Генетические алгоритмы

Компьютерные программы, «эволюционирующие» в ходе своеобразного естественного отбора, способны решать сложные задачи, разобраться в которых порой трудно даже авторам этих программ

ДЖОН Х. ХОЛЛАНД

ЖИВЫЕ организмы — прирожденные решатели задач. Универсальность их способностей не идет ни в какое сравнение даже с самыми лучшими компьютерными программами. Это особенно досадно для программистов-теоретиков, затрачивающих месяцы, а иногда и годы на создание какого-либо алгоритма, в то время как организмы легко справляются с задачей благодаря казалось бы слепому механизму эволюции и естественного отбора.

Ученые-прагматики рассматривают замечательные возможности эволюции как нечто, чему следовало бы скорее подражать, нежели завидовать. Естественный отбор устраняет одно из главных затруднений при создании программ — необходимость предусматривать заранее все особенности решаемой задачи и действия, которые должна предпринимать программа, когда она сталкивается с этими особенностями. Поставив себе на службу механизм эволюции, исследователи смогут «выводить» программы, способные решать такие задачи, структуру которых ни один человек не может полностью осознать. В действительности, эти так называемые генетические алгоритмы уже продемонстрировали способность находить новаторские решения при конструировании таких сложных систем, как реактивные двигатели.

Генетические алгоритмы позволяют исследовать значительно более широкое многообразие возможных решений задачи по сравнению с обычными компьютерными программами. Более того, когда исследователи изучают процесс естественного отбора программ в строго контролируемых и хорошо понимаемых условиях, достигаемые ими практические результаты могут пролить свет на некоторые аспекты природной эволюции, обусловившей возникновение жизни и разума на Земле.

Большинство организмов эволюционирует посредством двух основных процессов: естественного отбора и полового размножения. Первый определяет, какие особи популяции выживут, чтобы дать потомство, а второй обеспечивает перемешивание и рекомбинацию генов у этого потомства. При слиянии сперматозоида и яйцеклетки соответствующие хромосомы располагаются друг перед другом и затем перекрещиваются на какую-то часть своей длины, обмениваясь, таким образом, генетическим материалом. Такое перемешивание позволяет организмам эволюционировать быстрее, чем если бы каждый потомок просто имел копию генов одного родителя, которая время от времени подвергается мутациям. (Хотя одноклеточные организмы и не спариваются, как люди иногда себе представляют, они обмениваются генетическим материалом, и их эволюцию вполне можно описывать аналогичным образом.)

С отбором все обстоит очень просто: если организм не выдерживает того или иного испытания на жизнеспособность, например не распознает опасности в хищнике или не находит своей пищи, то он погибает. Поэтому у исследователей также не возникает особых трудностей в выбраковке неудачных алгоритмов. Если программа по замыслу ее создателя должна, например, сортировать числа в порядке возрастания, то нужно лишь проверить, чтобы каждый элемент, отобранный программой, был больше предыдущего.

Уже на протяжении тысячелетий люди пользуются сочетанием скрещивания и отбора для выведения лучших сортов сельскохозяйственных культур, скаковых лошадей или декоративных растений. Однако не так-то просто перенести эти процедуры на компьютерные программы. Главная трудность заключается в построении «генетического кода», представляющего структуру различных программ, подобно тому, как ДНК представляет структуру человека или мыши. Например, спаривание или мутация в текстах программы на Фортране в большинстве случаев не приведет к появлению лучшей или худшей программы на Фортране, скорее всего в результате мы вовсе не получим никакой программы.

Первые попытки скрестить информатику и эволюцию в конце 50-х — начале 60-х годов были не очень впечатляющими, поскольку в них реализовывались принципы, изложенные в биологической учебной литературе того времени, и для получения новых комбинаций генов в большей мере использовался механизм мутаций, нежели спаривания. Затем, также в начале 60-х годов, Ханс Дж. Бремерман из Калифорнийского университета в Беркли ввел механизм спаривания, при котором свойства потомства определяются суммированием соответствующих генов двух родителей. Однако процедура спаривания была ограниченной, поскольку применялась только к свойствам, которые можно было суммировать по смыслу этой операции.

В ТЕ ГОДЫ я занимался математическим анализом адаптации и пришел к убеждению, что рекомбинация групп генов при спаривании является важнейшим фактором эволюции. К середине 60-х годов я разработал метод программирования — генетический алгоритм — который хорошо подходил для эволюционного процесса, обусловленного как спариванием, так и мутациями. На протяжении следующего десятилетия я пытался расширить сферу применения генетических алгоритмов путем построения генетического кода, который мог бы представлять структуру любой компьютерной программы.

В результате была разработана классифицирующая система, состоящая из набора правил, по которым выполнялись определенные действия всякий раз, когда условия их применения удовлетворялись фрагментом информационного массива, анализировавшимся в данный момент времени. Условия и действия представлены цепочками битов, соответствующих присутствию или отсутствию конкретных характеристик на входе и вы-

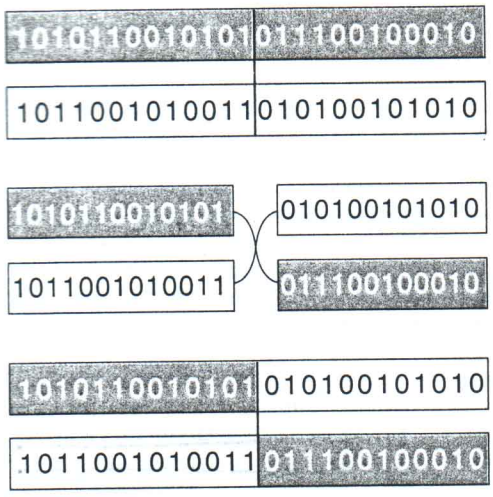
ходе правил. Для каждой присутствующей характеристики цепочка содержала единицу в соответствующей позиции, а для каждой отсутствующей — ноль. Например, классифицирующее правило, распознающее собак, могло содержать единицы в битах, соответствующих понятиям «поросший шерстью», «лающий», «преданный» и «бегающий за палкой», и нули в позициях, означающих «металлический», «говорящий по турецки» и «обладающий кредитной карточкой». В более реалистичных моделях программист может выбирать простейшие характеристики, так чтобы их можно было комбинировать (как, например, в игре «двадцать вопросов») с целью успешной классификации в широком диапазоне объектов и ситуаций.

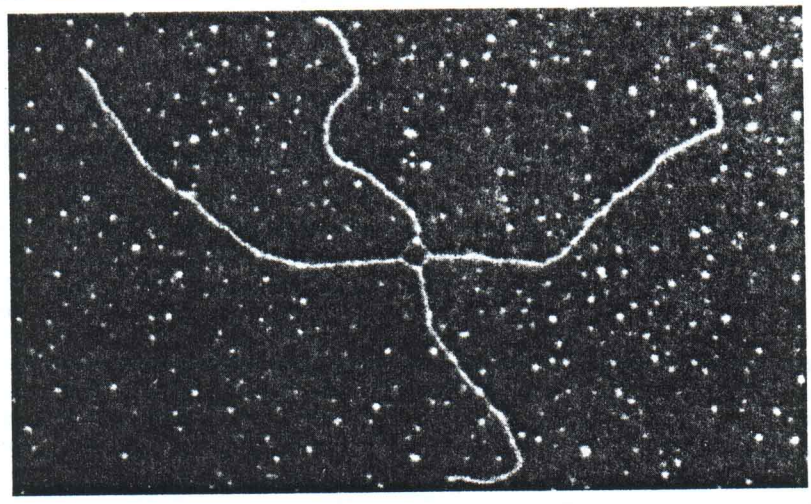
Эти правила, превосходно справляющиеся с задачами классификации, можно также приспособить для активации определенных действий, связывая отдельные биты их выхода с соответствующими поведенческими актами (см. рисунок на с. 35). Любую программу, написанную на каком-нибудь стандартном языке программирования, можно переписать, превратив ее в классификационную систему.

Чтобы вывести в ходе эволюционного процесса классификационные правила, позволяющие решить какую-то конкретную задачу, можно просто начать с популяции случайных цепочек нулей и единиц и оценивать каждую цепочку в зависимости от показываемых ею результатов. Судя по характеру решаемой задачи, в качестве критерия жизнеспособности можно принять прибыль в бизнесе, счет очков в игре, количество совершаемых ошибок или любые другие показатели. Высококачественные цепочки спариваются, низкокачественные погибают. По мере смены поколений цепочки, ассоциированные с лучшими вариантами решений, начнут доминировать. Кроме того, благодаря процессу спаривания эти цепочки будут комбинироваться новыми и новыми способами, порождая все более совершенные алгоритмы решения. Задачи, поддающиеся описанному методу, варьируют в широких пределах — от выработки новаторских стратегий в теории игр до конструирования сложных механических систем.

В ПЕРЕВОДЕ на язык генетических алгоритмов поиск хороших решений задачи сводится к поиску определенных двоичных цепочек. Множе-

ПЧЕЛИНАЯ ОРХИДЕЯ демонстрирует специфичность, с которой естественный генетический отбор может привязать организм к определенной нише. Цветок, напоминающий пчелу-самку, опыляется пчелами-самцами, пытающимися спариваться с ним. Механизмы, сходные с естественным отбором, по утверждению автора, могут порождать компьютерные программы (так называемые генетические алгоритмы), способные решать такие сложные задачи, как конструирование турбины реактивного двигателя или сетей связи.





КРОССИНГОВЕР — фундаментальный механизм перераспределения генов как для реальных организмов, так и для генетических алгоритмов. Хромосомы располагаются линия в линию и обмениваются порциями генетического кода, лежащими по одну из сторон от точки перекрещивания.

ство всевозможных цепочек можно рассматривать как некий ландшафт, в котором впадины отмечают расположение цепочек, кодирующих плохие решения, а высшая точка символизирует наилучшее возможное решение.

Области в пространстве решений можно также определять по наличию нулей или единиц в определенных позициях цепочек. По сути это своеобразный двоичный эквивалент системы координат. Например, множество всех цепочек, начинающихся с 1, представляет собой некую область в пространстве возможностей. То же самое можно сказать и обо всех цепочках, начинающихся с 0, или таких, которые содержат 1 в четвертой позиции, 0 в пятой и 1 в шестой, и т.д.

Одним из традиционных способов анализа таких поверхностей является метод наискорейшего подъема — начать в произвольной точке и, если небольшое изменение улучшает качество вашего решения, продолжать двигаться в том же направлении; в противном случае двигайтесь в противоположную сторону. Однако сложные задачи порождают поверхности со многими пиками. По мере того как возрастает число измерений в пространстве решений, у ландшафта могут появляться туннели, мосты и даже более сложные топологические образования. Найти нужный холм или даже определить, где верх, а где низ, становится все труднее. В добавок такие поисковые пространства обычно имеют колоссальные размеры. Если принять, например, что каждый шахматный ход имеет в среднем 10 альтернативных вариантов, а типичная игра состоит из 30 ходов с той и другой стороны, то существует около 10^{60} стратегий игры (большинство из которых, конечно, плохие).

Генетические алгоритмы покрывают этот ландшафт сетью. Множество цепочек в развивающейся популяции одновременно тестирует многие его области. Точнее, частота, с которой генетический алгоритм тестирует различные области, пропорциональна среднему «возвышению» области, то есть вероятности отыскания хорошего решения в ближайшей окрестности.

Эта примечательная способность генетических алгоритмов концентрировать внимание на наиболее перспективных частях пространства решений является прямым следствием их способности комбинировать цепочки, содержащие частные решения. Сначала оценивается каждая цепочка популяции, чтобы определить, насколько успешна стратегия, которую она кодирует. На втором этапе получившие наивысшие оценки цепочки спариваются. Две цепочки вытягиваются друг перед другом, на линии их соприкосновения случайным образом выбирается точка, и части, расположенные слева от этой точки, меняются местами, образуя двух потомков: один из них содержит символы первой цепочки вплоть до точки перекрещивания, а далее — символы второй; другой же потомок содержит дополнительный набор символов (см. рисунок вверху. Биологические хромосомы перекрещиваются друг с другом, когда две гаметы сливаются, образуя зиготу, и, таким образом, процесс кроссинговера в генетических алгоритмах, действительно, в значительной степени моделирует процесс биологический. Потомки не замещают родительскую пару; вместо этого они замещают цепочки с низкими показателями, которые отбрасываются в каждом поколении, так что численность популяции со временем остается одинаковой.

На третьем этапе мутации изменяют небольшой фрагмент цепочек: приблизительно один из каждых 10 000 символов переключается с 0 на 1 или наоборот. Сами по себе мутации обычно не дают прогресса в поиске решения, но они страхуют от возникновения однородной популяции, не способной к дальнейшей эволюции.

ГЕНЕТИЧЕСКИЙ алгоритм интенсивно исследует перспективные, или «целевые», области пространства решений, поскольку в результате многократного размножения и скрещивания в этих областях скапливается все большее количество цепочек. В качестве родителей алгоритм выбирает наилучшие цепочки, и поэтому цепочки с показателями выше средних (они то и попадают в целевые области) в следующем поколении будут иметь больше потомков.

На самом деле, численность цепочек в данной области возрастает со скоростью, пропорциональной статистической оценке жизнеспособности этой области. Следуя статистическим методам, мы были бы вынуждены оценивать десятки пробных цепочек из тысяч или даже миллионов областей, чтобы определить средние показатели каждой области. Генетический алгоритм достигает тех же результатов со значительно меньшим количеством цепочек и практически не производя никаких вычислений.

Ключом к пониманию этого довольно удивительного поведения является тот факт, что одна цепочка

принадлежит сразу всем областям, в которых присутствует любой из составляющих ее битов. Например, цепочка 11011001 одновременно принадлежит областям 11****** (здесь * означает, что значение данного бита безразлично), 1******1, **0**00* и т.д. Наиболее крупные области (содержащие много неуказанных битов) обычно тестируются значительной частью всех цепочек, составляющих популяцию. Так, генетический алгоритм, манипулирующий популяцией из нескольких тысяч цепочек, на самом деле тестирует немного большее количество областей. Этот неявный параллелизм обеспечивает генетическому алгоритму его главное преимущество по сравнению с другими процессами поиска решений задач.

Механизм скрещивания усложняет эффект неявного параллелизма. Цель скрещивания цепочек в генетическом алгоритме заключается в том, чтобы протестировать новые области, вместо того чтобы тестировать снова и снова одну и ту же цепочку в новых поколениях. Но этот процесс может также «переместить» потомство из одной области в другую, в результате чего частота тестирования различных областей отклоняется от строго пропорциональной зависимости от среднего показателя эффективности. Это отклонение замедляет темпы эволюции.

Вероятность того, что потомство двух цепочек покинет область обитания их родителей, зависит от расстояния между единицами и нулями, которые определяют границы этой области. Потомство цепочки, тестирующей область 10****, например, выйдет за пределы этой области только в том случае, если перекрещивание начнется во второй позиции цепочки — в одном случае из пяти для цепочки, содержащей шесть генов. (Тот же строительный блок подвергнется риску только в одном из 999 случаев, если он содержится в цепочке из 1000 генов.) Потомство шестигенной цепочки, тестирующей область 1****1, рискует выйти из области обитания родителей независимо от того, где произойдет перекрещивание их хромосом.

Близко соседствующие единицы и нули, определяющие пределы области, называются компактными строительными блоками. У них наибольшая вероятность пережить скрещивание в неизменном виде и, таким образом, распространиться в будущих поколениях со скоростью, пропорциональной средней эффективности цепочек, которые будут содержать в себе эти блоки. Хотя механизм размножения, включающий скрещивание, не дает возможности тестировать все области пространства с частотой, пропорциональной их качеству, он обеспечивает ее для всех областей, которые определяются кампактными строительными блоками. Количество же компактно определенных строительных блоков в популяции цепочек намного превосходит число самих цепочек, благодаря чему генетический алгоритм все же проявляет свойства неявного параллелизма.

Интересно отметить, что операция, называемая в генетике живых организмов инверсией, время от времени меняет расположение генов таким образом, что гены, расположенные у родителей далеко друг от друга, могут стать соседями в потомстве. В нашей модели это равносильно переопределению строительного блока, в результате которого он становится более компактным и менее уязвимым по отношению к перекресту. Если строительный блок определяет область с высокой средней эффективностью, то более компактная версия автоматически замещает менее компактную, поскольку теряет меньше потомков в результате ошибок, возникающих при воспроизводстве. Как результат система адаптации, использующая инверсию, может обнаруживать и отдавать преимущество компактным версиям полезных строительных блоков.

Неявный параллелизм генетического алгоритма позволяет ему тестировать и использовать большое количество областей в пространстве поиска, манипулируя относительно небольшим числом цепочек. Этот скрытый параллелизм помогает также генетическим алгоритмам справляться с нелинейными задачами, то есть в тех случаях, когда эффективность цепочки, содержащей два полезных строительных блока, может оказаться зна-

Как построить классификационную систему

Для построения компьютерного алгоритма, который может эволюционировать, необходим способ представления программы, с тем чтобы любое изменение в ее генотипе (биты, которые составляют программу) приводило к значимому изменению в ее фенотипе (тому, что делает программа). Классификатор состоит просто из цепочек, представляющих возможные характеристики входа программы и действий, которые необходимо выполнить (*внизу*). Изменение любого символа в цепочке изменяет ее поведение.

АЛФАВИТ КЛАССИФИКАТОРА	1 = ДА	0 = НЕТ	# = НЕ ИМЕЕТ ЗНАЧЕНИЯ

Классификационная система, действующая, например, подобно лягушке, могла бы содержать цепочки, которые реагировали бы на объекты, находящиеся в поле зрения лягушки. В зависимости от движения объекта, его размера, положения и других характеристик, лягушка нападает на него, преследует или проявляет спокойствие. Несколько цепочек могут составить вход, моделирующий такое же поведение; цепочка с наименьшим числом символов «не обращать внимания» задает поведение системы.

ДВИЖЕТСЯ	НА ЗЕМЛЕ	БОЛЬШОЙ	ДАЛЕКО	ПОЛОСАТЫЙ	СПАСАЙСЯ!	НАПАДАЙ!
					ВЫХОД	
ВХОД						
1	#	#	#	#	1	0

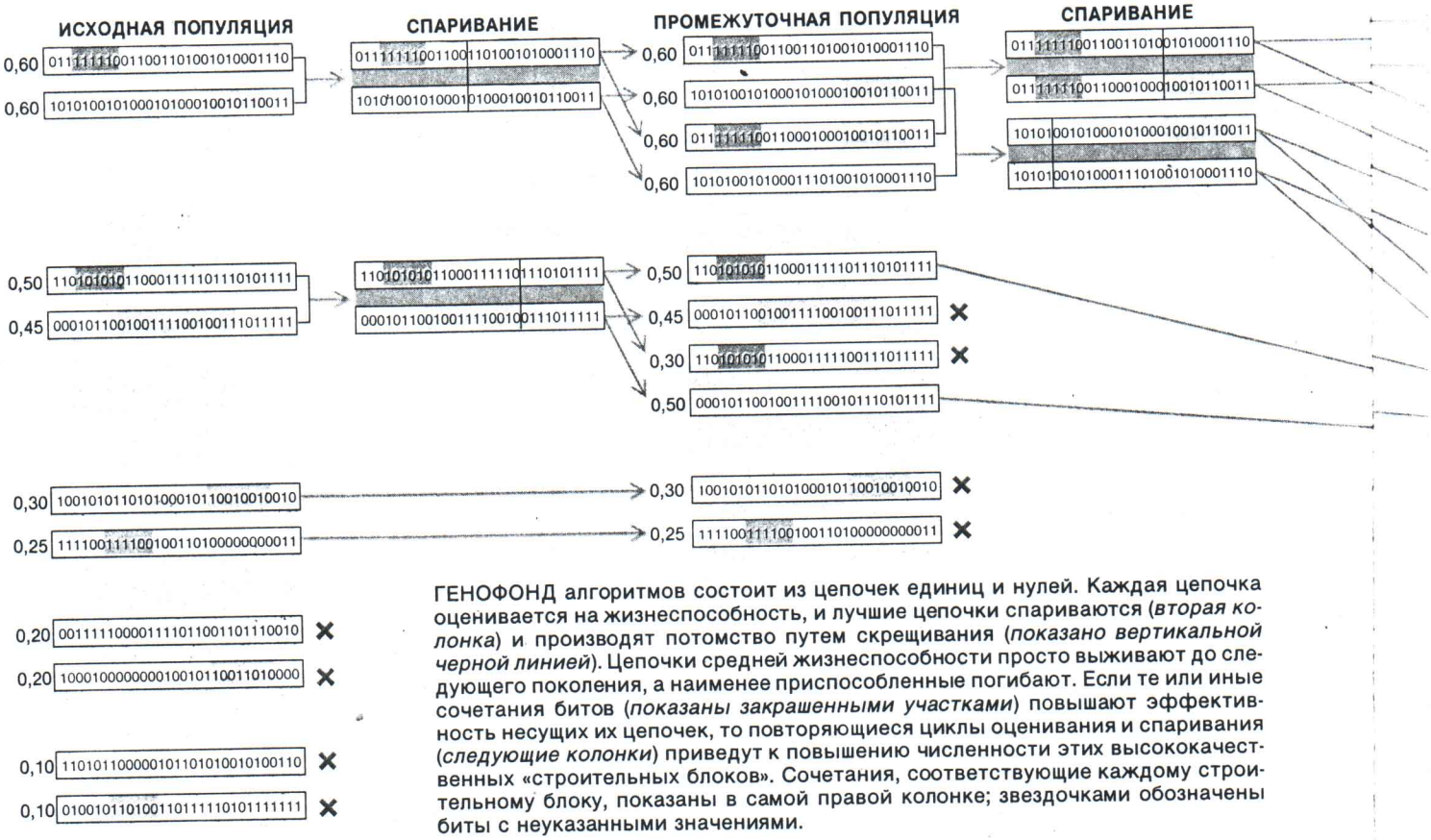
ЕСЛИ ОБЪЕКТ ДВИЖЕТСЯ, СПАСАЙСЯ

1	0	0	0	#	0	1

ЕСЛИ ОБЪЕКТ ДВИЖЕТСЯ, В ВОЗДУХЕ, НЕБОЛЬШОЙ И РЯДОМ, НАПАДАЙ

1	0	0	0	1	0	0

ЕСЛИ ОБЪЕКТ ДВИЖЕТСЯ, В ВОЗДУХЕ, НЕБОЛЬШОЙ И ПОЛОСАТЫЙ, НЕ ОБРАЩАЙ ВНИМАНИЯ



ГЕНОФОНД алгоритмов состоит из цепочек единиц и нулей. Каждая цепочка оценивается на жизнеспособность, и лучшие цепочки спариваются (*вторая колонка*) и производят потомство путем скрещивания (*показано вертикальной черной линией*). Цепочки средней жизнеспособности просто выживают до следующего поколения, а наименее приспособленные погибают. Если те или иные сочетания битов (*показаны закрашенными участками*) повышают эффективность несущих их цепочек, то повторяющиеся циклы оценивания и спаривания (*следующие колонки*) приведут к повышению численности этих высококачественных «строительных блоков». Сочетания, соответствующие каждому строительному блоку, показаны в самой правой колонке; звездочками обозначены биты с неуказанными значениями.

чительно выше (или значительно ниже), чем сумма эффективностей каждого из этих строительных блоков.

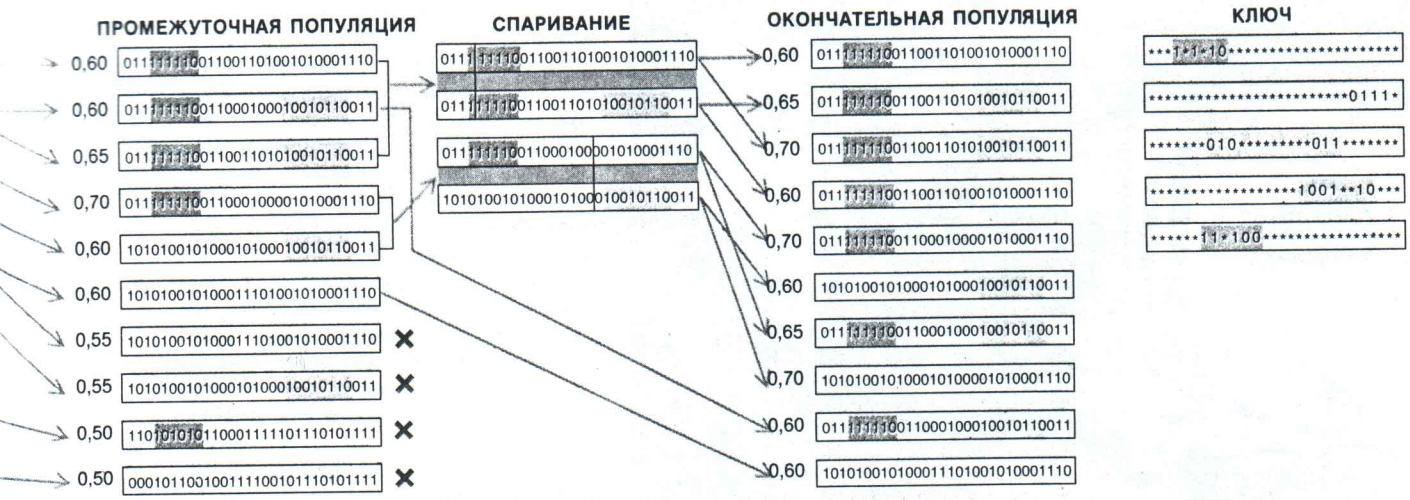
Линейные задачи представляют уменьшенное пространство поиска, поскольку присутствие 1 или 0 в определенной позиции цепочки не оказывает никакого влияния на эффективность, обусловленную присутствием 1 или 0 где-то еще. Пространство 1000-разрядных цепочек, например, содержит более 3^{1000} вариантов, но если задача линейна, то алгоритму необходимо исследовать лишь цепочки, содержащие 1 или 0 в каждой позиции, то есть всего 2000 вариантов.

Когда задача нелинейна, трудности резко возрастают. Например, эффективность области *01*** может быть выше средней для популяции, хотя степени эффективности, соответствующие областям *0**** и **1***, ниже средней для популяции. Нелинейность не означает, конечно, что полезных строительных блоков не существует, просто блоков, состоящих из одиночных единиц и нулей, уже недостаточно. А это свойство в свою очередь приводит к взрывному росту числа возможностей: множество всех цепочек длиной в 20 бит содержит уже более трех миллиардов строительных блоков. К счастью, скрытым параллелизмом и здесь можно воспользоваться. В популяции из нескольких тысяч цепочек многие компактные строительные блоки будут появляться в 100 и более цепочках, чего вполне достаточно, чтобы получить хорошую статистическую выборку. Строительные блоки, пользующиеся нелинейностью для достижения эффективности, превышающей средние показатели, автоматически будут чаще применяться в будущих поколениях.

Помимо того что генетический алгоритм справляется с нелинейностью, он помогает решить проблему, которая долго не поддавалась обычным методам решения задач: отыскание баланса между исследованием и эксплуатацией найденных стратегий. Например, как только программа находит хорошую стратегию в шахматной игре, возникает соблазн воспользоваться ею. Но за это приходится расплачиваться, поскольку эксплуатация найденной стратегии сделает открытие действительно новаторских стратегий уже маловероятным. Прогресс достигается путем опробирования новых, рискованных вариантов. Поскольку многие рискованные варианты оказываются неудачными, исследование влечет за собой снижение эффективности. Определить, в какой степени настоящее можно заложить на будущее, — это классическая проблема для всех систем, которые обучаются и адаптируются.

Подход генетического алгоритма к этой проблеме основан на кроссинговере, или перекрещивании, генных цепочек. Хотя перекрещивание может помешать эксплуатации строительного блока, разбив его, этот процесс рекомбинации тестирует строительные блоки в новых сочетаниях и новых контекстах. При скрещивании образуются новые выборки областей, обладающих качеством выше среднего, подтверждая или опровергая оценки, полученные в предыдущих тестированиях. Кроме того, когда при перекрещивании разбивается строительный блок, образуется новый блок, позволяющий генетическому алгоритму тестировать области, по которым ранее отсутствовали выборки.

Игра, известная под названием дилеммы заключенного, хорошо иллюстрирует способность генетического алгоритма находить компромисс между исследованием и эксплуатацией. Эта давно изучавшаяся игра предоставляет двум игрокам выбор между «кооперацией» и «выдаванием» соучастника. Если оба игрока солидарны (т.е. кооперируются), то получают некий выигрыш; если один игрок предает другого, он получает больший выигрыш, в то время как кооперирующийся не получает ничего; если оба выдают друг друга, они получают минимальный выигрыш (см. таб-



лицу внизу справа). Например, если игрок А молчит, а игрок В выдает его, то игрок А не получает очков, в то время как игрок В получает выигрыш в 5 очков.

Дилемму заключенного внимательно изучали ученые политологи и социологи, поскольку она дает простой и ясный пример, демонстрирующий трудности, связанные с кооперацией. Теория игр предсказывает, что каждый игрок должен стремиться минимизировать ущерб, который может ему нанести другой игрок: короче говоря, оба игрока должны нарушить договоренность. Однако когда два человека играют многократно, они, как правило, научаются сотрудничать, чтобы повысить свой общий выигрыш. Одну из наиболее эффективных известных стратегий в дилемме заключенного можно назвать «око за око»; она начинается с сотрудничества, но затем переходит к копированию поведения партнера в последней игре. Другими словами, следует «наказывать» предательство таким же предательством на следующем этапе, и наоборот, «поощрять» кооперацию партнера своей солидарностью на следующем этапе.

Роберт Аксельрод из Мичиганского университета, работая совместно со Стефани Форрест, теперь перешедшей в Университет Нью-Мексико, решили выяснить, сумеет ли генетический алгоритм открыть стратегию «око за око». Чтобы применить генетический алгоритм, необходимо сначала перевести возможные стратегии на язык цепочек символов. Один простой способ заключается в том, чтобы следующий ответ основывать на результатах последних трех партий. Каждая итерация имеет четыре возможных результата, и, таким образом, последовательность из трех партий дает 64 варианта. Для каждого из них в 64-битовой цепочке имеется один ген (или одна позиция цепочки). Например, первый ген мог бы соответствовать трем следующим друг за другом случаям взаимной кооперации, а последний — трем следующим один за одним взаимным предательствам. Значение каждого гена может быть равно 0 или 1 в зависимости от того, соответствует ли ему предпочитаемый на основе истории, отражаемой позицией гена, вариант солидарности или ее нарушения. Например, 64-битовая цепочка, состоящая из одних нулей, будет означать стратегию предательства во всех случаях. Даже для такой простой игры существует 2^{64} различных стратегий.

Аксельрод и Форрест снабдили генетический алгоритм небольшим набором случайно выбранных цепочек, представляющих стратегии. В качестве критерия эффективности цепочки было просто принято среднее количество очков, набранных при помощи соответствующей стратегии после нескольких сыгранных партий. Все эти цепочки обладали низкой эффективностью, потому что вообще большинство стратегий для игры в дилемму заключенного не дает очень хороших результатов. Довольно быстро генетический алгоритм открыл для себя стратегию «око за око», однако в дальнейшем эволюция привела к дополнительному прогрессу. Новая стратегия, открытая генетическим алгоритмом уже тогда, когда он и без того играл на достаточно высоком уровне, была основана на «блефе» — она заманивала партнера, заставляя его раз за разом проявлять солидарность и обманывая его. Однако она возвращалась к тактике «око за око», когда ход игры показывал, что партнер уже не поддается на обман.

Цель биологической эволюции, конечно, не состоит в том, чтобы породить один сверхвид, скорее она стремится создать взаимодействующие виды, хорошо приспособляющиеся друг к другу. (На самом деле, в биологическом мире не существует такого явления, как сверхиндивид.) Аналогичным образом можно воспользоваться модифицированным генетическим алгоритмом, который бы управлял эволюцией не просто индивидуальных правил или стратегий, но создавал бы «организмы» в рамках классификационной системы, состоящие из многих правил. Вместо того чтобы отбирать наиболее эффективные изолированные правила, внешняя среда своим давлением может приводить к возникновению более крупных систем, возможности которых закодированы в цепочках, являющихся их составными элементами.

Дилемма заключенного

ИГРОК	(В) СОЛИДАРНОСТЬ	(В) ПРЕДАТЕЛЬСТВО
(А) СОЛИДАРНОСТЬ	3/3	5/0
(А) ПРЕДАТЕЛЬСТВО	0/5	0/0

ДИЛЕММА ЗАКЛЮЧЕННОГО — это классическая задача в теории игр. Каждый игрок может либо проявить солидарность, либо выдать партнера, получая при этом выигрыш, зависящий от хода партнера. Хотя взаимное нарушение солидарности — самая безопасная стратегия, после нескольких партий партнеры часто приходят к кооперации.

УРБИНА РЕАКТИВНОГО ДВИГАТЕЛЯ была сконструирована с помощью генеического алгоритма. Введя в него в качестве исходных данных параметры есьма эффективной конструкции, инженеры достигли лучших результатов по равнению с тем, что позволяли сделать либо ручные методы, либо более тра-иционные программные средства.

Чтобы воссоздать эволюцию на том более высоком уровне, необхоимо внести несколько модификаций исходный генетический алгоритм. Цепочки по-прежнему представляют равила типа «условие—действие», и аждое правило, условия которого довлетворяются, вызывает определенное действие. Однако если оценивать каждое правило по числу прильных действий, генерируемых им, о эволюция приведет к развитию ин-ивидуальных «сверхправил», вместо ого чтобы находить кластеры пра-ил, полезно взаимодействующих руг с другом. Чтобы переориентировать поиск на взаимодействующие равила, процедура модифицируется аким образом, чтобы правила конурировали друг с другом за право правлять действиями системы. Каж-ое правило, условия применения ко-орого удовлетворены, конкурирует о всеми другими правилами, чьи ус-овия также удовлетворены, и силь-ейшее правило определяет в конеч-ом итоге, какое действие должна со-ершить система в данной ситуации. Если действия системы оказываются спешными, все выигравшие правила одкрепляются; в противном случае ни ослабляются.

Можно и по-другому взглянуть на тот метод, рассматривая каждое пра-ило-цепочку как гипотезу системы-лассификатора о мире. Правило ступает в конкурентную борьбу ишь в том случае, если оно «утверж-дает», что является уместным в данной ситуации. Его конкурентоспособность определяется тем, какой вклад оно внесло в решение аналогичных задач. По мере работы генетического алгоритма сильные правила спариваются и порождают правила-потомства, в которых сочетаются строительные блоки родителей. Эти потомки, замещающие правила, оказавшиеся слабейшими, можно рассматривать как вероятные, но еще не проверенные гипотезы.

Благодаря конкуренции между правилами система обладает изящным механизмом приспособления к постоянно меняющимся условиям. Когда у системы есть сильные правила, действующие в определенной ситуации, то можно сказать, что она имеет хорошо проверенные, надежные гипотезы. Правила-потомки, которые в начале своей жизни слабее родителей, могут выиграть конкурентную борьбу и повлиять на поведение системы только тогда когда нет сильных правил, условия применения которых удовлетворены, — другими словами, когда система не знает, что делать. Если действия, предпринятые по этим правилам, помогают, то правила выживают; если нет, их вскоре замещают другие. Таким образом, новое поколение не вмешивается в поведение системы, когда она находится в знакомых ситуациях, а терпеливо ожидает своего часа, когда возникнет потребность в гипотезах относительно того, что следует делать в новых, незнакомых условиях.

Фактор конкуренции, вводимый в классификационную систему рассмотренным образом, существенно влияет на ее эволюцию. На первом этапе после запуска у нее развиваются правила с простыми условиями (при этом ситуации довольно широкого спектра рассматриваются так, как будто они по существу идентичны). Система пользуется такими правилами, которые указывают, что следует сделать в отсутствие более подробной информации. О таких правилах говорят, что они действуют «по умолчанию». Однако, поскольку они позволяют проводить лишь очень грубое различие в ситуациях, они часто совершают ошибки и не набирают силу. По мере того как система приобретает опыт, процесс размножения и скрещивания приводит к появлению более сложных, более детальных правил, которые быстро становятся достаточно сильными, чтобы диктовать системе то или иное поведение в определенных ситуациях.

В конечном итоге в системе развивается иерархия: слои, состоящие из правил исключения и лежащие на нижних уровнях, вступают в действие в большинстве случаев, а правила, работающие по умолчанию и находящиеся на верхнем уровне иерархии, активируются, когда ни одно из детальных правил не располагает достаточной информацией для того, чтобы были удовлетворены условия их применения. Такая иерархия позволяет системе использовать накопленный опыт в незнакомых условиях и в то же время не дает ей увязнуть в трясине слишком детальных правил.

Те же характеристики, которые помогают классификационным системам уверенно справляться с постоянно возникающими новыми ситуациями, могут оказаться полезными и в тех случаях, когда награда за совершенное действие может быть получена лишь намного позже после того, как было совершено это действие. Первые ходы в шахматной игре, например, могут заложить основу победе или послужить причиной поражения в партии.

Чтобы приучить классификационную систему к таким отдаленным целям, программист поощряет ее всякий раз, когда она завершает решение задачи. Благодарность за успех (или упрек за поражение) может распространяться по иерархии, усиливая (или ослабляя) индивидуальные правила, даже если их действия имели лишь отдаленное влияние на исход. В течение многие поколений система развивает в себе правила, которые вступают в действие на все более ранних стадиях, чтобы подготовить по-

чву для более поздних выигрышей. Таким образом, она все в большей мере становится способной предвидеть последствия своих действий.

К НАСТОЯЩЕМУ времени генетические алгоритмы уже опробованы в разнообразных контекстах. Дейвид Е. Голдберг из Иллинойского университета, например, разработал алгоритмы, обучающиеся управлять системой газопровода, которая моделирует реальный газопровод, протянувшийся с юго-запада на северовосток США. Газопроводный комплекс состоит из многочисленных ответвлений, по которым проходит различное количество газа. Единственным средством управления являются компрессоры, увеличивающие давление в той или иной ветви газопровода, и клапаны, регулирующие поток газа, поступающего в хранилища и выходящего из них. Поскольку существует очень большая временна́я задержка между операциями с компрессорами или клапанами и результирующим изменением давления в линиях, то задача управления газопроводом не имеет аналитического решения, и диспетчеры, будь то люди или алгоритм Голдберга, должны постигать это ремесло на своем опыте.

Система Голдберга не только справляется с переменными потребностями в газе при затратах, сравнимых с реальными условиями, но и выработала иерархию правил, действующих при неполной информации и способных адекватно реагировать даже на внезапно пробитые отверстия в газопроводе (что, увы, так часло случается на практике в результате неосторожных действий бульдозеров). Лоуренс Дейвис из компании Tica Associates в Кембридже (шт. Массачусетс) воспользовался аналогичными методами для конструирования сетей коммуникации; цель его компьютерной программы заключается в том, чтобы обеспечить передачу максимального количества данных при минимальном числе линий передач и связывающих их коммутаторов.

Группа исследователей из компании General Electric и Ренсселеровского политехнического института недавно с успехом применила генетический алгоритм для конструирования турбины реактивного двигателя, который применяется в современных авиалайнерах. Эти турбины, состоящие из многочисленных рядов стационарных и вращающихся лопастей, размещенных в трубе цилиндрической формы, находятся в центре проектов новых реактивных двигателей, продолжающихся по пять и более лет и стоящих до 2 млрд. долл.

В конструкции турбины участвует по меньшей мере 100 переменных, каждая из которых может изменяться в различных диапазонах. В результате поисковое пространство содержит более 10^{387} точек. «Эффективность» турбины зависит от того, насколько хорошо она удовлетворяет набору примерно из 50 ограничений, таких, как гладкость ее внутренних и внешних контуров, а также давление, скорость и турбулентность потока в различных точках внутренней части цилиндра. Чтобы оценить одну конструкцию, требуется прогнать программу моделирования двигателя, на что уходит около 30 с на обычной рабочей станции.

В одном довольно типичном случае инженеру, работавшему в одиночку, потребовалось около восьми недель, чтобы достичь удовлетворительного варианта конструкции. Так называемые экспертные системы, использующие основанные на опыте правила логического вывода, чтобы предсказать эффект от изменения одной или двух переменных, могут помочь исследователю в его поисках полезных изменений. Конструктору, который воспользовался подобной экспертной системой, потребовалось менее одного дня, чтобы изобрести двигатель, который вдвое повышал эффективность существующей конструкции по сравнению с моделью, на создание которой ушло восемь недель ручного труда.

Однако экспертные системы подобного рода вскоре увязли там, где дальнейшие усовершенствования могут быть достигнуты только за счет одновременного изменения сразу многих переменных. Такие тупиковые ситуации возникают потому, что практически невозможно отсортировать все эффекты, вызванные различными многочисленными изменениями, не говоря уж о том, чтобы указать области пространства конструкций, в которых данные предшествующего опыта остаются справедливыми.

Чтобы выйти из тупиковой ситуации, конструктор должен найти новые строительные блоки, дающие решение задачи. Вот здесь то и вступает в действие генетический алгоритм. После того как в алгоритм были введены исходные данные конструкции, найденные экспертной системой, инженеру потребовалось лишь два дня, чтобы найти конструкцию, которая была втрое лучше по сравнению с той, которая была разработана вручную (и на 50% более эффективной по сравнению с версией, использовавшей только экспертную систему).

Этот пример указывает как на сильные стороны, так и на ограниченные возможности простых генетических алгоритмов: они хороши, когда нужно исследовать сложные поверхности, чтобы найти области максимальных возможностей. Но когда уже найденное частное решение можно усовершенствовать за счет мелких изменений в нескольких переменных, генетический алгоритм лучше дополнить другими, более стандартными методами.

ХОТЯ генетические алгоритмы имитируют механизм естественного отбора, до сих пор они действовали в значительно меньших масштабах по сравнению с биологической эволюцией. Я и мои коллеги прогоняли на компьютере классификационные системы, содержащие до 8000 правил, что соответствует лишь нижней границе численности, необходимой для жизнеспособности природных популяций. Крупные животные, не находящиеся на грани исчезновения, могут насчитывать миллионы особей, популяции насекомых измеряются триллионами, бактерий — квинтиллионами и более. Такая большая численность во много раз повышает преимущества, обусловленные неявным параллелизмом системы.

По мере того как все большее распространение получают высокопараллельные компьютеры, становится проще анализировать эволюцию программно моделируемых популяций, размеры которых приближаются к численности биологических видов. На самом деле генетический алгоритм очень удобно реализовать на таких машинах. Каждый процессор можно полностью отвести на одну цепочку, поскольку операции алгоритма концентрируются на одиночных цепочках или по крайней мере на паре цепочек во всемя скрещивания. В результате вся популяция целиком может обрабатываться параллельно.

Нам предстоит еще многое узнать о классификационных системах, но уже проведенные исследования показывают, что потенциально они способны на весьма сложное поведение. Рик Л. Риоло из Мичиганского университета уже наблюдал, как генетические алгоритмы демонстрируют «скрытое обучение» (явление, в котором животное, такое, как крыса, исследует лабиринт, не получая никакой награды, но зато позже находит пищу, спрятанную в лабиринте, значительно быстрее).

В Институте Санта-Фе Форрест, У. Байен Артур, Джон Х. Миллер, Ричард Дж. Палмер и я воспользовались классификационными системами для моделирования экономических агентов с ограниченно рациональным поведением. Эти агенты

эволюционируют, пока не становятся способными действовать сообразно меняющейся коньюнктуре на простой товарной бирже, порождая такие известные явления, как спекулятивное повышение цен и биржевые кризисы.

Миры, моделируемые и населяемые этими агентами, во многих отношениях проявляют сходство с природными экосистемами, демонстрируя аналоги таких явлений, как симбиоз, паразитизм, биологическая «гонка вооружений», подражание, образование ниши и специализация. Другие работы по генетическим алгоритмам проливают свет на условия, при которых эволюция поощряет половое или неполовое размножение. В конечном итоге искусственная адаптация может вернуть свой долг природе, позволив ученым лучше понять естественные экосистемы и другие сложные адаптирующиеся системы.

Наука и общество

Превращения в изображениях

ВЫ, ВЕРОЯТНО, увидите много такого, что не поверите своим глазам. Полицейских, которые могут превращаться в тех, к кому они прикасаются, руки, которые вытягиваются в длинные ножи и опять становятся руками, автомобили, видоизменяющиеся так, что становятся гепардами, танцующих людей из воды. Все это только некоторые из удивительных видеотрюков, осуществление которых стало возможным благодаря методу машинной графики, получившему название «морфинг». Под этим термином, образованным от слова «метаморфоза», понимается метод постепенного наплывания двух изображений на кинопленке или видеоленте таким образом, что одно из них плавно переходит в другое.

Люди, сведущие в организации развлекательных представлений, говорят, что отношение публики к морфингу постоянно менялось. «Когда мы впервые начали применять этот метод, он воспринимался просто как один из киноэффектов, — говорит Дуглас Смит, старший технический директор по машинной графике в фирме Industrial Light and Magic (ILM) в Сан-Рафаэле. — Сегодня морфинг используется очень широко».

Коммерческий дебют нового метода состоялся в 1987 г. в фильме «Ива». Когда постановщику Рону Говарду понадобилось, чтобы по ходу сюжета колдунья превращалась в разных животных, он обратился в фирму ILM, в студию спецэффектов, принадлежащую режиссеру Джорджу Лукасу. «Это был призыв принять участие в выполнении программы, невзирая на мнение соседей или руководства», — рассказывает Смит. С тех пор он использовал морфинг в нескольких рекламных роликах и фильме «Терминатор-2».

Основной метод демонстрировался в упрощенном виде несколько лет назад Томом Бригемом из Нью-Йоркского технологического института. Смит добавил к экрану сетки, упрощающие разложение изображений. В результате мозаичные детали можно перемещать так, чтобы, например, левый глаз или нос оказывался на соответствующем месте в синтезированном изображении. Такая процедура известна как перенос элементов сетки. «Это — процесс повторной дискретизации. Идея заключается в том, что мы имеем кривые, непрерывно дифференцируемые на данном отрезке в почти горизонтальном и почти вертикальном направлениях», — объясняет Смит. «Почти» означает, что допускается изгибание кривых вокруг любого из двух изображений, пока они не перекроют или не коснутся друг друга. Отсчеты элементов изображения берутся вдоль этих кривых в так называемом исходном изображении и переносятся на деформируемое изображение. Изменение частоты дискретизации вызывает большее или меньшее растяжение. Для выполнения трансформации, которая длится две секунды, может потребоваться работа в течение двух недель.

Были разработаны и более простые методы, как утверждает Тадеуш Бейер, специалист по разработке программного обеспечения компании Silicon Graphics Computer Systems в Маунтин-Вью (шт. Калифорния). Подробности метода, разработанного им совместно с Шоном Нили из фирмы Pacific Data Images (PDI) в Саннивейле (шт. Калифорния), были изложены в докладе на ежегодной конференции по машинной графике SIGGRAPH, состоявшейся в Чикаго в июле этого года. «Вместо того чтобы пытаться точно указать, как должно меняться изображение, мы предоставляем художнику самому определить это, задав только характерные черты нужного изображения, после чего компьютер «догадывается», что именно и как должно меняться», — объясняет Бейер. В его методе линии, добавленные художником-мультипликатором, окружены областью воздействия, в которой изображение автоматически деформируется на заранее определенную величину. Например, нанесение линий вокруг силуэта головы человека приводит, без какой-либо дополнительной работы, к появлению линии волос на лбу.

«Я думаю, что это в 1000 раз более эффективно: ваше внимание сосредоточивается на художественном эффекте, а не на раздражающих технических аспектах», — говорит Джеми Диксон, сотрудник PDI, отвечающий за видеоэффекты в Голливуде. «Это совершенно другой метод, который придает бóльшую динамичность смене изображения», — говорит он. Диксон применил его в видеоклипе Майкла Джексона «Черный или белый», в котором лица исполнителей плавно превращаются в лица людей разных народностей, а сам Джексон в конце ролика принимает вид пантеры.

Наибольшее применение морфинг, достигаемый любым методом, ве-

МЕТОДЫ МОРФИНГА, подобные тем, что использовались в фильме «Терминатор-2», позволяют создавать ошеломляющие видеоэффекты. Фото фирмы ILM.