Пространственный кроссовер для задач оптимизации выбора узлов при помощи генетического алгоритма

С.Л. СОТНИК

Днепродзержинский государственный технический университет

В статье предлагается вариант кроссовера, позволяющий увеличить эффективность генетических алгоритмов в задаче оптимизации выбора узлов.

У статті пропонується варіант кросоверу, що дозволяє збільшити ефективність генетичних алгоритмів у задачі оптимізації вибору вузлів.

Article describes modification of crossover operator, which increases efficiency of Genetic Algorithms for optimization of nodes selection task.

Введение. Задача оптимизации выбора узлов является огромной и хорошо проработанной теоретически областью, которая широко используется на практике. Это такие материальные применения, как обработка различных деталей в станках с ЧПУ, и такие информационные технологии, как выбор узлов для карт либо выбор опорных точек для 3D-моделей в компьютерных играх. Все трудно перечислить.

Однако, несмотря на огромное количество работ в данной области, остается очень много задач, математическое описание которых либо весьма сложно для аналитического решения, либо отсутствует. Именно для таких задач рекомендуется применять эвристические алгоритмы наподобие генетических.

Решаемая задача. При решении проблемы с использованием генетического алгоритма, часто встречается ситуация, когда неплохо было бы увеличить популяцию и сделать больше итераций алгоритма, создающих новые поколения. Этому препятствует весьма высокая ресурсоемкость генетических алгоритмов. Поэтому задача повышения поисковой эффективности ГА всегда является актуальной (под повышением поисковой эффективности здесь подразумевается нахождение решения с одним и тем же качеством за меньшее количество итераций и с меньшей популяцией, либо нахождение с теми же параметрами более качественного решения).

Для повышения эффективности ГА, как правило, в первую очередь подвергают настройке сами параметры алгоритма – различные вероятности (мутации, кроссовера и других генетических операторов), использование элитизма и т.п. Некоторые авторы для этого используют генетический же алгоритм. Однако и сами генетические операторы не являются догмой – для разных задач придумано великое множество различных разновидностей этих операторов.

После первых попыток решения задачи оптимизации выбора узлов, стало понятно, что используемый стандартный двухточечный оператор кроссовера страдает болезнью, известной как «проблема конкурирующих решений» («проблема перестановки») [1,2,3,4,5]. Суть её заключается в том, что одно и то же решение можно описать несколькими способами (в случае опорных точек – это точки с похожими координатами, закодированные в разных частях генома особи). Эти решения обладают похожими показателями качества, в то же время потомки, получаемые от скрещивания данных решений, имеют гораздо худшие показатели качества. Говоря образно, трудно получить хорошего потомка, если взять от родителей три ноги, и поставить одну из них вместо руки.

Другой причиной, затрудняющей поиск хорошего решения, является весьма большая разрушительная сила кроссовера. Дело в том, что очень часто опорные точки (узлы), находящиеся вблизи друг от друга, являются взаимосвязанными. В то же время, обычный кроссовер случайным образом выбирает отдельные из них, помещая в другой, уже сложившийся ансамбль точек (узлов).

Для того чтобы избежать описанных выше проблем, и была реализована новая разновидность кроссовера. Основным её отличием от известных вариантов, является использование информации о пространственном положении кодируемых в геноме опорных точек (узлов). Несмотря на то, что в вычислительном отношении пространственный кроссовер несколько сложнее простейших вариантов – одно- и двухточечного кроссовера, тот факт, что львиная доля вычислений в большинстве реальных задач приходится на процедуру оценки качества полученного решения, позволяет не учитывать это увеличение.

Пространственный кроссовер. Предлагаемая разновидность кроссовера во многом похожа на простой двухточечный кроссовер, за исключением того, что от родителей отбираются точки, близкие в пространственном отношении, а не в том, как они расположены в геноме. Кодирование генов соответствует схеме непрерывных генетических алгоритмов [6,7,8,9]. Параметры, относящиеся к одной точке (в первую очередь координаты), являются сцепленными (в генетическом смысле) признаками, т.е., разделение генотипа на элементы не производится внутри данных об одной точке, а только на границе между ними. Процедура синтеза потомка из генетического материала предков (кроссовера), в общем случае будет выглядеть следующим образом.

Есть два набора опорных точек (предки).

$$\Psi^K = \left(\psi_1^K, \psi_2^K \ldots \psi_n^K\right), \; k = 1,2 \qquad (1)$$

Введем классификатор, который позволяет разделить множества Ψ^K на 2 непересекающихся подмножества индексов элементов, объединение которых дает оригинальные множества:

$$
\begin{aligned}
&I^K = (1\ldots n),\\
&I_1^K \in I^K, I_2^K \in I^K,\\
&I_1^K \cap I_2^K = \varnothing, I_1^K \cup I_2^K = I^K,\\
&I_1^K = I_1^K(\Psi), I_2^K = I_2^K(\Psi),\ k = 1,2
\end{aligned}
\tag{2}
$$

В качестве классификатора в данной работе использовался (хотя, это конечно только один из возможных классификаторов) следующий:

Пусть $(\psi_1^K \ldots \psi_n^K)$ – наборы данных, которые помимо всего прочего, содержат также информацию о расположении точки в пространстве.

Ко множеству I_1^K мы отнесем индексы, соответствующие точкам, лежащим внутри гиперсферы с радиусом R и центром в точке P. R задаем случайным образом в диапазоне от 0 до D, где D – диаметр множества возможных точек для решаемой задачи. P является случайно выбранной точкой, находящейся в пределах, допустимых для задачи (обычно это случайная точка из множества допустимых точек решения).

Ко множеству I_2^K мы отнесем индексы, соответствующие точкам, лежащим вовне гиперсферы с радиусом R и центром в точке P.

Получить новое множество точек (потомка), мы можем путем объединения точек (точнее, участков генов, содержащих координаты одной точки и сцепленную с этой точкой информацию) предков.

$$
Z = (\psi_i^1 \in \Psi^1 : i \in I_1^1) \cup (\psi_i^2 \in \Psi^2 : i \in I_2^2) \tag{3}
$$

В случае если mes Z > n, то удаляем из Z случайно выбранные элементы, в количестве mes Z – n.

Если mes Z < n, то пополняем множество Z неиспользованными точками предков.

$$
\begin{aligned}
Z := Z \cup \{&(\psi_i^1 \in \Psi^1 : i \in I_2^1) \vee\\
&\vee (\psi_i^2 \in \Psi^2 : i \in I_1^2)\}
\end{aligned}
\tag{4}
$$

Процедура пополнения повторяется до тех пор, пока не будет выполнено равенство mes Z = n.

Графически, схема пространственного кроссовера (для двухмерного случая) изображена на рис. 1.

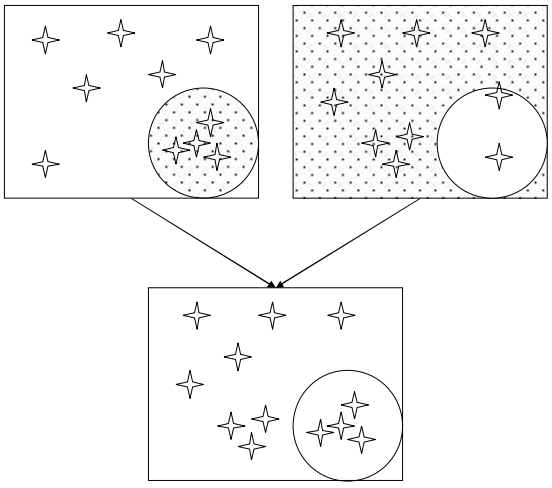


Рис. 1. Графическое изображение пространственного кроссовера.

После этапа синтеза, изображенного на рис. 1, происходит этап коррекции – ведь в общем случае при таком алгоритме объединения генов, их суммарное количество в предках и потомке совпадать не будет. Поэтому, если мы хотим, чтобы у нас оставалось одинаковое количество точек из поколения в поколение, мы убираем лишние точки, либо добавляем неиспользованные.

Проверка. Для проверки эффективности предлагаемого алгоритма объединения генетического материала, был создан соответствующий инструментарий на языке C# (платформа .NET Framework), позволяющий сравнить работу ГА с двухточечным и пространственным кроссовером на одной и той же задаче.

Задача заключалась в следующем – существует эталонное RGB изображение. Необходимо максимально точно восстановить его, используя ограниченное количество опорных точек. Точки, не являющиеся опорными, принимают цвет ближайшей опорной точки.

Мерой близости точек была выбрана метрика Минковского (5).

$$
d = |x_1 - x_2| + |y_1 - y_2| \tag{5}
$$

Как несложно заметить, для выбранной метрики, вырезаемое «окно» из рис.1 будет иметь форму ромба.

Показателем качества решения была выбрана ошибка восстановления изображения в пространстве L_2 (6).

$$
e = \sum_{x=1}^{Width} \sum_{y=1}^{Height} \sum_{c=1}^{3} \left(Y_{x,y,c} - \tilde{Y}_{x,y,c}\right)^2, \tag{6}
$$

где $Y_{x,y,c}$ и $\tilde{Y}_{x,y,c}$ - яркость цветовой компоненты с в точке с координатами (x, y) в оригинальном и восстановленном изображении соответственно.

Указанные ниже результаты получены на изображении, приведенном на рис.2а., однако аналогичные закономерности были получены и на многих других изображениях.

При восстановлении, ГА запускался при одних и тех же настройках. Единственным отличием был выбор того или иного варианта кроссовера. Предварительно было выяснено, что оптимальные (для эволюции в 200 поколений и популяции в 100 особей) настройки вероятности кроссовера лежат в районе p_C=0.5, а вероятность мутации p_M=0…0.1. Причем, эти значения одинаково хороши как для обычного двухточечного кроссовера, так и для пространственного.

Нулевая вероятность мутации видимо показала хороший результат потому, что за 200 поколений, потомки успевают исследовать еще далеко не все комбинации исходных генов, поэтому еще не сильно нуждаются в мутации, которая позволяет точнее подстроить параметры при поиске локального экстремума, в то же время в подавляющем числе случаем, нанося большой ущерб потомкам. Поэтому весьма вероятно, что при достаточно большом пуле опорных точек, имеющихся в решениях, более продуктивным будет применение (не обязательно для каждого поколения) градиентных методов поиска локального экстремума. Изучение этой модификации алгоритма планируется в ближайшем будущем.

Качество поиска решения (системы опорных точек) в шести экспериментах, представлено на графике 1. Для удобства визуального восприятия, ошибка восстановления пересчитана по формуле (7).

$$e = \sqrt{\sum_{x=1}^{Width} \sum_{y=1}^{Height} \sum_{c=1}^{3} \left(Y_{x,y,c} - \tilde{Y}_{x,y,c} \right)^2} \quad . \qquad (7)$$

Как несложно убедиться, производительность работы генетического алгоритма при использовании пространственного кроссовера выше. Уже было сказано выше, данные, приведенные на графике, иллюстрируют данную закономерность, стабильно выявляемую также на других изображениях и с другими настройками, как алгоритма, так и восстанавливающего инструмента (с разным количеством точек восстановления, с другими мерами близости точек). А именно – при использовании пространственного кроссовера решения находятся быстрее, а само пространство возможных решений исследуется более качественно.

Рис. 2 а. Тестовое изображение.

Рис. 2 б. Типичное восстановление изображения при 1000 опорных точек, расположенных случайным образом. Ошибка восстановления 1133676018.

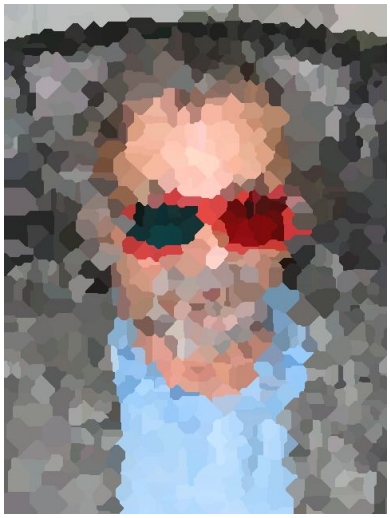
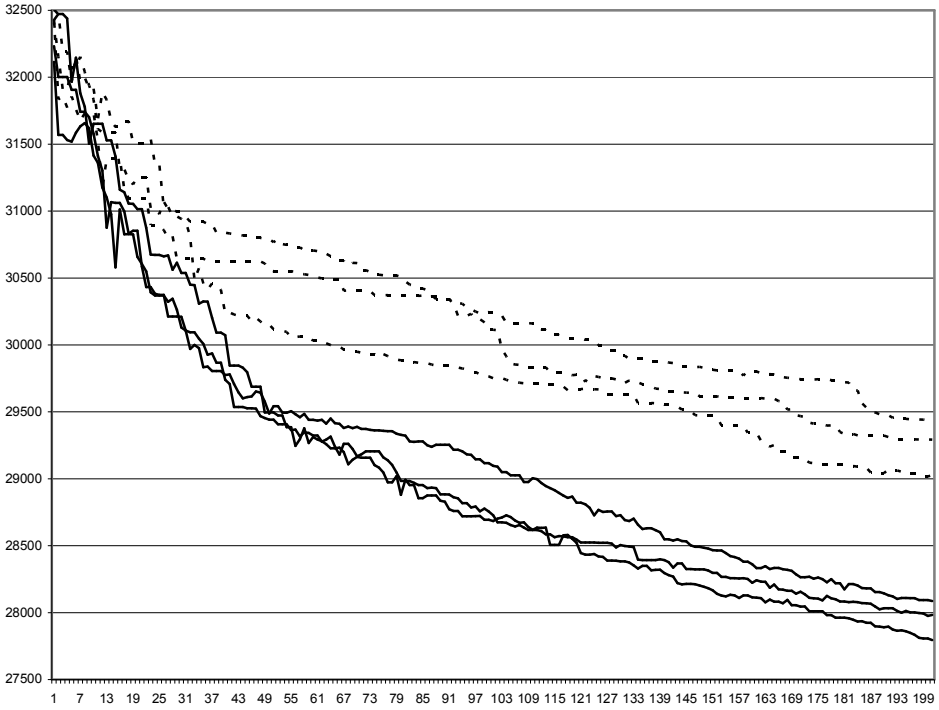


Рис. 2 в. Восстановленное изображение при ошибке восстановления 783019394. Контуры резких границ прорисованы более четко и точно.

График 1. Динамика ошибки лучшего экземпляра в зависимости от поколения (пунктир – двухточечный кроссовер, сплошные линии – пространственный).



Выводы. Работа над задачей оптимизации опорных точек при помощи генетического алгоритма показала, что тонкая настройка алгоритма, включающая в себя модификацию его базовых генетических операторов и схемы кодирования генетической информации, позволяет существенно повысить его производительность при поиске решений указанной задачи. В ближайших планах автора исследование различных вариантов реализации генетического оператора мутации и онтогенеза (изменение параметров решения в процессе существования особи) для улучшения производительности ГА при поиске локальных экстремумов у конкурирующих решений.

ЛИТЕРАТУРА

1. Perez-Bergquist A. S. Applying ESP and Region Specialists to Neuro-Evolution for Go. Technical Report CSTR01-24. The University of Texas at Austin, 2001. See also: http://www.nn.cs.utexas.edu/
2. Saravanan N., Fogel D. B. Evolving neural control systems // IEEE Expert, 1995, pp. 23-27.
3. Whitley D., Starkweather T., Bogart C. Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity // Parallel Computing, 1990, Vol. 14, pp. 341-361. See also: http://www.cs.colostate.edu/~genitor/
4. Shaffer J., Whitley D., Eshelmann L. Combination of Genetic Algorithms and Neural Networks: A Survey of the State of the Art // In Proceedings of the International Workshop on Combination of Genetic Algorithms and Neural Networks (COGANN-92). Los Alamos, CA: IEEE Computer Society Press, 1992. pp. 1-37. See also: http://www.cs.colostate.edu/~genitor/
5. Цой Ю.Р., Спицын В.Г., Эволюционный подход к настройке и обучению искусственных нейронных сетей. // Журнал «Нейроинформатика», 2006, том 1, №1, стр.34-61
6. Herrera F., Lozano M., Verdegay J.L. Tackling real-coded genetic algorithms: operators and tools for the behaviour analysis // Artificial Intelligence Review, Vol. 12, No. 4, 1998. – P. 265-319.
7. Herrera F., Lozano M., Sanchez A.M. Hybrid Crossover Operators for Real-Coded Genetic Algorithms: An Experimental Study // Soft Comput. 9(4): 280-298 (2005).
8. Wright A. Genetic algorithms for real parameter optimization // Foundations of Genetic Algorithms, V. 1. – 1991. – P. 205-218.
9. Deb, K. and Kumar, A. (1995). Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems. Complex Systems, 9(6), 431--454.

пост. 28.03.07

Дослідження *B*-сплайну п’ятого порядку та їх лінійної комбінації

П.О. ПРИСТАВКА, О.Г. ЧОЛИШКІНА

Дніпропетровський національний університет

Отримано властивості *B*-сплайну п’ятого порядку, визначеного на рівномірному розбитті вісі аргументу. Досліджено властивості поліноміального сплайну на основі *B*-сплайнів п’ятого порядку, що є близьким до інтерполяційного у середньому.

Получены свойства *B*-сплайна пятого порядка, определённого на равномерном разбиении оси аргумента. Исследованы свойства полиномиального сплайна на основе *B*-сплайнов пятого порядка, близького к интерполяционному в середнем.

The features of the *B*-spline of the fifth order have been obtained. This spline has been determined on the even breaking up of the axis of the argument. The features of the polynomial spline based on the *B*-spline of the fifth order which is close to the interpolar on an average have been investigate.

Постановка проблеми. В задачах обробки послідовностей відліків гладких функцій, що є результатами вимірювань чи подання сигналів у цифровому форматі, виникає потреба врахування факту наявних похибок в даних. В такій постановці задачі застосування мають два типи методів апроксимації: високоточні – такі, що враховують осциляції функцій та методи згладжування, спрямовані на оцінку тренду. Для останнього типу обчислювальний аспект застосування процедур локальної апроксимації є більш привабливим у порівнянні з класичним підходом побудови моделей на підставі методу найменших квадратів.

Останні десятиріччя обсяги даних, що підлягають обробці зростають і така тенденція є постійною. Для великих обсягів даних ширина вікна ковзного середнього (дискретної згортки функцій) вже не є вирішальним фактором стримування при застосуванні відповідного методу апроксимації. Більш того, ширина вікна (кількість членів дискретної згортки) для конкретних задач може мати інформативну змістовну інтерпретацію.

У зв’язку зі зробленими зазначеннями актуальною є задача отримання нових методів локальної апроксимації гладких функцій, заданих значеннями на рівномірних сітках вузлів, причому таких, що відповідають вимогам наявності високих згладжувальних властивостей при відносно низькій обчислювальній складності.

