Федеральное агентство по образованию
Государственное образовательное учреждение высшего профессионального образования
Омский государственный университет им. Ф. М. Достоевского
Математический факультет
Кафедра прикладной и вычислительной математики

МНОГОЭТАПНАЯ ЗАДАЧА О ПОСТАВКАХ ПРОДУКЦИИ

Дипломная работа
студентки группы МП-004
Шабакиной О.В.
_____ (подпись)

Научный руководитель
старший преподаватель
кафедры ПиВМ,
к. ф.- м. н.
Еремеев А.В.
_____ (подпись)

Омск - 2005

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Одной из задач, с которой в ходе своей деятельности приходится сталкиваться практически любой производственной или торговой организации, является выбор поставщика продукции с условием удовлетворения потребностей предприятия и минимизации затрат.

При решении одноэтапной задачи о поставках продукции может быть получен ответ на вопрос о том, какое количество продукции следует закупать у конкретного поставщика. В случае решения многоэтапной задачи можно узнать не только о том, услугами каких поставщиков лучше воспользоваться, но и получить информацию о наиболее выгодном времени поставки, которое совпадает с началом каждого интервала времени (например, еженедельно или ежемесячно).

Задача состоит в минимизации общих затрат, складывающихся из затрат на приобретение товара, оформление заказа и хранение запаса (для многоэтапной задачи). Затраты на приобретение становятся важным фактором, когда цена единицы продукции зависит от размера поставки, что обычно выражается в виде оптовых скидок в тех случаях, когда цена единицы товара убывает с возрастанием размера заказа. Затраты на оформление заказа являются постоянным расходом, связанным с его размещением. Затраты на хранение представляют собой расходы на содержание запаса на складе (например, процент на инвестированный капитал, затраты на переработку, амортизационные и эксплуатационные расходы), обычно возрастают с увеличением запаса.

Данная модель имеет много общего с хорошо известной транспортной задачей, но главное отличие состоит в том, что в задаче о поставках продукции продавец либо вообще ничего не поставляет, либо предлагает некоторое количество товара из определенного для него интервала.

В настоящей работе рассматривается одноэтапная задача о поставках продукции и ее обобщение для нескольких периодов планирования. Основная цель работы – построение алгоритма динамического программирования и

генетического алгоритма для решения данной задачи, исследование их эффективности. Постановка задач и их формальная запись представлены в §1. В §2 описаны алгоритмы динамического программирования для решения одноэтапной и многоэтапной задач о поставках продукции, рассматривается их трудоемкость. В §3 описаны общая схема генетического алгоритма и ее модификация для задачи о поставках продукции. В §4 представлены результаты вычислительных экспериментов. Приведены решения одноэтапной задачи о поставках с помощью алгоритма динамического программирования [3] и эвристического алгоритма [4], исследована зависимость времени работы алгоритма [3] от входных параметров. Здесь же описаны эксперименты по сравнению времени работы алгоритма динамического программирования и CPLEX MIP Solver 6.5 при решении многоэтапной задачи. Проведен ряд экспериментов по настройке параметров генетического алгоритма, исследована вероятность нахождения им оптимального решения.

1. ПОСТАНОВКА ЗАДАЧИ О ПОСТАВКАХ ПРОДУКЦИИ

1.1 Одноэтапная задача о поставках продукции

Рассмотрим задачу о поставках продукции следующего вида: имеется множество поставщиков, которые предлагают некоторую идентичную продукцию и один потребитель, предъявляющий спрос на данный товар. Необходимо удовлетворить спрос покупателя так, чтобы затраты на приобретение были минимальными, а поставленное количество товаров точно соответствовало объему заказа.

Для каждого поставщика определены нижняя и верхняя границы, в рамках которых он может предложить покупателю любое количество своего товара. Нижняя граница этого интервала устанавливается с точки зрения экономической целесообразности (поставлять меньшее количество продукции было бы просто невыгодно), а верхняя – связана с техническими ограничениями: это максимальное количество товара, которое может быть поставлено заказчику сразу после заключения договора.

Стоимость продукции каждого поставщика выражена вогнутой функцией, аргументом которой является количество проданного товара. Предположение вогнутости отражает общую ситуацию, имеющую место в промышленности, когда стоимость единицы продукции и стоимость ее транспортировки уменьшается с увеличением объема заказа.

В данной модели рассматривается один период планирования (например, неделя или месяц), т.е. нас не интересуют потребности покупателя и возможности поставщиков на следующих этапах их работы.

Формально эту задачу можно записать следующим образом:

$$\sum_{i=1}^{n} k_i\ (x_i\ ) \to \min, \qquad (1)$$

$$\sum_{i=1}^{n} x_i = A, \qquad (2)$$

$x_i \in \{0\} \cup [m_i, M_i]$ *для* $i = 1,2,\ldots,n.$ (3)

Здесь n - количество поставщиков,

x_i - количество продукции, которое получаем от поставщика с номером i,

A - количество продукции, в котором нуждается потребитель,

m_i - минимальное количество товара, которое готов предложить поставщик с номером i,

M_i - максимальное количество товара, которое может предложить поставщик с номером i.

Все описанные выше величины относятся к стандартному периоду планирования, который рассматривается в данной задаче.

Стоимость поставки выражается функцией

$$k_i(x_i) = \begin{cases} 0, & \text{если} \quad x_i = 0, \\ a_i + g_i(x_i), & \text{если} \quad x_i > 0, \end{cases}$$

где $a_i \geq 0$ можно рассматривать как затраты на оформление заказа,

$g_i(x_i) > 0$ – вогнутая, возрастающая функция положительного аргумента x_i, $i=1,\ldots,n$.

В [3] показано, что задача (1) – (3) с линейной функцией, выражающей стоимость поставки, и рациональными входными параметрами является *NP*-трудной.

1.2 Многоэтапная задача о поставках продукции

Данная модель является обобщением задачи (1) – (3). Отличие состоит в том, что после удовлетворения спроса на первом этапе, деятельность поставщиков и потребителя не прекращается, начинается новый период планирования, в котором снова возникает потребность покупателя в продукции и предложения поставщиков. При рассмотрении каждого этапа в отдельности многоэтапная задача о поставках продукции сводится к задаче (1) – (3). Таким образом, для каждого периода планирования мы по-прежнему имеем некоторое множество поставщиков с определенными интервалами возможных поставок и

функциями стоимости, а также потребителя, предъявляющего спрос на данный товар. От последовательности независимых задач (1) – (3) модель отличается тем, что на каждом этапе может быть заказано такое количество продукции, что после удовлетворения текущего спроса, остаток переходит на удовлетворение спроса последующих этапов. В связи с этим возникают затраты на хранение продукции, оставшейся после данного этапа и переходящей в следующий. Общие затраты складываются из затрат на закупку, оформление заказа и хранение по всем этапам. Задача состоит в минимизации общих затрат. Предполагается также, что дефицит не допускается. Многоэтапная задача о поставках продукции сходна с задачей управления запасами [2], но на каждом этапе формулируется как одноэтапная задача о поставках продукции.

Формально задача может быть записана в следующем виде:

Пусть n - количество поставщиков,

T - количество периодов планирования,

x_{it} - количество продукции, которое получаем от поставщика с номером i на этапе t,

A - общее количество продукции, в котором нуждается потребитель,

m_{it} - минимальное количество товара, которое готов предложить поставщик с номером i на этапе t,

M_{it} - максимальное количество товара, которое может предложить поставщик с номером i на этапе t.

z_t – остаток, переходящий из этапа t в этап $t + 1$,

ξ_t – спрос на этапе t,

Стоимость поставки выражается функцией

$$k_{it}(x_{it}) = \begin{cases} 0, & \text{если} \quad x_{it} = 0, \\ a_{it} + g_{it}(x_{it}), & \text{если} \quad x_{it} > 0, \end{cases}$$

$a_{it} \geq 0$ – стоимость оформления заказа,

$g_{it}(x_{it}) > 0$ – вогнутая, возрастающая функция положительного аргумента x_{it}, $i=1,\ldots,n$, $t = 1,\ldots,T$.

Стоимость хранения выражается заданной функцией $h_t(z_t)$.

$$\sum_{t=1}^{T}\sum_{i=1}^{n} k_{it}(x_{it}) + \sum_{t=1}^{T-1} h_t(z_t) \to \min, \qquad (1')$$

$$z_t = z_{t-1} + \sum_{i=1}^{n} x_{it} - \xi_t, \quad t = 1,2,...,T, \qquad (2')$$

$$z_t \geq 0,\ t = 1,2,...,T-1, \qquad (3')$$

$$z_T = 0, \qquad (4')$$

$$x_{it} \in \{0\} \cup [m_{it}, M_{it}], \text{ для } i = 1,2,...,n,\ t = 1,2,...,T. \qquad (5')$$

В последующих параграфах будут рассмотрены алгоритмы для решения данной задачи.

2. АЛГОРИТМ ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ ДЛЯ ЗАДАЧИ О ПОСТАВКАХ ПРОДУКЦИИ

Динамическое программирование представляет собой математический аппарат, разработанный для решения некоторого класса задач математического программирования. Характерным для этого метода является подход к решению задачи по этапам, с каждым из которых ассоциированы одна или несколько переменных. Набор рекуррентных вычислительных процедур, связывающих различные этапы, обеспечивает получение допустимого оптимального решения задачи в целом при достижении последнего этапа.

2.1 Алгоритм динамического программирования для одноэтапной задачи о поставках продукции

В дальнейшем будем полагать, что A, m_i, M_i – целые числа. Это не является существенным ограничением, так как для каждой задачи можно подобрать достаточно малые единицы измерения, в которых условие целочисленности входных данных будет выполнено.

Если A, m_i, M_i - целые числа, то существует оптимальное решение, где все компоненты x_i, $i=1,\ldots,n$ также целые, тогда исходную задачу можно рассматривать как задачу дискретной оптимизации (см., например, [3]), для решения которой применима техника стандартного динамического программирования. Основанный на этой технике псевдополиномиальный алгоритм нахождения оптимального решения ($ДП_1$) предложили в [3] С.С. Чаухан, А.В. Еремеев, А.А. Колоколов и В.В. Сервах, в своей работе опиравшиеся на результат, полученный в [4]:

Лемма 1. *Существует оптимальное решение* $X=\{x_1, x_2, \ldots, x_n\}$ *такое, что* $x_i = m_i$ *или* $x_i = M_i$ *или* $x_i = 0$ *для* $i = 1,2,\ldots,n$, *за исключением не более одного* $j\in\{1,2,\ldots,n\}$, *для которого* $m_j < x_j < M_j$.

Каждое состояние системы определяется двумя параметрами: количеством поставщиков, которые участвуют в удовлетворении спроса потребителя, и количеством продукции, которое они поставляют. Для каждого количества поставщиков $p=\{1,2,\ldots,n\}$ будем определять величину $\varphi(p,a)$ - минимальную стоимость поставки a единиц продукции, $a=\{0,1,2,\ldots,A\}$.

Начальное множество состояний схемы динамического программирования задается следующим образом: $\varphi(0,0)=0$, $\varphi(0,a)=\infty$, $a=1,2,\ldots,A$, таким образом, описано начальное краевое условие для уравнения Беллмана «вперед».

Переход в следующее состояние осуществляется по рекурсивному соотношению:

$$\varphi(p,a)=\min\left\{\varphi(p-1,a),\ \min_{m_p\le x_p\le M_p;\,x_p\le a}(\varphi(p-1,a-x_p)+k_p(x_p))\right\},$$

$$p=1,2,\ldots,n;\ \ a=0,1,2,\ldots,A.$$

Покажем корректность данного уравнения. Возможны два случая: $x_n=0$ или $m_n\le x_n\le M_n$.

1. Если $x_n=0$, то $k_n(0)=0$ и, следовательно, задача (1) – (3) сводится к следующей:

$$\sum_{i=1}^{n-1}k_i(x_i)\to\min,$$

$$\sum_{i=1}^{n-1}x_i=A,$$

$x_i\in\{0\}\cup[m_i,M_i]$ *для* $i=1,2,\ldots,n-1$.

В случае 1 получаем $\varphi(n,A)=\varphi(n-1,A)$.

2. Если мы рассматриваем некоторое фиксированное значение x_n при выполнении $m_n\le x_n\le M_n$, то задача (1) - (3) сводится к следующей:

$$\sum_{i=1}^{n-1}k_i(x_i)+k_n(x_n)=\varphi(n-1,A-x_n)\to\min,$$

$$\sum_{i=1}^{n-1}x_i=A-x_n,$$

$x_i\in\{0\}\cup[m_i,M_i]$ *для* $i=1,2,\ldots,n-1$.

Следовательно, для вычисления минимальной стоимости в случае $x_n > 0$ необходимо знать решение задачи для *n-1* поставщика.

Рассмотрев оба случая, найдем значение x_n, минимизирующее целевую функцию:

$$\varphi(n, A) = \min\left\{\varphi(n-1, A), \min_{x_n \le x_n \le M_n} (\varphi(n-1, A - x_n) + k_n(x_n))\right\}.$$

Для вычисления $\varphi(n, A)$ требуется решение задачи размерности *n-1*, для нахождения которого необходимо решить задачу (1) – (3) для *n-2* поставщиков и так далее. Таким образом, получаем описанное выше рекурсивное соотношение, по которому осуществляется переход в следующее состояние. Корректность уравнения Беллмана доказана.

2.2 Алгоритм динамического программирования для многоэтапной задачи о поставках продукции

Применим технику динамического программирования и опишем алгоритм (ДП) для решения задачи (1') - (5'). Состояние системы на каждом этапе, как и в случае одноэтапной задачи, будет определяться двумя параметрами: количеством поставщиков, которые участвуют в удовлетворении спроса потребителя, и количеством продукции, которое они поставляют. Для каждого количества поставщиков *p={1,2,…,n}* будем определять величину $\varphi_t(p, a)$ - минимальную стоимость поставки a единиц продукции, $a = \{0,1,2,..., A - \sum_{\tau=1}^{t-1} \xi_\tau\}$ для периода планирования $t = \{1,2,...,T\}$.

Начальное множество состояний схемы динамического программирования задается следующим образом:

$$\varphi_1(0,0) = 0,\ \varphi_1(0,a) = \infty,\ \ a = 1,2,..., A,$$

$$\varphi_t(0,a) = \varphi_{t-1}(n, a + \xi_{t-1}) + h_{t-1}(a),\ \ t \neq 1,\ \ a = 0,1,2,...,\ A - \sum_{\tau=1}^{t-1} \xi_\tau .$$

Переход в следующее состояние осуществляется по рекурсивному соотношению:

$$\varphi_t(p,a) = \min\left\{\varphi_t(p-1,a), \min_{m_{pt}\le x_{pt}\le M_{pt}; x_{pt}\le a}(\varphi_t(p-1,a-x_{pt}) + k_{pt}(x_{pt}))\right\},$$

$$t = 1,2,...,T, \quad p = 1,2,...,n, \quad a = 0,1,2,..., A - \sum_{\tau=1}^{t-1}\xi_\tau.$$

При вычислении общих затрат задачи (1') - (5') с применением данной схемы динамического программирования на каждом этапе решается задача (1) - (3). В зависимости от решения задачи на предыдущем этапе формируются начальные условия и ограничения на максимальный размер поставки продукции для следующего этапа. Покажем, что, действуя таким образом, сможем минимизировать общие затраты.

В рамках одного периода планирования задача (1') - (5') сводится к задаче (1) - (3), остаются справедливыми лемма 1 и доказательство корректности уравнения Беллмана для одного этапа.

Составим вспомогательную задачу (1) - (3) для периода планирования *T*:

$$\sum_{i=1}^{n} k_{iT}(x_{iT}) \to \min,$$

$$\sum_{i=1}^{n} x_{iT} = \xi_T - z_{T-1},$$

$x_{iT} \in \{0\} \cup [m_{iT}, M_{iT}]$ *для* $i = 1,2,\ldots,n$.

Согласно доказательству, приведенному в §3.1, для решения данной задачи требуется решение задачи для *n-1* поставщика, для которого необходимо решение (1) - (3) для *n-2* поставщиков и т. д. Предложенная схема ДП позволяет производить необходимые рекурсивные вычисления. В итоге приходим к необходимости формирования начальных условий, от которых будут производиться дальнейшие расчеты. За счет начальных условий в данной схеме осуществляется связь между этапами.

Формируем задачу (1) – (3) для этапа *T-1*:

$$\sum_{i=1}^{n} k_{i(T-1)}(x_{i(T-1)}) \to \min,$$

$$\sum_{i=1}^{n} x_{i(T-1)} = \xi_{(T-1)} + \xi_T - z_{T-2},$$

$x_{i(T-1)} \in \{0\} \cup [m_{i(T-1)}, M_{i(T-1)}]$ *для* $i = 1,2,\ldots,n.$

Из решения данной задачи можно получить информацию о стоимости поставки любого количества продукции $a = \{0,1,2,\ldots,\xi_{T-1} + \xi_T\}$.

Если после удовлетворения потребностей этапа *T-1* остается некоторый объем продукции $z_{T-1} = \{0,1,2,\ldots,\xi_T\}$, то он переходит в следующий этап, на начало этапа *T* имеем z_{T-1} единиц продукции. Стоимость поставки соответствующего количества продукции в периоде *T-1* известна, но при переходе к периоду *T* она должна быть увеличена на затраты, связанные с хранением товара. Таким образом формируются начальные условия для этапа *T*. Для решения задачи на этапе *T-1* требуются начальные условия, сформированные на основании вычисления минимальной стоимости поставок для этапа *T-2* и т.д. Формирование начальных условий на первом этапе производится как в схеме $ДП_1$.

2.3 Трудоемкость алгоритмов

Утверждение 1. Пусть для вычисления значения функции $k_i(x_i)$ для любых $i = 1,\ldots,n, \quad x_i \in [m_i, M_i]$ требуется не более *C* операций, где *C* – *const*, тогда трудоемкость алгоритма $ДП_1$ равна $O(A \cdot \sum_{p=1}^{n} (M_p - m_p + 2))$.

Доказательство: Алгоритму $ДП_1$ при каждом значении *a* для нахождения минимальной стоимости $\varphi(p,a)$ требуется не более, чем M_p - m_p + *2* операций сравнения, таким образом, общее количество операций сравнения, которые необходимо выполнить для решения задачи (1) – (3), ограничено величиной

$$A \cdot \sum_{p=1}^{n} (M_p - m_p + 2).$$

Утверждение 2. Пусть для вычисления значения функции $k_{it}(x_{it})$ для любых $i = 1,...,n, \quad t = 1,...,T, \quad x_{it} \in [m_{it}, M_{it}]$ требуется не более *C* операций, где *C* – *const*, тогда трудоемкость алгоритма ДП равна $O(A \cdot T \cdot \max_{1 \le t \le T} \sum_{p=1}^{n} (M_{pt} - m_{pt} + 2))$.

Доказательство: Алгоритму ДП в любом периоде планирования *t* требуется не более $A \cdot \sum_{p=1}^{n} (M_{pt} - m_{pt} + 2)$ операций сравнения. Таким образом, за *T* периодов планирования будет произведено не более $A \cdot T \cdot \max_{1 \le t \le T} \sum_{p=1}^{n} (M_{pt} - m_{pt} + 2)$ операций сравнения.

3. ГЕНЕТИЧЕСКИЕ АЛГОРИТМЫ

Одним из широко используемых в настоящее время классов эвристических алгоритмов оптимизации являются генетические алгоритмы (ГА), предложенные Голландом [5]. В основе ГА лежит идея моделирования развития популяции живых организмов в процессе эволюции или при искусственной селекции. Принципы наследственности, изменчивости и отбора, сформулированные Ч. Дарвином, в ГА реализуются в построении новых решений - потомков посредством случайных операторов, модифицирующих полученные ранее точки в пространстве решений задачи оптимизации. Отбор таких точек для модификации производится с учетом значения в них целевой функции (чем выше приспособленность – тем больше шансы точки на отбор в качестве «родительской» особи).

3.1 Общая схема генетического алгоритма

Пусть оптимизационная задача состоит в максимизации (минимизации) функционала f на области допустимых решений D в пространстве решений X. Поиск решений задачи в ГА ведется с помощью популяции особей, причем каждая особь рассматривается как пара, состоящая из *генотипа* g и *фенотипа* $x(g)$, представляющая некоторую пробную точку пространства X. Генотип g – строка из l символов некоторого алфавита, используемого для кодировки решений в ГА. Функция $x(g)$ отображает генотип в пространство X, определяя *представление* решений в ГА.

Процесс поиска идет с учетом значения функции пригодности $\Phi(g)$, которая определяет степень «приспособленности» особей, и при $x(g) \in D$ имеет вид $\Phi(g) = \phi(f(x(g)))$, где $\phi: \boldsymbol{R} \to \boldsymbol{R}$ – некоторая монотонная (в случае задачи максимизации возрастающая, иначе - убывающая) функция, используемая для повышения эффективности поиска. Если $x(g) \notin D$, то Φ может действовать как функция штрафа.

Перенумеровав особи популяции, ее можно представить как вектор из генотипов $G = (g_1, g_2,\ldots,g_N)$, N – численность популяции. На практике в ГА достаточно хранить только генотипы особей, так как соответствующие им фенотипы всегда могут быть восстановлены.

Общая схема ГА

1. Построить начальную популяцию $П^0$;
2. Положить $t = 0$;
3. Пока не осуществится условие остановки, выполнять:
   - 3.1. Используя оператор селекции, выбрать родительские особи g и h;
   - 3.2. С помощью оператора скрещивания получить потомков g' и h';
   - 3.3. Применить к g' и h' оператор мутации;
   - 3.4. Добавить полученные генотипы в популяцию;
   - 3.5. Положить $t = t + 1$;
4. Ответ: лучшее из найденных решений.

Генотипы начальной популяции генерируются случайным образом в соответствии с заранее выбранным распределением вероятностей.

Один из наиболее распространенных вариантов организации ГА – *популяционная* схема воспроизводства, когда на каждой итерации ГА все генотипы новой популяции $П^{(t+1)}$ порождаются в соответствии с одним и тем же распределением вероятностей, определяемым текущей популяцией $П^{(t)}$ и распределениями вероятностей операторов селекции, мутации и кроссинговера.

Другой способ организации ГА – стационарная схема воспроизводства, когда на каждой итерации создается только одна новая особь, замещающая в популяции особь с наименьшим значением функции пригодности, или другую особь, выбранную по некоторому эвристическому правилу, и численность популяции остается постоянной. Опыт использования ГА показывает, что стационарная схема воспроизводства во многих случаях оказывается предпочтительнее популяционной с точки зрения среднего времени поиска решений приемлемого качества.

Построение генотипа новой особи (потомка) начинается с выбора пары родительских особей из текущей популяции G с помощью вероятностного оператора селекции, отдающего предпочтение наиболее пригодным особям. Выбор одной родительской особи не зависит от выбора другой.

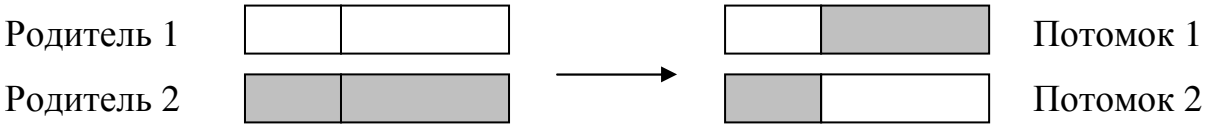
Среди наиболее распространенных операторов отбора – операторы *пропорциональной* и *турнирной* селекции. При пропорциональной селекции вероятность $P\left(g_k^{(t)}\right)$ выбора особи $g_k^{(t)}$ из популяции $\Pi^{(t)}$ пропорциональна ее пригодности:

$$P\left(g_k^{(t)}\right)=\frac{\Phi\left(g_k^{(t)}\right)}{\sum_{j=1}^{N}\Phi\left(g_j^{(t)}\right)}.$$

При использовании данного оператора предполагается, что функция пригодности принимает только положительные значения. Основной недостаток пропорциональной селекции состоит в том, что, как правило, после достаточно большого числа итераций все особи популяции получают достаточно близкие значения, и вероятности их выбора становятся почти одинаковыми.

Оператор турнирной селекции с размером турнира s при построении очередного решения извлекает из текущий популяции s особей с равномерным распределением и выбирает лучшую из них. Избирательность этого способа не изменяется в процессе работы алгоритма и не зависит от вида функции пригодности. Посредством выбора размера турнира s можно регулировать интенсивность селекции.

К выбранным генотипам с фиксированной вероятностью p_{cross} применяется оператор кроссинговера, который заменяет часть генов одного генотипа генами другого. Оператор *одноточечного* кроссиноговера действует по схеме:



Случайным образом выбирается точка разрыва j от 1 до $l-1$. Каждый родительский генотип делится на две части (первая – с номерами генов, не превосходящими j, во вторую попадают оставшиеся гены). Генотипы потомков получаются присоединением второй части одного из родителей к первой части другой родительской особи.

Двухточечный кроссинговер выбирает две точки разрыва, и родительские генотипы «обмениваются» той частью, которая находится между этими точками.

При использовании *униформного* кроссинговера обмен происходит в каждом гене независимо с вероятностью ½.

Полученный после кроссинговера генотип подвергается мутации, которая вносит случайные изменения в особь. Чаще всего в ГА используется оператор мутации, который в соответствии со схемой Бернулли с фиксированной вероятностью мутации p_{mut} изменяет значения генов, заменяя их случайными символами соответствующих алфавитов.

В качестве критерия остановки ГА используют ограничение по числу итераций, времени работы или достижение оптимума. Результатом работы ГА является лучшее из найденных решений.

3.2 Генетический алгоритм для задачи о поставках продукции

Для применения описанной в §3.1 схемы ГА при решении конкретной задачи необходимо определить способ представления решения в виде генотипа, функцию пригодности, операторы мутации, селекции и кроссинговера.

Опишем особенности ГА, разработанного для решения задачи (1') - (5').

Каждому решению ставится в соответствие генотип, представляющий собой вектор длины $n \cdot T$ и полученный с помощью алфавита из 4-х символов: 0, 1, 2, 3. Генотип условно можно разбить на T частей, соответствующих T периодам планирования. В каждой из этих частей содержится n генов, соответствующих

поставщикам. Таким образом, ген $g_{i+n\cdot(t-1)}$ кодирует информацию об открытии поставки предприятия i в периоде t.

$g_{i+n\cdot(t-1)} = 0 \Leftrightarrow x_{it} = 0,$

$g_{i+n\cdot(t-1)} = 1 \Leftrightarrow x_{it} = m_{it},$

$g_{i+n\cdot(t-1)} = 2 \Leftrightarrow x_{it} = M_{it},$

$g_{i+n\cdot(t-1)} = 3 \Leftrightarrow \{0\} \cup [m_{it},\ M_{it}],$

где $i = 1,\ldots,n,\ t = 1,\ldots,T.$

Согласно лемме 1 существует оптимальное решение, генотип которого имеет следующую структуру: в каждом периоде t существует единственное предприятие i, для которого $g_{i+n\cdot(t-1)} = 3$.

Если $g_{i+n\cdot(t-1)} \neq 3$, то ген с равной вероятностью может принимать любое значение от 0 до 2. Однако, если потребность на данном этапе близка к максимальному объему продукции, который могут предложить все поставщики, то имеет смысл отказаться от равномерного распределения и с большей вероятностью присваивать генам значение 2. Аналогично, чем меньше потребность, тем с меньшей вероятностью ген принимает значение 2.

При таком способе кодировки могут возникать особи, не соответствующие допустимым решениям. Возможны два варианта несоответствия: количество поставленной продукции может быть больше необходимого объема, а может быть недостающим, даже с учетом поставок, закодированных символом «3». В этом случае к функции пригодности особи добавляется штраф, пропорциональный отклонению от необходимого размера поставки.

Выбранная кодировка предполагает на каждом этапе одну поставку, размер которой не определен. Зная размеры остальных поставок и потребности на всех этапах, формируем новые потребности, удовлетворение которых будет производиться за счет тех поставщиков, для которых $g_{i+n\cdot(t-1)} = 3$. Размеры заказов, закодированных символом «3», необходимо определить так, чтобы получить как можно меньшее отклонение от требуемого количества товара и минимизировать затраты на его приобретение. Таким образом, вычисление

значения функции пригодности для каждой особи сводится к задаче управления запасами:

$$\sum_{t=1}^{T} k_t(x_t) + \sum_{t=1}^{T-1} h_t(z_t) \to \min, \qquad (1'')$$

$$z_t = z_{t-1} + x_t - \xi_t, \quad t = 1,2,...,T, \qquad (2'')$$

$$z_t \geq 0, \quad t = 1,2,...,T-1, \qquad (3'')$$

$$z_T = 0, \qquad (4'')$$

$$x_t \in \{0\} \cup [m_t, M_t], \quad t = 1,2,...,T. \qquad (5'')$$

Решение задачи (1'') - (5'') может быть реализовано с помощью алгоритма ДП.

Для формирования новой популяции родительские генотипы выбираются с помощью оператора 2-турнирной селекции.

С фиксированной вероятностью p_{cross} к родительским генотипам применяется оператор скрещивания. В данной задаче уместно использование одноточечного или двухточечного кроссинговера. В обоих случаях разрыв родительской особи происходит между двумя этапами. Положение точки разрыва выбирается случайным образом с равномерным распределением.

Действие оператора мутации поочередно распространяется на части генотипа, соответствующие периодам планирования. На каждом из этих этапов оператор мутации выполняет два главных шага:

Шаг 1. Оператор мутации с определенной вероятностью p_{mut} изменяет положение символа «3».

Шаг 2. В соответствии со схемой Бернулли с определенной вероятностью p_{mut} оператор мутации любой символ от 0 до 2 с равной вероятностью заменяет каким-либо символом алфавита от 0 до 2.

Полученные в результате применения операторов селекции, скрещивания и мутации особи добавляются в текущую популяцию, заменяя в ней особей с наихудшей пригодностью.

4. ВЫЧИСЛИТЕЛЬНЫЕ ЭКСПЕРИМЕНТЫ

В настоящем параграфе приведены результаты тестирования рассмотренных ранее алгоритмов и их сравнение с другими известными алгоритмами, использующимися для решения рассматриваемых задач. Для проведения экспериментов алгоритмы были реализованы на языке C++ и тестировались на ЭВМ с тактовой частотой процессора 1,8 ГГц. В качестве тестовых примеров использовались задачи, данные в которых генерировались случайным образом.

Для решения задачи (1) – (3) С.С. Чаухан и Ж.-М. Прот в [4] предложили эвристический алгоритм *NPSM* (*N Providers and Single Manufacturing unit*), находящий приближенное решение.

В [4] приводятся результаты численных экспериментов по решению одноэтапной задачи о поставках с помощью предложенного приближенного алгоритма *NPSM*.

По условиям задачи, описанной в [4] мы имеем 6 поставщиков, ниже для них приведены функции стоимости, а также нижняя и верхняя границы интервала возможных поставок:

a. $$k_a(x)=\begin{cases} 0 \text{ если } x=0 \\ p_a x+2 \text{ если } x>0 \end{cases}, \qquad m_a=20,\ M_a=100,$$

b. $$k_b(x)=\begin{cases} 0 \text{ если } x=0 \\ 4-e^{-p_b x} \text{ если } x>0 \end{cases}, \qquad m_b=30,\ M_b=80,$$

c. $$k_c(x)=\begin{cases} 0 \text{ если } x=0 \\ 7-e^{-p_c x} \text{ если } x>0 \end{cases}, \qquad m_c=10,\ M_c=50,$$

d. $$k_d(x)=\begin{cases} 0 \text{ если } x=0 \\ \dfrac{p_d x+2}{x+1} \text{ если } x>0 \end{cases}, \qquad m_d=40,\ M_d=120,$$

e. $$k_e(x)=\begin{cases} 0 \text{ если } x=0 \\ 1+p_e x \text{ если } x>0 \end{cases}, \qquad m_e=15,\ M_e=70,$$

f. $$k_f(x) = \begin{cases} 0 \text{ если } x = 0 \\ 5 - e^{-p_f x} \text{ если } x > 0 \end{cases}, \qquad m_f = 5,\ M_f = 60.$$

С помощью алгоритма $ДП_1$ было решено 20 задач (1) - (3). В каждой задаче вручную вводились параметры p_a, p_b, p_c, p_d, p_e и p_f. Решения, полученные точным алгоритмом, сравнивались с решениями, приведенными в [4]. Результаты эксперимента приведены в таблице 1 в приложении (в скобках указаны значения, полученные с помощью алгоритма $ДП_1$). Из этой таблицы видно, что решения, найденные с помощью алгоритма $ДП_1$ совпадают с точностью до второго знака с решениями, найденными приближенным алгоритмом. В задаче №17 замечено существенное расхождение полученных значений целевых функций (30,02 - для *NPSM* и 39,00 – для $ДП_1$). Векторы оптимальных поставок в обоих случаях одинаковы.

В ходе экспериментов исследована зависимость между временем работы алгоритма $ДП_1$ и верхней границей поставки - параметром *M*. Входным параметрам были присвоены следующие значения: $A = 10000$, $n = 100$, $k_i = 1$, $m_i = 100$, $i = 1,2,\ldots,n$.

В ходе эксперимента было проведено 20 тестов, значения *M* при этом изменялись от 0 до 20000. Результаты серии тестов представлены на рис. 1 в приложении.

Пока *M* меньше *A,* количество операций сравнения равно $A \cdot \sum_{p=1}^{n} (M_p - m_p + 2)$ зависимость близка к линейной, как только *A* становится больше *M,* количество операций сравнения становится равным $A \cdot \sum_{p=1}^{n} (A - m_p + 2)$, данная величина является постоянной, поэтому график принимает соответствующий вид.

После обобщения схемы динамического программирования на несколько этапов алгоритм ДП был запрограммирован. Для проверки работы алгоритма решения тестовых задач находились с помощью алгоритма ДП и программного пакета OPL Studio (CPLEX MIP Solver 6.5). Во всех рассмотренных примерах

полученные ответы совпадали. Одновременно проводился вычислительный эксперимент, в ходе которого сравнивалось время, необходимое для решения задачи (1') - (5') с помощью алгоритма ДП и CPLEX MIP Solver 6.5. Эксперимент проводился в двух направлениях:

1. Количество периодов постоянно (T=10), количество поставщиков возрастает, на протяжении эксперимента n изменялось от 300 до 10000;

2. В этом случае количество периодов возрастает и изменяется от 300 до 1000, количество поставщиков на протяжении эксперимента постоянно.

Результаты приведены в таблице:

В двух последних столбцах каждой таблицы указано время решения задачи, (в секундах).

n	T	CPLEX	ДП
2000	10	2,59	1,734
3000	10	3,30	2,594
4000	10	4,39	3,453
5000	10	5,55	4,313
10000	10	12,53	8,672

n	T	CPLEX	ДП
10	300	0,55	10,3
10	400	0,66	18,3
10	500	1,15	28,7
10	600	1,43	41,5
10	700	1,60	56,5

На задачах малых размерностей время нахождения решения в обоих случаях менее секунды. При постоянном T и увеличении n ДП дает лучшие результаты (при n=10000 алгоритм ДП находит ответ в 1,49 раза быстрее). Однако при увеличении количества этапов появляется существенная разница во времени, затраченном на решение задачи, доминирует CPLEX MIP Solver 6.5 (при T = 700 алгоритм ДП работает в 35,3 раза дольше).

На работу ГА значительное влияние оказывают значения параметров. В ходе экспериментов опробованы различные настройки параметров ГА.

Во всех тестовых задачах для ГА n = 10, T = 10. Остальные параметры задавались случайным образом из определенного интервала. Каждый раз при запуске программы ГА стартовал 100 раз, после завершения работы программа

выдавала среднее арифметическое ста полученных решений и количество найденных оптимальных решений.

Определение влияния параметра p_{mut} на работу алгоритма. В ходе эксперимента p_{mut} изменялся от 0,001 до 0,3. Численность популяции $N = 90$. На основании полученных данных построен график (рис. 2), показывающий влияние уровня мутации на отклонение найденного решения от оптимального. Наилучших результатов удалось достичь при $0{,}0114 < p_{mut} < 0{,}0256$. При значениях p_{mut}, входящих в данный интервал, среднее отклонение от оптимального решения составляет не более 1,2%, а вероятность нахождения оптимального решения - 70%.

При $p_{mut} = 0{,}015$ и описанных выше условиях определялось влияние численности популяции N на работу ГА. Количество особей в популяции изменялось от 15 до 400. Наименьшее отклонение от оптимального решения достигалось при $N \in (40,130)$. В этом же интервале наблюдалась наибольшая вероятность нахождения оптимального решения. Результаты представлены на рис. 3 в приложении.

В ГА с некоторой вероятностью p_{cross} используется оператор скрещивания. Проведен эксперимент по определению зависимости между полученным решением и вероятностью кроссинговера. В ходе эксперимента значение p_{cross} изменялось от 0 до 1. В таблице 5 приведены результаты эксперимента. Наличие или отсутствие оператора скрещивания не оказывает сильного влияния на среднюю стоимость решения, полученного ГА, оно достаточно близко к оптимальному. Однако, при использовании кроссинговера в данном примере увеличивается количество найденных оптимальных решений. На основании полученных экспериментальных данных построен график, показывающий влияние вероятности кроссинговера на частоту нахождения оптимального решения (рис. 4). В таблице 5 указана частота нахождения оптимального решения при ста запусках ГА.

Время работы построенного ГА существенно превышает время, затраченное на поиск решения алгоритмом ДП. Так, например, для решения

задачи (1) – (5) с $n = 20$, $T = 10$ алгоритму ДП требуется 0,11 секунды, а ГА находит оптимальное решение с вероятностью 90% в среднем за 33,7 секунды (продолжительность одного запуска колебалась от 19 до 45 секунд). Дополнительные эксперименты показали, что при увеличении размерности задачи разница во времени работы алгоритмов становится еще более ощутимой.

Представляется нецелесообразным применение построенного варианта ГА к решению задач данной структуры. Можно предположить, что подобная ситуация является типичной для задач, разрешимых псевдополиномиальным алгоритмом.

ЗАКЛЮЧЕНИЕ

Проведено исследование одноэтапной задачи о поставках продукции, реализован алгоритм динамического программирования для решения данной задачи.

Одноэтапная задача обобщена для нескольких периодов планирования, приведены содержательная постановка и формальная запись многоэтапной задачи о поставках продукции. Предложен алгоритм ее решения, основанный на технике динамического программирования, выполнена оценка его трудоемкости. Разработан и реализован генетический алгоритм для решения рассматриваемой задачи, схема которого учитывает особенности постановки.

В ходе вычислительных экспериментов установлено, что существуют задачи, на которых лучшие результаты дает использование программного пакета OPL Studio, и задачи, при решении которых преимущество имеет предложенный алгоритм динамического программирования. Рассмотренные примеры показали, что любой из этих методов работает эффективнее, чем построенный генетический алгоритм. Однако, возможно, что применение генетического алгоритма окажется полезным при решении многоэтапной задачи о поставках продукции в случае многих потребителей.

СПИСОК ЛИТЕРАТУРЫ

1. Гэри М., Джонсон Д.. Вычислительные машины и труднорешаемые задачи. М.:Мир,1982.

2. Таха Х. Введение в исследование операций. М.:Мир,1985.

3. S.S. Chauhan, A.V. Eremeev, A.A. Kolokolov, V.V. Servakh. Concave cost supply management for single manufacturing unit. In: Supply chain optimization. Ed. By A. Dolgui, J. Soldek, O. Zaikin. Springer, 2005. 167-174.

4. S.S. Chauhan, J.-M. Proth. The concave cost supply problem. European Journal of Operation Research 148. (2)(2003), 374-383.

5. J. H. Holland. Adaptation in Natural and Artificial System. University of Michigan Press, Ann Arbor, 1975.