ОПЫТ ПРОВЕРКИ ВОЗМОЖНОСТЕЙ ЭВОЛЮЦИИ АЛГОРИТМОВ
НА ПРИМЕРЕ МАШИН ТЬЮРИНГА.

С. Д. Мешвелиани
Институт программных систем РАН, г.Переславль-Залесский
5 декабря 2016 года.

§1. Введение

John R. Koza.
«Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems». 1990.
Stanford University Comp.Sci.Dep. technical report STAN-CS-90-1314.
http://www.genetic-programming.com/jkpdf/tr1314.pdf

1.1. ЭВОЛЮЦИЯ ЛИНЕЙНОГО ВИДА. Особь — квадратная матрица размера `n` **над** $\mathbb{Z}_2$.

СРЕДА:
Problem $-$ n **векторов размера** n **над** $\mathbb{Z}_2$.

ВЫСОТА ВЕКТОРА:
$H(v) =$ **расстояние от первой единицы до конца вектора.**

СВЕДЁНИЕ NF вектора v **по матрице** M**:**
найти строку r **в** M **такую, что** $H(v-r) < H(v)$.
Если такой нет то матрица сведена.
Иначе продолжить сведéние с вектора $v-r$.

ОТКЛОНЕНИЕ особи от требований среды:

$$dev(M) = \Sigma_{v \in Problem}\ H(NF(v))$$

ПОПУЛЯЦИЯ есть список пар $(M,\ d)$,
где M $-$ **особь,** d $-$ **её отклонение.**
Пары должны быть упорядочены по неубыванию отклонения (наиболее успешные находятся впереди).

КОЛИЧЕСТВО ОСОБЕЙ в популяции ограничено параметром pBound.

ИЗМЕНЧИВОСТЬ (мутация) особи:
случайно выбирается позиция $(i,\ j)$ **в матрице** M,
и элемент $M(i,\ j)$ **перевёртывается.**

ШАГ ЭВОЛЮЦИИ:

(1) из популяции случайно выбирается особь M;
(2) особь $M\prime$ **получается мутацией особи** M;
(3) вычисляется $d = dev(M\prime)$;
(4) $(M\prime, d)$ **вставляется в популяцию согласно значению** d.
(5) если граница pBound превышена, то последняя особь списка удаляется.

ЗАМЕЧАНИЕ О ТРЕУГОЛЬНОЙ МАТРИЦЕ:
очевидно, любая треугольная матрица размера $n \times n$,
или отличающаяся от неё перестановкой строк,
наилучшим образом приспособлена к любой среде.

Вопрос состоит в том, как быстро она получится в ходе эволюции.

ЗАДАНИЕ СРЕДЫ: случайная матрица Problem размера $n \times n$.

НАЧАЛЬНАЯ ПОПУЛЯЦИЯ
состоит из одной нулевой матрицы размера $n \times n$.

1.2. ПРИМЕР. Для $n = 20, \quad pBound = 1000$

100 тысяч шагов эволюции
(80 секунд счёта на персональном компьютере частоты 3 гигагерц)
достаточно для получения перестановочно-треугольной матрицы
(с нулевым отклонением).

1.3. О МЕСТНОМ МИНИМУМЕ. Положим в этом примере $pBound = 100$.
Тогда и миллиона шагов не хватит.
Причина тому: сваливание популяции в устойчивый местный минимум.

При увеличении границы популяции уменьшается монотонность процесса и увеличивается возможность блуждания в обход местного минимума.
При этом скорость спуска уменьшается.

1.4. ЗАКЛЮЧЕНИЕ О ЛИНЕЙНОМ ПРИМЕРЕ ЭВОЛЮЦИИ. Почти очевидно, что

в данном примере эволюция достигает налучшего итога
в среднем за небольшое количество шагов.
Можно было бы вывести приличную формулу оценки сверху
для этого количества.

Почему так легко ?
Мощность пространства матриц = $2^{(n^2)}$ **...**
?

Ниже рассмотрен пример эволюции на машинах Тьюринга.
И вот для этого более содержательного примера вопрос
о достижимости успеха эволюции мне пока не ясен.

В отчёте J. Koza содержится (среди многих других)
средний по сложности пример: эволюция на булевых выражениях.
Он важен, его следует разобрать.

ЭВОЛЮЦИЯ НА МАШИНАХ ТЬЮРИНГА.

alphabet = {A, B, C, Blank},

alphabet-0 = **множество букв отличных от пробела.**

Имеется множество

State = $\{q_0, q_1, \ldots,$ **maxQ**$\}$.

состояний машины.

Пока положим maxQ = 15.

РАСШИРЕННОЕ СОСТОЯНИЕ: (q, l),

где q **есть состояние, а** l **— буква алфавита.**

Машина занимается переработкой слов, написанных на ленте в данном алфавите.

Сдвиги по ленте: ToL, ToR.

СОБСТВЕННОЕ ДЕЙСТВИЕ машины на очередном шаге может быть:

ToL, ToR, l.

ОЧЕРЕДНОЕ ДЕЙСТВИЕ машины задаётся парой

(q, pAct),

где pAct **есть собственное действие.**

КОМАНДА есть пара

```
(eState, act),
```

На каждом шаге работы машины ЛЕНТА содержит конечный отрезок клеток,
и каждая клетка занята некоторой буквой.

Лента заканчивается слева на первой букве текущего слова
и она заканчивается справа на последней букве текущего слова.

Но лента потенциально бесконечна: всякий раз, когда головка выходит за край слова на ленте, в новую клетку автоматически ставится пробел.

МАШИНА задаётся частичным отображением

```
tab : ExtState → Action.
```

Его можно задать списком команд, в котором расширенные состояния не повторяются.

Расширенное состояние также назовём КЛЮЧОМ.

Сначала лента содержит слово, на первую непробельную букву которого указывает головка.

ШАГ МАШИНЫ:

```
(q, l)  ->  tab(q, l)
```

Машина ОСТАНАВЛИВАЕТСЯ, когда попадает в расширенное состояние,
которого нет в таблице.

1.5. БОЛЬШИЕ КОМАНДЫ.

```
[((q, l), act) |  l <- alphabet]
```

означает безусловное действие.

§2. О ПРОГРАММЕ ИНТЕРПРЕТАТОРА МАШИН И ЭВОЛЮЦИИ

Программа моделирования эволюции написана на языке Haskell, она испытывалась в системе реализации Glasgow Haskell 7.10.2.

Машина Тьюринга (данное `TM`**), лента (данное** `Tape`**), состояние машины вместе с состоянием ленты (данное** `Configuration`**), представлены в виде данных языка Haskell.**

На языке Haskell запрограммирован интерпретатор машины:
функции `step` **перехода к следующему полному состоянию,**
`wordToTrace` **выдачи последовательности состояний вычисления,**
`applyToWord` **выдачи итогового слова для данного в аргументе слова.**

Последней функции даётся ограничение `timeBound`**.**
Если конечное состояние не достигнуто за `timeBound` **шагов, то итогом считается данное** `Nothing`**.**

§3. ПРИМЕРЫ МАШИН ТЬЮРИНГА

Пример 1. Машина, которая всюду заменяет букву A на букву B:

```
swap_AB :: TM
swap_AB =
 Tm $ Map.fromList $ concat

 [ [((Q0,A), (Q1,Lett B)), ((Q0,B), (Q1,Lett A)), ((Q0,C), (Q0,ToR))],

   forallLetters Q1 (Q0,ToR)
 ]
```

Функция `concat` **соединяет списки,**
`Map.fromList` **укладывает список пар в таблицу конечного отображения.**

СОБСТВЕННОЕ ДЕЙСТВИЕ может быть записано как данное в виде

`Lett l, ToL, ToR.`

ПРИМЕР: последовательность вычисления на слове `CBABC`**.**

```
 0         0         1         0         1         0         1         0         0
 CBABC -- CBABC -- CAABC -- CAABC -- CABBC -- CABBC -- CABAC -- CABAC -- CABAC
```

Пример 2. Машина, которая в каждом непустом слове оставляет первую букву:

```
head' =
   Tm $ Map.fromList $ concat

   [forNonblank Q0 (Q1, ToR),   forNonblank Q1 (Q2, Lett Blank),

    forallLetters Q2 (Q1, ToR)
   ]
```

Большая команда `forNonblank`

относится к буквам отличным от пробела.

ПРИМЕР вычисления:

```
            0         1         2         1         2         1
            ABC  --  ABC  --  A C  --  A C  --  A    -- A
```

Пример 3. Машина, которая оставляет последнюю букву слова:

```
last' =  Tm $ Map.fromList $ concat

         [forNonblank Q0 (Q1,ToR),          forNonblank Q1 (Q2,ToL),

          forNonblank Q2 (Q3,Lett Blank),   [((Q3,Blank), (Q0,ToR))]
         ]
```

ПРИМЕР вычисления:

```
 0       1     2      3       0      1    2     3     0     1
 ABC -- ABC -- ABC --  BC --  BC -- BC -- BC --  C -- C -- C
```

Пример 4. Машина, которая переставляет первые две буквы слова:

```
swapFirstTwo =
 Tm $ Map.fromList $ concat
 [[((Q0,A), (Q1,ToR)),  ((Q0,B), (Q2,ToR)),  ((Q0,C), (Q3,ToR)),

   ((Q1,Blank), (Q10,ToL)),  ((Q1,A), (Q10,ToL)),    ((Q1,B), (Q5,Lett A)),
                                                     ((Q1,C), (Q6,Lett A)),

   ((Q2,Blank), (Q10,ToL)),  ((Q2,A), (Q4,Lett B)),  ((Q2,B), (Q10,ToL)),
                                                     ((Q2,C), (Q6,Lett B)),

   ((Q3,Blank), (Q10,ToL)),  ((Q3,A), (Q4,Lett C)),  ((Q3,B), (Q5,Lett C)),
                                                     ((Q3,C), (Q10,ToL))
   ],
   forallLetters Q4 (Q7, ToL),
   forallLetters Q5 (Q8, ToL),
   forallLetters Q6 (Q9, ToL),
   forallLetters Q7 (Q10, Lett A),
   forallLetters Q8 (Q10, Lett B),
   forallLetters Q9 (Q10, Lett C)
 ]
```

Здесь состояние `Q10` является конечным, оно равносильно приказу «стой».

ПРИМЕР вычисления:

```
 0       1      5     8      10
 ABC -- ABC -- AAC -- AAC -- BAC
```

Пример 5. Машина, которая переставляет первую букву слова в конец:

```
appendHead =
 Tm $ Map.fromList $ concat

 [[((Q0,A), (Q1,ToR)),  ((Q0,B), (Q2,ToR)),  ((Q0,C), (Q3,ToR)),

   ((Q1,A), (Q0,Lett A)), ((Q1,B), (Q5,Lett A)), ((Q1,C), (Q6,Lett A)),
   ((Q2,A), (Q4,Lett B)), ((Q2,B), (Q0,Lett B)), ((Q2,C), (Q6,Lett B)),
   ((Q3,A), (Q4,Lett C)), ((Q3,B), (Q5,Lett C)), ((Q3,C), (Q0,Lett C))
  ],
  concat
  [[((Q4,l), (Q7,ToL)),     ((Q5,l), (Q8,ToL)),
                                      ((Q6,l), (Q9,ToL)),
    ((Q7,l), (Q10,Lett A)), ((Q8,l), (Q10,Lett B)),
                                       ((Q9,l), (Q10,Lett C)),
    ((Q10,l), (Q0,ToR))
   ] | l <- alphabet
  ]
 ]
```

ПРИМЕР вычисления:

```
0       1      5     8      10      0       1      6     9      10      0
ABC -- ABC -- AAC -- AAC -- BAC -- BAC -- BAC -- BAA -- BAA -- BCA -- BCA --

                                                                        1
                                                                      BCA
```

§4. ОТОБРАЖЕНИЕ СЛОВ. ГРАНИЦЫ ВРЕМЕНИ И МНОЖЕСТВА ВХОДНЫХ СЛОВ

Машина определяет частичное отображение на множестве слов.

ГРАНИЦА ДЛЯ ЧИСЛА ШАГОВ МАШИНЫ M:

```
tBound :: Natural.
```

Отображение

```
wMap(M, b) :  Word → Maybe Word
```

называем ОТОБРАЖЕНИЕМ СЛОВ ограниченной по времени машины `M`.

`wMap(w) = Just w'` **или** `Nothing`.

Также ограничим отображение `wMap(M, b)` **на множество слов** `Word(wBound)` **длины не более** `wBound`.

Параметры `tBound` **и** `wBound`, **определяют таблицу**

```
wMap(M, tBound, wBound)
```

отображения слов.
Она вычисляется интерпретатором машины Тьюринга.

§5. РАССТОЯНИЕ МЕЖДУ МАШИНАМИ, МЕЖДУ ОТОБРАЖЕНИЯМИ СЛОВ

РАССТОЯНИЕМ называем количество мест, в которых эти слова имеют разные буквы, ещё добавляется разность длин этих слов.

Расстоянием между отображениями f и g слов на множестве `Word(n)`**:**

$$\Sigma_{w \in Word(n)}\ dist(f(w), g(w)).$$

Расстояние между машинами есть расстояние между их отображениями слов на множестве `Word(n)`**,**
с тем добавлением, что если какая-то из двух машин не завершается на некотором слове из `Word(n)`**, то расстояние не определено.**

§6. БЛИЗНЕЦЫ ПО ПЕРЕИМЕНОВАНИЮ. СВЕДЕ́НИЕ МАШИНЫ

Близнецами назовём те машины, которые отличаются переименованием состояний по некоторой перестановке.

В ходе эволюции порождается много случайных машин, а среди них — много «близнецов».

На каждом шаге удаляет близнецовые повторы из популяции.

Распознавание близнецов делается путём сведе́ния каждой появляющейся машины к наименьшему виду по перестановкам состояний.

СВЕДЕ́НИЕ делается относительно словарного упорядочения машин.

Применяется *частичный перебор*, устанавливается граница `permutationBound`**.**

Делается такая перестановка, что получаемая машина имеет множество состояний без пропуска индексов:

$$\{\mathbf{q}_i \mid \mathbf{0} \leqslant \mathbf{i} \leqslant \mathbf{k}\}.$$

Затем применяется сведе́ние по перестановкам на множестве состояний

$$\{\mathbf{q}_i \mid \mathbf{0} \leqslant \mathbf{i} \leqslant \mathbf{m}\}, \quad \text{где} \quad \texttt{m = min k permutationBound}.$$

Затем применяется сведе́ние только по траспозициям пар $\mathbf{q} < \mathbf{q}_i$ состояний для $\mathbf{i} > \mathbf{m}$.

Применяется «спуск к местному минимуму по транспозициям».

§7. ПРЕДСТАВЛЕНИЕ ЭВОЛЮЦИИ

7.1. ЦЕЛЕВЫЕ МАШИНА И ТАБЛИЦА. Эволюция машин определена для любой так называемой целевой машины gM.

Целевая таблица есть таблица отображения целевой машины на множестве
`Word(wBound)`.

`wBound` и `tBound` таковы, что целевая машина завершает работу на всяком слове из `Word(wBound)` не более, чем за `tBound` шагов.

7.2. ОЦЕНКА МАШИНЫ. ОТКЛОНЕНИЕМ dev(M) машины назовём расстояние между целевой таблицей и таблицей отображения машины M на множестве
`Word(n)`.

Если M не останавливается на некотором слове из `Word(n)` за `tBound` шагов, то отклонение считается большим, например — квадрат суммы длин слов из `Word(n)`.

Отклонения вычисляется в программе вызовом функции `deviation`.

Машину с нулевым отклонением называем ТОЧНОЙ. (для данной таблицы).

ТРИ ЧАСТИ ЗАКОНА СРЕДЫ:

- **целевые машина и таблица,**
- **закон оценки произвольной машины (отклонение можно задать по-разному),**
- **закон изменчивости произвольной машины.**

7.3. ПОПУЛЯЦИЯ И ЗАКОН ОТБОРА. ПОПУЛЯЦИЕЙ назовём конечный список
машин, упорядоченный по неубыванию отклонения таблицы машины от целевой таблицы.

Зададим параметр

```
pBound :: Natural,
```

так что список популяции имеет длину не более `pBound`.

На каждом шагу эволюции в список вставляются несколько порождённых случайно машин, и из списка популяции удаляется хвост после границы `pBound`.

Чем меньше отклонение машины, тем она ближе к началу списка (успешнее приспособилась к закону среды).

7.4. ТЕКУЩАЯ ГРАНИЦА СОСТОЯНИЙ. Подмножества множества $\{q_0, \ldots, \mathrm{maxQ}\}$ состояний градуируются по мощности.

Введён параметр текущего ограничения на число состояний:

$$q_0 \leqslant \mathrm{qBound} \leqslant \mathrm{maxQ}.$$

Все нижеописываемые функции имеют дополнительный аргумент `qBound`.

Все состояния в появляющихся машинах не больше, чем `qBound`.

Будем считать, что в описаниях всех этих функций ко всякому построению добавлены слова
“при ограничении qBound на появляющиеся состояния”.

Программа эволюции запускается последовательно для `qBound` = $q_0, q_1, \ldots,$
каждый раз на некоторое большое количество шагов.

Испытатель наблюдает: какого значения `qBound` **достаточно для порождения машины с довольно малым отклонением.**

7.5. ЗАКОН ИЗМЕНЧИВОСТИ. На каждом шагу эволюции из популяции случайным образом выбирается некоторая машина M, и к ней применяется изменение («мутация»).

Изменение приводит к появлению нескольких новых машин.

Опишем СОСТАВНЫЕ ЧАСТИ ДЕЙСТВИЯ ИЗМЕНЕНИЯ.

7.5.1. *УДАЛЕНИЕ КОМАНДЫ.* **Из машины удаляется случайно выбранная команда.**

7.5.2. *ДОБАВЛЕНИЕ БОЛЬШОЙ КОМАНДЫ.* **Функция** `randomMacrocommandForAllLetters`
выбирает случайным образом состояние `q` **и действие** `act`,
и возвращает большую команду `[((q, l), act) | l <- alphabet]`.

7.5.3. *КОМАНДЫ НОВОГО КЛЮЧА.* **Функция** `moreCommandsOfRandomKey`

по машине M выбирает случайный ключ `eState`**, не принадлежащий M, и выдаёт**

- **список** `comms` **всех возможных команд** `(eState, act'))` **для этого ключа (их имеется qBound · (|alphabet| + 2) штук),**
- **список машин** `[insert comm M | comm <- comms]`**, в котором каждая машина получается из** `M` **добавлением команды из** `comms`**.**

Здесь `act'` **пробегает все возможные действия, то есть пары: состояние** `q'` **и запись буквы или сдвиг головки.**

7.5.4. *РАСЩЕПЛЕНИЕ МАШИНЫ НА КОМАНДЕ.* **Функция** `splitTM_atCommand`

выдаёт список машин, которые отличаются от M только командой ключа key, и добавляет ещё машину M без команды ключа key (всего выдаётся список длины qBound · |alphabet|).

Далее порождаются все команды вида `(eState, act')`**, для всх возможных действий.**
Число этих команд есть

$$\textbf{qBound} \cdot (|\textbf{alphabet}| + 2).$$

Для каждой из этих команд $\textbf{comm}_i$ **составляется машина** $\textbf{M1}_i$**, получаемая из M заменой команды comm на** $\textbf{comm}_i$**.**
Каждая из полученных машин сводится по перестановочным переименованиям состояний.

7.5.5. *ИЗМЕНЕНИЕ МАШИНЫ.* **Функция** `mutateTM`

выдаёт список всех машин, полученных из данной машины M вышеописанными действиями:

(1) удаление случайно выбранной команды,
(2) добавление большой команды для случайно выбранных состояния и действия,
(3) добавление команды для случайно выбранного ключа,
(4) расщепление машины M на ключе, случайно выбранном из команд M.

7.6. ПОПУЛЯЦИЯ. Она представлена списком пар `(M, qual)` **длины не более** `pBound`.

`qual` **содержит поля:**

- **отклонение,**
- **число состояний, входящих в M,**
- **список слов из целевых аргументов, на которых M не остановилась в течение** `timeBound` **шагов.**

При этом

- **Пары в популяции упорядочены по неубыванию величины отклонения,**
- **Пары с одинаковым отклонением упорядочены в этом списке по неубыванию числа состояний машин,**
- **машины в этом списке не повторяются, и каждая находится в виде, сведённом по перестановкам состояний.**

7.7. ВЫБОР МАШИНЫ ДЛЯ ИЗМЕНЕНИЯ. Очередной шаг эволюции начинается со случайного выбора машины из популяции.

Распределение случайной величины номера i машины в этом списке сделано неравномерным.

Добавлен всеобщий параметр:

$$\text{exponentBound} \geqslant 1.$$

И при выборе машины сначала случайно выбирается степень — натуральное число $1 \leqslant e \leqslant \text{exponentBound}$.

Потом случайно выбирается номер i машины. Затем вычисляется

$$j = \text{целая часть корня степени } e \text{ из } i.$$

И берётся j-я машина из списка.

7.8. ШАГ ЭВОЛЮЦИИ. Функция эволюции выдаёт бесконечный список популяций, полученных из начальной популяции, состоящей из одной машины.

Очередная популяция получается из последней популяции действиями, описываемыми ниже, и она добавляется к списку.

Головная функция может взять первые N членов этого списка, или, например, выдать первую популяцию, в которой имеется машина с достаточно хорошей оценкой.

Шаг эволюции выполняется функцией `evolutionStep`**.**

Случайно выбирается из последней популяции машина M.
К M применяется функция `mutateTM`**.**
Она выдаёт список Ms машин.
Машины из этого списка приводятся к сведённому виду, удаляются близнецы,
полученный список упорядочивается по величине отклонения.

Получается упорядоченный список пар pairs'.
Каждая пара в нём имеет вид (M', qual),
где M' есть новая машина, qual есть её оценка качества.
Затем функция `mergeQualTMs` **сливает упорядоченные списки pairs (популяции) и pairs' согласно упорядочению по оценке.**
При этом распознаются возникающие повторы
(повторяться могут только машины с одинаковой оценкой), и эти повторы удаляются.

Наконец, из полученной популяции удаляются члены списка с номерами бóльшими границы pBound.
Это удаление выражает естественный отбор.

§8. СМЫСЛ ОПИСАННОЙ МОДЕЛИ

**Машины Тьюринга (очень условно) изображают *особи*
— простейшие живые организмы, скажем, наподобие вирусов.**

Закон изменения изображает закон мутации особей.

Популяция машин изображает популяцию особей.

Эволюция популяции машин изображает последовательность состояний популяции особей во времени.

ЗАКОН ПРИРОДЫ изображается

- **целевой машиной и таблицей,**
- **законом оценки произвольной машины,**
- **законом изменчивости машины и законом изменчивости в популяции машин.**

ПРИСПОСОБЛЕННОСТЬ особи к среде выражена в виде той точности, с которой машина приближает закон среды.

§9. ОЦЕНКА КОЛИЧЕСТВА МАШИН

Оценим снизу мощность множества перестановочных классов машин, в котором происходит эволюция.

Обозначим $\mathrm{qB} = \mathbf{qBound}$, $|\,\mathrm{Al}\,| = |\mathbf{alphabet}|$.

Мощность множества ExtState ключей есть $|\,\mathrm{ES}\,| = \mathrm{qB}\cdot|\,\mathrm{Al}\,|$.

Число подмножеств мощности n множества ExtState есть $C^n_{|\,\mathrm{ES}\,|}$.

Число действий $= |\,\mathrm{Act}\,| = qB\cdot(|\,\mathrm{Al}\,|+2) \geqslant |\,\mathrm{ES}\,|$.

Число машин для данного подмножества ключей мощности n есть число отображений из соответствующего подмножества ключей во множество действий $= |\,\mathrm{Act}\,|^n$.

Число машин, имеющих множество ключей мощности n есть

$$C^n_{|\,\mathrm{ES}\,|}\cdot|\,\mathrm{Act}\,|^n.$$

Число машин для множества состояний State $= \{\mathbf{q}_0, \ldots, \mathbf{qB}\}$ и алфавита alphabet равно

$\Sigma_{0\leqslant n\leqslant|\,\mathrm{ES}\,|}$ **|машины с количеством ключей n|** $=$

$\Sigma_{0\leqslant n\leqslant|\,\mathrm{ES}\,|}\ (C^n_{|\,\mathrm{ES}\,|}\cdot|\,\mathrm{Act}\,|^n)$ **=приблизительно**

$C^{|\,\mathrm{ES}\,|/2}_{|\,\mathrm{ES}\,|}\cdot(\Sigma_{0\leqslant n\leqslant|\,\mathrm{ES}\,|}|\,\mathrm{Act}\,|^n) \geqslant C^{|\,\mathrm{ES}\,|/2}_{|\,\mathrm{ES}\,|}\cdot|\,\mathrm{Act}\,|^{|\,\mathrm{ES}\,|}$.

Число классов машин по модулю перестановок состояний больше чем

$$(C^{|\,\mathrm{ES}\,|/2}_{|\,\mathrm{ES}\,|}\cdot|\,\mathrm{Act}\,|^{|\,\mathrm{ES}\,|})/\,\mathrm{qB}!$$

Пример 6. Для $\mathrm{qB}=7$, $|\,\mathrm{Al}\,|=4$ имеем

$$|\,\mathrm{ES}\,| = 7\cdot 4 = 28,\ |\,\mathrm{Act}\,| = 7\cdot 6 = 42,$$

и число классов больше чем

$(\binom{28}{14}\cdot 42^{28})/7! = (\binom{28}{14}\cdot 6^{28}\cdot 7^{21}\cdot 7^7)/7! > \binom{28}{14}\cdot 6^{28}\cdot 7^{21} > \binom{28}{14}\cdot 6^{49} =$

$((15\cdot\ldots\cdot 28)/14!)\cdot 6^{49} > ((18^{14})/14!)\cdot 6^{49} >$

$((18^{14})/(12^{14}))\cdot 6^{49} = ((3/2)^{14})\cdot 6^{49} =$

$3^{14}\cdot(1/2^{14})\cdot 2^{49}\cdot 3^{49} > 3^{63}\cdot 2^{35} > 2^{98}$.

То есть эволюция происходит в очень большом множестве классов машин.

Например, за миллиард шагов эволюции будет испытано меньше
одной миллиардной части этого множества машин.
Появление успешных машин за практически достижимое число
шагов для рассмотренных выше примеров объяснимо наличием
законов отбора и наследственности.

НАСЛЕДСТВЕННОСТЬ выражается в том, что один шаг мутации
для машины M порождает машины близкие к M.
И после такого шага в популяции всегда есть машины с оценкой
не хуже оценки M,
а иногда появляются и машины с лучшей оценкой.

§10. ИСПЫТАНИЕ МОДЕЛИ

Вышеприведённые примеры машин

`swap_AB, head', last', swapFirstTwo, appendHead`

были поставлены в качестве целевых.

Отображения слов задавались для всех слов длины не больше 3.

Граница завершаемости работы каждой машины задана как

`timeBound = 30`

шагов машины.

Степень неравномерности случайного выбора машины из популяции для мутации задана как

`exponentBound = 3.`

Начальная машина полагалась случайной машиной для выбранного множества состояний.

Число состояний, по которым выявлялись машины – близнецы, задано как

`permutBound = 4.`

Наибольшее число состояний в выбранных для примера целевых машинах равно 11.

Граница размера популяции задана параметром `pBound`, **для которого на удачу пробовались значения от 300 до 1000 с шагом 100.**

При запуске задавалось ограничение `eStepBound` **на количество шагов эволюции.**
При появлении точной машины процесс завершается досрочно.

10.1. ЗАКЛЮЧИТЕЛЬНАЯ ПРОВЕРКА. Если в ходе эволюции появилась точная машина, то она дальше проверяется на увеличенной таблице слов: на всех словах длины 5, 8.

Во всех этих примерах оказалось, что эта машина точна и на увеличенной таблице.
Так что очень вероятно, что найденная машина вообще равносильна целевой.

Доказательство равносильности надо бы проводить вручную для каждого примера.
Но правило отбора действует именно согласно конечной таблице слов, а не согласно равносильности машин.

10.2. ИТОГИ. В первых трёх примерах точная машина возникала не больше, чем за

5000 шагов эволюции

(несколько секунд счёта на компьютере частоты 3 гигагерц).

В примере `swapFirstTwo`
точная машина появилась приблизительно через 30 тысяч шагов и 60 секунд времени.
Причём эта машина имеет 7 состояний вместо одиннадцати состояний целевой машины.

В примере `appendHead`,
для ограничения шестью состояниями и размером популяции 1000, после 600 тысяч шагов появилась машина с отклонением 10.
(что довольно мало).
Но в дальнейшем 1 миллиона шагов (8 часов счёта) не хватило для порождения более точной машины
(хотя была другим способом случайно найдена точная машина для шести состояний).

§11. ЧТО ПРЕПЯТСТВУЕТ УСПЕХУ

1) То же, что в примере линейной эволюции.
2) Почему в простом примере `appendHead` эволюция неудачна ?

11.1. МНОЖЕСТВО ПОПУЛЯЦИЙ. **В природе имеется много популяций,**
в которых независимо протекает эволюция.

В нашей модели этому соответствует некоторое конечное множество популяций машин, так, что для каждой из этих популяций происходит спуск к своему локальному минимуму. Это воплощено в виде функции (программы) `evolveUntil`.

Назовём популяцию насыщенной,
если она содержит наибольшее возможное число `pBound` машин и у всех этих машины одно и то же отклонение и одно и то же число состояний.

Функция `evolveUntil` выполняет процесс эволюции, вызывая повторно функцию `evolutionStep`.

Когда получается насыщенная популяция, функция случайно выбирает машину из этой популяции и запускает эволюцию заново, с выбранной машины.

Датчик случайных чисел передаётся в новом состоянии после каждого своего применения.
Поэтому две популяции, полученные из одного корня при разных начальных состояниях датчика будут, вообще говоря, развиваться по-разному, и как правило дадут разные насыщенные популяции, разные локальные минимумы.

Поэтому эта модель изображает множественные популяции, что в природе соответствуют независимому развитию популяций из особей одного вида в изолированных друг от друга местах с разными условиями.

В этой модели появляется много насыщенных популяций. Чем меньше граница `pBound`, тем чаще возникают новые насыщенные популяции и новые локальные минимумы.

В этой модели для примера `appendHead`
оценка отклонения 37 за 1 миллион шагов снизилась до 10.

Но для появления точной машины миллиона шагов не хватило.

§12. ВЫВОДЫ

Пока из этой модели не видны по-настоящему воможности эволюции.

Для оценки этих возможностей надо рассмотреть больше примеров, и притом увеличить скорость счёта.

Ведь в природе некоторые популяции состоят из триллионов особей и развиваются в течение миллиадра лет.