CONCURRENT APPROACH TO THE PROBLEM OF MODELING MAN-MACHINE SYSTEMS

A.S. Bogomolov, candidate of physical-mathematical Sciences, senior lecturer
Saratov State University named after N.G. Chernyshevsky
A.Yu. Gapchenko, student
K.A. Vasylev, student
Togliatti State University, Togliatti (Russia)

Keywords: man-machine systems; resources consumption; optimal path; concurrent programming.

Annotation: Mathematical modeling of man-machine systems is an essential tool for identifying the preconditions of pre-emergency situations and their prevention. Parallel approach to the software implementation of the designated models can greatly reduce the time required to obtain the final result.

УДК 519.713(043)

ПРИМЕНЕНИЕ МУЛЬТИЭВРИСТИЧЕСКОГО ПОДХОДА ПРИ АЛГОРИТМИЗАЦИИ РЕШЕНИЯ ГОЛОВОЛОМОК

А.А. Боргардт, студент
А.М. Лысенко, аспирант
Б.Ф. Мельников, доктор физико-математических наук, профессор
Тольяттинский государственный университет, Тольятти (Россия)

Ключевые слова: задача упаковки; эвристика наибольшего касания; трудно решаемые задачи; генетический алгоритм.

Аннотация: В статье рассмотрены некоторые решения задач дискретной оптимизации на основе мультиэвристического подхода. В работе представлен мультиэвристический подход к алгоритмизации решения головоломок с конечным числом ситуаций и чётко определёнными правилами, таких как маджонг и трёхмерный тетрис.

ВВЕДЕНИЕ. МУЛЬТИЭВРИСТИЧЕСКИЙ ПОДХОД ДЛЯ РЕШЕНИЯ ЗАДАЧ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Применение мультиэвристического подхода для решения различных задач дискретной оптимизации (ЗДО) было рассмотрено в нескольких наших предыдущих публикациях ([1–4] и др.). Несколько упрощая ситуацию, можно сказать, что мультиэвристический подход является развитием незавершённого метода ветвей и границ (классический вариант которого описан, например, в [5]).

Одна из многих возникающих при этом подзадач – такова. В [1,2] была отмечена необходимость применения разных наборов-геномов (при этом возможно даже применение и не совсем одинаковых алгоритмов самообучения) – для разных классов задач. Но это – простой вопрос, точнее, вопрос, основная сложность в решении которого ложится не столько на программиста, сколько на эксперта (т.е. человека, очень хорошо представляющего специфику рассматриваемой ЗДО). Гораздо важнее была бы возможность автоматической генерации условий принадлежности конкретной ЗДО классу задач, подлежащих отдельному от других классов рассмотрению[1].

[1] Отметим ещё, что в случае программирование игр к этой задаче вплотную примыкает задача автоматической генерации некоторого нового параметра для статической оценки позиции.

После одного из двух таких вариантов автоматической генерации идёт обычное самообучение с помощью генетических методов, а после самообучения – проверка, действительно ли мы сгенерировали некоторый новый класс подзадач для рассматриваемой нами ЗДО (или же шли по «ложному следу»). Отметим, что возможные алгоритмы такой проверки более-менее очевидны (они похожи на обычные алгоритмы кластеризации), гораздо более важной представляется первая часть задачи – про автоматическую генерацию описания класса. (И, по мнению авторов, именно такие алгоритмы и стоит называть искусственным интеллектом в чистом виде!)

Все используемые нами эвристики можно считать примерами т.н. подхода, основанного на правилах (на знаниях – т.н. rule-based approach). По этому поводу у авторов статьи имеется следующее мнение, подтверждённое в различных публикациях[2]: применение подходов, не основанных на знаниях эксперта (или «не сильно» от них зависящих; здесь мы имеем в виду, прежде всего, нейросетевые алгоритмы), очень часто сначала даёт значительно лучшие результаты, чем применение одного из подходов, основанных на знаниях эксперта; однако для разработки наиболее удачных алгоритмов необходимо применение подхода, основанного на правилах (rule-based approach) – или, по крайней мере, обоих этих подходов вместе[3]. Отметим, что по этому поводу нередко появляются соответствующие статьи в журнале ассоциации "International Computer Games Association", в котором рассматриваются статьи связанные с программированием не только игр, но и головоломок; среди статей этого журнала, связанных с применением подхода, основанного на правилах, отметим [7] и выходящую в 2012 г. статью [8].

РАЗДЕЛЯЮЩИЕ ЭЛЕМЕНТЫ В ОДНОМ ИЗ ВАРИАНТОВ ЗАДАЧИ УПАКОВКИ

Классический вариант задачи упаковки приведён в классической же монографии [9] [4]. Впоследствии рассматривалось много различных вариантов этой задачи ([10] и мн. др.) – ниже мы используем один из этих вариантов, который может быть получен из общего случая путём добавления специальных ограничений.

Итак, мы будем рассматривать задачу упаковки в контейнер на примере различных фигурок трёхмерного тетриса. Пусть каждая фигура состоит из пяти элементов – кубов, соединённых своими гранями (таких фигур всего 29). Контейнер представляет собой куб с ребром в 5 элементов. Требуется полностью заполнить контейнер уникальными фигурами.

Перед началом решения данной задачи нам требуется построить все эти уникальные фигуры и определить – может ли вообще задача иметь решение (нетрудно посчитать, что для заполнения контейнера необходимо 125 элементов, или 25 фигур из 29).

Так как само построение фигур не является основной целью работы, был применён очень простой рекурсивный алгоритм генерации фигур. Строить фигуры будем в декартовой системе координат. Разместим первый куб в начале системы координат (0,0,0). Каждый последующий куб будет размещаться со сдвигом на единицу (+1 либо –1) по каждой из осей относительно каждого уже размещённого куба. После размещения пятого элемента мы сравниваем получившуюся фигуру с уже имеющимися[5] и, если она уникальна, добавляем её к ним. После этого мы возвращаемся на шаг назад (т.е. убираем поставленный последним куб) и ставим элемент на новую позицию, или, если все позиции уже «пройдены», возвращаемся ещё на один шаг назад. Перебор останавливается тогда, когда не останется не пройдённых позиций для первого элемента.

Остановимся подробнее на вспомогательном алгоритме сравнения фигур. Перед сравнением фигур сдвинем их таким образом (на такой вектор), чтобы каждый элемент имел минимально возможные неотрицательные координаты по каждой из осей. При этом нужно учитывать, что две фигуры могут быть одинаковы, но развёрнуты относительно друг друга. Поэтому сравнивать каждую новую фигуру следует со всеми возможными разворотами уже имеющихся фигур. После описанных преобразований само сравнение будет представлять из себя сравнение координат кубов (для каждого куба из одной фигуры должен найтись куб с такими же координатами из другой). Применив вышеописанный подход, мы в итоге получаем 29 различных фигур. Так как количество найденных уникальных фигур превышает минимально возможное для заполнения контейнера, будем считать, что задача решение имеет.

Перед заполнением контейнера проведём с фигурами следующие преобразования. Так как во время размещения фигур нам понадобятся их различные ортогональные «развороты» – для ускорения работы программы – мы сгенерируем для каждой фигуры все возможные уникальные развороты. Для этого сместим исходную фигуру в положительную часть пространства координат и добавим её в набор доступных разворотов. После этого строим все возможные развороты данной фигуры, выравниваем их в системе координат и добавляем к уникальным разворотам, если сгенерированный разворот не совпал ни с одним из имеющихся. Сгенерировав таким способом все развороты для всех фигур, мы смещаем все эти развороты таким образом, чтобы первый куб оказался в начале координат.

Контейнер будет представлять собой трёхмерный массив размером 5 для каждого из 3 измерений. Нулём будем помечать пустые ячейки, числом от единицы до двадцати девяти – ячейки, занятые фигурой с соответствующим номером.

Итеративный процесс размещения фигур будет происходить следующим образом. Выбирается очередная свободная ячейка контейнера[6]. В ней размещается ну-

[2] Вероятно, впервые, – в одной из публикаций М.Минского, на русский язык, по-видимому, не переведённой; см., например, [6].

[3] На эту тему в Википедии (http://ru.wikipedia.org/, в статье, посвящённой М.Минскому), имеется следующая история про Минского и его ученика Сассмена: «Однажды, когда Сассмен был ещё стажёром, к нему зашёл Минский и застал его в момент отладки очередной программы для PDP-6. – Что ты делаешь? – спросил Минский. – Обучаю случайно связанную нейросеть играть в крестики-нолики, – ответил Сассмен. – А почему случайно связанную? – спросил Минский. – Не хочу, чтобы у неё было заложенное заранее мнение о том, как играть, – сказал Сассмен. Минский закрыл глаза. – Зачем ты закрыл глаза? – спросил Сассмен учителя. – Чтобы комната стала пустой. Тут Сассмен стал просветлённым». По-видимому, эту историю следует рассматривать не как анекдот, а как часть руководства по проектированию нейросетей (в частности) и алгоритмов искусственного интеллекта (вообще) – с применением подхода, основанного на правилах (rule-based approach).

[4] Также эта задача хорошо описана в Википедии – однако, по-видимому, на английской странице (http://en.wikipedia.org/wiki/Bin_packing_problem) это описание существенно подробнее, чем на соответствующей русской.

[5] Это делается с помощью специального вспомогательного алгоритма, включающего все возможные повороты, – подробнее см. далее.

[6] Важно отметить, что её можно считать частью структуры, описывающей очередной разделяющий элемент.

левой элемент фигуры. Далее высчитывается положение в контейнере для остальных элементов и проверяется, не накладывается ли оно на другие фигуры и не выходит ли за стенки контейнера. Если данные условия выполняются – считается, что фигура может быть размещена в данной позиции.

Размещаться фигуры будут посредством т.н. бэктрекинга – алгоритма с возвратами. На каждом шаге алгоритма мы размещаем определённую, не выбранную ранее фигуру (определённый её «разворот») – на определённой, допустимой позиции в контейнере, после чего переходим к следующему шагу. Данный процесс заканчивается, либо когда мы получили решение (в контейнере заняты все ячейки), либо когда свободные ячейки все еще есть, но разместить ни одну из свободных фигур мы уже не можем. В последнем случае мы возвращаемся на шаг назад, убираем поставленную на нем фигуру, и пробуем поставить очередную, не опробованную фигуру. Если список неопробованных фигур пуст, либо из имеющихся фигур нельзя разместить ни одной, то мы возвращаемся ещё на 1 шаг назад – и так далее.

Так как количество вариантов для перебора даже в такой «игрушечной» задаче очень велико, нам потребуются дополнительные критерии выбора фигуры для размещения на очередном шаге. Таким критерием стала «эвристика наибольшего касания» (подробнее см. [3]). Суть этой эвристики заключается в том, что мы считаем для каждого разворота каждой из доступных фигур общее количество соприкосновений ее элементов с элементами других размещенных фигур и «стенками» контейнера. После того как данная величина будет доступна для каждого разворота каждой «не размещенной» фигуры, мы сможем определить порядок фигур для размещения на этом шаге – фигуры с наибольшим значением будут использоваться в первую очередь.

В итоге, применяя описанный подход, удалось получить допустимые решения данной задачи. Время получения очередного подходящего решения (как функция от некоторых параметров эвристических алгоритмов) будет приведено в одной из следующих публикаций. Одно из возможных подходящих решений приведено на следующем рисунке:

D	A	A	A	A
E	E	B	B	B
E	F	B	C	B
E	F	C	C	C
F	F	C	G	G

D	H	H	H	A
E	I	K	K	L
N	I	I	L	L
N	N	I	L	G
F	N	N	L	G

D	H	O	J	J
P	I	K	S	R
P	K	K	R	R
P	P	Q	R	T
P		Q	Q	G

D	H	O	J	J
X	X	V	S	R
V	V	V	S	U
V	Y	S	S	T
Y	Z	Q	T	T

D	O	O	O	J
X	X	M	U	U
X	M	M	M	U
Y	Y	M	Z	U
Y	Z	Z	Z	T

Рис. 1. Допустимое решение задачи «Упаковка в контейнер».

ОБ ОДНОМ ПОДХОДЕ К РЕШЕНИЮ ГОЛОВОЛОМКИ MAHJONGG SOLITAIRE (麻將)

Маджонг (анг. Mahjongg) – игра, созданная ещё во времена древнего Китая. В данной статье рассматривается один из походов к решению головоломки Mahjongg solitaire[14]. При запуске программы на вход подаётся очередная комбинация Маджонга. При этом заранее известно, что такая комбинация раскладывается.

Сразу после подачи данных организуется их проверка, что обеспечивает их корректность. На следующем этапе осуществляется снятие фишек по специально определённому генетическому алгоритму ([12] и мн. др.). А именно, пусть существует число, являющееся оптимальным числовым показателем для снятия фишек, т.е. рейтинг. При этом для определения параметров этого рейтинга используются следующие числовые характеристики:

1) длина ряда фишек;
2) высота уровня;
3) координаты фишек в разных полушариях;
4) координаты фишек в одном полушарии;
5) вероятность появление новой пары фишек после удаления исходной.

Из этих характеристик составляются различные группы. На основе фитнесс-функции организуется интеллектуальный турнир, каждой группе выдаётся задание с головоломкой. Группа, давшая больше всего решений, объявляется победителем, проигравшие же отсеиваются.

Для нахождения рейтинга следует отыскать комбинацию числовых показателей фишки с наибольшим процентом решённых головоломок. Популяция представляет собой массив размера *size*, элементами которого являются особи типа <структура>. При создании особи мы прибегаем к вероятностному подходу.

Отбор особей для размножения

Создаётся временный массив, в который заносятся «победители» турнира.

1. В турнире участвуют:
   a) особь с порядковым номером счётчика;
   b) особь, выбранная случайным образом.
2. Проводится турнир на основе показаний фитнесс-функций. Во временный массив записывается особь с наибольшим показателем функции, ведь она более приспособленная; цикл повторяется *size* раз.
3. По окончании цикла во временном массиве содержится популяция, которая будет использоваться в дальнейшем.

Скрещивание особей

После отбора особей необходимо скрестить, чтобы получить новую популяцию. Чтобы в скрещивании участвовали все особи, была применена следующая интерпретация генетического алгоритма:

1) создаются одномерные массивы *mas* и *mas_tmp* размером *size*, заполненные полностью –1 и числами по порядку от 0 до *size* соответственно, *i=0*;
2) начинаем движение по массиву *mas_tmp*, , при *i=size* движение прекращается;
3) случайным образом выбирается значение *k* от 0 до *size*. Пока *mas[k]* не равно –1, продолжаем выбор *k*;
4) если *mas[k]=–1*, то *mas[k]=mas_tmp[i]*, *i=i+1*.

Таким образом, получается массив размера size, заполненный числами от 0 до size.

Алгоритм кроссовера

1. Устанавливаем *l=0*. Создаём «временную» популяцию. Из массива *mas* берутся элементы с номерами *l* и *l+1*.
2. Случайным образом выбираем «барьер» (число, выше которого происходит обмен между генами родителей).

3. Происходит стандартный для одноточечного кроссовера обмен хромосомами.

4. Полученные особи записываются во «временную» популяцию, счётчик *l* увеличивается на 2.

5. работаем с популяцией, хранящейся во временном массиве.

Мутация, создающая новую популяцию

После скрещивания M лучших особей переходят в новую популяцию, остальные *(size – M)* подвергаются мутации.

Завершение работы алгоритма

В са́мом начале работы алгоритма устанавливается особь с набором коэффициентов и фитнесс-функцией не являющимися искомым решением задача. В структуре хранится особь с наилучшей популяцией *max*.

На каждом этапе мы выполняем отбор, скрещиванием особи, мутируют те, у кого значение фитнесс-функции наибольшее – *max_tmp*. Значения *max* и *max_tmp* сравниваются. Самая приспособленная особь записывается в *max*.

ЗАКЛЮЧЕНИЕ

Выше уже были приведены ссылки на классиков искусственного интеллекта (М.Минского). Также у М.Минского встречается следующая мысль – о том, что именно «игрушки», головоломки – это и есть «самое важное в искусственном интеллекте» (см. также [11]). *Именно здесь* в наиболее простом и ясном виде проявляются *все методы* решения задач искусственного интеллекта – а «всё остальное» (в том числе и экспертные системы) – это уже «второстепенное».

В связи с этим интересно вспомнить проводившийся в течение нескольких лет открытый Зеленоградский турнир по программированию (http://zcontest.ru): практически каждая предлагавшаяся на нём задача (из 40 штук) является, с одной стороны, головоломкой, а с другой стороны – полноценной задачей дискретной оптимизации. Основная трудность решения каждой из них (с применением мультиэвристического подхода) состоит в выборе разделяющего элемента; приведём лишь некоторые примеры:

- http://zcontest.ru/2006.02/zttr.php – упрощённый вариант задачи, рассматривавшейся в данной статье; описание разделяющих элементов может быть получено на основе приведённого выше алгоритма;
- http://zcontest.ru/2006.02/zlib.php – разделяющими элементами являются команды;
- http://zcontest.ru/2008.02/zjawb.php – разделяющими элементами являются блоки фишек;
- http://zcontest.ru/2008.02/zrex.php – разделяющими элементами являются очередные главные операции формируемых регулярных выражений.

См. на эту тему также некоторые задачи, приведённые в [13].

Работа первого автора частично поддержана Федеральной целевой программой «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы (соглашение N 14.B37.21. 0233).

Работа второго и третьего авторов частично поддержана программой Министерства образования и науки РФ в рамках Госзадания Тольяттинского государственного университета на 2012 год (шифр 6.3072.2011).

СПИСОК ЛИТЕРАТУРЫ

1. B. Melnikov. “Discrete optimization problems – some new heuristic approaches”, Proceedings of the Eighth International Conference on High-Performance Computing in Asia-Pacific Region, IEEE Computer Society Washington, 2005, 73–80.
2. Б. Мельников. «Мультиэвристический подход к задачам дискретной оптимизации», Кибернетика и системный анализ (НАН Украины), 2006, № 3, 32–42.
3. А. Лысенко. «Об алгоритмах принятия решений в NP-полных задачах дискретной оптимизации», Вектор науки ТГУ, 2010, № 4 (14), стр. 33–35.
4. С. Баумгертнер, Б. Мельников. «Мультиэвристический подход к проблеме звездно-высотной минимизации недетерминированных конечных автоматов», Вестник Воронежского гос. унив., сер. Системный анализ и информационные технологии, 2010, № 1, С. 5–7.
5. С. Гудман, С. Хидетниеми. «Введение в разработку и анализ алгоритмов». – М., Мир, 1981.
6. M. Minsky, S. Papert. “Artificial intelligence”. – Univ. of Oregon Press, 1972.
7. U. Lorenz. “A new implementation of error analysis in game trees”, ICGA Journal (SCI), 2006, Vol. 29, No. 2, 55–64.
8. Kuo-Yuan Kao, I-Chen Wu, Yi-Chang Shan, Shi-Jim Yen. “Selection search for mean and temperature of multi-branch combinatorial games”, ICGA Journal (SCI), July 2012, in printing.
9. М. Гэри, Д. Джонсон. «Вычислительные машины и труднорешаемые задачи». – М., Мир, 1982.
10. S. Martello, P. Toth. “Knapsack Problems Algorithms and Computer Implementations”. – NY, John Wiley & Sons, 1990.
11. И. Братко. «Программирование на языке Пролог для искусственного интеллекта». – М., Мир, 1990.
12. С. Рассел, П. Норвиг. «Искусственный интеллект. Современный подход». – М., Вильямс, 2006.
13. B. Melnikov, E. Melnikova, “Some competition programming problems as the beginning of artificial intelligence”, Informatics in Education, 2007, Vol. 6, No. 2, 385–396.
14. Mahjongg Solitaire games [Электронный ресурс] : Mahjongg games и др. — Режим доступа: http://www.mah-jongg.ch/mahjongg/mahjongg.html, свободный. — Яз. англ.

THE USE OF MULTI-HEURISTIC APPROACH WHEN PROGRAMMING SOLUTIONS PUZZLES

A.A. Borgardt, student
A.M. Lysenko, postgraduate student
B.F. Melnikov, doctor of physics and mathematics sciences, professor
Togliatti State University, Togliatti (Russia)

Keyword: Bin packing problem; heuristics most touching; intractable problems; genetic algorithm

Annotation: The article discusses some solutions of discrete optimization based Multi-heuristic approach. This paper presents an approach to algorithmic Multi-heuristic solving puzzles. They have a finite number of situations and clearly defined rules. On the example of a three-dimensional Mahjong and Tetris.

УДК 004.456.45, 004.414.23

МНОГОПОТОЧНЫЙ АЛГОРИТМ МОНТЕ-КАРЛО МОДЕЛИРОВАНИЯ РОСТА НАНОКРИСТАЛЛОВ

А.А. Боргардт, студент
Ю.С. Нагорнов, кандидат физико-математических наук, старший научный сотрудник, доцент кафедры «Высшая математика»
Тольяттинский государственный университет, Тольятти (Россия)

Ключевые слова: моделирование методом Монте-Карло; рост нанокристаллов; многопоточный алгоритм; сетевое хранилище данных.

Аннотация. В статье рассмотрен однопоточный алгоритм на основе метода Монте-Карло, позволяющий моделировать образование нанокристаллов карбида кремния. С целью совершенствования процесса моделирования был разработан и апробирован алгоритм на основе принципа многопоточности. В работе также представлен алгоритм корректировки и синхронизации всех спорных точек поверхности.

ВВЕДЕНИЕ

Для моделирования процессов, происходящих во время роста кристаллов, их отжига, выращивания различных пленок и соединений на кристаллах широко используют метод Монте-Карло (МК) [1 - 6]. Преимуществами метода МК являются простота получаемых моделей и высокая производительность алгоритмов, что в свою очередь позволяет исследовать значительные по размерам объекты.

Физическая модель кинетической вариации метода Монте-Карло роста кристалла сводится к введению понятий событий или элементарных процессов реализуемых в системе. Каждое событие в реальной системе происходит с заданной вероятностью, при этом величина, равная числу событий в единицу времени является темпом данного процесса. Пусть N возможное число всех событий в данной конфигурации системы *C* в момент времени *t*, а темпы процессов R_a, где $a=1..N$, тогда полный темп определяется выражением:

$$Q = Q(C) = \sum_{a=1}^{N} R_a V^a (C \to C'), \qquad (1)$$

где $V^a(C \to C')$ — стохастическая матрица, определяющая возможность перехода системы из состояния C в C' путем реализации события a. Соответственно с этим, вероятность изменения конфигурации системы путем реализации события a равно $R_a/Q(C)$. Таким образом, элементарная реализация метода Монте-Карло заключается в том, что в каждый момент времени рассчитывается темп каждого элементарного события в системе, определяется их вероятность, в соответствии с которыми и выбирается выполняемое событие.

Основными рассматриваемыми типами процессов в моделируемой системе являлись стандартные события