РЕАЛИЗИРАНЕ НА ГЕНЕТИЧНИ АЛГОРИТМИ В СРЕДАТА MATLAB

гл. ас. д-р Пенка Вълкова Георгиева
Бургаски свободен университет

IMPLEMENTATION OF GENETIC ALGORITHMS IN MATLAB

Penka Georgieva, PhD
Burgas Free University

***Abstract**: Some of the main characteristics of genetic algorithms (GA) and their implementation in MatLab are discussed in this paper. A conceptual model for applying GA in portfolio selection is proposed.*

***Key words:** genetic algorithms, portfolio selection, financial asset management.*

I. Генетични алгоритми и микро- генетични алгоритми

Генетичните алгоритми (GA) са предложени за пръв път от Холанд в [11] чрез следните две основни характеристики:

1) възможност за стрингово представяне на сложни структури;
2) използването на елементарни трансформации на стринговете с цел подобряване на представянето на тези структури.

По-късно Голдбърг [10] развива идеятана Холанд, като въвежда целева функция, наречена от него фитнес функция.

Генетичните алгоритми (GA) са алгоритми за търсене, в които наподобяването на процесите в естествената еволюция се реализира чрез имитиране на принципите на генетичните изменения в природата, като целта е намиране на оптимални решения в пространството от възможни решения (популация). В [10] е доказано, че GA са ефективно средство за подробно търсене в сложни пространства. Процедурите, използвани в GA, са основани на манипулиране на възможните решения за получаването на по-добри решения.

Всеки GA започва с първоначално генерирани решения (хромозоми) и подобряването на решенията се достига след определен брой итерации (поколения). След генериране на съответното поколение се оценява пригодността му, т.е. до каква степен е достигнато изискването на целевата функция. Във всяко поколение, относително добрите решения имат по-голям вероятност да бъдат избрани от генетичните оператори (кръстосване и мутация) за възпроизвеждане. [14]

При кръстосването се съчетават гените от *N*-тото поколение (родители) за получаване на гените на *N+1*-то поколение (деца). Така при кръстосването се форсира бърза промяна, докато при мутацията се правят малки локални промени на възможното решения. Възпроизводството продължава до достигане на предварително зададен максималния брой повторения или до достигане на невъзможност за по-нататъшно подобрение в следващите поколения [7]. При случаен избор на първото поко-

ление, реализирането на даден GA до голяма степен зависи от обема на популацията. Основен недостатък при малки популации е преждевременно намиране на „решение", което не винаги е оптимално, докато при големи популации се изискват по-големи изчислителни ресурси.

Задълбочено изследване на посочения по-горе проблем е направено в [5] при различни генетични алгоритми с различни параметри. Според това изследване, при по-големи популации оптималното решение трябва да бъде намирано след по-малко на брой популации в сравнение с по-малките полпулации. От друга страна обаче, достигането на това оптимално решение става за много по-дълго време, тъй като изчисленията при всяка популация са значително повече.

Поради тази причина е предложен микро-GA (μGA). При μGA се работи с малка популация, но при липса на сходимост към оптимално решение се прибягва към генериране на нови случайни хромозоми. Основните етапи на μGA са:

1) инициализиране на първоначална популация (случайна);
2) оценяване на пригодността на всяка хромозома от популацията;
3) избор на най-пригодните индивиди („елит");
4) избор на хромозоми за репродукция;
5) кръстосване;
6) мутация;
7) следващо поколение;
8) оценяване на пригодността на новите хромозомите от популацията;
9) ако условието за прекратяване е изпълнено, се извежда най-доброто решение; а ако не е изпълнено – най-пригодните индивиди се запазват за пренареждане;
10) пренареждане на най-пригодните индивиди и случайно генериране на останалите индивиди;
11) връщане в 4).

При реализирането на всеки от етапите съществуват различни проблеми, например: как се генерират хромозомите, какво кодиране се използва, какви параметри на кръстосването и мутацията да бъдат избрани, как да се избират родители и пр.

II. Основни операции на генетичните алгоритми

1. Кодиране

В различните приложения се използват различни кодирания на хромозомите – двоично кодиране, кодиране по пермутации, кодиране по стойност, кодиране в дърво и други. При кодирането на хромозомите основно изискване е всяка хромозома да съдържа информация за решението, което тя задава.

В GA най-често се използва двоично кодиране, при което *k*-тата хромозома има вида:

$$11011001\ldots.001101101,$$

като всяка група от нули и единици може да задава някоя характеристика на решението.

Кодирането по пермутации е подходящо предимно за задачи, свързани с намиране на наредредба.

Кодирането по стойност може да бъде използвано в задачи, в които е наложително използването на по-сложни представяния. Използването на двоично кодиране при този вид задачи би било твърде сложно. При кодиране по стойност, всяка хромозома е стринг от някакви стойности. Стойностите могат да бъдат числа, реални числа или символи, дори в някои случаи – обекти.

Кодиране в дърво се използва при главно динамично програмиране. При кодиране в дърво всяка хромозома представлява дърво от някакви обекти, например функции или команди в програмен език. В приложения от този вид е често използван програмния език LISP, защото програмите се представят в тази форма и кръстосването и мутацията могат да се извършат относително лесно.

2. Параметри на GA

GA имат три основни параметъра:

- обем на популацията M;
- вероятност за кръстосване p_c;
- вероятност за мутация p_m.

Обемът на популацията M е равен на броят хромозоми, включени в съответното поколение. При недостатъчно голям обем, при реализиране на GA се осъществяват малко на брой кръстосвания и решението се търси само в малка част от пространството на евентуални решения, а при твърде голям брой – GA става бавен.

Вероятността за кръстосване задава честотата на осъществяване на кръстосването, като при вероятност p_c=0, цялата нова популация се получава като точно копие от гените на предходната, а при p_c=1, цялата популация е с нови гени, които са части от родителските хромозоми.

Вероятността за мутация задава честотата на мутиране на гени от хромозомата. При p_m=0, поколение е точно копие на полученото след кръстосване, а при p_m=1 цялата хромозома напълно се променя. При стойности на p_m, близки до 1, GA става алгоритъм за произволно търсене.

3. Селекция

Съгласно еволюционната теория на Дарвин [6], най-добрите индивиди оцеляват и създават ново поколение. Изборът на хромозоми за репродукция може да бъде осъществен с различни методи: селекция по рулетка, селекция по ранг, селекция на устойчивите състояния и други.

При селекция по рулетка, вероятността за избор на дадена хромозома е правопропорционална на нейната пригодност, т.е. колкото по-добро решение на проблема дава дадена хромозома, толкова по-голяма е вероятността тя да бъде избрана за родител.

При селекция по ранг, първо всички хромозоми се подреждат според пригодността ими на всяка от тях се приписва ранг от 1 до M. Тогава вероятността за избор на дадена хромозома е правопропорционална на нейният ранг.

При селекция по устойчивост на състоянието от всяко поколение се избират няколко „добри“ хромозоми за създаването на новото потомство, а няколко „лоши“ се заместват с получените деца от „добрите“ хромозоми. Останалата част от популацията се запазва.

4. Кръстосване

При кръстосването се избират гени от родителските хромозоми за създаване на новото поколение. Един възможен подход за това е случаен избор на позиция, като всички гени преди тази позиция се запазват, а всички гени след нея се унаследяват от другия родител, например:

k-та хромозома (родител 1)	11011001….00 1101101
k+1-ва хромозома (родител 2)	11100001….01 *1001011*
k'-та хромозома (дете 1)	11011001….00 *1001011*
(k+1)'-ва хромозома (дете 2)	11100001….01 1101101

5. Мутация

При мутацията новото поколение се променя случайно, като това е предпазна мярка против попадане в локален оптимум. Един често използван метод за мутация е промяна на случайно избран ген от 1 в 0 или от 0 в 1, например:

k-та хромозома	1101*1*001….001101101
k'-та хромозома	1101*0*001….001101101

III. GA **в програмна среда MatLab**

1. Терминология

В генетичните алгоритми, реализирани в MATLAB, е прието да се използват термините от еволюционната теория на биологията. Основните термини [4], които се използват са:

- *Individual* – (индивид) точка от пространството на възможни решения
- *Fitness Function* – фитнес функция, осигуряваща оценка на отделния индивид (целева функция);
- *Fitness Value* – стойността на фитнес функцията за даден индивид (стойността на целевата функция в дадена точка);
- *Population* – популация (множество от точки, получено при дадена итерация);
- *Best Fitness Value* – най-добрата оценка измежду оценките на отделните индивиди в популацията;
- *Parents* – родители, избрани индивиди от една популация, които създават индивиди (деца) от следващото потомство;
- *Generation* – нова генерация или потомство, създадено по определени правила от текущата популация (следващо множество от точки, доближаващо се до оптималното решение);
- *Diversity* – различие, несходство между индивидите в една популация (характеризира разпръснатостта на текущата популация от точки в допустимата област). Този параметър е основен за генетичните алгоритми, защото дава възможност за претърсване на по-голяма област от пространството на възможни решения.

2. Синтаксис на процедурата ga в програмна среда MatLab

Процедурата **ga** има следния синтаксис:

[x, fval, flag, output, *lastPop*, *scores*] =
ga(*@fitnesFcn*, *nVars*, A, b, Aeq, beq, lb, ub, *@nlConsFcn*, *intConstr*, *options*),

където:

@fitnesFcn – указател на фитнес функцията;
nVars – брой на независимите променливи;
@nlConsFcn – указател на функцията на нелинейните ограничения;
intConstr – вектор с индексите на целочислените променливи;
options – запис с опциите;
lastPop – последната популация;
scores – оценките на последната популация.

Важна особеност на процедурата **ga** е, че когато някои от параметрите трябва да приемат само целочислени стойности, процедурата **ga** не допуска ограничения от тип равенства. Затова, когато аргументът *intConstr* не е празен, аргументите *Aeq* и *beq* трябва да бъдат празни (означени с []), а функцията *nlConsFcn* трябва да връща за *ceq* (нелинейни ограничения от тип равенства) също символа за празен масив [].

Опции

Оперирането с опциите се извършва с функцията gaoptimset:

gaoptimset(@ga) – извежда на екрана всички опции с техните подразбиращи се стойности;

```
>> gaoptimset(@ga)
ans =
      PopulationType: 'doubleVector'
        PopInitRange: [2x1 double]
      PopulationSize: 20
         EliteCount: 2
    CrossoverFraction: 0.8000
      ParetoFraction: []
   MigrationDirection: 'forward'
    MigrationInterval: 20
    MigrationFraction: 0.2000
        Generations: 100
          TimeLimit: Inf
        FitnessLimit: -Inf
       StallGenLimit: 50
      StallTimeLimit: Inf
            TolFun: 1.0000e-06
            TolCon: 1.0000e-06
    InitialPopulation: []
       InitialScores: []
      InitialPenalty: 10
       PenaltyFactor: 100
```

```
        PlotInterval: 1
     CreationFcn: @gacreationuniform
 FitnessScalingFcn: @fitscalingrank
    SelectionFcn: @selectionstochunif
    CrossoverFcn: @crossoverscattered
     MutationFcn: {[@mutationgaussian]  [1]  [1]}
DistanceMeasureFcn: []
      HybridFcn: []
       Display: 'final'
      PlotFcns: []
     OutputFcns: []
     Vectorized: 'off'
    UseParallel: 'never'
```

options = gaoptimset('opt1', val1,'opt2', val2, ...) – задаване на стойности на опции по двойки „име-стойност";

options = gaoptimset(options,'opt1', val1, ...) – обновяване на съществуващ запис с опции *options*.

Често използвани опции са:

InitialPopulation – задава началната популация във вид на матрица с координатите на началните точки, разположени по редове. По подразбиране се генерира автоматично с генератор на случайни числа;

PopulationSize – брой на индивидите в популацията (в случая 20);

Generations – брой на генерираните популации (в случая 100);

ElitCount – брой на елитните индивиди, които попадат директно в следващата популация (в случая 2);

CrossoverFraction – определя каква част от родителите да се кръстосват. Приема стойности от 0 до 1 (в случая 0.8).

TimeLimit – максимално време на изчисленията (в случая *Inf*);

TolFun – допуск по стойност на функцията (1.e-06).

Повторното изпълнение на процедурата **ga** с начална популация, съвпадаща с последната популация *lastPop* от предишното изпълнение може да подобри получените резултати:

```
options = gaoptimset(options, 'InitialPop', lastPop);
[x, fval, ...] = ga(@fitnesFcn, nVars, ..., options);
```

Последната популация от второто изпълнение може да послужи като начална в третото и т. н.

IV. Конструиране на портфейли чрез GA

В случая трите основните оператора в генетичен алгоритъм са:

- *селекция* – пресмята се репродуктивната вероятност p_i за всеки индивид:

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

където f_i е стойността на фитнес функцията за *i*-ия индивид и *n* е обемът на популацията. При всяка итерация се избира единствена нова хромозома [8], така че

за произволно случайно число $r \in [0, 1]$, ако $r \leq p_1$ да се избира първата, а ако е изпълнено

$$p_{j-1} < r \leq p_j \ ,$$

се избира j-тата;

- *кръстосване* – популацията, получена след селекция се разделя на две равни части, избират се двойки хромозоми от всяка и дадена двойка се кръстосва с вероятност P_c;
- *мутация* – след кръстосването случайно избрани битове се променят с вероятност P_m.

За търсене на оптимален инвестиционен портфейл [1] се предлага следната процедура:

- фиксира се n – желаният брой финансови активи в инвестиционния портфейл и едновременно обем на полулацията;
- прилага се модела FLQM [9], откъдето се получава множество от портфейли и това са хромозомите;
- дефинира се целевата функция:

$$G(x_1, x_2, \ldots x_n) = R_p - \sigma_p,$$

където x_j са дяловете на активите в портфейла, R_p е възвращаемостта на портфейла и σ_p е инвестиционният риск на портфейла [12], [13];

- търси се оптимум на целевата функция $G(x_1, x_2, \ldots x_n)$ с генетичен алгоритъм. [3]

V. Заключение

Реализирането на генетичен алгорутъм в процеса на управление на инвестиционни портфейли е в състояние значително да редуцира ресурсите, които обикновено се влагат в решаването на портфейлната задача.

Необходимо е провеждането на редица експерименти, за прецизно получаване на стойностите на параметрите на процедурата ga.

Литература:

[1]. Георгиева, П. Софт компютинг като направление в изкуствения интелект. Годишник БСУ, том XIX, 2008.
[2]. Георгиева П., Оценка на капитала при конструиране на инвестиционни портфейли. ММ XXI (5), 2013
[3]. Георгиева П., Генетичните алгоритми като средство за решаване на оптимизационни задачи. Списание „Компютърни науки и комуникации”, Том 2, № 3
[4]. Цонев Ст., Витлиемов В., Коев П. Методи за оптимизация. (http://www.uni-ruse.bg/faculties/mtf/tm/site/vgv/VGV-12_book2-2004.pdf)
[5]. Carroll, D.L. (1996): Genetic Algorithms and Optimizing Chemical Oxygen-Iodine Lasers, Developments in Theoretical and Applied Mechanics, Vol. XVIII, eds. H.B. Wilson, R.C. Batra, C.W. Bert, A.M.J. Davis, R.A. Schapery, D.S. Stewart, and F.F. Swinson, School of Engineering, The University of Alabama, pp.411-424.
[6]. Darvin Ch., The Origin of Species. John Murray, 1859.

[7]. Fogel L., Owens A., Walsh M., Artificial Intelligence through a Simulation of Evolution". Spartan, Washington DC, 1965, pp. 131-156.
[8]. Garkaz M., The Selection and Optimization of Stock Portfolio using Genetic Algorithm based on Mean-semi Variance Model, International Conference on Portfolio Selection Using Genetic A Economics and Finance Reaserch, IPEDR, LACSIT Press, Singapore, 4, (2011), 379-381.
[9]. Georgieva P., I. Popchev, „Fuzzy Q-measure Model for Managing Financial Investments", Compus Rendus Acad. Bulg. Sci., vol. 66(5), pp. 651-658, 2013
[10]. Goldberg, D. E.(1989): Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989
[11]. Holland, John H. (1975): Adaption in Natural and Artificial Systems: An Introductory Analysis with Application to Biology, Control and Artificial Intelligence, Univ. of Michigan Press, 1975.
[12]. Lin D., Xiaoming Li, Mingiang Li, A Genetic Algorithm for Solving Portfolio Optimization Problems with Transactions Costs and Minimum Transactions Lots. Proceedings of the First International Conference on Advances in Natural Computation ICNC'05, Publisher Springer-Verlag Berlin, Heidelberg, 3, (2005), 808-811.
[13]. Markowitz H., Portfolio selection. Journal of Finance, 7, (March, 1952), 77-91.
[14]. Poli R., etc., „A Field Guide to Genetic Programming". Published via: http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008 (with contributions by J. R. Koza).