Решение задачи о рюкзаке методами генетического программирования

Ледак Е.В.

СОДЕРЖАНИЕ

Введение

Современный уровень научных исследований и развитие рыночной экономики предполагает быструю и эффективную обработку информации. Множество прикладных задач относится к теории дискретного программирования: например, задача коммивояжера и задача о рюкзаке. Точное решение таких задач возможно найти за конечное число итераций. Поэтому, к таким задачам применим метод простого перебора. Однако, в случае, когда множество допустимых решений состоит из очень большого числа точек, нахождение решения этим методом может затянуться на годы. Поэтому, для решения задач дискретного программирования были разработаны специальные, в том числе, и переборные методы, которые пытаются сузить количество перебираемых точек.

Существует два подхода к решению задач дискретного программирования. Первый получил название комбинаторного и основан на целенаправленном, частичном переборе частных решений. Методы этой группы делятся на точные и приближенные. К точным относится метод ветвей и границ. Приближенные алгоритмы основаны на некоторых эвристических идеях, учитывающих специфику решаемой задачи. Естественно, приближенные методы не дают точного решения, но поиск решения такими методами занимает в десятки раз меньше времени.

Второй подход — методы отсечения. Идея этих методов состоит в том, что множество допустимых решений образует некоторый выпуклый замкнутый контур. В противном случае, такой контур строится искусственно. Экстремальная задача решается на этом множестве путем отсечения частей множества. Часть, не содержащая экстремального решения отбрасывается по некоторому алгоритму. Оставшаяся часть подвергается рассечению до тех пор, пока не останется контур, не поддающийся дальнейшему рассечению и содержащий оптимальное решение. Представителем данной группы является алгоритм Гомори для решения задач целочисленного линейного программирования.

1. ОПИСАНИЕ ЗАДАЧИ О РЮКЗАКЕ

Турист собирается в поход. У него имеется рюкзак, в который можно упаковать вещи общим весом в p единиц, и набор из n неделимых вещей, которые турист хотел бы взять с собой в поход. Каждая вещь обладает весом a_i единиц и стоимостью c_i денежных единиц. Требуется найти такой набор вещей, который был бы максимальным по суммарной стоимости, но общий вес которого не превышал вместимости рюкзака.

Пусть

$$x_i = \begin{cases} 1, \textit{если } i-\textit{ая вещь упакована}; \\ 0, \textit{если } i-\textit{ая вещь не упакована}. \end{cases}$$

Тогда величиной, характеризующей стоимость некоторой i-ой вещи, будет $c_i x_i$, а величиной, характеризующей вес некоторой i-ой вещи, будет $a_i x_i$. Набор в целом характеризуется общей стоимостью и общим весом. Общая стоимость является суммой всех величин, характеризующих стоимости вещей. В свою очередь, общий вес является суммой всех величин, характеризующих веса вещей.

Постановка задачи в математическом виде будет выглядеть следующим образом:

$$\sum c_i x_i \to \max;$$
$$\begin{cases} \sum a_i x_i \le p; \\ x_i \in \{0, 1\}; \\ i = \overline{1, n}. \end{cases}$$

2. МЕТОД МОНТЕ-КАРЛО ПРОТИВ МЕТОДА ЭЛИТНЫХ ПОПУЛЯЦИЙ

В проведенной работе было проведено сравнение методов Монте-Карло и метода генетического программирования, основанного на элитных популяциях.

Для начала, необходимо сказать несколько слов о причинах сравнения именно этих алгоритмов, ведь наиболее традиционным для решения задачи о рюкзаке является метод ветвей и границ, а не метод Монте-Карло. В основе этого выбора лежит тот факт, что метод Ветвей и границ не может быть использован при использовании большого количества отбираемых вещей. При наличии в «кладовой» более 50 – 100 предметов метод ветвей и границ не превосходит по точности метод случайного выбора предметов, т.е. метод Монте-Карло. Очевидно, что программирование метода ветвей и границ представляет гораздо большую сложность, чем программирование метода Монте-Карло. Это обстоятельство и сыграло ведущую роль в выборе конкурента для генетического алгоритма. Конечно, существует множество разновидностей метода Монте-Карло, но автор на период создания программы с ними не был знаком. Кроме того, во время программирования метода элитных популяций автор столкнулся с непредвиденными осложнениями, что значительно сократило время, потраченное на поиск и выбор алгоритмов решения данной задачи.

3. ОПИСАНИЕ АЛГОРИТМОВ

3.1. Метод Монте-Карло.

1. Маленькая хитрость: метод случайного отбора предметов будет показывать лучшие результаты, если отсортировать все множество предметов, хранящихся в кладовой по критерию полезности. Полезность u i-го предмета находится как следующее отношение:

$$u_i = \frac{c_i}{p_i}, \text{ где}$$

c_i – *стоимость i-го предмета в кладовой,*

p_i – *масса i-го предмета в кладовой.*

2. Начиная с самой полезной вещи ($u_j = \max u_i$, где $i=\overline{1,10}$) перебираем предметы, доступные для выбора. С вероятность 50% вещь попадает в искомый набор.

3. После попадания вещи в набор, количество вещей данного типа уменьшается на единицу. Вещь перестает быть доступной для выбора, как только количество вещей данного типа становится равным нулю.

4. Процесс продолжается до тех пор, пока в наборе не останется ни одной вещи, которая могла бы поместиться в рюкзаке.

3.2. Метод элитных популяций.

3.2.1. Общие положения.

В 1975г. Джон Холланд предложил алгоритм, заданием которого было моделирование процесса эволюции. Данный алгоритм преобразует две популяции: B(t), называемую базовой популяцией, и B(t+1), называемую популяцией потомков. В этих популяциях содержится одинаковое количество особей. На шаге первоначальной инициализации, базовая популяция заполняется сгенерированными случайным образом особями. На этапе оценки приспособленности для каждой из них подсчитывается значение функции приспособленности. После инициализации базовой популяции B(0)

переходим к основному циклу программы, в котором определен процесс искусственной эволюции.

В начале происходит репродукция, основанная на скрещивании случайно выбранных особей из базовой популяции. Вероятности выбора, однако, не являются одинаковыми: особи с большим значением функции приспособленности имеют большие шансы на репродукцию. Кроме того, происходит выбор с возвращением. В результате репродукции можно ожидать, что для скрещивания будет выбрано большее количество копий наиболее приспособленных особей. Выбранные особи помещаются в популяцию потомков B(1).

Далее, особи из B(1) подвергаются генетическим операциям – скрещиванию и мутации. В процессе скрещивания особи группируются в пары, и для каждой отдельной пары принимается решение о проведении скрещивания. Если решение положительно**,** то происходит скрещивание, и полученные особи-потомки, заменяют своих родителей. В случае отрицательного решения**,** пара остается без изменений. Вероятность проведения скрещивания $\boldsymbol{p_c}$ является параметром алгоритма. Далее, для каждой особи из B(1) производится мутация, являющаяся изменением ее генотипа. Полученные таким образом особи оцениваются вычислением функции приспособленности и образуют популяцию потомков B(1). Эта популяция становится новой базовой популяцией для следующего витка цикла. Происходит увеличение счетчика итераций *t*.

Процесс продолжается циклически до тех пор, пока не выполнятся условия остановки. Ими может быть, к примеру, выполнение заданного числа итераций, или также нахождение особи с относительно высоким значением функции приспособленности**.** Каждое прохождение основного цикла часто называется генерацией и рассматривается как одна его итерация, а состояние популяции B(t) в момент времени *t* называют поколением.

Рассмотрим более подробно понятия кодирования особей и генетических операции. Схема алгоритма, предложенного Холландом**,** абстрагируется от определенного способа кодирования особей и выполнения операций скрещивания и мутации, однако данный алгоритм наиболее известен при использовании бинарной кодировки. В данной версии хромосом является n-мерным вектором генов, каждый из которых является нулем или единицей (рис.3.1).

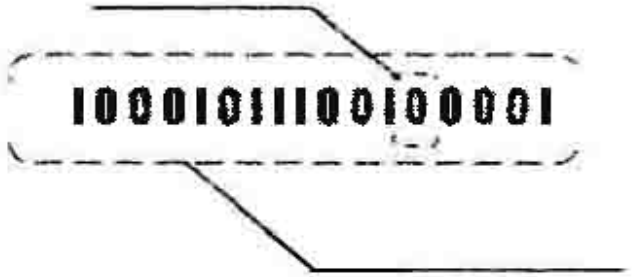

рис 3.1. Хромосома с бинарной кодировкой

Мутация особи: является оператором, производимым над каждым геном, по отдельности. Она основана на том, что с вероятностью p_m, величина которой является параметром алгоритма, значения гена изменяется на противоположное.

В свою очередь, в процессе скрещивания берут участие две родительские особи, ход этой операции подражает наблюдаемому в природе процессу изменений в ДНК, происходящим, во время генеративного размножения.

Родительские хромосомы разделяются на два фрагмента. Далее, первый фрагмент первой хромосомы склеивается со вторым фрагментом второй хромосомы, аналогично первый фрагмент второй хромосомы склеивается со вторым фрагментом первой хромосомы. Полученные таким образом цепочки генов образуют потомственные особи (рис.3.2).

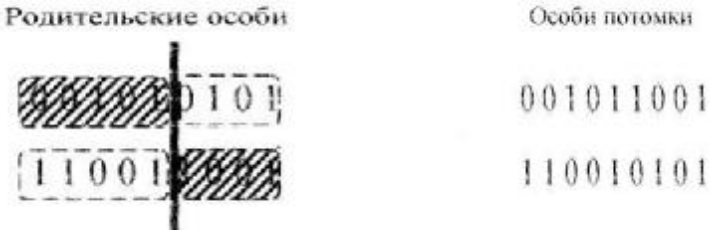

рис. 3.2. Скрещивание в бинарной кодировке

Так как все хромосомы содержат одинаковое число генов, то разделение должно происходить в одном и том же месте для обеих родительских хромосом. Место разделение выбирается случайно с равномерным распределением.

Оператор инверсии, примененный к отдельной хромосоме, меняет значение каждого из генов на противоположное.

Одним из важнейших аспектов генетического алгоритма является механизм выбора наиболее приспособленных особей для дальнейшей репродукции. Популяция B(1) состоит из особей, среди которых преобладают наиболее приспособленные. Для отбора особей для скрещивания Дж. Холланд предложил алгоритм, называемый пропорциональной репродукцией или репродукцией рулетки. Она описывается вероятностью на множестве элементов популяции B(t) с распределением

$$p_i(X) = \frac{\Phi(X_i)}{\sum_{j=1}^{N} \Phi(Y_j)}$$

где *X* обозначает особь,

Ф{X) — значение ее функции приспособленности,

N — количество выбранных вещей.

Для определения особи, подлежащей репродукции, производим случайный выбор (с равномерным распределение) числа *a* из промежутка

(0.1). Репродуцируется особь X^i. для которой выполняется следующее условие:

$$P_г(x^{i-1})<a<P_г(X^i).$$

Необходимо отметить, что средние значение функции приспособленности особей в популяции B(1) больше чем среднее средние значение приспособленности в базовой популяции B(0). Этого можно было ожидать, т.к. алгоритм построен таким образом., что во временную популяцию переводятся более приспособленные особи. Однако ценой, которую приходится за это платить, является уменьшение разнородности B(0) по отношению к B(1). Генетические операторы возвращают разнородность в B(1) не гарантируя, однако, улучшения, среднего значения приспособленности.

Когда остановить генетический алгоритм? Этот вопрос об условиях, остановки алгоритма является очень важным, и в выше приведенных рассуждениях не было получено ответа на него. Наилучшим (хоть и не совсем практичным) ответом будет ... никогда, т.к. генетический алгоритм является случайным, процессом и нельзя гарантировать достижения результата за конечное число генераций.

3.2.2. Некоторые аспекты реализации алгоритма.

В рамках решаемой задачи бинарная хромосома является удобным способом кодирования. Ее можно рассматривать как вектор упакованных в рюкзак предметов.

Для преобразования хромосом будем использовать генетические операторы, определенные выше, за исключением, быть может, оператора инверсии.

Вычисление целевой функции будем производить таким образом, чтобы наиболее приспособленными оказались наборы, имеющие максимальную суммарную стоимость и общий вес которых был бы меньше лимита рюкзака:

$$\Phi(X) = \begin{cases} \sum c_i x_i, \text{ если } \sum a_i x_i \le p; \\ 0, \text{ в противном случае}. \end{cases}$$

Будем использовать пропорциональную репродукцию. Но при этом требуется, чтобы значения функции приспособленности были положительными, что, вообще говоря, не является истинным для принятого вида функции. Для преодоления этого, применим, метод масштабированной приспособленности (fitness scaling), при котором вероятность репродукции описывается соотношением:

$$p_i(X) = \frac{\Phi(X_i) - \Phi_{\min}}{\sum_{j=1}^{N} \Phi(Y_j) \Phi_{\min}}$$

где Φ_{min} является, значением функции приспособленности наихудшей особи в популяции B(0):

$$\Phi_{\min} = \min \Phi(Y)$$

Благодаря этому мы избегаем опасности, получения «отрицательной вероятности».

Остановка алгоритма происходит, если в течение 100 последовательных генераций не наступило улучшение результата.

4. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Было проведено тестирование на случайном наборе вещей, результаты которого были сведены в таблицу (табл. 4.1.)

Table 4.1.

Item No	Item's value	Item's weight	1)	2)
1	32	9	0	0
2	29	6	1	0
3	99	3	1	3
4	23	1	1	4
5	36	6	1	0
6	74	10	1	0
7	6	9	1	0
8	44	6	1	0
9	57	3	1	0
10	19	4	1	0
11	99	1	1	7
12	30	2	0	2
13	35	1	1	4
14	2	3	0	0
15	50	3	0	1
16	39	4	0	0
17	19	6	0	0
18	12	9	0	0
19	54	2	0	2
20	39	9	0	0
21	100	7	0	0
22	81	6	0	0
23	1	1	0	0
24	27	9	0	0
25	21	4	0	0
26	80	7	0	0
27	49	4	0	0
28	78	3	0	5
29	20	3	0	0
30	56	6	0	0
Sum			521	1830
Freespace			0	
Quantity of iterations				

Результаты сравнения эффективности двух алгоритмов

1) – How many items of this type was selected with Monte-Carlo method?
2) – How many items of this type was selected Simple Genetic Algorithm?

В соответствии с выполнением вычислительного процесса были построены графики по выполнению каждого из алгоритмов (рис 4.1).

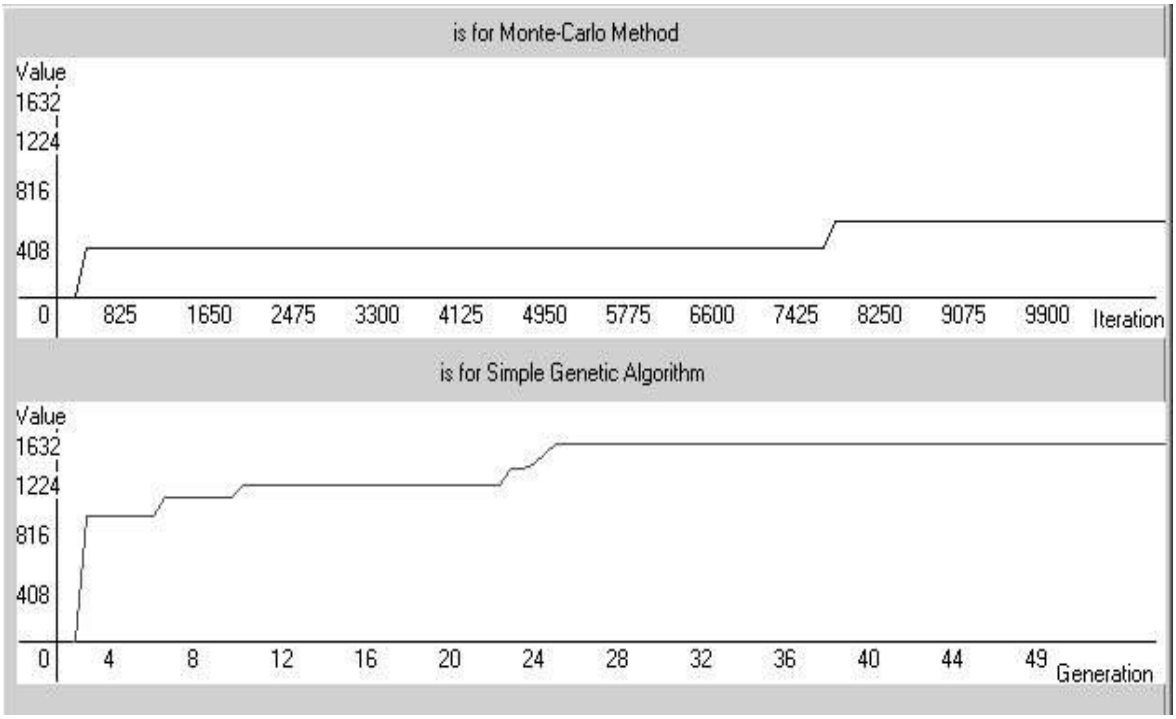

рис 4.1. Динамика вычислительного процесса по каждому из алгоритмов

На графике хорошо показано, что в простом генетическом алгоритме был использован метод элитных популяций: график имеет неубывающий вид. Это происходит потому, что в каждую последующую популяцию входят наиболее приспособленные особи из предыдущей популяции. Таким образом, приспособленность каждой последующей популяции получается не меньше, чем приспособленность предыдущих популяций.

ВЫВОД

В ходе выполнения данной работы были рассмотрены следующие методы решения задач дискретного программирования: простой генетический алгоритм и метод Монте-Карло. Простой генетический алгоритм позволяет найти лучшее, по сравнению с методом Монте-Карло, значение. Зато метод Монте-Карло производит поиск решения значительно быстрее. Однако, данный метод не всегда предоставляет решение, равное решению, полученному в результате функционирования простого генетического алгоритма.

Повысить точность решения с помощью простого генетического алгоритма можно, увеличив размер популяций. В нашем случае, размер популяции в два раза больше количества доступных туристу предметов. При увеличении численности популяций, возрастает время решения задачи, но найденное решение является лучшим, в большинстве случаев, чем решение, найденное при меньшей численности популяции.

Анализируя результаты тестирования, приведенные выше, можно сделать вывод, что простой генетический алгоритм лучше метода Монте-Карло. Это очевидно, так как если во втором алгоритме происходит случайный выбор элементов с определенной вероятностью, то в первом алгоритме происходит целенаправленный перебор, приводящий к неухудшению результата, накопленного на предыдущих этапах.

ЛИТЕРАТУРА

1. Воронковский Г.К., Махотило К.В., Петрашев С.Н., Сергеев С.А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности – Харьков: ”Основа”, 1997.