Сервис высокопроизводительных вычислений для труднорешаемых задач

А.А. Горбенко, М.Л. Морнев, В.Ю. Попов

Уральский государственный университет им. А.М.Горького
gorbenko.ann@gmail.com, max.mornev@gmail.com, Vladimir.Popov@usu.ru

Как известно, многие задачи, имеющие практические приложения, относятся к классу сложности NP. В свою очередь, все задачи из NP полиномиально сводятся к проблеме выполнимости, сокращенно обозначаемой SAT. В последние годы в области разработки быстрых алгоритмов для решения SAT был достигнут существенный прогресс [1]. Преимущественно делается акцент на генетические алгоритмы и алгоритмы локального поиска. На сегодняшний день предложено несколько вариантов генетических алгоритмов [2] – [5]. Рассматривались гибридные алгоритмы, в которых подход генетических алгоритмов совмещен с локальным поиском [6]. Довольно высокой эффективности удается достичь и для алгоритмов, основанных исключительно на локальном поиске. Конечно, эти алгоритмы работают за экспоненциальное время в худшем случае. Но они могут относительно быстро получать решение для многих булевых функций, возникающих на практике. Поэтому, идея использования сведения к SAT для решения трудных задач обретает практический смысл. Активно ведутся исследования по созданию программ для решения SAT. Новые алгоритмы позволяют обычным настольным компьютерам справляться с булевой функцией в конъюнктивной нормальной форме, имеющей более 10000 конъюнктов (см., например, [7]).

На сегодняшний день существует общепризнанный сайт, на котором размещаются решатели SAT [8]. В настоящее время на сайте опубликовано 16 реализаций алгоритмов решения SAT. Они делятся на два основных класса: алгоритмы стохастического локального поиска и алгоритмы усовершенствованного полного перебора. Все решатели допускают общепринятый формат DIMACS для записи булевой функции в конъюнктивной нормальной форме и решения соответствующей проблемы [9]. Помимо собственно решателей, на сайте также представлен большой набор тестовых проблем в формате DIMACS. Этот набор включает случайно сгенерированные проблемы 3-SAT, а также конъюнктивные нормальные формы, полученные сведением комбинаторных задач к SAT или сведением задач логистики, планирования, верификации интегральных схем. Последние представляют особый интерес, поскольку все эти тесты построены для применения к реальным дизайнам интегральных схем. Это проблемы ограниченной верификации моделей [10], возникшие при проверке микросхем, производимых IBM и Galileo, а также группа проблем, связанных с анализом сетевых сбоев, полученных с помощью инструмента Nemesis, применяемого для анализа коррекности работы CMOS-микросхем [11], [12].

В классическом генетическом алгоритме в эволюционном процессе используются конкретные данные. Традиционно каждая хромосома представляет некоторое решение исходной задачи и состоит из последовательности двоичных чисел - значений переменных заданной булевой функции. После запуска алгоритма обычно формируется случайная популяция потенциальных решений и при помощи стандартных операций для эволюционных исчислений, таких как скрещивание хромосом, операции реверса и мутации, через некоторое количество циклов мы находим решение данной задачи.

Такой подход имеет ряд недостатков. В первую очередь, поиск решения в данном случае это попытка «подобрать» значения переменных, не принимая во внимание особенности функции, например, количество вхождений той или иной переменной, а также ее знак. Вторым существенным недостатком является неспособность накапливать

знания и опыта решения. В данном случае мы находим только решение для конкретной функции. Если несколько видоизменить исходную функцию и запустить алгоритм, то эволюционный процесс начнется заново, при этом он не будет иметь информации о решении подобных задач. Это является серьезным препятствием при увеличении размерности задачи. Кроме того, такой подход не позволяет использовать опыт, полученный при решении других задач, даже в случае массового применения алгоритма. Таким образом, необходим некоторый интеллектуальный алгоритм, способный накапливать знания и применять их для решения аналогичных задач.

Рассматривая особенности строения некоторой булевой функции, мы можем выработать некоторые правила, которые описывают «правильное» значение той или иной переменной в данной функции. Это позволит обходиться без поиска значения для этой переменной и, как следствие, сократить общее время работы алгоритма. Например, во многих случаях не обязательно искать значение переменной, если она встречается только один раз. При этом, учитывая, с каким знаком она входит в функцию, мы можем сразу установить правильное значение для нее. Таких переменных может быть несколько. Применяя даже такое простое правило к булевой функции мы можем очевидным образом уменьшить общее время работы алгоритма. Количество подобных правил может быть большим. На сегодняшний день разработано большое количество эвристических методов для решения SAT (см., например, [1]), основанных на использовании различных правил.

Зная как задавать значение переменной в том или иной случае, мы можем задать порядок, в котором переменные будут просматриваться. Здесь также существует множество различных вариантов. Правила выбора значений переменных и порядок, в котором происходит их просмотр, образуют некоторую стратегию поиска решения задачи. Правил обоих видов может быть большое количество, соответственно различных вариантов стратегий также может быть очень много. При этом некоторые из них могут быть совместимы, а некоторые нет. Исходя из этого, мы рассматриваем генетический алгоритм эволюции популяции стратегий, а не конкретных решений. Работа такого алгоритма, вообще говоря, не предполагает остановки. Если в некоторый момент времени нам необходимо найти выполняющий набор для той или иной булевой функции, то мы пытаемся применить к ней текущую популяцию стратегий. Если решение не удается найти сразу, то мы эволюционируем популяцию с учетом вида функции до нахождения решения. После этого та же популяция продолжает развиваться, исходя из ранее накопленного опыта до поступления новой булевой функции. Для полноценного функционирования основного алгоритма необходимы два вспомогательных интеллектуальных алгоритма. Один из них прогнозирует отсутствие выполняющего набора для булевой функции. Второй применяется для защиты от переобучения популяции.

Основное преимущество подхода, основанного на эволюции стратегий решения, заключается в возможности накопления опыта решения задач.

Наряду с правилами, разрабатываемыми исходя из соображений "здравого смысла" в рамках различных эвристических методов, можно использовать универсальные методы построения правил и даже целых стратегий. Например, для создания правила можно рассмотреть некоторую модель универсального вычислительного устройства, которая получает на вход булеву функцию и номер переменной, а в качестве результата выдает "правильное" значение соответствующей переменной или порядковый номер при ее перечислении. Соответственно, модель, формирующая стратегию, получает на вход булеву функцию, а в качестве ответа перечисляет порядок переменных и правила выбора их значений. Учитывая результат о полноте по Тьюрингу рекуррентных нейронных сетей [13], [14], мы в качестве универсальной модели правил и стратегий используем именно рекуррентные нейронные сети с рациональными коэффициентами.

Преимущество подхода к порождению правил и стратегий, основанного на использовании универсальной вычислительной модели, очевидным образом заключается в полноте выбора стратегий и, соответственно, в возможности нахождения эвристик, которые трудно спрогнозировать, исходя из соображений "здравого смысла". Однако слишком масштабное пространство решений не способствует быстрой сходимости алгоритма, эволюционирующего популяцию создаваемых таким образом правил и стратегий. Поэтому естественно рассматривать в качестве исходной популяции набор стратегий, полученный исходя из анализа и исследования задачи. А в процессе эволюции в исходную популяцию внедрять правила и стратегии, синтезированные нейронными сетями. При этом рукотворные стратегии выступают в качестве инструмента обучения и оценки работы рекуррентных нейронных сетей, обеспечивая эффективность их обучения.

Одновременное использование и правил, и стратегий, синтезированных нейронными сетями, обусловлено тем, что получить эффективное правило существенно проще, чем целую стратегию, что дает существенное преимущество на начальной стадии эволюции. Однако с развитием эволюционного процесса преимущество использования отдельных правил постепенно отходит на второй план из-за отсутствия взаимной обусловленности в их использовании. Соответственно необходим интеллектуальный алгоритм подавления присутствия отдельных правил в общей популяции.

В данном случае мы можем рассмотреть генетический алгоритм с двухкритериальной функцией приспособленности. С одной стороны мы выбираем наиболее приспособленные особи в популяции. Здесь присутствуют правила, синтезированные нейронными сетями. Они обеспечивают поиск новых правил и возможность ускорить процесс нахождения выполняющего набора. С другой стороны, нам необходимо наблюдение за количество тех или иных правил на данном этапе эволюции. Большое содержание правил полученных нейронными сетями может существенно замедлить эволюционный процесс. При этом присутствие в текущей популяции правил, основанных на исследовании функции, способствует дальнейшему развитию эволюционного процесса.

Выбор как детерминированных, так и стохастических решателей, включая решатели на основе генетических алгоритмов, весьма широк. Эти решатели демонстрируют существенно различную производительность в зависимости от исходных данных. Поэтому представляется весьма актуальным вопрос о выборе подходящего решателя для данной конкретной задачи и возможности оперативного реагирования на изменение характеристик входного потока данных. При достаточно большом потоке задач на входе сервиса, поддерживающего решатели, запуск всех возможных решателей для одной задачи естественно будет приводить к неэффективному использованию имеющихся вычислительных ресурсов. Кроме того, стохастические решатели, как правило, не дают однозначного ответа, если конъюнктивная нормальная форма не имеет решения, или прекращают поиск даже в том случае, когда решение существует. Анализ свойств решателя в плане применимости его для обработки некоторого класса конъюнктивных нормальных форм является примером задачи распознавания образца. Такого рода задачи давно исследуются специалистами в области искусственного интеллекта. Поэтому, естественно использовать интеллектуальный алгоритм для решения этой задачи. Проблема составления набора решателей может быть сформулирована как проблема поиска для генетического алгоритма, где в качестве хромосомы будет выступать описание класса конъюнктивных нормальных форм и класса решателей, предположительно оптимальных для их обработки. В то же время, свойства алгоритмов локального поиска, алгоритмов полного перебора и генетических алгоритмов решения SAT различаются весьма существенно. Поэтому имеет смысл применение отдельных генетических алгоритмов выбора для перечисленных трех классов. Соответственно, необходим генетиче-

ский алгоритм-супервизор, комбинирующий три набора решателей на основании требований к качеству решения и собранной статистики о свойствах наборов решателей.

Тестирование стратегий решения SAT, полученных при помощи генетических алгоритмов, требует больших вычислительных ресурсов. Действительно, обработка даже одного тестового набора с сайта [8] для одного решателя может занимать до 30 часов на одном процессоре, с эффективной возможностью масштабирования до состояния, когда одна конъюнктивная нормальная форма приходится на один процессор. В то же время число стратегий, сгенерированных генетическим алгоритмом в процессе работы может быть весьма значительным. Поэтому для завершения работы генетических алгоритмов за разумное время, требуется массивный параллельный вычислитель.

Для сервиса SAT, кроме собственно функции решения задач, представляется весьма полезным кеширование задач и результатов их решения. Конечно, при числе конъюнктивных нормальных форм порядка нескольких тысяч, появление точно такой же или близкой конъюнктивной нормальной формы на входе весьма маловероятно. Тем не менее, сохранение конъюнктивных нормальных форм с решениями в постоянную память позволит, в перспективе, получить большую тестовую базу, пригодную для сравнения и усовершенствования алгоритмов решения SAT, а также для обучения интеллектуальных алгоритмов выбора решателей. Таким образом, применение централизованной системы хранения является весьма полезным для реализации сервиса решения SAT. В то же время, поскольку рассматриваемые нами решатели не требуют вспомогательного сетевого обмена, а средний размер конъюнктивных нормальных форм из реального тестового набора [8] составляет примерно 8.5 Кб, ширина канала общения между узлами параллельного вычислителя не является критичной все время работы решателя. Следовтельно, в качестве вычислительной платформы могут использоваться не только классические многопроцессорные вычислители, но и слабо связанные GRID-системы, а также системы облачных вычислений.

При выборе вычислительной платформы, следует также обратить внимание на получившие в последнее время широкое распространение вычислители на базе FPGA. Использование FPGA для реализации решателя NP-полной задачи позволяет достичь многократного ускорения (см., например, [13] для гамильтонова цикла). Даже персональный компьютер, процессор которого работает на существенно большей частоте, чем FPGA, проигрывает последней по скорости в несколько раз. Следует ожидать, что применение кластеров FPGA для решения трудных задач позволит достичь еще более впечатляющих результатов.

Литература

1. J. Gu, P. Purdom, J. Franco, B. Wah. Algorithms for the Satisfiability (SAT) Problem: A Survey. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1996. P.19-152.
2. J. Fleurent. Genetic algorithms and hybrids for graph coloring. Annals of Operations Research. 1996. V.63. P.437-461.
3. J. Hao, R. Dorne. A new population-based method for satisfiability problems. Proc. of 11th European Conf. on Artificial Intelligence. Amsterdam: John Wiley & Sons, 1994. P.135-139.
4. K. Jong, W. Spears. Using genetic algorithms to solve np-complete problems. Proc. of the Int. Conference on Genetic Algorithms. 1989. P.124-132.
5. R. Voorn, M. Dastani, E. Marchiori. Finding simplest pattern structures using genetic programming. Proc. of the Genetic and Evolutionary Computation Conference. 2001. P.3-10.
6. J. Hao, F. Lardeux, F. Saubion. A hybrid genetic algorithm for the satisfiability problem. Proc. of the 1rst International Workshop on Heuristics. 2002. http://www.info.univ-angers.fr/pub/lardeux/papers/IWH02.pdf

7. K. Iwama, S. Miyazaki. SAR-variable complexity of hard combinatorial problems. Information Processing ’94, (Hamburg, 1994), IFIP Trans. A Comput. Sci. Tech., vol. I, North-Holland, Amsterdam, 1994. P.253–258.
8. http://people.cs.ubc.ca/~hoos/SATLIB/index-ubc.html
9. http://www.cs.ubc.ca/~hoos/SATLIB/Benchmarks/SAT/satformat.ps
10. A. Biere, A. Cimatti, E. Clarke, Y. Zhu. Symbolic Model Checking without BDDs. Proc. of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS99), lncs, springer, 1999. P.193-207.
11. F. Ferguson, T. Larrabee. Test Pattern Generation for Realistic Bridging Faults in CMOS ICS. Proceedings of the International Testing Conference. 1991. P.492-499.
12. T. Larrabee. Test Pattern Generation Using Boolean Satisfiability. IEEE Transactions on Computer-Aided Design. 1992. V.11. P.6-22.
13. P. Indyk. Optimal simulation of automata by neural nets. Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science STACS'95. LNCS 900, Springer-Verlag, Berlin, 1995. P.337-348.
14. H. Siegelmann, E. Sontag. Computational power of neural networks. Journal of Computer System Science. 1995. V.50. P.132-150.
15. M. Serra, K. Kent. Using FPGAs to solve the Hamiltonian cycle problem. Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03. 2003. V.3. P.III-228-III-231.