Государственное образовательное учреждение
высшего профессионального образования
«Уральский государственный университет им. А.М. Горького»

На правах рукописи

СКВОРЦОВ Евгений Сергеевич

Об эффективных алгоритмах для задачи CSP и их программной реализации

05.13.18 – математическое моделирование,
численные методы и комплексы программ

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Екатеринбург
2008

Работа выполнена в государственном образовательном учреждении высшего профессионального образования «Уральский государственный университет им. А. М. Горького» на кафедре алгебры и дискретной математики

Научный руководитель:	доктор физико-математических наук, профессор М. В. Волков
Официальные оппоненты:	доктор физико-математических наук, профессор М. Ю. Хачай
	кандидат физико-математических наук, доцент Э. А. Гирш
Ведущая организация:	Саратовский государственный университет им. Н. Г. Чернышевского

Защита состоится 18 июня 2008 г. в _____ часов на заседании диссертационного совета Д 212.286.10 по защите докторских и кандидатских диссертаций при Уральском государственном университете им. А. М. Горького по адресу: 620083, г. Екатеринбург, пр. Ленина, 51, комн. 248.

С диссертацией можно ознакомиться в научной библиотеке Уральского государственного университета им. А. М. Горького.

Автореферат разослан « » _____ 2008 г.

Ученый секретарь диссертационного совета,
доктор физико-математических наук,
профессор В. Г. Пименов

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. С расширением области применения компьютерных технологий растет количество комбинаторных задач, используемых в приложениях. В этой ситуации оказываются востребованы эффективные универсальные алгоритмы, применимые к широким классам задач.

Данная работа относится к исследованию комбинаторной задачи CSP (от английского 'Constraint Satisfaction Problem', в немногочисленной русскоязычной литературе встречается также название Обобщенная выполнимость [1]). Цель данного направления – разработать эффективные универсальные алгоритмы для задач из класса NP. Задача CSP является лишь одной из многих известных NP-полных задач, однако в последнее десятилетие стало ясно, что она занимает особое место. Хотя по самому определению NP-полной задачи к любой такой задаче сводится любая задача из класса NP, соответствующее сведение зачастую оказывается громоздким и искусственным. Преимущество задачи CSP состоит в том, что большинство комбинаторных задач может быть представлено в виде CSP просто и естественно. Многие комбинаторные задачи могут быть естественным образом охарактеризованы как подклассы задачи CSP.

В задаче CSP даны множество переменных, множество их возможных значений, а также заданы ограничения на значения переменных. Каждое ограничение состоит из вектора переменных $\vec{s}$ и отношения ρ на множестве значений переменных. Ограничение считается выполненным, если вектор значений переменных $\vec{s}$ принадлежит ρ. Требуется указать значения переменных так, чтобы выполнялись все ограничения. Впервые общая задача CSP была введена Монтанари в 1974 г. [43] при решении одной из задач машинного зрения – распознавания формы многогранников по их двумерному изображению. В последующие годы этот формализм с успехом использовался для моделирования как дискретных, так и непрерывных прикладных задач, а затем и разработки универсальных алгоритмов для их решения. Теория задачи CSP находит применение в таких областях, как теория реляционных баз данных [37, 54], временная и пространственная логика [52], распознавание образов [43], автоматическое доказательство

теорем [14], техническое проектирование [46], анализ языков программирования [45] и естественных языков [3], биоинформатика [39] и многих других.

Для многих задач очевиден естественный способ представления в виде CSP. Например, k-раскраска графа – это в точности задача CSP на k-элементном множестве, в которой в качестве отношения ограничения используется только неравенство. Популярная игра Судоку может быть представлена в виде задачи CSP, при помощи отношения $\mathcal{S}_v(\vec{s})$ = «ровно одна из переменных из $\vec{s}$ равна v». Легко видеть, что любая система уравнений есть не что иное, как задача CSP, отношениями ограничения которой являются множества векторов, удовлетворяющие каждому отдельному уравнению. Разумеется, существуют задачи, для которых способ представления в виде CSP неочевиден. К таковым относится, например, задача о пространственной структуре молекул белка, однако и в таких случаях задача CSP зачастую оказывается весьма полезной [5, 39].

В настоящее время существуют специальные декларативные языки программирования: ECLiPSe [4], Oz [49], 2LP [41], CHIP [26], в которых для решения задачи достаточно записать ее в виде CSP. Существуют библиотеки с подобной функциональностью для C++ (например ILOG [48]), расширениями, позволяющими решать задачу CSP, снабжено большинство современных версий Prolog [44]. Язык Newton [27] позволяет решать разновидность задачи CSP в которой областью значений переменных является множество рациональных чисел. Практическая полезность таких программных продуктов напрямую зависит от эффективности алгоритмов, применяемых для решения задачи CSP.

Частным случаем задачи CSP, в котором множество значений переменных двухэлементно, а разрешенные ограничения – дизъюнкции литералов (*клозы*), является задача Выполнимость, одна из первых задач, NP-полнота которых была доказана [2, 13]. Задача Выполнимость представляет самостоятельный интерес, так как формулируется она проще, чем CSP, и в то же время сведение задачи CSP к задаче Выполнимость не составляет труда.

Сама задача Выполнимость является в настоящее время объектом активного исследования. Ей посвящена ежегодная конференция SAT (это

название является сокращением от SATISFIABILITY – английского названия задачи ВЫПОЛНИМОСТЬ), проводящаяся с 1996 г. С 2005 г. в Нидерландах выпускается специализированный журнал JSAT (Journal on Satisfiability, Boolean Modeling and Computation). Важное место в исследовании задачи ВЫПОЛНИМОСТЬ занимает разработка реально работающих программ-решателей. При конференции SAT регулярно проходит соревнование таких программ, в котором принимают участие десятки программных продуктов, созданных специалистами со всего мира. Прогресс, достигнутый в разработке алгоритмов для практических задач огромен: многие задачи, еще 10 лет назад считавшиеся практически неразрешимыми, современными программами могут быть решены за доли секунды.

Возможный путь решения задачи CSP – это сведение ее к задаче ВЫПОЛНИМОСТЬ и использование эффективных эвристических алгоритмов для последней. Однако богатый язык общей задачи CSP позволяет сохранить при моделировании структуру исходной задачи, что дает ряд преимуществ. Во-первых, как было отмечено выше, многие задачи могут быть естественным образом охарактеризованы как ограниченные задачи CSP. Во-вторых, бóльшая структурированность задачи позволяет переносить в теорию CSP алгоритмы, разработанные для других комбинаторных задач. Кроме того, язык задачи CSP оказывается проще для понимания, поэтому запись задачи в виде CSP на практике оказывается предпочтительнее кодирования в виде задачи ВЫПОЛНИМОСТЬ с точки зрения взаимодействия с заказчиком при моделировании предметной области.

Постановка основных задач. Для CSP, как и для любой NP-полной задачи, не известно алгоритмов, эффективных в худшем случае, однако существует несколько подходов, использование которых позволяет решать за приемлемое время многие классы задачи CSP, возникающие в практических приложениях. В данной диссертации затрагиваются три таких подхода.

1) На языке CSP может быть сформулированы все задачи из класса NP, а значит и полиномиальные, поэтому важным направлением в исследовании задачи CSP является идентификация полиномиальных подклассов и разработка для них эффективных алгоритмов.

Естественный способ выделения подзадач CSP состоит в задании множества отношений, разрешенных для использования в ограничениях. Для каждого такого множества Γ через CSP(Γ) обозначается множество задач CSP, в которых множество отношений, используемых в ограничениях, содержится в Γ.

В 1978 г. Шефер [51], изучая ограниченные классы задачи CSP на двухэлементном множестве, показал, что каждый такой класс либо NP-полон, либо имеет полиномиальный алгоритм, решающий его. Кроме того, он нашел критерий, разделяющий легкие и трудные задачи. В этой же работе Шефер поставил проблему поиска аналогичного критерия для общей задачи CSP: для каждого множества отношений Γ определить сложность задачи CSP(Γ) и найти для нее полиномиальный алгоритм, если таковой существует. Множество отношений Γ, для которого задача CSP(Γ) полиномиальна, называется *полиномиальным*.

Полиномиальные множества отношений могут быть заданы многими разными способами. Например, в [17] с использованием техники логического программирования и методов теории групп удалось выявить два больших класса задач CSP, решаемых за полиномиальное время. В частности, в эти классы попадают все «легкие» задачи CSP, найденные Шефером. Иной подход был предложен в [34, 35], где полиномиальные множества задавались с помощью некоторых инвариантов. Этот подход был развит в [8, 9].

Предпринимались также попытки выделить классы задач CSP, которые могут быть сконструированы из более простых задач так, чтобы решение каждой конкретной задачи сводилось к решению составляющих ее компонент. Такому разложению могут быть подвергнуты как диофантовы формулы из задачи CSP [12, 15, 19, 21, 22], так и множества отношений [10, 11].

В [10, 11] новые множества отношений строятся с помощью операции *амальгамирования*. Амальгамой множеств отношений R_A, R_B на множествах A и B соответственно называется множество отношений $\mathbb{A}(R_A, R_B)$ на множестве $A \cup B$, состоящее из всевозможных объединений $\varrho_A \cup \varrho_B$, где $\varrho_A \in R_A$, $\varrho_B \in R_B$ – отношения одинаковой арности. Нетрудно показать, что задача CSP, соответствующая амальгаме двух множеств, не менее сложна, чем ее составляющие. Кроме того, амальгама двух полино-

миальных множеств не всегда полиномиальна. Даже тогда, когда она полиномиальна, алгоритм, сводящий ее к составляющим, может быть весьма нетривиальным.

В [10] был рассмотрен случай, когда множества A и B дизъюнктны. В этом случае при некоторых необременительных ограничениях амальгама полиномиальна тогда и только тогда, когда исходные множества отношений полиномиальны, а алгоритм декомпозиции тривиален. В работе [11] в предположении равенства множеств, на которых заданы отношения, указано несколько частных типов полиномиальных амальгам, важных с прикладной точки зрения. **Первой основной задачей** является вопрос об условиях полиномиальности амальгамы двух клонов в общем случае.

2) Вероятностное распределение интересующей нас задачи может быть таким, что существует алгоритм, получающий решение (или хорошее приближение решения) за приемлемое время *с большой вероятностью*, в то время как в худшем случае задача может оказаться очень сложной или даже алгоритмически неразрешимой. Более того, оценка эффективности алгоритма с позиции анализа худшего случая зачастую не отражает ни результаты экспериментов, ни его практическую полезность. Классическим примером является алгоритм решения задач линейного программирования: симплекс-метод, являющийся экспоненциальным с точки зрения анализа худшего случая, широко используется на практике.

Вероятностный анализ способен дать более адекватную оценку эффективности алгоритма. В этот анализ могут быть вовлечены два вероятностных пространства, во-первых, сам алгоритм может быть стохастическим и выдавать различные результаты в применении к одной и той же задаче, во-вторых, в анализе может быть учтено вероятностное распределение поступающих задач. Распространенным подходом является комбинирование этих пространств и вычисление вероятности того или иного результата при поступлении на вход стохастического алгоритма случайной задачи. В диссертации вероятностный анализ применяется к простейшему варианту локального поиска решения задачи 3-Выполнимость.

Локальный Поиск (ЛП) относится к числу стандартных приемов, используемых для решения NP-полных задач, в частности задачи Выпол-

нимость. В отличие от систематического поиска, в котором происходит выбор значений переменных с откатом при появлении противоречия, в локальном поиске идет работа с векторами – наборами значений всех переменных. Начальный набор значений выбирается случайно или следуя некоторой эвристике. Затем на каждом шаге новый набор значений выбирается случайным образом из окрестности текущего в соответствии с некоторым вероятностным распределением.

Локальный Поиск – один из первых алгоритмов, использованных для решения задачи Выполнимость и ее оптимизационного варианта – задачи Максимальная Выполнимость. С конца 1980-х гг. внимание исследователей привлекали разные версии алгоритма ЛП, см., например, [23, 50]. Алгоритм UnitWalk [29], построенный на принципах локального поиска, в 2003 г. занял первое место в одной из номинаций соревнования SAT. Доказано [30], что некоторые варианты локального поиска даже в худшем случае находят решение задачи Выполнимость за время, ограниченное экспонентой от количества переменных с основанием строго меньшим, чем 2.

В простейшей версии ЛП, известной как *Итеративное Улучшение* [33], мы начинаем со случайного набора значений переменных и затем на каждом шаге изменяем значение одной из переменных, для которых это приводит к увеличению количества выполненных клозов. Если таких переменных больше нет, то работа прекращается и возвращается найденный вектор. Таким образом, работа алгоритма заканчивается, когда текущий набор значений переменных оказывается локальным максимумом количества выполненных клозов для данной формулы.

Куцупиас и Пападимитриу [38] провели вероятностный анализ работы простейшей версии ЛП для модели, в которой равновероятны все формулы, истинные на заданном наборе значений, показав, что в этом случае ЛП находит решение. **Второй основной задачей** диссертации является исследование эффективности алгоритма ЛП на задачах 3-Выполнимость, распределенных по другому закону, а именно, по закону $\Phi(n, \varrho n)$: для заданного количества переменных n и константы ϱ равновероятна любая задача, содержащая n переменных и $\lfloor \rho n \rfloor$ клозов. Это распределение при-

влекает особое внимание, поскольку известно (см., например, [42]), что в зависимости от параметра ρ оно позволяет генерировать задачи различной сложности с точки зрения как доказательства невыполнимости, так и нахождения решения.

3) Популярным и продуктивным подходом к решению комбинаторных задач вообще, и в частности задачи Выполнимость, является применение генетических алгоритмов. Генетические алгоритмы впервые рассматривались в [31] и позже стали широко известными благодаря работе [20]. В генетических алгоритмах моделируются два основных механизма эволюции – наследование и естественный отбор, или выживание наилучших, которое приводит к исчезновению из популяции неприспособленных к среде индивидов. В роли индивидов, как правило, выступают элементы области определения целевой функции, а их приспособленность к окружающей среде задается значением целевой функции.

Генетические алгоритмы находят широкое применение как в практических, так и теоретических областях компьютерных наук. Генетический подход успешно применялся при конструировании самолетов [7], маршрутизации водопроводов [53], составлении расписаний [28]. В то же время на основе генетического подхода разрабатываются эффективные эвристические алгоритмы для решения NP-трудных комбинаторных задач Выполнимость [6], Задача Коммивояжера [24] и других.

Различные варианты генетических алгоритмов применялись для решения задачи Выполнимость (см., например, [16,18,25,36,55]). В статье [40] Лардо и др. предложили гибридный алгоритм GASAT, использующий Локальный Поиск при образовании новых индивидов в ходе эволюции. Сравнение, проведенное Лардо и др. на задачах библиотеки SATLIB, показало, что такое совмещение оказывается более эффективным, чем каждый из подходов по отдельности.

Работа генетического алгоритма, решающего оптимизационную задачу, строится по следующей схеме. Прежде всего, некоторым образом (в простейшем варианте – случайным) генерируется начальная популяция. Каждый индивид в ней – это строка, представляющая из себя код элемента множества, в котором ищется решение. Затем популяция итеративно обнов-

ляется за счет того, что в нее добавляются потомки присутствующих в ней индивидов и удаляются «погибшие» особи. Вероятность гибели индивида тем больше, чем меньше индивид приспособлен к среде, что соответствует худшему значению оптимизируемой функции.

Третьей основной задачей диссертации является разработка эвристического алгоритма для задачи Выполнимость, основанного на генетическом подходе и учитывающего влияние популяции на среду обитания, и оценка эффективности этого алгоритма путем его тестирования на общепринятом наборе тестовых задач.

Цель работы, таким образом, состоит в решении следующих задач:

- выяснить, каково устройство клона функций, сохраняющих амальгаму заданных клонов отношений, а также определить, в каких случаях амальгама клонов отношений полиномиальна;
- установить, каков наиболее вероятный результат применения алгоритма Локальный Поиск к случайной задаче Выполнимость, поступающей в соответствии с распределением $\Phi(n, \rho n)$;
- разработать и протестировать эффективный генетический алгоритм для задачи Выполнимость, учитывающий воздействие популяции на среду обитания.

Методы исследования. В диссертации используются методы теории клонов, теории вероятностей, а также экспериментальные данные по эффективности алгоритмов Локальный Поиск, GASAT и VEGAS.

Научная новизна. Основные результаты работы являются новыми и состоят в следующем: дано описание устройства клона функций, сохраняющих амальгаму и установлен критерий полиномиальности амальгамы; получена система дифференциальных уравнений, описывающая процесс исполнения алгоритма Локальный Поиск; разработан генетический алгоритм для задачи Выполнимость, учитывающий влияние популяции на среду обитания, и проведены эксперименты свидетельствующие о его эффективности.

Теоретическая и практическая ценность. Результаты, полученные в областях поиска полиномиальных подзадач и вероятностного анализа алгоритмов, носят теоретический характер и могут быть использованы в теории алгоритмов.

Результаты диссертации в области разработки генетических алгоритмов представляют практический интерес и могут быть использованы для создания программ-решателей практических задач.

Апробация работы. Результаты диссертации представлены на Всероссийской конференции «Дискретный анализ и исследование операций» (Новосибирск, 2002), на Мальцевских чтениях (Новосибирск, 2002), Объединенной конференции по искусственному интеллекту (Акапулько, Мексика 2003), Объединенной конференции по логике (Сиэтл, США, 2006), международной конференции «Компьютерные науки в России» (Екатеринбург, 2007). Отметим, что отбор работ на последние три из перечисленных конференции осуществлялся в соответствии с принятой в компьютерных науках практикой, т. е. на конкурсной основе на базе отзывов трех анонимных рецензентов отбиралась в среднем одна работа из трех представленных. Результаты диссертации докладывались также на алгебраических и алгоритмических семинарах Уральского государственного университета и университета Саймона Фрэйзера (Ванкувер, Канада).

Публикации по теме диссертации. Основные результаты диссертации отражены в семи публикациях автора [57–63]. В трех совместных работах с А. А. Булатовым последнему принадлежит постановка задачи и общая методика исследований; доказательства всех основных утверждений принадлежат диссертанту. Совместная работа с Е. Амири выполнена в неразрывном соавторстве. Работа [58] опубликована в издании, входившем в перечень ВАК на момент публикации.

Структура и объем работы. Диссертационная работа состоит из 97 страниц машинописного текста, включающего введение, три главы, а также рисунки, таблицы и список литературы из 74 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обсуждается история проблем, решаемых в диссертации, даются общие определения.

В **главе 1** решается вопрос о полиномиальности амальгамы множеств отношений. Для решения поставленной задачи мы используем предложенную в [34, 35] методику, опирающуюся на достижения теории *клонов*. Напомним, что *клоном функций* на множестве A называется семейство функций, замкнутое относительно суперпозиции и содержащее все проекции, т. е. функции вида $e_n^i(x_1, \ldots, x_n) = x_i$. Далее, каждому n-местному отношению ϱ на A естественным образом соответствует n-местный предикат P_ϱ. Это позволяет ввести следующее понятие. Множество отношений Γ называется *клоном отношений*, если оно содержит отношение равенства и для любых $\varrho_1, \ldots, \varrho_k \in \Gamma$ содержит отношение ϱ определяемое предикатом

$$P_\varrho(x_1, \ldots, x_n) = \exists y_1, \ldots, y_m \Phi(x_1, \ldots, x_n, y_1, \ldots, y_m),$$

где формула в правой части диофантова и ее атомарные формулы суть $P_{\varrho_1}, \ldots, P_{\varrho_k}$. Говорят, что n-местная функция f сохраняет m-местное отношение ρ, если результат построчного применения f к любой матрице $m \times n$, столбцами которой являются элементы ρ, принадлежит ρ. Множество всех функций, сохраняющих все отношения из Γ, обозначается через $\mathsf{Pol}(\Gamma)$; множество всех отношений, сохраняемых всеми функциями из C, обозначается через $\mathsf{Inv}(C)$. Как хорошо известно [47], множества вида $\mathsf{Pol}(\Gamma)$ – это в точности клоны функций, а вида $\mathsf{Inv}(C)$ – клоны отношений, причем операции Inv и Pol задают соответствие Галуа между решетками клонов функций и отношений.

Говорят, что конечное множество отношений Γ полиномиально, если полиномиален класс задач $\mathrm{CSP}(\Gamma)$, а бесконечное множество отношений Γ полиномиально, если полиномиальны все его конечные подмножества. Клоны отношений играют важную роль в изучении задачи CSP ввиду того, что, как было установлено в [34], если множество отношений Γ полиномиально, то полиномиален и клон отношений $\langle\Gamma\rangle$, порожденный этим множеством. В то же время полиномиальность клона отношений R может быть проверена по наличию определенных функций в клоне $\mathsf{Pol}(R)$.

Основным рабочими понятием в изучении клона функций, соответствующего амальгаме, является понятие *схемы, собранной из функций клонов* $F_1, \ldots, F_n$. Для краткости мы приведем здесь определение схемы в случае $n = 2$. (В диссертации схема определяется для произвольного n.) Пусть клоны функций F_A, F_B определены на множествах A и B соответственно. Через Z мы будем обозначать множество $\{A, B\}$, через C – множество $A \cap B$, а через D – множество $A \cup B$. Вектор $(X_1, \ldots, X_k) \in Z^k$ мы будем называть *шаблоном* вектора $\vec{x} = (x_1, \ldots, x_k) \in D^k$, если $x_i \in X_i$. Множество векторов, являющихся шаблонами $\vec{x}$, мы будем обозначать через $\Phi(\vec{x})$.

Если Ψ – шаблон $\vec{x}$, и X – элемент Z, то через $\vec{x}_{\Psi \searrow X}$ мы будем обозначать вектор, полученный из $\vec{x}$ опусканием координат, на местах которых в Ψ стоит не X. Через $\sharp_X \Psi$ мы обозначаем количество вхождений X в Ψ.

Функция $f : C^n \to C$ арности n называется *схемой, собранной из функций клонов* F_A, F_B, если существуют функция $\mathbf{f} : Z^n \to Z$ и семейство функций

$$\mathbf{D} = \{f_\Psi \mid \Psi \in Z^n \wedge ar(f_\Psi) = \sharp_{\mathbf{f}(\Psi)}(\Psi)\},$$

содержащееся в $F_A \cup F_B$, таких, что для всех $\vec{x}$ и всех $\Psi \in \Phi(\vec{x})$ выполняется равенство $f(\vec{x}) = f_\Psi(\vec{x}_{\Psi \searrow \mathbf{f}(\Psi)})$. Функция $\mathbf{f}$ называется *метафункцией* функции f, а функции из множества $\mathbf{D}$ – ее *деталями*.

Другими словами, схемой называется такая функция, которая на векторах данного шаблона действует как некоторая определяемая по шаблону функция f_Ψ, принадлежащая одному из клонов F_A, F_B. Множество всех схем, собранных из функций клонов F_A, F_B, мы обозначаем через $S(F_A, F_B)$. Первым основным результатом главы 1 является

Теорема 1.3 *Имеет место равенство*

$$\mathsf{Pol}(\mathbb{A}(R_1, \ldots, R_n)) = S(\mathsf{Pol}(R_1), \ldots, \mathsf{Pol}(R_n)).$$

Обозначим клоны $\langle \mathbb{A}(R_A, R_A|C) \rangle$ и $\langle \mathbb{A}(R_B, R_B|C) \rangle$ через R_A^+ и R_B^+ соответственно. Клон R_A называется *монолитным*, ни одно из его унарных отношений не содержится в $A \backslash C$. Оказывается, если клон R_A монолитен, то задача $\mathrm{CSP}(\mathbb{A}(R_A, R_B))$ сводится к $\mathrm{CSP}(R_B^+)$. Таким образом, наибольший интерес представляет случай, когда существуют такие унарные отношения

$E_A \in R_A,\ E_B \in R_B$, что $E_A \cap C = E_B \cap C = \varnothing$. В этом случае мы указываем следующий критерий полиномиальности амальгамы, который является вторым основным результатом главы 2.

Теорема 1.6 *Если клоны R_A и R_B немонолитны, то клон $\langle \mathbb{A}(R_A, R_B)\rangle$ полиномиален тогда и только тогда, когда полиномиальны клоны R_A^+, R_B^+ и $\langle \mathbf{R}_A \cup \mathbf{R}_B\rangle$.*

В **главе 2** исследуются алгоритм Локальный Поиск (ЛП) и его ослабленная версия Локальный Поиск за Один Просмотр (ЛПОП). На вход им подается булева формула Φ, сгенерированная в соответствии с распределением $\Phi(n, \varrho n)$: для заданного количества переменных n и константы ϱ равновероятна любая 3-КНФ, содержащая n переменных и $\lfloor \rho n \rfloor$ клозов. Псевдокод алгоритмов представлен на рис. 1. В нем через $W(\Phi, \vec{x})$ обозначено множество переменных, изменение значения которых в $\vec{x}$ приводит к увеличению количества выполненных клозов в $\Phi(\vec{x})$.

Алгоритм Локальный Поиск (ЛП)

Выборать набор значений $\vec{x}$ равномерно случайно
Пока $W(\Phi, \vec{x}) \neq \varnothing$
  Выбрать $x_t \in W(\Phi, \vec{x})$ равномерно случайно
  Изменить значение x_t на противоположное

Алгоритм Локальный Поиск за Один Просмотр (ЛПОП)

Выборать набор значений $\vec{x}$ равномерно случайно
Для каждой переменной x_t от x_1 до x_n
  Если $x_t \in W(\Phi, \vec{x})$ **то**
    изменить значение x_t на противоположное

Рис. 1: Псевдокод алгоритмов ЛП и ЛПОП

Мы используем принципиально различные модели для анализа ЛП и ЛПОП, однако и в том и в другом случае ключевым моментом нашего анализа является применение теоремы Вормальда [56], позволяющей доказать близость заданных параметров случайного процесса к решению некоторой системы дифференциальных уравнений.

Мы пользуемся следующими определениями из теории вероятности. Событие $A(n)$ происходит *почти наверняка при n, стремящемся к бесконечности*, если $\mathsf{P}(A(n)) \underset{n\to\infty}{\longrightarrow} 0$. Пусть $X(n)$ – некоторая случайная величина, а c – некоторая константа. Говорят, что *равенство $X(n) = cn + o(n)$ выполняется почти наверняка при n, стремящемся к бесконечности*, если существуют бесконечно малые последовательности $\alpha(n), \beta(n)$ такие, что выполняется неравенство

$$\mathsf{P}\big(|X(n) - cn| > \alpha(n)\big) < \beta(n).$$

В диссертации мы сокращаем «почти наверняка при количестве переменных в задаче стремящемся к бесконечности» до «почти наверняка».

При работе алгоритма ЛПОП после выполнения шага t переменные $\{x_1, \ldots, x_t\}$ были рассмотрены и в дальнейшем изменяться не будут. Литералы, содержащие эти переменные, мы называем *обработанными*. В модели алгоритма ЛПОП рассматриваются следующие множества:

$E_\varnothing$ – множество клозов, не содержащих обработанных литералов;

E_1 – множество клозов, содержащих выполненный обработанный литерал;

E_0 – множество клозов, содержащих три невыполненных обработанных литерала;

E_{++} (E_{+-}, E_{--}) – множества клозов, которые содержат ровно один обработанный невыполненный литерал и два (ровно один, ноль, соответственно) других литерала выполнены;

E_+ (E_-) – множество клозов, содержащих два невыполненных обработанных литерала, в которых необработанный литерал выполнен (невыполнен), соответственно.

Через $e_\tau, \tau \in \{\varnothing, 1, 0, ++, +-, --, +, -\}$ мы обозначаем мощность множества E_τ.

Мы показываем, что параметры e_τ удовлетворяют условиям теоремы Вормальда, а следовательно, близки к решению некоторой явно выписываемой системы дифференциальных уравнений (см. систему (2.8) в диссертации). Поскольку в конце работы алгоритма все литералы обработа-

ны, параметр e_0 оказывается равным количеству невыполненных клозов. Таким образом, мы получаем первый основной результат главы 2:

Теорема 2.2 *Для любого положительного ϱ существует константа c_1 такая, что для случайной 3-КНФ, распределенной по закону $\Phi(n, \varrho n)$, количество клозов, выполненных набором значений, получаемым по окончании работы алгоритма ЛПОП, почти наверняка равно $c_1 n + o(n)$.*

Поскольку во время работы ЛП каждая переменная может изменять значение несколько раз, модель, разработанная для ЛПОП, оказывается неприменимой. В модели, описывающей работу ЛП, мы рассматриваем множества $\{E_{ab}\}$, где a, b – неотрицательные целые числа. Множество E_{ab} содержит такие переменные x_i, что если значение x_i изменится, то ровно a невыполненных клозов станут выполненными и ровно b выполненных клозов перестанут быть выполненными. Через e_{ab} мы обозначаем мощность множества E_{ab} и через $\vec{e}$ совокупность значений e_{ab} по всем неотрицательным целым a, b. Ясно, что $\vec{e}$ изменяется в ходе работы ЛП.

Наш анализ алгоритма ЛП основан на следующем допущении:

Допущение 1 *В предположении истории $\vec{e}(1), \ldots, \vec{e}(t)$ процесса работы алгоритма ЛП для произвольного клоза C текущей формулы, произвольных позиций p, r в нем, $p \neq r$, любых a_1, a_2, b_1, b_2 и любых переменных $x_i \in E_{a_1,b_1}, x_j \in E_{a_2,b_2}$, события "$x_i$ занимает позицию p клоза C" и "x_j занимает позицию r клоза C" независимы.*

Оно представляется вполне правдоподобным, но мы пока не смогли дать чисто математического обоснования его справедливости. Тем не менее мы считаем возможным принять это допущение, поскольку получаемые на его основе результаты весьма хорошо соответствуют экспериментальным данным. Принимая допущение 1, мы можем применить теорему Вормальда к набору параметров $\vec{e}$ и получить описывающую их эволюцию систему дифференциальных уравнений (система (2.16) в диссертации). Изучая решения этой системы дифференциальных уравнений, мы получаем второй основной результат главы 2.

Теорема 2.3 *Для любого положительного ϱ существует константа c_2 такая, что для случайной 3-КНФ, распределенной по закону $\Phi(n, \varrho n)$, количество клозов, выполненных набором значений, получаемым по окончании работы алгоритма ЛП, почти наверняка равно $c_2 n + o(n)$.*

Константы c_1 и c_2 были определены численным решением соответствующих систем дифференциальных уравнений, при этом был использован метод Рунге-Кутты четвертого порядка, реализованный в пакете Matlab. При численном решении системы дифференциальных уравнений для ЛП мы ограничиваем число переменных e_{ab} условиями $a \leq 20, b \leq 20$. Сравнение полученных значений с экспериментальными данными представлено в таблицах 1 и 2.

ϱ	c (эксперимент)	c (система (2.8))
3	2.95	2.95
4	3.91	3.91
10	9.53	9.52

Таблица 1: Зависимость качества решения, найденного алгоритмом ЛПОП, от плотности задачи. Экспериментальные данные и предсказание системы (2.8).

ϱ	c (эксперимент)	c (система (2.16))
3	2.936	2.933
4	3.884	3.878
10	9.430	9.429

Таблица 2: Зависимость качества решения, найденного алгоритмом ЛП, от плотности задачи. Экспериментальные данные и предсказание системы (2.16).

На рисунке 2 изображены графики зависимости количества неудовлетворенных клозов от времени работы алгоритма ЛП, полученные экспериментально и решением системы дифференциальных уравнений. Можно сделать вывод, что решение системы дает очень точное приближение экспериментальных данных.

В **главе 3** описывается модификация генетического алгоритма решения задачи Выполнимость, учитывающая влияние индивидов на окружаю-

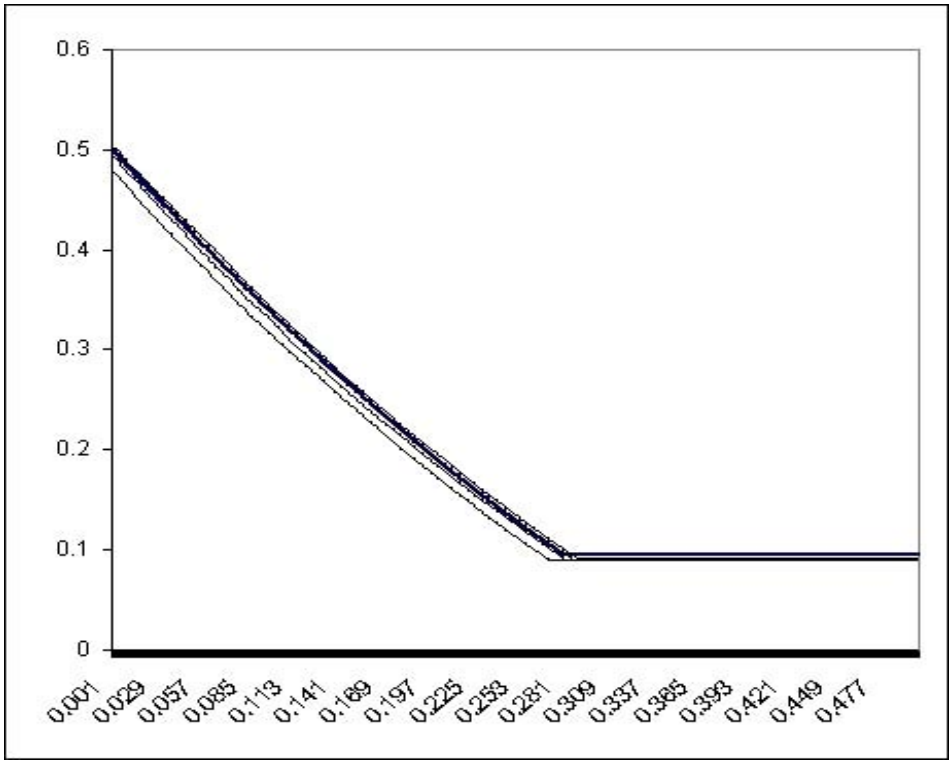

Рис. 2: Графики зависимости количества неудовлетворенных клозов от времени работы алгоритма ЛП. Тонкие линии отражают значения, полученные из запусков алгоритма на конкретных задачах, а жирная линия отвечает результатам численного решения системы дифференциальных уравнений. На оси ординат откладывается отношение числа невыполненных клозов к числу переменных, а на оси абсцисс – отношение числа проделанных шагов к числу переменных. Наклонная часть графиков отражает работу алгоритма до попадания в локальный минимум, далее каждый график продолжается горизонтальной линией, соответствующей количеству клозов, неудовлетворенных в конце работы.

щую среду. Разработанный нами алгоритм мы назвали VEGAS (сокращенно от Valuation Enhanced Genetic Algorithm for Satisfability[1]). В то время как в классическом варианте популяция обитает и развивается в неизменном окружении, в предлагаемом нами алгоритме популяция и окружающая среда активно влияют друг на друга, эволюционируя вместе. В предлагаемой модели каждый клоз C мы считаем формой *хищника* и интерпретируем удовлетворение вектором $\vec{x}$ клоза C как способность существа $\vec{x}$ защититься от хищника C. Эволюция среды в алгоритме VEGAS реализована следующим образом. Когда в результате размножения появляется существо с генотипом $\vec{x}$, хищники, от которых это существо не защищено, размножаются, что в терминах задачи выражается в увеличении веса клозов, которые нарушаются вектором $\vec{x}$. Увеличения весов невыполненных клозов производится на величину δ, которая является настроечным параметром алгоритма.

Алгоритм был реализован на языке C++ в среде Microsoft Visual Studio. Тестирование эффективности проводились на персональном компьютере Pentium IV тактовой частотой 3GHz. Для того, чтобы оценить насколько включение воздействия индивидов на окружающую среду влияет на эффективность вычислений, мы сравниваем VEGAS с одним из лучших генетических алгоритмов – алгоритмом GASAT на ряде известных тестовых задач [32]. Результаты тестирования представлены в таблице 3. В колонке *время* указано среднее время вычисления закончившегося решением задачи (в секундах), если задача была решена хотя бы один раз. Если решение не было найдено, то в скобках указывается число невыполненых ограничений для наилучшего найденного индивида. В таблице приведены результаты тестирования версии GASAT, выложенной на сайте авторов (колонка GASAT), а также результаты, полученные самими авторами алгоритма GASAT (колонка GASAT(авт)).

Проведенные тесты показывают, что VEGAS может решить больший класс задач, чем GASAT, а задачи, посильные GASAT, алгоритм VEGAS решает за меньшее время, оказываясь таким образом сильнее своего предшественника. Представляется, что предложенная нами модификация гене-

[1]Усиленный взвешиванием генетический алгоритм для задачи Выполнимость

Задачи			VEGAS		GASAT		GASAT(авт)	
Файл	Кол-во перем.	Кол-во клозов	Успех	Время	Успех	Время	Успех	Время
aim-100-1_6-yes1-1.cnf	100	160	100%	0,43	0%	(1)	2%	2
aim-200-1_6-yes1-1.cnf	200	320	85%	2,48	0%	(1)	—	—
par8-1-c.cnf	64	254	100%	0,06	10%	0,17	100%	0,028
par8-1.cnf	350	1149	100%	1,04	10%	41,87	17%	8,01
par8-2.cnf	350	1157	85%	1,45	0%	(1)	25%	11,12
par32-5-c.cnf	1339	5350	0%	(4)	0%	(13)	0%	(5)
par32-5.cnf	3176	10325	0%	(8)	0%	(129)	0%	(16)
color-15-4.cnf	900	45675	100%	14,91	100%	15,22	100%	479,248
color-22-5.cnf	2420	272129	40%	255,45	0%	(3)	0%	(5)
dp11u10.shuffled.cnf	9197	25271	0%	(2)	0%	(81)	0%	(36)
mat25.shuffled.cnf	588	1968	0%	(3)	0%	(8)	0%	(39)
mat26.shuffled.cnf	744	2464	0%	(2)	0%	(8)	0%	(56)
g125.18.cnf	2250	70163	100%	5,92	100%	30,19	98%	92,0
g250.29.cnf	7250	454622	5%	313,08	0%	(2)	0%	(2)

Таблица 3: Таблица сравнения эффективности алгоритмов VEGAS и GASAT.

тической модели заслуживает внимания и может быть применена и при решении других задач.

Автор благодарит своего научного руководителя профессора М. В. Волкова за внимание к работе и помощь в подготовке текста диссертации, а также доцента А. А. Булатова за постановку задач и постоянное внимание к работе. Автор с чувством глубокой благодарности вспоминает своего первого научного руководителя профессора Е. В. Суханова.

Список литературы

[1] Гэри М., Джонсон Д. *Вычислительные машины и труднорешаемые задачи*. М.: Мир, 1982.

[2] Левин Л. А. *Универсальные задачи перебора*// Проблемы передачи информации. 1973. Т. 9, № 3. С. 115–116.

[3] Allen J. *Natural Language Understanding*. Benjamin Cummings, 1994.

[4] Apt K., Wallace M. *Constraint Logic Programming using Eclipse*. Cambridge University Press, 2007.

[5] Backofen R., Will S. *A constraint-based approach to fast and exact structure prediction in three-dimensional protein models*// Constraints. 2006. Vol.11. № 1. P.5–30.

[6] Bertoni A., Carpentieri M., Campadelli P., Grossi G. *A genetic model: Analysis and application to MAX-SAT*// Evolutionary Computation. 2000. Vol.8. P.291–310.

[7] Bramelette M., Bouchard E. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.

[8] Bulatov A., Jeavons P., Krokhin A. *Classifying the complexity of constraints using finite algebras*// SIAM J. Comput. 2005. Vol.34, № 3. P.720–742.

[9] Bulatov A., Krokhin A., Jeavons P. *Constraint satisfaction problems and finite algebras*// Proc. 27th Int. Colloq. Automata, Languages and Programming (ICALP'00). LNCS Vol. 1853. Springer, 2000. P.272–282.

[10] Cohen D., Jeavons P., Gault R. *New tractable classes from old*// Principles and Practice of Constraint Programming CP'00. LNCS Vol. 1894. Springer, 2000. P.160–171.

[11] Cohen D., Jeavons P., Koubarakis M. *Building tractable disjunctive constraints*// J. ACM. 2000. Vol.47. P. 826–853.

[12] Cohen D.A. Jeavons P., Gyssens M. *A structural decomposition for hypergraphs*// Contemporary Mathematics. 1994. Vol.178. P.161–177.

[13] Cook S. *The complexity of theorem-proving procedures*// 3rd IEEE Symp. the Foundations Comput. Sci. 1971. P.151–158.

[14] Dechter R., Dechter A. *Structure-driven algorithms for truth maintenance*// Artificial Intelligence. 1996. Vol. 82, № 1-2. P.1–20.

[15] Dechter R., Pearl J. *Tree clustering for constraint networks*// Artificial Intelligence. 1989. Vol.38. P.353–366.

[16] Eiben A.E., Ven Der Hauw J.K., Van Hemert J.I. *Graph coloring with adaptive evolutionary algorithms*// J. Heuristics. 1998. Vol.4, № 1. P.25–46.

[17] Feder T., Vardi M. *The computational structure of monotone monadic SNP and constraint satisfaction: A study through datalog and group theory*// SIAM J. Comput. 1998. Vol. 28. P.57–104.

[18] Fleurent. J., Ferland J. *Genetic algorithms and hybrids for graph coloring*// Ann. Operations Research. 1996. Vol. 63. P.437–461.

[19] Freuder E. *Complexity of k-tree structured constraint satisfaction problems*// 8th Nat. Conf. Artificial Intelligence AAAI'90. 1990. P. 4–9.

[20] Goldberg D. *Genetic Algorithms in Search, Oprimization, and Machine Learning.* Addison-Wesley, 1989.

[21] Gottlob G., Leone L., Scarcello F. *A comparison of structural CSP decomposition methods*// Artificial Intelligence. 2000. Vol. 124, № 2. P.243–282.

[22] Grohe M., Schwentick T., Segoufin L. *When is the evaluation of conjunctive queries tractable?*// 33rd Annual ACM Symp. Theory of Computing. ACM Press, 2001. P.657–666.

[23] Gu. J. *Efficient local search for very large-scale satisfiability problem*// ACM SIGART Bull. 1992. Vol. 3, № 1. P. 8–12.

[24] Han H., Xiaowei Y., Zhifeng H. Chunguo W., Yanchun L., Xi Z. *Hybrid chromosome genetic algorithm for generalized traveling salesman problems*// Adv. Natural Comput. LNCS Vol. 3612. Springer, 2005. P.137–140.

[25] Hao J., Dorne R. *A new population-based method for satisfiability problems*// 11th European Conf. Artificial Intelligence. John Wiley & Sons, 1994. P.135–139.

[26] Hentenryck V. *The CLP language CHIP: constraint solving and applications*// Proc. IEEE Comput. Soc. Int. Conf. 1991. P. 382–387.

[27] Hentenryck V., Michel L. *Newton: Constraint programming over nonlinear real constraints*// Sci. Comput. Programming. 1997. Vol. 30. P. 83–118.

[28] Hiroaki U., Ouchi D., Takahashi K., Miyahara T. *A co-evolving timeslot/room assignment genetic algorithm technique for university timetabling*// 3rd Int. Conf. Practice and Theory of Automated Timetabling. LNCS Vol. 2079. Springer, 2000. P.48–63.

[29] Hirsch E., Kojevnikov A. *UnitWalk: A new SAT solver that uses local search guided by unit clause elimination*// Ann. Math. Artificial Intelligence. 2001. Vol. 43, № 1–4. P. 91–111.

[30] Hirsch E. A. *Worst-case study of local search for Max-k-Sat*// Discrete Appl. Math. 2003. Vol. 130. № 2. P.173–184.

[31] Holland J. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[32] Hoos H. *Satisfiability Library*// http://www.satlib.org (Электронный ресурс).

[33] Hoos H., Stutzle T. *Stochastic Local Search, foundations and applications*. Elsevier, 2005.

[34] Jeavons P. *On the algebraic structure of combinatorial problems*// Theor. Comput. Sci.. 1998. Vol. 200. P.185–204.

[35] Jeavons P., Cohen D., Gyssens M. *Closure properties of constraints*// J. ACM. 1997. Vol. 44. P.527–548.

[36] Jong K. D., Spears W. *Using genetic algorithms to solve np-complete problems*// Proc. Int. Conf. Genetic Algorithms. 1989. P.124–132.

[37] Kolaitis P., Vardi M. *Conjunctive-query containment and constraint satisfaction*// J. Comput. Syst. Sci. 2000. Vol. 61, №2. P.302–332.

[38] Koutsoupias E., Papadimitriou C. H. *On the greedy algorithm for satisfiability*// Information Processing Letters. 1992. Vol.43, № 1. P.53–55.

[39] Krippahl L., Barahona P. *Applying constraint programming to protein structure determination*// Proc. 5th Int. Conf. Constraint Programming. LNCS Vol. 1713. Springer, 1999. P. 289–302.

[40] Lardeux F., Saubion F., Hao J.-K. *A hybrid genetic algorithm for the satisfiability problem*// 1st Int. Workshop on Heuristics. 2002. P. 69–77.

[41] McAloon K., Tretkoff C. *2LP: Linear programming and logic programming*// Principles and Practice of Constraint Programming. MIT Press. P. 101–116.

[42] Mitchell D. G., Selman B., Levesque H. J. *Hard and easy distributions for SAT problems*// Proc. 10th Nat. Conf. Artificial Intelligence. AAAI Press, 1992. P. 459–465.

[43] Montanari U. *Networks of constraints: Fundamental properties and applications to picture processing*// Information Sciences. 1974. Vol. 7. P. 95–132.

[44] Narboni G. A. *From Prolog III to Prolog IV: The logic of constraint programming revisited*// Constraints. 1999. Vol. 4, № 4. P.313–335.

[45] Nadel B. *Constraint satisfaction in Prolog: Complexity and theory-based heuristics*// Information Sciences. 1995. Vol. 83, № 3–4. P.113–131.

[46] Nadel B., Lin J. *Automobile transmission design as a constraint satisfaction problem: Modeling the kinematik level*// Artificial Intelligence for Engineering Design, Anaysis and Manufacturing (AI EDAM). 1991. Vol. 5, № 3. P. 137–171.

[47] Pöschel R., Kalužnin L. *Funktionen- und Relationenalgebren*. DVW, Berlin, 1979.

[48] Puget J. F. *A C++ implementation of CLP*// Proc. 2nd Singapore Int. Conf. Intelligent Systems. 1994. P. 256-261.

[49] Roy P.V. *Logic programming in Oz with Mozart*// Proc. Int. Conf. on Logic programming. 1999. P. 38–51.

[50] Selman B., Levesque H., Mitchell D. *GSAT – A new method for solving hard satisfiability problems*// 10th Nat. Conf. Artificial Intelligence (AAAI-92). 1992. P.440–446.

[51] Schaefer T. *The complexity of satisfiability problems*// Proc. 10th ACM Symp. Theory Comp. STOC'78. 1978. P. 216–226.

[52] Schwalb E., Vila L. *Temporal constraints: a survey*// Constraints. 1998. Vol. 3, № 2-3. P. 129–149.

[53] Simpson A., Dandy G., Murphy L. *Genetic algorithms compared to other techniques for pipe optimization*// J. Water Resources Planning and Management. 1994. Vol. 120. № 4. P. 423–443.

[54] Vardi M. *Constraint satisfaction and database theory: a tutorial*// Proc. 19th ACM Symp.Priciples of Database Systems (PODS'00). 2000. P. 76–85.

[55] Voorn R., Dastani M., Marchiori E. *Finding simplest pattern structures using genetic programming*// Proc. Genetic and Evolutionary Comput. Conf. 2001. P.3–10.

[56] Wormald N. *Differential equations for random processes and random graphs*// Ann. Appl. Probability. 1995. Vol.5, № 4. P.1217–1235.

Работы автора по теме диссертации

[57] Булатов А., Скворцов Е. *Амальгамы комбинаторных задач*// «Дискретный анализ и исследование операций». Мат. конф. Новосибирск. 2002. С. 136.

[58] Скворцов Е. *О клонах на множестве и его частях*// Алгебра и логика. 2005. Т. 44, № 1. С. 97–113.

[59] Скворцов Е. С. *VEGAS — новый генетический алгоритм для задачи Выполнимость*// Изв. Урал. гос. ун-та. 2008. № 62. С. 192–207.

[60] Скворцов Е. *Решение задачи* Выполнимость *генетическим алгоритмом*// Тр. 36-й молодеж. школы-конф. «Проблемы теоретической и прикладной математики». 2005. С. 373–375.

[61] Amiri E., Skvortsov E. S. *Pushing random walk beyond golden ratio* // Computer Science – Theory and Applications. CSR 2007. LNCS Vol. 4649. Springer, 2007. P. 44–55.

[62] Bulatov A., Skvortsov E. *Amalgams of constraint satisfaction problems*// 18th Int. Joint Conf. Artificial Intelligence (IJCAI'03), Acapulco, Mexico. 2003. P. 197–202.

[63] Bulatov A., Skvortsov E. *Efficiency of local search*// Theory and Applications of Satisfiability Testing – SAT 2006. LNCS Vol. 4121. Springer, 2007. P. 297–310.