_{212} (рис. 9). В первом оптимальной вершиной является F, а во втором – вершина E с координатами соответственно (2, 2) и (3, 1) и одинаковым значением x_{0max}, равным 4. Поскольку полученное значение x_{0max} для целочисленных переменных x_1 и x_2 больше, чем оценки x_{0max} в нерассмотренных симплексах, то оно тем более больше самих значений x_{0max} в этих симплексах. Это означает, что последнее значение x_{0max} является наибольшим для всего симплекса S и, следовательно, является окончательным решением задачи. На рис. 10 приведено традиционное изображение графа решения методом ветвей и границ рассмотренной задачи, в котором наглядно видно, что последнее значение x_{0max} больше, чем оценка x_{0max} для любой неразвернутой вершины графа. Если бы нашлась такая, у которой оценка x_{0max} больше, чем найденное значение x_{0max}, то пришлось бы вернуться к соответствующему симплексу и искать в нем решение задачи.

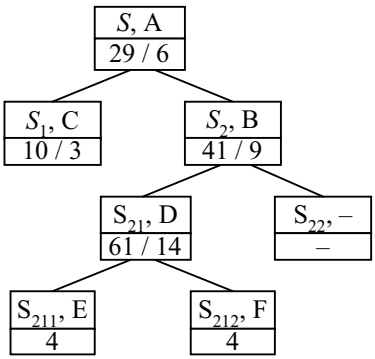

Рис. 10. Граф решения задачи методом ветвей и границ

В реальных задачах такие возвраты приходится выполнять довольно часто, поэтому решение задачи затягивается. В большинстве крупных машинных программ, реализующих частично целочисленные алгоритмы, используется

критерий останова, гарантирующий отклонение результата $x_{0\max}$ на величину, не более наперед заданной.

16. Бинарное решение задач линейного программирования методом ветвей и частичных решений (теоретическое обоснование)

Выбор оборудования и распределение операций по рабочим местам решаются с помощью задач линейного программирования, в которых переменные являются бинарными, т. е. принимают двоичные значения. В общем виде задача ставится так:

максимизировать

$$\sum_{j=1}^{n} c_j x_j \tag{52}$$

при ограничениях

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \ i = 1, 2, \ldots, m, \tag{53}$$

где условия целочисленности сведены к

$$x_j = \{0, 1\}, j = 1, 2, 3, \ldots, n. \tag{54}$$

Если учитывать лишь условие (54), то существует 2^n возможных выбора значений $(x_1, x_2, \ldots, x_n)$. Разумеется, многие из них недопустимы из-за линейных ограничений (53), и лишь очень небольшое число являются допустимыми. Рассмотрим некоторое подмножество x_j, в котором каждому x_j поставлено в соответствие определенное числовое значение (нуль или единица). Такое подмножество называется частичным решением. Переменные, не вошедшие в частичное решение, называются свободными. Любой конкретный набор числовых значений свободных переменных называется дополнением соответствующего частичного решения. Если частичное решение содержит s переменных, то существует 2^{n-s} дополнения. В рассматриваемом алгоритме каждому частичному решению соответствует неразвернутая вершина дерева решений. Возможные дополнения для некоторой неразвернутой вершины порождают дополняющие ветви. Если к некоторому моменту получено

несколько частичных решений, а также известна некоторая оценка оптимального решения и зафиксировано допустимое решение, дающее эту оценку, то для каждого частичного решения можно сделать проверку на существование допустимого дополнения, для которого значение целевой функции превышает текущую нижнюю оценку. Если такого дополнения нет, то рассматриваемое частичное решение считается прозондированным. Для него нет необходимости дальнейшего ветвления дерева решений. При этом из дальнейшего рассмотрения исключаются 2^{n-s} возможных назначения свободных переменных. Если же такие дополнения есть, то при заданном частичном решении значения остальных переменных должны выбираться так, чтобы дополнение этого решения было оптимальным. Существует несколько способов определения бесперспективности дополнения частичного решения. Рассмотрим один из простых для задачи с таким ограничением:

$$-x_1 + 2x_2 - 6x_3 + 4x_4 - x_5 \le 0. \qquad (55)$$

Для частичного решения $(x_1, x_3, x_4) = (1, 0, 1)$ ни одно дополнение не допустимо по условию (55). Пусть целевая функция для этого примера описывается уравнением $2x_1 + x_2 + 5x_3 + 3x_4 - x_5$, а текущая оценка целевой функции $x_0^t = 7$. Чтобы улучшить эту оценку, дополнение должно удовлетворять ограничению, полученному из выражения для целевой функции и оценки $x_0^t = 7$:

$$2x_1 + x_2 + 5x_3 + 3x_4 - x_5 \ge 8 \text{ или } -2x_1 - x_2 - 5x_3 - 3x_4 + x_5 \le -8. \qquad (56)$$

Последняя форма этого неравенства более удобна для проверки дополнений, так как в нем используется тот же знак неравенства, что и в каждом ограничительном неравенстве. Итак, для частичного решения $(x_2, x_3, x_5) = (1, 0, 1)$ ни одно дополнение не допустимо по условию (56).

Смысл рассмотренных примеров в следующем. Для заданного частичного решения представим линейные ограничения в виде

$$\sum_{\substack{\text{По всем}\\ \text{свободным}\\ \text{переменным}}} a_{ij}x_j \le b_i - \sum_{\substack{\text{По всем}\\ \text{частичным}\\ \text{переменным}}} a_{ij}x_j,\ i = 0, 1, 2, \ldots, m, \qquad (57)$$

где каждой переменной x_j в частичном решении поставлено в соответствие определенное значение, входящее под знак суммы в правой части неравенства (57), а коэффициенты в ограничении при $i = 0$ равны $a_{0j} = -c_j$ и $b_0 = -x_0^t - 1$. Тогда, если для заданного частичного решения

$$\sum_{\substack{\text{По всем}\\ \text{свободным}\\ \text{переменным}}} \min(a_{ij}, 0) > b_i - \sum_{\substack{\text{По всем}\\ \text{частичным}\\ \text{переменным}}} a_{ij} x_j$$

для любого $i = 0, 1, 2, \ldots, m.$ (58)

то допустимого дополнения, которому соответствует значение целевой функции, превосходящее нижнюю оценку x_0^t, не существует.

Член в левой части неравенства (58) – сумма всех отрицательных коэффициентов при свободных переменных. Если эта сумма больше правой части неравенства, то даже полагая $x_j = 1$, для каждой свободной переменной с $a_{ij} < 0$ невозможно выполнить i-е ограничение (57).

Другим важным моментом является то, что при заданном частичном решении иногда удается определить, какое значение должна иметь свободная переменная при любом допустимом дополнении в случае, когда значение целевой функции превосходит текущую нижнюю оценку. Например, для частичного решения $(x_1, x_2) = (1, 1)$ для задачи с ограничением

$$x_1 - x_2 + 2x_3 - x_4 \le 0 \quad (59)$$

в любом дополнении должно выполняться условие $x_3 = 0$, при котором оно допустимо по ограничению (59). Можно сформулировать общее правило.

Если для частичного решения и свободной переменной x_k

$$\sum_{\substack{\text{По всем}\\ \text{свободным}\\ \text{переменным}}} \min(a_{ij}, 0) + |a_{ik}| > b_i - \sum_{\substack{\text{По всем}\\ \text{частичным}\\ \text{переменным}}} a_{ij} x_j \quad \text{для любого } i, \quad (60)$$

то для $a_{ik} > 0$ переменная $x_k = 0$, а для $a_{ik} < 0$ переменная $x_k = 1$.

Перейдем к описанию алгоритма. На итерации t выполняются следующие шаги.

Шаг 1. Прекратить вычисления, если основной список частичных решений пуст. Иначе выбрать частичное решение из основного списка и вычеркнуть его из списка.

Шаг 2. Если можно найти свободные переменные, которые должны иметь определенные значения при любом допустимом дополнении, когда значение целевой функции превосходит текущую ее оценку x_0^t, то расширить частичное решение включением значений найденных свобод-

ных переменных. Если можно установить, что не существует допустимого дополнения со значением целевой функции, превосходящим оценку x_0^t, то положить оценку $x_0^{t+1} = x_0^t$ и вернуться к шагу 1. Иначе перейти к шагу 3.

Шаг 3. Если расширенное частичное решение – полное (содержит все *n* переменных), то зафиксировать его, положить оценку x_0^{t+1} равной значению целевой функции и вернуться к шагу 1. Иначе перейти к шагу 4.

Шаг 4. Выбрать любую свободную переменную x_k, не входящую в расширенное частичное решение. Внести две задачи в основной список. В одной из них положить $x_k = 0$, в другой $x_k = 1$. Положить $x_0^{t+1} = x_0^t$ и вернуться к шагу 1. После опустошения основного списка частичных решений полученная оценка x_0^t является оптимальной.

17. Бинарное решение задач линейного программирования методом ветвей и частичных решений (пример решения)

Максимизировать

$$3x_1 + 6x_2 + 3x_3 + 6x_4 + 13x_5 \tag{61}$$

при ограничениях

$$-3x_1 - 6x_2 + 6x_3 + 12x_4 + 7x_5 \le 8,\ i = 1; \tag{62}$$

$$6x_1 + 12x_2 - 3x_3 - 6x_4 + 7x_5 \le 8,\ i = 2; \tag{63}$$

$$x_j = \{0, 1\},\ j = 1, 2, \ldots, 5. \tag{64}$$

Примем $x_0^1 = 0$, что соответствует допустимому решению, в котором все x_j равны нулю ($x_j = 0$). Введем в основной список два частичных решения (ЧР): ЧР1 ($x_5 = 1$) и ЧР2 ($x_5 = 0$). Выбор x_5 объясняется тем, что при $x_5 = 1$ приращение оценки целевой функции является максимальным.

На шаге 1 рассмотрим ЧР1. Допустимое дополнение со значением целевой функции (ЦФ), превосходящим оценку $x_0^1 = 0$, должно удовлетворять условиям (62) – (64), а также условию

$$-3x_1 - 6x_2 - 3x_3 - 6x_4 - 13x_5 \le -1,\ i = 0. \tag{65}$$

На шаге 2 найдем свободные переменные, значения которых определяются только соблюдением условий (62)–(65). Применим условие (60).

Для $i = 1$ и $k = 4$ условие (60) примет вид

$$-3-6+12>8-7x_5=8-7=1, \quad (66)$$

а для $i = 2$ и $k = 2$ условие (60) примет тот же вид

$$-3-6+12>8-7x_5=8-7=1. \quad (67)$$

Из формулы (66) следует, что только при $x_4 = 1$ может быть нарушено условие (62), а из формулы (67) следует, что только при $x_2 = 1$ может быть нарушено условие (63). Следовательно, для ЧР1 во всех допустимых дополнениях x_2 и x_4 должны быть равны нулю. Из ЧР1 автоматически следует ЧР11: $(x_2, x_4, x_5) = (0, 0, 1)$. Для ЧР11 можно снова применить условие (60):

для $i = 1$ и $k = 3$ имеем

$$-3+6>8-7=1; \quad (68)$$

для $i = 2$ и $k = 1$ имеем

$$6>8-7=1. \quad (69)$$

Следовательно, повторное применение условия (60) позволяет получить полное решение (ПР) ПР1 $(x_1, x_2, x_3, x_4, x_5) = (0, 0, 0, 0, 1)$ с текущей оценкой максимума ЦФ $x_0^1 = 13$, удовлетворяющее условиям (62) и (63).

Полагаем $x_0^2 = x_0^1 = 13$, возвращаемся к шагу 1 и рассматриваем ЧР2, где $x_5 = 0$. Применение условия (60) для $i = 0, 1, 2$ не приводит к расширению ЧР2. Применение же условия (58) для $i = 0, 1, 2$ не исключает появление допустимого дополнения со значением ЦФ, превосходящим 13. Переходим к шагу 3, а затем к шагу 4, поскольку ЧР2 не является полным. На шаге 4 выбираем переменную x_1 и расчленяем ЧР2 на ЧР21: $(x_1, x_5) = (1, 0)$ и ЧР22: $(x_1, x_5) = (0, 0)$.

Возвращаясь к шагу 1 при $x_0^3 = 13$, рассматриваем ЧР21. Применение проверки (60) для $i = 0, 1$ не определяет однозначно ни одну свободную переменную. Для $i = 2$ проверка (60) дает $x_2 = 0$ для любого допустимого дополнения:

$$12x_2-3-6>8-6=2 \text{ для } i=2,\ x_2=1. \quad (70)$$

Расширяя ЧР21 до ЧР211: $(x_5, x_1, x_2) = (0, 1, 1)$, и повторно применяя (60) для $i = 0$, можно убедиться, что допустимого дополнения не существует:

$$-3(x_3=1)-6(x_4=1)\le -13+3=-10. \quad (71)$$

Поэтому шаг 2 заканчивается, и нужно вернуться к шагу 1, приняв $x_0^4 = 13$ и рассматривая ЧР22 $(x_1, x_5) = (0, 0)$. Проверка (60) для $i = 0$ показывает, что для любого допустимого дополнения $x_3 = 1$

$$-6(x_2 = 1) - 3(x_3 = 1) - 6(x_4 = 1) \le -13. \quad (72)$$

Применяя условие (60) для $i = 1, 2$, получаем, что $x_4 = 0$ и $x_2 = 0$: для $i = 1$ имеем

$$-6(x_2 = 1) + 12(x_4 = 0) \le 8 - 6(x_3 = 1) = 2; \quad (73)$$

для $i = 2$ имеем

$$+12(x_2 = 0) - 6(x_4 = 1) \le 8 + 3(x_3 = 1). \quad (74)$$

В результате проверки (58) для $i = 0$ получаем, что не существует допустимого дополнения со значением ЦФ, превосходящим $x_0^4 = 13$. Возвращаясь к шагу 1, прекращаем вычисления, так как все ЧР из дерева ветвей исчерпаны. Оптимальное значение целевой функции равно 13.

18. Оптимизация запуска процессов обработки с одинаковыми маршрутами динамическим программированием по векторному критерию

Рассмотрим применение метода динамического программирования на примере обработки пяти деталей a, b, c, d, e на пяти станках A, B, C, D, E (табл. 3).

Таблица 3

Время выполнения операций пяти деталей на пяти станках

Деталь	Время выполнения операции на станке				
	A	B	C	D	E
a	2	5	3	4	6
b	4	3	5	4	5
c	5	4	3	6	3
d	4	3	6	4	4
e	3	6	4	5	4

Как видно из таблицы, маршруты обработки всех деталей одинаковы. Время окончания процессов обработки всех деталей T зависит от порядка запуска их обработки $T = T(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$, где σ_i – номер детали, запускаемой в обработку. Необходимо определить такой порядок обработки деталей $(\sigma_1^{opt}, \sigma_2^{opt}, \sigma_3^{opt}, \sigma_4^{opt}, \sigma_5^{opt})$, при котором $T(\sigma_1^{opt}, \sigma_2^{opt}, \sigma_3^{opt}, \sigma_4^{opt}, \sigma_5^{opt}) = T_{min}$.

На первом этапе для всех пар деталей производится сравнение времени их обработки для двух случаев обработки сначала первой детали (σ_1, σ_2), а потом – второй (σ_2, σ_1). Из каждой пары вариантов оставляется тот, для которого время завершения обработки обеих деталей на каждом станке меньше или равно соответствующему времени для другого варианта. В противном случае оставляются оба варианта для рассматриваемой пары деталей.

Рассмотрим пару деталей a, b. Ниже приведены табл. 4, 5, соответствующие обоим порядкам их запуска, полученные с помощью приложения Excel. Время завершения обработки первой детали равно времени окончания ее обработки на предыдущем станке плюс время обработки на текущем станке. Время завершения обработки второй детали равно максимуму времени окончания ее обработки на предыдущем станке и времени окончания обработки первой детали на текущем станке плюс время обработки второй детали на текущем станке.

Таблица 4

Время обработки деталей a, b

Деталь	Время выполнения и завершения операции на станке				
	A	B	C	D	E
a – выполнение	2	5	3	4	6
a – завершение	2	7	10	14	20
b – выполнение	4	3	5	4	5
b – завершение	6	10	15	19	25

Таблица 5

Время обработки деталей b, a

Деталь	Время выполнения и завершения операции на станке				
	A	B	C	D	E
a – выполнение	4	3	5	4	5
a – завершение	4	7	12	16	21
b – выполнение	2	5	3	4	6
b – завершение	6	12	15	20	27

Сравнение времени окончания обработки обеих деталей для двух порядков показывает, что это время меньше для порядка обработки сначала детали a, затем детали b (a, b). Произведя сравнение для всех пар деталей, получим следующие результаты (табл. 6).

На основании этих данных производится сравнение последовательностей обработки каждой пары и третьей детали. Оптимальные порядки запуска трех деталей приведены в табл. 7.

На основании этих данных производится сравнение последовательностей обработки каждой тройки и четвертой детали. Оптимальные порядки запуска четырех деталей приведены в табл. 8. На основании этих данных производится сравнение последовательностей обработки каждой четверки и пятой детали.

Оптимальные порядки запуска пяти деталей приведены в табл. 9. Сравнение времени завершения обработки пятерки деталей на станках показывает, что оптимальными являются два порядка запуска деталей: (a, b, c, d, e) и (a, c, d, e, b) с одинаковым временем окончания обработки на последнем станке, равным 38 условным единицам времени.

Для сравнения приведем полученную в Excel таблицу с одним из неоптимальных порядков следования запусков деталей (e, d, c, b, a) (табл. 10).

Таблица 6

Минимальное время окончания обработки пар деталей

Пары деталей	Время завершения операций на станке				
	A	B	C	D	E
ab	6	10	15	19	25
ac	7	11	14	20	23
ad	6	10	16	20	24
ae	5	13	17	22	26
bc	9	13	16	22	25
bd	8	11	18	22	26
be	7	13	17	22	26
cd	9	12	18	22	26
ce	8	15	19	24	28
de	7	13	17	22	26

Таблица 7

Минимальное время окончания обработки трех деталей

Тройки деталей	Время завершения операций на станке				
	A	B	C	D	E
abc	11	15	18	25	28
acd	11	14	20	24	28
ade	9	16	20	25	29
bcd	13	16	22	26	30
bde	11	17	22	27	31
cde	12	18	22	27	31

Таблица 8

Минимальное время окончания обработки четырех деталей

Группы деталей	Время завершения операций на станке				
	A	B	C	D	E
abcd	15	18	24	29	33
acde	14	20	24	29	33
bcde	16	22	26	31	35
cdeb	16	21	27	31	36

Таблица 9

Время обработки пяти деталей для оптимальных запусков

Группы деталей	Время завершения операций на станке				
	A	B	C	D	E
abcde	18	24	28	34	38
acdeb	18	23	29	33	38
bcdea	18	27	30	35	41

Таблица 10

Время обработки пяти деталей для неоптимальных запусков

Деталь	Время выполнения и завершения операции на станке				
	A	B	C	D	E
e – выполнение	3	6	4	5	4
e – завершение	3	9	13	18	22
d – выполнение	4	3	6	4	4
d – завершение	7	12	19	23	27
c – выполнение	5	4	3	6	3
c – завершение	12	16	22	29	32
b – выполнение	4	3	5	4	5
b – завершение	16	19	27	33	38
a – выполнение	2	5	3	4	6
a – завершение	18	24	30	37	44

Выигрыш во времени окончания всех процессов для оптимального запуска по сравнению с неоптимальным составляет $\delta T = 100(44 - 38)/38 = 15{,}8\ \%$.

19. Оптимизация запуска процессов обработки с неодинаковыми маршрутами методом ветвей и границ

Рассмотрим задачу оптимизации порядка запуска операций нескольких процессов с неодинаковыми маршрутами после распределения их на рабочих местах. Она решается методом ветвей и границ. На рис. 11 изображен граф, определяющий порядок прохождения станков A, B, C деталями a, b, c.

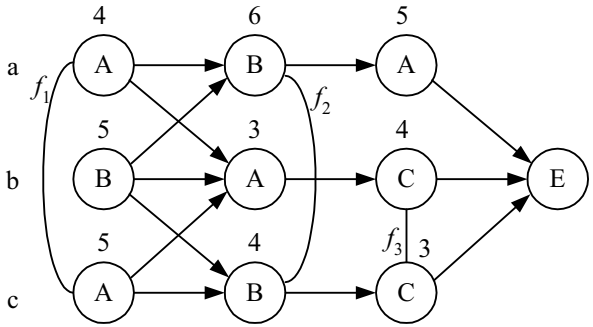

Рис. 11. Граф порядка обработки деталей a, b, c на станках A, B, C

Операции изображены кружочками, а порядок их выполнения – линиями со стрелками (дугами). Поскольку станков меньше, чем операций, операции aA и cA соединены линией без стрелок (ребром), означающей неодновременность их выполнения с пока еще неизвестным порядком. То же самое относится к парам операций aB и cB, а также к операциям bC и cC. В данном примере таких конфликтов три. Они порождают $2^3 = 8$ вариантов их разрешения. В реальных задачах конфликтов – десятки, и число их разрешения – это два в степени, равной числу конфликтов. Для всего множества решений (в данном случае 8) в качестве нижней границы можно взять время окончания всех процессов, в которых конфликтные операции выполняются одновременно. Полученное время будет меньше реального минимального, так как конфликтные операции должны выпол-

няться последовательно. Поэтому это время может быть нижней границей (НГ) искомого минимума. Для подсчета НГ воспользуемся приложением Excel. В табл. 11 приведены данные, соответствующие исходному графу (см. рис. 11). В табл. 11 вместо времени длительности операций записаны выражения для расчета времени их завершения.

Таблица 11

Время обработки деталей без учета конфликтных ситуаций

Деталь	Время выполнения и завершения операции на станке						
	A	B	A	B	A	C	E
a – выполнение	4			6	5		
a – завершение	4			11	16		
b – выполнение		5	3			4	
b – завершение		5	8			12	16
c – выполнение	5			4		3	
c – завершение	5			9		12	

Время завершения всех процессов, равное 16, является заниженным, так как в выражениях конфликтные операции считаются выполняемыми одновременно. Поэтому это время является нижней границей завершения процессов и одной и той же для всех восьми вариантов разрешения трех конфликтов. Далее начинаем заниматься разрешением первого конфликта. Он имеет два решения: *а*) операция aA выполняется перед операцией cA; *б*) операция aA выполняется после операции cA. Нижние границы для каждого решения могут измениться и быть неодинаковыми. Они постоянны для всех четырех вариантов разрешения двух остальных конфликтов. Перспективным для поиска минимума является то множество неразрешенных конфликтов, для которого нижняя граница меньше.

В табл. 12 и 13 приведено время завершения процессов для решений *а*) и *б*) первого конфликта.

Из анализа табл. 12 и 13 следует, что второй конфликт надо разрешать для случая выполнения операции aA перед операцией cA. Второй конфликт также имеет два решения: *а*) операция aB выполняется перед операцией cB; *б*) операция aB выполняется после операции cB. Каждый вариант будет характеризоваться своей нижней границей, постоянной для двух вариантов решения последнего конфликта.

Приведем результаты, полученные для вариантов *а*) и *б*) для второго конфликта (табл. 14, 15).

Из анализа табл. 14 и 15 следует, что третий конфликт надо разрешать для случая выполнения операции aB перед cB. Третий конф-

Таблица 12

Время обработки деталей с учетом выполнения aA перед cA

Деталь	Время выполнения и завершения операции на станке						
	A	B	A	B	A	C	E
a – выполнение	4			6	5		
a – завершение	9			15	20		
b – выполнение		5	3			4	
b – завершение		5	8			12	20
c – выполнение	5			4		3	
c – завершение	5			9		12	

Таблица 13

Время обработки деталей с учетом выполнения aA после cA

Деталь	Время выполнения и завершения операции на станке					
	B	A	B	A	C	E
a – выполнение			6	5		
a – завершение			11	17		
b – выполнение	5	3			4	
b – завершение	5	12			16	17
c – выполнение			4		3	
c – завершение			13		16	

Таблица 14

Время обработки деталей с учетом выполнения aB перед cB

Деталь	Время выполнения и завершения операции на станке					
	A	B	A	B	A	C
a – выполнение	4			6	5	
a – завершение	4			11	17	
b – выполнение		5	3			4
b – завершение		5	12			16
c – выполнение	5			4		3
c – завершение	9			15		18

Таблица 15

Время обработки деталей с учетом выполнения aB после cB

Деталь	Время выполнения и завершения операции на станке						
	A	B	A	B	A	C	E
a – выполнение	4			6	5		
a – завершение	4			19	24		
b – выполнение		5	3			4	
b – завершение		5	12			16	24
c – выполнение	5			4		3	
c – завершение	9			13		16	

ликт также имеет два решения: *а*) операция bC выполняется перед операцией cC; *б*) операция bC выполняется после операции cC. Для этих вариантов приведены табл. 16 и 17.

Из анализа таблиц следует, что преимуществом по времени завершения всех процессов обладает вариант выполнения операции bC перед cC. Таким образом, за три деления множества всех вариантов решения задачи получено оптимальное решение, приведенное в табл. 16.

Таблица 16

Время обработки деталей с учетом выполнения bC перед cC

Деталь	Время выполнения и завершения операции на станке						
	A	B	A	B	A	C	E
a – выполнение	4			6	5		
a – завершение	4			11	17		
b – выполнение		5	3			4	
b – завершение		5	12			16	19
c – выполнение	5			4		3	
c – завершение	9			15		19	
Порядок	aA	cA	aB	cB	bC	cC	

Таблица 17

Время обработки деталей с учетом выполнения bC после cC

Деталь	Время выполнения и завершения операции на станке						
	A	B	A	B	A	C	E
a – выполнение	4			6	5		
a – завершение	4			11	17		
b – выполнение		5	3			4	
b – завершение		5	12			22	22
c – выполнение	5			4		3	
c – завершение	9			15		18	

Таблица 18

Время обработки деталей с учетом выполнения bC после cC

Деталь	Время выполнения и завершения операции на станке						
	A	B	A	B	A	C	E
a – выполнение	4			6	5		
a – завершение	9			15	20		
b – выполнение		5	3			4	
b – завершение		5	8			26	26
c – выполнение	5			4		3	
c – завершение	5			19		22	
Неоптимальный порядок	cA	aA	aB	cB	cC	bC	

Время завершения процессов равно 19 единицам, в первом конфликте начинается операция aA, во втором – операция aB и в третьем – операция bC. Сравним его с самым неоптимальным решением. Для этого при первом решении конфликта выберем неоптимальное. То же самое повторим для остальных двух конфликтов. Данные неоптимального решения приведены в табл. 18. Время завершения процессов равно 26 единицам, в первом конфликте начинается операция cA, во втором – операция aB и в третьем – операция cC.

Выигрыш во времени для оптимального запуска составляет по сравнению с неоптимальным $\delta T = (T_{n\text{opt}} - T_{\text{opt}})/ T_{\text{opt}} = 100(26 - 19)/19 = 36{,}8$ %.

20. Оптимизация сборочно-монтажных процессов с неизменяемыми режимами выполнения операций

Модель с неизменяемыми режимами выполнения операций (5) – (11) можно представить в виде двух подсистем уравнений и неравенств. В первой сгруппированы компоненты общей системы, связанные только со временем выполнения операций, во второй – только со временем ожидания партий перед выполнением операций. Для этого стоимость общего процесса C представим в виде суммы стоимости выполнения процесса и стоимости ожидания операций

$$C = C_R + C_W, \tag{75}$$

где стоимость выполнения операций

$$C_R = \sum_{p\in P}\sum_{b\in B} C_{pb} t_{pb} N_p \delta_{pb} \to \min, \tag{76}$$

а стоимость ожидания операций

$$C_W = \sum_{p\in P}\sum_{b\in B} C_p \tau_p(\boldsymbol{\Delta},\boldsymbol{\pi}) k_p \delta_{pb} \to \min. \tag{77}$$

Аналогично поступим со сроком выпуска изделия, создаваемого процессом $\mathbf{p}\in\mathbf{P}$.

Общий срок

$$T_{\mathbf{p}} = T_{R\mathbf{p}} + T_{W\mathbf{p}}, \tag{78}$$

где $T_{R\mathbf{p}}$ – ограничение на время выполнения любой из цепочек операций от начальной p_0 до конечной (p_{21} или p_{22}, см. рис. 3) без ожиданий:

$$\sum_{p\in P_{p_0}}\sum_{b\in B} t_{pb} N_p \delta_{pb} \le T_{R\mathbf{p}}, \quad p_0\in P_0, \mathbf{p}\in\mathbf{P}, \tag{79}$$

где $T_{W\mathbf{p}}$ – ограничение на время ожидания операции в любой из цепочек операций от начальной p_0 до конечных p_{21} и p_{22}:

$$\sum_{p\in P_{p_0}}\sum_{b\in B} \tau_p(\boldsymbol{\Delta},\boldsymbol{\pi}) k_p \delta_{pb} \le T_{W\mathbf{p}}, \quad p_0\in P_0, \mathbf{p}\in\mathbf{P}. \tag{80}$$

Остальные неравенства не связаны со временем ожидания операций. Поэтому первую подсистему образуют отношения (76), (79), (7) – (11), а вторую – отношения (77), (80).

Первая представляет собой постановку задачи линейного программирования с бинарными переменными выбора оборудования $\mathbf{\Delta}$, целочисленными переменными распределения операций по рабочим местам $k = \{k_{pb}, p \in P, b \in B\}$ и числом партий для всех операций n_p. Решение этой задачи методом ветвей и частичных решений позволит оптимально выбрать оборудование, распределить его по рабочим местам и определить количество партий для каждой операции, т. е. получить $\mathbf{\Delta}^{\text{opt}}$ и $\mathbf{k}^{\text{opt}}$. Вторая же является постановкой задачи определения оптимальной очередности $\boldsymbol{\pi}^{\text{opt}}$ выполнения операций на каждом рабочем месте. Эту задачу можно решить методом ветвей и границ. При этом минимизируется или стоимость, или время ожидания выполнения операций. В первом случае не контролируется общее время ожидания $T_{W\mathbf{p}}$, $\mathbf{p} \in \mathbf{P}$, а во втором случае – минимальная стоимость ожидания $C_{W.}$ Это вынуждает повторять решение обеих задач с разным соотношением между $T_{R\mathbf{p}}$ и $T_{W\mathbf{p}}$ до тех пор, пока не будет получено минимальное значение $C_{\min}$ при заданном значении вектора сроков выпуска изделий $\mathbf{T}$.

В настоящее время простые модели могут быть решены с привлечением офисного приложения Excel-97. Эта работа проводится с целью отработки методик решения таких задач, а также выяснения трудоемкости решения и других факторов, влияющих на решение подобных задач.

21. Оптимизатор задач линейного программирования Lindo

Описание приложения Lindo (*MSDOS*).

При проектировании оптимальных технологических процессов требуется решать задачи линейного программирования с большим числом переменных и неравенств. Для их решения используются оптимизаторы. Одним из них является интерактивная программа Lindo.

Программа решает задачи линейного и целочисленного программирования с числом вещественных переменных до 299 и числом функциональных ограничений до 118. Целочисленные переменные могут принимать значения только 0 и 1. Предельное число целочисленных переменных не превышает 110. Исходные данные могут вводиться с клавиатуры или из текстового файла. Они записываются в виде одной или нескольких строк, например:

max 2x + 6y – 3z
st
10x – 2y + 4z < 15
2x + 9y + z < 10
– 4x + 3y – 10z < 20
end
go

В первой строке указана целевая функция, затем функциональные ограничения. Слова st и end – начало и конец неравенств на ограничение переменных, слово go запускает задачу на решение. При вводе допустимы переносы на следующую строку в любом месте целевой функции или ограничений.

Управление пакетом осуществляется специальными командами, сгруппированными по следующим 11 функциональным признакам:

информация (help, com, cat),

ввод (max, min, retr, take, leave),

вывод на экран (pic, tabl, look, nonz, shoc, solu, range, bpic),

вывод файла (save, dive, rvrt, sdbc),

решение (go, piv),

редактирование (lt, ext, del, sub, appc),

выход (quit),

целочисленные, квадратичные и параметрические переменные (int, para, bip),

диалоговые параметры (widht, ters, verb, bat, page),

программы для пользователя, уточнение решения (inv, stat).

С помощью команды help можно получить краткие сведения о пакете в целом, а с помощью двух слов, например, help help или help xxx, где xxx — одна из команд, можно получить сведения о назначении и правилах выполнения команды. Рассмотрим назначение наиболее важных команд пакета:

com – вызов списка команд;

retr – чтение с диска файла с постановкой задачи, записанной редактором пакета и сохраненной на диске командой save;

take – чтение с диска файла с постановкой задачи и командами Lindo, записанными на диск любым текстовым редактором (последней командой в нем должна быть leave);

look – вывод на экран одной или нескольких строк постановки задачи. Команда look 1 5 выведет первые пять строк задачи;

solu – повторный вывод на экран результата решения задачи;

save – запись на диск постановки задачи, введенной редактором Lindo;

sdbc – запись на диск текстового файла с результатом решения задачи;

dive – запись на диск списка команд оптимизатора;

go – команда запуска решения задачи;

piv – команда запуска пошагового решения задачи;

alt – команда редакции постановки задачи. При запросе row вводится номер строки, а при запросе variable вводится имя переменной, перед которой меняется коэффициент, или rhs, если меняется правая часть, или dir, если меняется знак неравенства в неравенстве или направление оптимизации (min/max) в первой строке;

ext – команда добавления функциональных неравенств к ранее введенной постановке задачи;

del – команда удаления строки из постановки задачи;

appc – команда добавления в постановку задачи коэффициентов для новой переменной или изменения правых частей всех неравенств. В ответ на name вводят имя новой переменной или rhs для редакции правой части. В ответ на ? вводят номер строки, а в ответ на coef вводят значение коэффициента или значение правой части неравенства в режиме rhs;

quit – команда выхода из оптимизатора;

int – команда указания в постановке задачи бинарных переменных, принимающих значения 0 или 1. Эта команда указывается в конце постановки задачи. Если за ней следует число M, то оптимизатор считает первые M переменных бинарными. Если же после int следует список переменных, то оптимизатор считает бинарными переменные из этого списка.

Решение задач линейного программирования с помощью Lindo.

В первом случае нужно выбрать курсором небольшой файл (1 – 2 кбайт) и запустить встроенный редактор, нажав клавишу F4. На экране высветится содержимое файла. Если он нужен, то надо создать его копию, нажав одновременно клавиши Shift+F2 и изменив его имя в появившемся окне. После нажатия Enter нужно очистить копию одновременным нажатием клавиш Ctrl+Y и приступить к вводу задачи. Для задачи, рассмотренной в разд. 11, текст описания будет иметь вид

```
max X1+ 2X2
ST
X1 < 7
X1 + X2 < 9
X1 + 3X2 < 15
X2 < 4 (9)
END
```

Затем нужно запустить оптимизатор, установив курсор на имя Lindo.exe и нажав Enter. На экране появится заставка с курсором и двоеточием перед ним. Это означает, что оптимизатор ждет ввода описания задачи или одной из его команд. Для описываемого случая необходимо ввести команду take и имя файла с описанием задачи или просто take. Тогда после нажатия Enter появится окно с именами файлов текущего каталога, и можно установить курсор на требуемый файл и нажать Enter. В ответ появятся двоеточие и курсор. Чтобы убедиться в том, что файл загружен, нужно ввести команду look 1 5. Тогда на экране появится описание введенной задачи и в конце – приглашение к вводу команды или описание новой задачи.

Для второго случая после запуска оптимизатора и появления двоеточия приступают к вводу задачи. После ввода рекомендуется записать описание на диск командой save. Повторный вызов с диска этой задачи отличается от вызова задачи в первом случае тем, что вместо команды take необходимо ввести команду retr и выбрать из каталога сохраненный командой save файл задачи.

После ввода описания задачи можно приступать к ее решению. Для этого нужно ввести команду go или piv для полного или пошагового решения задачи симплекс-методом. В первом случае появляется сообщение о числе шагов решения задачи, максимальном (или минимальном) значении целевой функции, оптимальных значениях переменных и приглашение к анализу чувствительности решения к изменению правых частей функциональных ограничений. Во втором случае, после каждого шага, появляется сообщение о значениях основных переменных, номере опорного уравнения и значении целевой функции для текущего базиса. После выполнения всех шагов появляется сообщение, совпадающее с сообщением в случае полного решения задачи, и приглашение к анализу чувствительности ограничительных функций. В ответ на Y появляется сообщение о допустимых значениях увеличения и уменьшения коэффициентов при переменных и правых частей функциональных ограничений.

Подготовка исследовательских задач.

Изучение симплекс-метода заключается в решении двумерной задачи линейного программирования, заданной графически (см. рис. 4) симплекс-методом, описанным в разд. 11. Исследование оптимизатора заключается в пошаговом решении заданной задачи оптимизатором, решении задачи с бинарными переменными и исследовании предельных воз-

можностей оптимизатора по числу переменных, обычных и бинарных, а также по числу ограничений. Чтобы облегчить процесс описания исследовательских задач с большим числом переменных и ограничений, предлагается пример программы на Турбо-Паскале (загрузочный модуль TASK.EXE), иллюстрирующий принцип создания таких описаний. Ниже приводится текст программы:

```
program Task;
uses Crt;
const Sc1='C1 – правая часть неравенств: (5…15)';
      Sc2='C2 – число членов в неравенстве: (1...3)';
      Sn1='N1 – число переменных: (2...210)';
      Sn2='N2 – число неравенств: (2...130)';
      Sl1='L1 – длина строки целевой функции: (2...10)';
      Sl2='L2 – длина строки неравенств: (2...4)';
      Svv='Введите параметры задачи: C1, C2, N1, N2, L1, L2';
var I, J, K:word;F:text;S:string;C1, C2, N1, N2, L1, L2:byte;
procedure Zend;
begin WriteLn('END');WriteLn(F, 'END'); Close(F) end;
begin TextAttr:=Blue shl 4+Yellow;ClrScr;Randomize;
GotoXY(13, 2); Write(Svv);
repeat GotoXY(15, 3);ClrEol;Write(Sc1);ReadLn(C1);
until (C1>4) and (C1<16);
repeat GotoXY(14, 4);ClrEol;Write(Sc2);ReadLn(C2);
until (C2>0) and (C2<4);
repeat GotoXY(20, 5);ClrEol;Write(Sn1);ReadLn(N1);
until (N1>1) and (N1<211);
repeat GotoXY(20, 6);ClrEol;Write(Sn2);ReadLn(N2);
until (N2>1) and (N2<131);
repeat GotoXY(14, 7);ClrEol;Write(Sl1);ReadLn(L1);
until (L1>1) and (L1<11);
repeat GotoXY(17, 8);ClrEol;Write(Sl2);ReadLn(L2);
until (L2>1) and (L2<5);
GotoXY(25, 9);Write('Введите имя файла:');ReadLn(S);
Assign(F, S);ReWrite(F);
Write('MAX'); Write(F, 'MAX');
for I:=1 to N1 – 1 do
begin if I mod L1 = 0 then begin WriteLn;WriteLn(F) end;
Write('X', I, '+');Write(F, 'X', I, '+');
```

```
end;WriteLn('X', I+1);WriteLn(F, 'X', I+1);
WriteLn('ST');WriteLn(F, 'ST');I:=0;K:=0;
repeat I:=0;
 repeat Inc(I);
  if I mod L2 = 0 then begin WriteLn;WriteLn(F) end;
  if (K>N2) or (I>N1 – 1) then begin Zend; exit end;
   Write(F, 'X', I, '+'); Write('X', I, '+');
  for J:=1 to Random(C2) do
  begin Inc(I);Write(F, 'X', I, '+');Write('X', I, '+') end;Inc(I);
   Write(F, 'X', I, '<', Random(C1 – 1)+1, ' ');
   Write('X', I, '<', Random(C1 – 1)+1, ' ');Inc(K);
 until I > N1 – 1
until K > N2;Zend;
ReadLn
end.
```

В процедуре программы описывается запись в файл постановки задачи слова END. В основном блоке первая строка устанавливает цвет фона и символов, очищает экран и устанавливает случайное значение генератора случайных чисел. Следующие четырнадцать запрашивают предельное значение C1 случайно изменяющейся правой части функциональных неравенств, случайное число слагаемых в неравенстве C2, число переменных N1, число неравенств N2, число слагаемых в строке целевой функции L1 и число неравенств в одной строке L2.

Две строки ниже запрашивают имя и открывают текстовый файл с постановкой задачи. Очередная пишет на экран и в файл команду MAX. Четыре следующих пишут на экран и в файл целевую функцию из N1 переменных по L1 членов в строке. Ниже пишет разделитель ST, означающий «при условии». Далее следуют два вложенных друг в друга цикла. Внешний пишет N2 функциональных ограничений по L2 в строку, а внутренний пишет случайное число слагаемых в каждое функциональное ограничение. Оператор Close закрывает файл, а ReadLn задерживает описание постановки задачи на экране до нажатия Enter. Ниже приводится текст постановки задачи, создаваемый этой программой:

```
MAX X1+X2+X3+X4+X5+X6+X7+X8+X9+X10+X11+X12+
X13+X14+X15+X16+X17+X18+X19+X20
ST
```

X1+X2<9 X3+X4<6 X5+X6<2 X7+X8+X9<6
X10+X11+X12<4 X13+X14<5 X15+X16+X17<4
X18+X19+X2<4 X1+X2<5 X3+X4<9 X5+X6<5
X7+X8<7 X9+X10+X11<7
X12+X13+X14<2 X15+X16+X17<8 X18+X19<4
END

Исследование оптимизатора.

Исследование оптимизатора заключается в определении зависимости времени выполнения задачи от числа переменных при предельном числе функциональных ограничений и от числа ограничений при предельном числе переменных. Это исследование нужно выполнить для вещественных переменных и для целочисленных переменных, если общее число переменных меньше 110. Если же оно больше 110, то остальные переменные должны быть вещественными. Чтобы выполнить решение, необходимо запустить задачу TASK.EXE, ввести требуемые значения параметров задачи, включая число переменных и ограничений, ввести имя файла, например T_N, где N – номер решения, а T – имя файла с решением. Задача создаст файл с решением и передаст управление операционной системе. Затем нужно запустить программу LINDO.EXE, загрузить файл с решением T_N командой take, проверить правильность загрузки командой look 1 M, где M – число строк загруженной постановки задачи, и запустить решение задачи командой go. Если в задаче должны быть K целочисленных переменных, то перед командой go нужно запустить команду int K. В каждом решении нужно определять время загрузки файла с задачей и время решения задачи.

22. Использование Lindo для распределения во времени и по рабочим местам сборочно-монтажных процессов

С помощью программы Sborka23.pas создается текстовый файл с табл. 19.

В таблице P1 – сборочная единица (СЕ), собираемая из СЕ P10 и P11, первая из которых собирается из P100 и P101, а вторая – из P110 и P111 и т. д.; V – число сборок; B1 – B4 – рабочие места; C – стоимость операции в единицу времени; *T* – длительность операции; 0 – запрет на выполнение операции на данном рабочем месте.

Таблица 19

Стоимость C и длительность *T* операций P на рабочих местах B

Сборочные операции	Число операций V	B1		B2		B3		B4	
		C	*T*	C	*T*	C	*T*	C	*T*
P1	4	0	0	6	6	7	6	0	0
P10	4	8	5	0	0	6	7	5	8
P11	4	3	10	8	6	0	0	7	6
P110	4	0	0	8	5	3	10	6	8
P111	4	0	0	4	10	3	10	0	0
P1100	4	4	10	7	7	0	0	0	0
P1101	4	6	8	4	9	5	7	0	0
P11010	4	4	9	7	6	0	0	3	10
P11011	4	0	0	4	8	0	0	5	7
P2	8	4	8	5	8	0	0	5	8
P20	8	8	5	0	0	3	10	5	9
P21	8	4	9	3	11	0	0	5	8
P210	8	5	8	5	7	0	0	0	0
P211	8	3	9	0	0	5	9	5	8
P2110	8	7	5	0	0	0	0	4	8
P2111	8	0	0	3	11	5	7	0	0
P21110	8	6	7	6	8	0	0	3	11
P21111	8	0	0	5	8	6	7	6	8

С помощью этой же программы создается файл с постановкой задачи распределения операций по рабочим местам, минимизирующий общую стоимость сборки при ограничении на неравномерность загрузки рабочих мест и время выполнения всех операций. Поскольку Lindo не воспринимает сложные обозначения переменных, приходится обозначать переменные B2P1 и B3P1, как X1 и X2, B1P10, B3P10 и B4P10, как X3, X4 и X5 и т. д. Программа Sborka23.pas создает третий файл соответствия между этими обозначениями переменных. Затем загружается Lindo, командой take загружается файл с постановкой задачи, командой go получается решение, командой sdbc создается файл с решением задачи, командой quit выгружается Lindo. Загружается Lexicon, в два разных окна загружаются файл с решением задачи и файл с обозначениями переменных.

Блоковыми командами {Shift-F3}, {RIGHT}, {DOWN}, {Ctrl-F3} и {Shift-F4}, {Alt-2} и {Alt-1} эти файлы объединяются и командой {Alt-y} из файла удаляются строки с нулевыми значениями бинарных переменных. Должны остаться переменные, соответствующие назначению каждой операции на одно из возможных рабочих мест. В итоге нужно сохранить файл с исходной таблицей, в которой перед обозначением каждой операции вставлен префикс обозначения выбранного рабочего места, например, вместо P1 должно быть указано обозначение B2P1 или B3P1 и т. д. Затем запускается программа Spodg25.pas, которая запрашивает файл с этой таблицей и выдает файл с графиком загрузки оборудования операциями сборочных процессов (рис. 12).

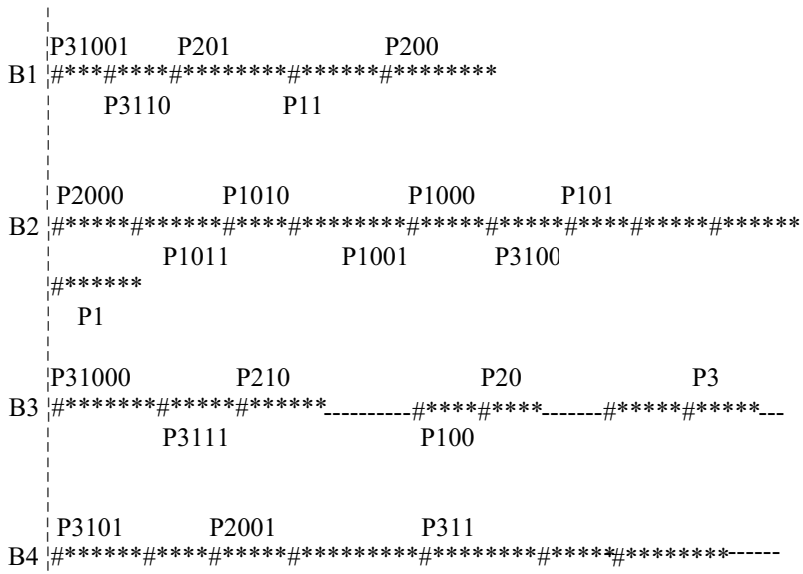

Рис. 12. График запуска сборочных операций на рабочих местах

В графике имеются окна с ожиданием начала операций. Необходимо с помощью блоковых команд редактора Word сократить размеры окон до минимума, а с помощью переназначения операциям рабочих мест и перезапуском программы Spodg25.pas сократить до минимума неравномерность загрузки оборудования.

Библиографический список

1. *Смирнов О. Л.*, *Соколова А. Н.* Проектирование технологического процесса сборки и монтажа печатных узлов на ЭВМ: Учеб. пособие/ ЛИАП. Л., 1991. 48 с.

2. *Смирнов О. Л.*, *Соколова А. Н.*, *Баринов А. Е.* Основы проектирования оптимальной сборки модулей РЭА: Текст лекций/СПбГААП. СПб., 1992. 48 с.

Программа формирования постановки задачи распределения сборочных операций по рабочим местам Sborka23.pas.

```
uses Crt;
  const Pg = 22; Ism = 50; Ism1 = 30; Z = 4; Ls = 8;
        Im = 8; Rm = 9; Ir = 6; Bm = 4; Nbr = 1; Npr = 1;
        Imr = 4; Npm = 10; Nbm = 4; NbO =2*(Nbr+Nbm+1);
        Csm = 5;Imm = 7; PgM = 4*(Imm+Imr+1);Jr = 5;
        Ng : array [1..6] of byte = (4, 8, 13, 15, 20, 24);
        B : array [1..5] of string[2] =
                    ('B1', 'B2', 'B3', 'B4', 'B5');
  TabH : array [1..2] of string[65] =
('          V   B1    B2   B3   B4   B5',
'            C   T   C   T   C   T   C   T   C   T');
  TabH1 : array [1..2] of string[65] =
('          V   B1    B2   B3   B4',
'            C   T   C   T   C   T   C   T');
  {типы данных}
  type TabL = record Nam : string[Ls];
              Dat : array[1..NbO] of byte
             end;
       Dig = 1..5;
       Bnn = set of Dig;
  {переменные}
  var I, Ip, Imt, I1, I2, Iz, Is, J, Ji, Jk, Jm, Jst:byte;
  K, Kb, Kbv, Nb, Np, Cs, Lsv, Pgt:byte;
  Otm:boolean; Ch:char;
  J1:word; Fname : string[14];
  F:text; Cw, Cwt : string[2];
  Stm : longint; St : word;
  S : string[1];
  Cc, Jmx : integer;
  Tab : array [1..PgM] of TabL;
  Bn : Bnn;
  VarX: array[1..100] of string[15];
                {процедуры и функции}
  procedure F2o; {формирование имен операций}
```

```
  begin
    Inc(I); Tab[I].Nam:=Tab[Iz].Nam+'0';
    Inc(I); Tab[I].Nam:=Tab[Iz].Nam+'1';
  end;
procedure NamP; {запись операции P}
begin Lsv:=Ls-Ord(Tab[I].Nam[0]);
  Write(Tab[I].Nam, ":Lsv); Write(F, Tab[I].Nam, ":Lsv)
end;
    {запись объема выпуска V}
procedure WriteV;
begin Cwt:=Tab[I].Nam; if Cwt <> Cw then
  begin Kb:=Random(Rm-2)+2; if Kb <= 3 then Inc(Kb); Cw:=Cwt end;
  Write(":Z, Kb);Write(F, ":Z, Kb);Tab[I].Dat[2]:=Kb
end;
procedure WriteS; {запись состава рабочих мест для операции}
var Ik, Jk, Kb : byte; Usl : boolean;
begin Bn:=[]; if random(Npm)<Npm div 2 then Kb:=2 else Kb:=3;
  for Ik:=1 to Kb do begin
    repeat Jk:=random(Nb)+1 until not (Jk in Bn); Bn:=Bn+[Jk]
  end
end;
procedure WriteN; {запись запрета рабочего места}
  begin Write(":Z, '-'); Write(F, ":Z, '-');
    Write(":Z, '-'); Write(F, ":Z, '-');
  Tab[I].Dat[1+2*J]:= Rm+1; Tab[I].Dat[2+2*J]:= Rm+1
end;
procedure Wtabl; {заполнение таблицы процессов и рабочих мест}
const UrK = 3;
function Kt(C:byte):byte;
begin Kt:=12-C+Round(Random)+Round(Random); end;
begin NamP; WriteV; WriteS;
  for J:=1 to Nb do if J in Bn then for Jk:=1 to 2 do
  begin if Jk=1 then K:=Random(Rm-UrK)+UrK else K:=Kt(K);
    Write(K:Z+1); Write(F, K:Z+1);
    Tab[I].Dat[Jk+2*J]:= K
end else WriteN; WriteLn; WriteLn(F)
end;
```

```
procedure WriteK; {отметка концевых операций}
begin
repeat Cs:=random(Csm);
  Case Cs of
    0, 1:begin Tab[Iz+2].Dat[1]:=1;Inc(Iz);F2o;Inc(Iz) end;
    4, 5:begin Tab[Iz+1].Dat[1]:=1;Inc(Iz);Inc(Iz);F2o end;
    else begin Inc(Iz);F2o;Inc(Iz);F2o end
  end;
until I > Imt;
  Imt:=I; for I:=Iz+1 to Imt do Tab[I].Dat[1]:=1; Iz:=Imt;
end;
procedure TabHd(V : string); {заголовок таблицы}
begin WriteLn(V); WriteLn(F, V) end;
procedure WriteT; {запись таблицы}
begin
  for I:=1 to Pg do Wtabl; {начало таблицы}
  ReadLn; ClrScr;
  for I:=1 to 2 do
  if Nb = 4 then WriteLn(TabH1[I]) else WriteLn(TabH[I]);
  for I:=Pg+1 to Pgt do Wtabl; {конец таблицы}
  Close(F);
end;
procedure ConPg;{размер строк и страниц}
begin Inc(J1);
  if J1 mod Jr = 0 then {Jr слагаемых в строку}
  begin Writeln; Writeln(F); J1:=0;
    Inc(I2);if I2 mod 23 = 0 then
    begin ReadLn;ReadLn end
  end
end;
function Sgn(N:integer):char; {знаки коэффициентов}
begin
  if Jst=1 then Sgn:=' ' else Sgn:='+'
end;
procedure ObjF; {целевая функция}
begin Jst:=0;WriteLn('min'); WriteLn(F, 'min');
  for I:=1 to Pgt do for J:=1 to Nb do
```

```
  begin
    if Tab[I].Dat[1+2*J] < Rm+1 then {запись целевой функции}
    begin St:=Tab[I].Dat[2]*Tab[I].Dat[1+2*J]*Tab[I].Dat[2+2*J];
          Inc(Jst);
          VarX[Jst]:=B[J]+Tab[I].Nam;
          Write(Sgn(J), St, 'X', Jst);
          Write(F, Sgn(J), St, 'X', Jst);
          ConPg
    end;
  end;
  Jmx:=Jst
end;
function Indx(Str:string):integer; {индекс имени операции}
var I:integer;
begin
I:=0;
repeat
  Inc(I);
until (VarX[I]=Str) or (I>Jmx);
if I>Jmx then
begin
  Write('О Ш И Б К А В I n d x'); Halt(1)
  end;
  Indx:=I
end;
procedure SubR; {выполнение операции на одном рабочем месте}
begin
for I:=1 to Pgt do
 begin for J:=1 to Nb do
  if Tab[I].Dat[1+2*J] < Rm+1 then
  begin
    Write(Sgn(J), 'X', Indx(B[J]+Tab[I].Nam));
    Write(F, Sgn(J), 'X', Indx(B[J]+Tab[I].Nam))
  end;
  Write('=1'); Write(F, '=1'); Writeln; Writeln(F);
  Inc(I2);if I2 mod 23 = 0 then ReadLn
  end
end;
```

```
    procedure SubTr; {загрузка рабочего места}
    begin
    for J:=1 to Nb do
    begin J1:=0; Stm:=0; for I:=1 to Pgt do
      if Tab[I].Dat[1+2*J] < Rm+1 then
      begin ConPg;
        St:=Tab[I].dat[2]*Tab[I].Dat[2+2*J]; Stm:=Stm+St;
        Write(Sgn(I), St, 'X', Indx(B[J]+Tab[I].Nam));
        Write(F, Sgn(I), St, 'X', Indx(B[J]+Tab[I].Nam))
      end; Stm:=Stm div 4; WriteLn; WriteLn(F);
      WriteLn(' <= ', Stm); WriteLn(F, ' <= ', Stm)
      end;
    end;
    procedure SubPr; {одновременное выполнение операций}
    begin
      for I:=1 to Pgt do
      if Length(Tab[I].Nam) > 2 then
      begin
        for J:=1 to Nb do
        if (Tab[I].Dat[1+2*J] < Rm+1) and (Tab[I+1].Dat[1+2*J] < Rm+1)
          then
          begin {параллельные операции на разных рабочих местах}
            WriteLn('X', Indx(B[J]+Tab[I].Nam), '+', 'X',
Indx(B[J]+Tab[I+1].Nam), ' = 1');
            WriteLn(F, 'X', Indx(B[J]+Tab[I].Nam), '+', 'X',
Indx(B[J]+Tab[I+1].Nam), ' = 1');
            Inc(I2);if Iz mod 23 = 0 then ReadLn
          end;
          Inc(I);
        end;
      end;
    procedure FormTask; {файл с постановкой задачи}
    begin
          Write(':15, 'ИМЯ ФАЙЛА С ПОСТАНОВКОЙ ЗАДАЧИ =>');
          Read(Fname);WriteLn;
          Assign(F, Fname); ReWrite(F); {открытие файла}
          J1:=0;I2:=0; ObjF; WriteLn; WriteLn(F);
          Inc(I2);if I2 mod 23 = 0 then ReadLn;
```

```
        WriteLn('st'); WriteLn(F, 'st');
        Inc(I2);if I2 mod 23 = 0 then ReadLn;
        SubTr; SubR; SubPr;
        WriteLn('end'); WriteLn(F, 'end');
        Close(F);
    end;
    procedure VarName; {файл с именами переменных}
    var I:integer;
    begin
        Write(':15, 'ИМЯ ФАЙЛА С ИМЕНАМИ ПЕРЕМЕННЫХ => ');
        Read(Fname);WriteLn;
        Assign(F, Fname); ReWrite(F); {открытие файла}
        for I:=1 to Jmx do
        begin
          WriteLn(VarX[I]); WriteLn(F, VarX[I]);
        end;
        Close(F);
    end;
    procedure FirstData; {исходные данные: число процессов и рабочих
мест}
    begin
        Write(':25, 'ИМЯ ФАЙЛА С ИСХОДНЫМИ ДАННЫМИ => ');
        Read(Fname);
        Assign(F, Fname); ReWrite(F); {открытие файла}
        for I:=1 to PgM do begin Tab[I].Nam:=' ';
          for J:=1 to NbO do Tab[I].Dat[J]:=0
        end;
        repeat WriteLn;
          Write(':27, 'ЧИСЛО ПРОЦЕССОВ ? <3-4>'); ReaδΛn(Np);
        until (Np = 3) or (Np = 4);
        repeat WriteLn;
          Write(':27, 'ЧИСЛО РАБОЧИХ МЕСТ ? <4-5>'); ReadLn(Nb);
        untiλ (Nb = 4) or (Nb = 5);
        Imt:=0;I:=0;Iz:=0;
        for Ip:=1 to Np do {имена операций}
        begin Inc(I); Str(Ip, S); Tab[I].Nam:='P'+S; Inc(Iz);
          F2o; Imt:=Imt+Imm+random(Imr); WriteK;
```

```
        end; Pgt:=I;
        for I:=1 to 2 do if Nb = 4 then {шапка таблицы}
        TabHd(TabH1[I]) else TabHd(TabH[I]);
        WriteT;
end;
procedure Install; {инициализация}
begin
        TextBackground(LightBlue); ClrScr;
        Otm := false;I1:=0; Cw:=' '; Kb:=0; Randomize;
        Is:=Random(Ism); {ввод случайности}
        for I:=1 to Is do J:=Random(Ism1);
end;
begin {операторы программы}
     Install;
     repeat Inc(I1);WriteLn;
        FirstData;
        FormTask;
        VarName;
        ReadLn;ClrScr;
        Write(':27, 'ЗАКОНЧИТЬ ? <Y/N>'); Ch:=ReadKey;Write(Ch);
     until (Ch = 'y') or (Ch = 'Y')
     end.
```

Программа выдачи файла с графиком загрузки оборудования операциями сборочных процессов Spodg25.pas.

```
uses Crt;
var I, Im:byte;
    F:text;Nm:string[15];
    Arstr:array[1..40] of string[10];
    Arpt:array[1..40] of byte;
procedure OpenRd;
begin
  Write(#13, #10, 'Имя файла для чтения данных');
  ReadLn(Nm); Assign(F, Nm); Reset(F);
end;
procedure OpenWr;
begin
  Write(#13, #10, 'Имя файла для записи результатов');
  ReadLn(Nm);
  Assign(F, Nm);
  Rewrite(F);
end;
function Pt(N:byte):byte;
var St:string[10];I:byte;
begin Pt:=0;
  St:=Arstr[N];
  if Pos(' ', St)=3 then Pt:=N else
  begin
    St[Pos(' ', St)-1]:=' ';
    for I:=1 to N-1 do if St=Arstr[I] then
    begin Pt:=I; I:=N-1 end;
  end;
end;
procedure SetArst;
begin
  OpenRd; ReadLn(F); ReadLn(F);
I:=0;
while not Eof(F) do
begin
```

```
  Inc(I);
  ReadLn(F, Arstr[I]);
end;
Im:=I;
Close(F);
end;
procedure SetArpt;
begin
  for I:=1 to Im do Arpt[I]:=Pt(I);
end;
procedure WrArpt;
begin
  OpenWr;
  for I:=1 to Im do
  begin
    if WhereX > 55 then begin WriteLn;WriteLn(F); end;
    Write(' Arpt[', I, '] = ', Arpt[I]:2);
    Write(F, ' Arpt[', I, '] = ', Arpt[I]:2);
  end;
  Close(F);
end;
begin {программа}
  SetArst;
  SetArpt;
  WrArpt;
end.
```

Оглавление

Учебное издание

Смирнов Олег Леонидович

АВТОМАТИЗАЦИЯ ТЕХНОЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ

Учебное пособие

Редактор *А. Г. Ларионова*
Компьютерная верстка *А. Н. Колешко, Ю. С. Бардуковой*

Лицензия ЛР № 020341 от 07.05.97. Сдано в набор 28.09.01. Подписано к печати 24.12.01. Формат 60×84 1/16. Бумага тип. № 3. Печать офсетная. Усл. печ. л. 3,8. Усл. кр.-отт. 4,1. Уч. -изд. л. 4,0. Тираж 100 экз. Заказ №

Редакционно-издательский отдел
Лаборатория компьютерно-издательских технологий
Отдел оперативной полиграфии
СПбГУАП
190000, Санкт-Петербург, ул. Б. Морская, 67