УДК 004.94

АНАЛИЗ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ

Галькина Е.В., Мунько В.В.

Омский государственный технический университет, г. Омск, Россия, prikladnay@yandex.ru;

Аннотация. Проанализированы задачи дискретной оптимизации, а также методы их решения. Разработана математическая модель и генетический алгоритм нахождения связи между дисциплинами и компетенциями в учебном плане.

Ключевые слова. Дискретная оптимизация, генетический алгоритм, учебный план.

В настоящее время задачи дискретной оптимизации актуальны и востребованы во многих сферах и областях. Для их эффективного решения используются различные методы, включая генетические алгоритмы, основанные на идеях биологической эволюции. Они являются эффективным инструментом для поиска оптимального решения в случаях, где традиционные алгоритмы могут быть неэффективными или непригодными.

Задача дискретной оптимизации (1) заключается в поиске такого вектора $x=(x_1,x_2,\ldots,x_n)\in\mathbb{R}$, который доставляет максимум или минимум целевой функции $f(x)=f(x_1,x_2,\ldots,x_n)$, определённой на конечном или счётном множестве D, элементы которого называются допустимыми решениями:

$$f(x)\cdot extr, x\in D; \quad (1)$$

Если множество D задаётся системой ограничений (2):

$$g_i(x)\le 0, i=1,\ldots p,$$

$$g_i(x)=0, i=p+1,\ldots m, \quad (2)$$

$$x_i\in D_i\subset\mathbb{R}, i=1,\ldots n,$$

где каждое D_i – либо конечное множество, содержащее не менее двух элементов, либо счётное множество, то задача (1) называется задачей дискретного математического программирования.

Принято выделять следующие классы задач дискретной оптимизации: транспортная задача и ее варианты, задачи с неделимостями, экстремальные комбинаторные задачи, задачи на неклассических областях, задачи с разрывной целевой функцией.

В классическом виде, транспортная задача формулируется как задача организации оптимального маршрута доставки однородного товара из пунктов хранения в пункты потребления с использованием заданных транспортных средств, с учетом статических данных и линейного подхода. Такие условия присущи основным задачам данного типа, включая задачи назначений и задачи направления потоков в сетях.

Задачи с неделимостями – это математические модели прикладных задач, переменные в которых представляют физически неделимые величины. Такие задачи описывают, например, планирование выпуска неделимых видов продукции или использования неделимых производственных факторов, это задачи распределения ресурсов и капиталовложений, сетевого планирования и управления, производственного планирования и т. п.

В комбинаторных задачах оптимизируется функция, заданная на конечном множестве, элементами которого служат выборки (перестановки) из n объектов. Из комбинаторных задач, имеющих большое прикладное значение, следует отметить задачу о коммивояжёре и задачи теории расписаний. При постановке комбинаторных задач в виде задач целочисленного математического программирования часто вводятся булевы переменные $x_i\in\{0,1\}$, носящие логический характер:

– $x_i=1$, если выполняется некоторое условие;

– $x_i=0$, в противном случае.

Задачи на неклассических областях представляют собой задачи нахождения экстремума линейной функции на невыпуклой или несвязной области, задаваемой, например, с использованием логических условий вида «либо-либо».

В задачах с разрывными целевыми функциями, напротив, допустимое множество – выпуклый многогранник, но целевая функция не является непрерывной. К появлению подобных целевых функций приводит, в частности, учет в моделях постоянных затрат, которые должны быть произведены независимо от объема производства.

Далее будут рассмотрены методы решения задач целочисленного линейного программирования, подразумевающие временное отбрасывание условий дискретности и решение соответствующих непрерывных задач. В связи с этим кратко осветим основные из теории линейного программирования и сформулируем постановку задачи.

Задача линейного программирования формулируется следующим образом:

$$c,x\to min; \quad (3.1)$$

$$Ax=b; \quad (3.2)$$

$$x\ge 0; \quad (3.3)$$

где $A=(a_{ij})$ – матрица размером $m\times n$. При этом $m<n, c\in\mathbb{R}^n, b\in\mathbb{R}^m$. Будем считать, что ранг матрицы A равен m. Столбцы матрицы А обозначим $a^j, j=1,\ldots,n.$

Допустимым решением или планом задачи называется решение $x\in\mathbb{R}^n$ системы уравнений (3.2), удовлетворяющее условиям неотрицательности (3.3).

Множество допустимых решений D

$$D \subseteq \mathbb{R}^n, D = \{x \in \mathbb{R}^n \mid f_i(x) \le 0, i = 1,\ldots,m\} \quad (4)$$

Решение системы (3.2) $x \in \mathbb{R}^n$ называется базисным, если система столбцов a^j матрицы A, соответствующих ненулевым координатам вектора x, линейно независима. Базисное решение системы линейных уравнений (3.2) называется опорным вектором задачи линейного программирования (3.1) – (3.3).

Вектор $x \in D$ называется опорным планом, если система столбцов a^j матрицы A соответствующих положительным координатам вектора x, линейно независима. Таким образом, опорный план задачи (3.1) - (3.3) – это опорный вектор, удовлетворяющий условию неотрицательности (3.3), то есть являющийся планом задачи.

Теорема 1. Если задача линейного программирования допустима ($D \neq \varnothing$), то у неё существует опорный план.

Теорема 2. Если задача линейного программирования имеет решение, то среди её опорных планов найдется оптимальный план.

Любая задача дискретной оптимизации может быть решена полным перебором. Такой метод относится к классу методов поиска решения исчерпыванием всевозможных вариантов. Сложность полного перебора зависит от количества всех возможных решений задачи. Если пространство решений очень велико, то полный перебор может не дать результатов в течение нескольких лет или даже столетий.

Используя метод отсечения, сначала решается задача линейного программирования без условия целочисленности. Если полученный ответ удовлетворяет условию целочисленности, то задача решена. В противном случае к ограничениям задачи добавляется новое ограничение, обладающее следующими свойствами:

– оно должно быть линейным;

– оно должно отсекать найденный оптимальный нецелочисленный план;

– оно не должно отсекать ни одного целочисленного плана.

Такие ограничения называются правильными отсечениями. После введения нового ограничения вновь решается задача линейного программирования. Если вновь полученный план целочисленный, то задача решена. Если это не так, то к задаче добавляется новое ограничение. Процесс повторяется до тех пор, пока полученный оптимальный план не будет полностью целочисленным.

Метод ветвей и границ является развитием метода полного перебора, в отличие от последнего — с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений. Для метода ветвей и границ необходимы две процедуры: ветвление и нахождение оценок (границ).

Процедура ветвления состоит в разбиении множества допустимых значений переменной x на подобласти (подмножества) меньших размеров. Процедуру можно рекурсивно применять к подобластям. Полученные подобласти образуют дерево, называемое деревом поиска или деревом ветвей и границ. Узлами этого дерева являются построенные подобласти (подмножества множества значений переменной x).

Процедура нахождения оценок заключается в поиске верхних и нижних границ для решения задачи на подобласти допустимых значений переменной x .

В основе метода ветвей и границ лежит следующая идея: если нижняя граница значений функции на подобласти A дерева поиска больше, чем верхняя граница на какой-либо ранее просмотренной подобласти B , то A может быть исключена из дальнейшего рассмотрения (правило отсева).

Обычно минимальную из полученных верхних оценок записывают в глобальную переменную m ; любой узел дерева поиска, нижняя граница которого больше значения m , может быть исключён из дальнейшего рассмотрения.

Если нижняя граница для узла дерева совпадает с верхней границей, то это значение является минимумом функции и достигается на соответствующей подобласти.

Метод динамического программирования – это способ решения сложных задач путём разбиения их на более простые подзадачи. Он применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной.

Условия для решения задачи методом динамического программирования:

1. задача может быть разбита на подзадачи более простой структуры (меньшей размерности);

2. ограничения общего вида должны быть в небольшом количестве (в идеале одно).

Как правило, чтобы решить поставленную задачу, требуется решить отдельные части задачи (подзадачи), после чего объединить решения подзадач в одно общее решение. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Это особенно полезно в случаях, когда число повторяющихся подзадач экспоненциально велико.

Эвристический алгоритм – это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев. В действительности может быть даже известно (то есть доказано), что эвристический алгоритм формально неверен. Его всё равно можно применять, если при этом он даёт неверный результат только в отдельных, достаточно редких и хорошо выделяемых случаях или же даёт неточный, но всё же приемлемый результат.

Проще говоря, эвристика – это не полностью математически обоснованный (или даже «не совсем корректный»), но при этом практически полезный алгоритм.

Генетический алгоритм – представитель эволюционных алгоритмов, который базируется на прин-

ципах и механизмах, аналогичных тем, которые происходят в процессе естественного отбора в природе. Он процедурно применяет операторы скрещивания, мутации, отбора и размножения для итеративного улучшения популяции решений к заданной оптимизационной цели.

Генетические алгоритмы с большей вероятностью, чем традиционные алгоритмы способны находить глобальные оптимумы. Их можно применять в задачах, которые сложно формализовать математически, либо даже совсем не имеющие математического описания. Однако необходима специфичная формализация исходной задачи на уровне генов и хромосомы, а также операторов отбора, скрещивания и мутации.

Далее будет рассмотрена математическая модель взаимосвязи дисциплин и компетенций, на которой будет представлен генетический алгоритм нахождения связи между дисциплинами и компетенциями в учебном плане.

Будем рассматривать следующую структуру учебного плана:

$k = 1$ – блок 1 (дисциплины);

$k = 2$ – блок 2 (факультативы).

Блок «Практика» и «Государственная итоговая аттестация» не учитываем, так как структурные элементы этих блоков обязательны. Введем следующие обозначения:

j – номер дисциплины (без учета практик и ГИА), $j = \overline{1,n}$;

i – номер компетенции, $i = \overline{1,m}$;

$k_{ij} \in [0,1]$ – коэффициент закрытия i-й компетенции j-й дисциплиной;

v_j – количество зачетных единиц, выделяемых на j-ю дисциплину;

$\underline{V}^1$ – нижняя граница зачетных единиц блока 1 учебного плана (без учета обязательных дисциплин);

$\overline{V}^1$ – верхняя граница зачетных единиц блока 1 учебного плана (без учета обязательных дисциплин);

J^i – множество индексов дисциплин, каждая из которых обеспечивает закрытие i-й компетенции, $i = \overline{1,m}$, не менее, чем на величину α;

J_1 – множество индексов дисциплин, использование которых возможно в формировании блока 1;

J_2 – множество индексов дисциплин, используемых в формировании блока 2;

Если j-я дисциплина включена в k-й блок учебного плана, то переменная x_{kj} будет принимать значение 1, иначе 0.

Решением задачи является матрица X размерности $2 \times n$

1. Введем ограничения на единственность вхождения j-й дисциплины в учебный план:

$$\sum_{k=1}^{2} x_{kj} \le 1, \quad j = \overline{1,n}; \tag{5}$$

2. Введем ограничения, обеспечивающие закрытие i-й компетенции хотя бы одной дисциплиной на величину не менее, чем на величину α:

$$\sum_{j \in J^i} \sum_{k=1}^{2} x_{kj} \ge 1, \; i = \overline{1,m}; \tag{6}$$

3. Введем ограничения на количество зачетных единиц по 1 блоку учебного плана:

$$\underline{V}^1 \le \sum_{j \in J_1} v_j x_{1j} \le \overline{V}^1; \tag{7}$$

Требуется максимизировать обеспеченность каждой i-й компетенции в учебном плане:

$$\sum_{j=1}^{n} k_{ij} (x_{1j} + x_{2j}) \to max, \qquad i = \overline{1,m}. \tag{8}$$

Оптимальное решение для каждого критерия обозначим $y^{i*} = \left(y_1^{i*}, \ldots, y_l^{i*}, \ldots, y_N^{i*} \right)$.

Для нахождения оптимального решения применим принцип работы генетического алгоритма.

Шаг 1. Создаем начальную популяцию (нулевое поколение).

Каждая популяция включает набор хромосом – набор потенциальных решений. Генерируем матрицы (произвольное количество) размерностью $k \times n$, где k – количество блоков учебного плана, n – количество дисциплин (в нашем случае $k = 2$, так как для моделирования необходимы только блоки «Дисциплины» и «Факультативы»). Пусть h_u – хромосома в каждом поколении, $u = \overline{1,t}$. В каждом поколении должно быть не менее 2 хромосом ($t \ge 2$). Каждое значение в хромосоме называется геном – значение переменной x_{kj}, которое принимает значение равное 1, если дисциплина включена в k-й блок учебного плана, иначе – 0.

Примечание: при генерации произвольных матриц сразу учитываем ограничение (5). Правило: в 1 строке произвольно задаем 0 или 1, во второй строке 1 можно ставить только там, где в первой были 0.

Шаг 2. Вычисляем приспособленность каждой хромосомы в нулевом поколении.

В качестве функции приспособленности выступает целевая функция – нахождение максимальной обеспеченности каждой компетенции в учебном плане (8). Кроме того, при оценке приспособленности учитываются ограничения (5)-(7). Различают мягкие и жесткие ограничения. При невыполнении мягкого ограничения решение может являться допустимым. Ограничения (6), обеспечивающие закрытие каждой компетенции хотя бы одной дисциплиной на величину не менее, чем на величину α, будем считать мягкими. К жёстким ограничениям отнесем ограничения (5), (8): ограничения на единственность вхождения каждый дисциплины в 1 или 2 блок учебного плана и ограничение на количество зачетных единиц по блоку «Дисциплины». Каждой компоненте приспособленности присваиваем коэффициент важности от 0 до 1.

В задаче «Связь дисциплин и компетенций» для каждой компетенции составляем целевую функцию (количество задач равно заданному количеству ком-

петенций), при этом ограничения для каждой задачи остаются одинаковыми.

Приспособленность вычисляется один раз для начальной популяции, а затем для каждого нового поколения после применения операторов отбора, скрещивания и мутации. Поскольку приспособленность любой хромосомы не зависит от всех остальных, эти вычисления можно производить параллельно.

Шаг 3. Применяем *оператор отбора* хромосом для создания нового поколения. Процесс отбора основан на оценке приспособленности каждой хромосомы в популяции. Исключаем плохо приспособленные хромосомы (недопустимые решения или решения, чьи оценки ниже).

Шаг 4. Применяем *оператор скрещивания.* Скрещивание реализуется в результате обмена генов. Обычно для этого берутся две хромосомы, и части их хромосом меняются местами, в результате чего создаются две новые хромосомы, представляющие двух потомков.

Применяем метод равномерного скрещивания. При равномерном скрещивании каждый ген обоих родителей определяется независимо путем случайного выбора с равномерным распределением. Когда выбирается половина генов, оба родителя имеют одинаковые шансы повлиять на потомков.

Шаг 5. Применяем *оператор мутации*. Оператор мутации вносит случайные изменения в один или несколько генов хромосомы.

Применяем метод «Инвертирование бита». Для двоичной хромосомы можно случайным образом выбрать ген и инвертировать его, т.е. взять двоичное дополнение (например, 101**1**01 $\rightarrow$ 101**0**01). Правило инвертирования бита: единица из первой строки меняется с нулем из второй строки соответствующего столбца. Тем самым сохраняется жёсткое ограничение (5).

Шаг 6. Вычисляем приспособленность каждой хромосомы нового поколения путем возврата к шагу 2.

Повторные операции отбора, скрещивания и мутации приводят к появлению лучших хромосом, передающих эти гены следующему поколению, так как оптимальное решение задачи может быть собрано из небольших структурных элементов (генов), и чем их больше, тем быстрее находим оптимальное решение.

Цикл (отбор, скрещивание, мутация) продолжается, пока не будут выполнены условия остановки:

– достигнуто максимальное заданное количество поколений;

– на протяжении нескольких последних поколений не наблюдается заметных улучшений. Для этого запоминаем наилучшую приспособленность, достигнутую в каждом поколении, и сравниваем наилучшие текущее значения со значениями в нескольких предыдущих поколениях. Если разница меньше заранее заданного порога, то алгоритм завершит работу.

Шаг 7. Если условия остановки выполнены, выбираем хромосому с максимальной приспособленностью, т.е. находим оптимальное решение.

В заключении стоит отметить, что различные задачи могут иметь свои особенности и требовать специфических операторов и параметров, таких как выбор функции приспособленности, операторы скрещивания и мутации, методы отбора и оценки поколений. Это делает каждую программную реализацию уникальной и специально адаптированной к решению конкретной задачи.

Разработка программного обеспечения для апробации подходов и алгоритмов решения задач дискретной оптимизации обусловлена теоретической и практической важностью задач, встречающихся в различных областях науки и техники. В терминах дискретной оптимизации формулируются многие важные задачи экономики, управления, планирования, проектирования и т.д.

Существует возможность внедрения разработанных подходов, методов и алгоритмов в отдельные модули программ обучения бакалавриата, магистратуры, ДПО.

Для дальнейшего исследования есть возможность модифицировать представленный генетический алгоритм, применив различные операторы отбора, скрещивания и мутации. В последствии необходимо провести анализ полученных алгоритмов на основе скорости сходимости и выявить наилучшую комбинацию методов.

Литература

1. Т 98 Тюхтина А.А. Методы дискретной оптимизации: Часть 1: Учебно-методическое пособие. – Нижний Новгород: Нижегородский госуниверситет, 2014. – 62 с.

2. Корбут А.А., Финкельштейн Ю.Ю. Дискретное программирование. – М.: Наука, 1969. – 368 с.

3. Карманов, В. Г. Математическое программирование: учеб. пособие / В. Г. Карманов. – М.: ФИЗМАТЛИТ, 2004. – 264 с.

ANALYSIS OF GENETIC ALGORITHMS FOR SOLVING DISCRETE OPTIMIZATION PROBLEMS

E.V. Galkina, V.V. Munko

Omsk State Technical University, Omsk, Russia, prikladnay@yandex.ru ;

Abstract. Discrete optimization problems, as well as methods for solving them, are analyzed. A mathematical model and a genetic algorithm for finding connections between disciplines and competencies in the curriculum have been developed.

Keywords. Discrete optimization, genetic method, curriculum.