Методы распространения ограничений: основные концепции

Семенов А.Л.

Институт систем информатики им. А.П. Ершова СО
РАН Новосибирск, пр. ак. Лаврентьева, 6
semenov@iis.nsksu

Аннотация. В работе рассматриваются основные понятия и алгоритмы программирования в ограничениях — одной из областей искусственного интеллекта. Даются понятия совместности для дискретных и непрерывных областей и приводятся алгоритмы достижения совместности. Кратко описаны три системы на базе этого подхода.

1 Введение

Распространение ограничений (иногда также называемое удовлетворением ограничениям или программированием в ограничениях) является одной из наиболее интенсивно развивающихся областей искусственного интеллекта, связанной с решением разнообразных задач. Многие проблемы в различных приложениях могут быть сформулированы как задачи удовлетворения ограничений, хотя люди, не являющиеся специалистами в этой области, не всегда могут это увидеть. Представление же проблемы в виде задачи распространения ограничений позволяет применять для ее решения наряду со специальными методами прикладной области достаточно эффективные и универсальные методы решения задач распространения ограничений. В настоящее время техника распространения ограничений все чаще используется как основа для решения различных прикладных задач, таких как временные рассуждения, задачи ресурсного и календарного планирования, математическое и инженерное моделирование, задачи на графах и т.д. Поэтому естественным является большое внимание, уделяемое исследованию и методов решения задач удовлетворения ограничений. В рамках этих исследований получено достаточно много интересных и важных результатов, а также создан ряд прикладных систем, успешно применяемых для решения широкого спектра конкретных задач.

В методах распространения ограничений выделяют два класса методов — методы для задач с дискретными переменными, называемые распространением ограничений над дискретными областями [5,6,7,9,11-15,17], и методы для задач с вещественными переменными, называемые распространением ограничений над непрерывными областями [1,3,4,8,10,18,19]. Для каждого из этих классов разработан свой набор эффективных алгоритмов и построены соответствующие прикладные системы [2,16,18,20]. В настоящее время методы распространения ограничений все чаще объединяются с другими подходами, порождая новые эффективные алгоритмы. Например, распространение ограничений над конечными областями используется в комбинации с методами исследования операций и линейного программирования, а распространение ограничений над непрерывными областями использует методы интервальной математики.

Говоря неформально, задача удовлетворения ограничений — это задача, состоящая из конечного числа переменных, каждая из которых связана с некоторой областью ее определения (дискретной или непрерывной), и конечного множества ограничений, которые ограничивают множества значений переменных, принимаемых одновременно. Решение задачи удовлетворения ограничений состоит в присваивании значения каждой переменной так, чтобы одновременно удовлетворялись все ограничения. Целью может являться как нахождение одного такого набора (любого или с заданным свойством), так и нахождение всех наборов.

В данной статье будут рассмотрены основы теории распространения ограничений, ее основные алгоритмы и примеры использования данного подхода для решения прикладных проблем. Статья организована следующим образом. Во второй главе мы дадим основные сведения — базовые определения и алгоритмы для методов распространения ограничений над конечными областями. Третья глава будет посвящена понятиям и алгоритмам для задач над непрерывными областями. В четвертой главе мы рассмотрим некоторые наиболее известные системы на основе методов распространения ограничений. В заключение будет обсуждена дополнительная информация о методах удовлетворения ограничений.

2. Методы распространения ограничений над конечными областями

2.1 Формальное определение задачи удовлетворения ограничений

Определение 2.1. *Областью определения* переменной называется множество всех возможных значений, которые могут быть присвоены этой переменной. В дальнейшем мы будем говорить просто «область переменной» и обозначать эту область переменной *x* через D_x.

Если область переменной содержит только числовые значения, то эта переменная называется *числовой переменной*. В качестве области числовых переменных обычно рассматривается кольцо целых чисел ***Z***, точнее ее конечные подмножества. Если область переменной содержит два значения, то такая переменная называется *булевой переменной* и ее значениями считают 0 и 1. Если область переменной есть множество перечислимых значений (например, дни недели, цвета и т.д.), то переменная называется *символьной переменной*.

Определение 2.2. *Меткой* называется пара переменная-значение, которая определяет присваивание некоторого значения переменной. Метка, определяющая присваивание значения *v* переменной *x*, обозначается <*x,v*>. Метка <*x,v*> осмысленна только в том случае, когда $v \in D_x$.

Определение 2.3. *Составной меткой* называется одновременное присваивание значений некоторому множеству переменных. Составная метка, присваивающая значения $v_1,\ldots,v_n$ соответственно переменным $x_1,\ldots,x_n$, обозначается $(< x_1,v_1 >,\ldots,< x_n,v_n >)$. Здесь также полагается, что $v_i \in D_{x_i}$ для *i=1,...,n*.

Составная метка рассматривается как множество, поэтому порядок переменных в ней не важен, и она не может содержать двух одинаковых меток.

Определение 2.4. *k-составной меткой* называется составная метка, присваивающая *k* значений одновременно *k* переменным.

Определение 2.5. Пусть *m* и *n* — целые числа и $m \le n$. Будем говорить, что *m*-составная метка *M* есть *проекция n*-составной метки *N*, если все метки из *M* являются метками в *N*. Проекция обозначается *Pr(N,M)*.

Например, *(<a,1>,<c,3>)* является проекцией *(<a,1>,<b,2>,<c,3>)*.

Определение 2.6. *Ограничением* на некотором множестве переменных *S* называется набор составных меток для всех переменных множества S. Будем обозначать ограничение C_S.

Определение 2.7. О*граничение называется k-арным*, если его переменные образуют множество из *k* элементов. На практике особо выделяются случаи *k=1* и *k=2*. В первом случае ограничение называется *унарным*, а во втором — *бинарным*.

Определение 2.8. Пусть переменные составной метки *N* совпадают с переменными ограничения *C*. Будем говорить, что *метка N удовлетворяет ограничение C* тогда и только тогда, когда *N* является элементом *C*:

Определение 2.9. Пусть даны составная метка *N* и ограничение *C*, такие, что переменные ограничения *C* являются подмножеством переменных составной метки *N*. Будем говорить, что *метка N удовлетворяет ограничение C* тогда и только тогда, когда проекция *N* на переменные *C* является элементом *C*.

Другими словами, говоря, что *N* удовлетворяет *C*, мы подразумеваем, что если *C* есть ограничение на множестве переменных $\{x_1,\ldots,x_k\}$ или его подмножестве, то метки из *N* для соответствующих переменных образуют составную метку, которая удовлетворяет *C*.

Определение 2.10. *Задача удовлетворения ограничений* (ЗУО) — это тройка ***P = (X, D, C),*** обозначаемая **CSP(P)**, где

X — конечное множество переменных $\{x_1,\ldots,x_k\}$,

D — функция, отображающая каждую переменную из *X* на множество объектов произвольного типа:

D*: **X** –>* {конечное множество объектов некоторого типа}

Будем рассматривать D_{x_i} как множество объектов, отображенных из x_i функцией ***D***. Эти объекты называются значениями переменной x_i, а множество D_{x_i} — областью x_i.

C — конечное (возможно пустое) множество ограничений на произвольном подмножестве переменных из ***X***, то есть ***C*** -это множество наборов составных меток.

Говоря неформально, основная цель в задачах удовлетворения ограничений состоит в присваивании каждой переменной некоторого значения таким образом, чтобы одновременно удовлетворялись все ограничения. Формально это можно сформулировать следующим образом.

Определение 2.11. *Решением задачи удовлетворения ограничений* называется составная метка для всех переменных задачи, которая удовлетворяет все ограничения задачи.

Определение 2.12. Задача удовлетворения ограничений называется *разрешимой*, если она имеет хотя бы одно решение.

Все задачи удовлетворения ограничений можно разделить на следующие категории:

Задачи, в которых достаточно найти одно любое решение;

Задачи, в которых требуется найти все решения;

Задачи, в которых надо найти оптимальное решение, где оптимальность определяется в соответствии с некоторым критерием. В таких задачах часто бывает достаточно найти почти оптимальное решение.

2.2 Сведение задач и совместность

Одной из базовых техник решения задач удовлетворения ограничений является сведение. Сведением называется класс методов преобразования одних задач к другим, которые легче решать или о которых известно, что они неразрешимы. Несмотря на то, что одно только сведение задач обычно не приводит к решению, оно может облегчить последующие действия.

Определение 2.13. Две ЗУО называются *эквивалентными*, если они имеют одно и то же множество переменных и одно и то же множество решений.

Определение 2.14. Задача *P=(X,D,C) сводится* к задаче *P′=(X′,D′,C′)* если

а) *P* и *P′* эквивалентны;

б) каждая область переменных в *D′* является подмножеством области в *D*;

в) C' является более ограничивающей или также ограничивающей как *C*, (то есть все составные метки, которые удовлетворяют *C′*, будут удовлетворять *C*).

Так как мы рассматриваем ограничения как множества составных меток, то сведение задачи означает удаление элементов из ограничений тех составных меток, которые не появляются ни в одном решении. Если ограничения рассматриваются как функции, то сведение задачи означает изменение функции ограничения.

Определение 2.15. *Значение* в области является *лишним*, если оно не является частью ни одного решения:

Заметим, что удаление "лишних" значений из областей переменных не меняет множество решений задачи.

Определение 2.16. *Составная метка* в ограничении является *лишней*, если она не является проекцией ни одного решения:

Удаление "лишних" меток из ограничений также не меняет множество решений задачи.

Техника сведения преобразует ЗУО к эквивалентной более легкой задаче путем уменьшения размеров областей переменных и ограничений в исходной задаче. Сведение задач включает две возможных цели:

a. удаление лишних значений из областей определения переменных;
b. сужение ограничений так, чтобы меньшее число составных меток удовлетворяли им.

Если в результате область какой-либо переменной или какое-либо ограничение оказались сведены к пустому множеству, то задача является неразрешимой.

Сведение задачи предполагает умение определять лишние значения и лишние составные метки. Эта информация может быть выведена из ограничений. Например, если задано ограничение $x > y$, то мы можем удалить из области переменной *x* все значения, которые меньше наименьшего значения переменной *y*, а из области переменной y — все значения, которые больше наибольшего значения переменной x.

Существует ряд концепций совместности, которые помогают в определении лишних значений и составных меток. Эти понятия основываются на том, что если значение в области переменной или составная метка в ограничение нарушает ограничения, то можно показать, что они являются лишними. Далее мы рассмотрим несколько наиболее используемых видов совместностей.

Определение 2.17. *Выражением из ограничений* на множестве переменных *S* в ЗУО *P*, обозначаемое *CE(S, P)*, называется набор всех соответствующих ограничений из *P* на *S* и его подмножествах.

Определение 2.18. Будем говорить, что *составная метка CL удовлетворяет* выражение из ограничений *CE*,

если *CL* удовлетворяет все ограничения в *CE*.

Определение 2.19. *k-составная метка CL k-удовлетворяет* выражение из ограничений *CE*, если и только если *CL* удовлетворяет все ограничения в *CE*.

Определение 2.20. ЗУО *CSP(X,D,C)* называется *k-удовлетворяемой*, если и только если для всех подмножеств из *k* переменных из *X* существует множество меток для них, которое удовлетворяет все соответствующие ограничения в *CE(X, (X,D,C))*.

Определение 2.21. Задача удовлетворения ограничений *CSP(X,D,C)* с *n* переменными называется *удовлетворяемой*, если она является *n*-удовлетворяемой.

Определение 2.22. ЗУО *CSP(X,D,C)* является *1-совместной*, если и только если каждое значение в каждой области удовлетворяет унарные ограничения на соответствующую переменную. ЗУО является *k-совместной* для $k > 1$, если и только если все *(k-1)*-составные метки, которые удовлетворяют все соответствующие ограничения, могут быть расширены включением какой-нибудь дополнительной переменной до *k*-составной метки, которая *k*-удовлетворяет все соответствующие ограничения.

Стоит отметить, что 1-удовлетворяемая задача не обязана быть 1-совместной. Например, это будет не так в случае, когда некоторые значения в некоторых областях нарушают ограничения на эту переменную. 1-совместная задача также может быть 1-неудовлетворяемой. Примером такой ситуации является задача, в которой области некоторых переменных пустые, а все значения в непустых областях удовлетворяют ограничения на соответствующие переменные.

Если для всех переменных *x* в ЗУО мы удалим все значения, которые не удовлетворяют ограничению C_x, то полученная ЗУО будет эквивалентна исходной задаче. Это является результатом того, что при этом никакое решение не может быть удалено или добавлено (любое значение, которое не появляется в C_x, не может появиться в полном решении). Полученная ЗУО будет 1-совместной по определению.

Бинарные задачи составляют очень широкий класс и задачи общего вида часто можно свести к бинарным. Поэтому для бинарных задач был разработано несколько особых видов совместностей и, так как каждой бинарной ЗУО можно сопоставить граф, в этих совместностях используется терминология теории графов.

Определение 2.23. ЗУО называется *совместной в узле (node consistency)* если и только если для всех переменных все значения из их областей удовлетворяют унарным ограничениям на данную переменную. Для обозначения совместной в узле задачи *P* используется запись *NC(P)*.

Определение 2.24. Дуга *(x,y)* в графе, представляющем ЗУО, называется *совместной по дугам* (обозначается AC), если и только если для каждого значения *a* из области D_x, которое удовлетворяет ограничение на *x*, существует значение из области D_y, которое совместно с $<x,a>$.

Определение 2.25. ЗУО называется *совместной по дугам* (arc consistency), если и только если каждая дуга в графе ограничений является совместной.

Другими словами, ЗУО является совместной по дугам, если и только если для каждой переменной *x*, для каждой метки $<x, a>$, которая удовлетворяет ограничения на *x*, существует значение *b* для каждой переменной *y*, такое что составная метка $(<x,a>,<y,b>)$ удовлетворяет всем ограничениям на *x* и на *y*. Это полностью совпадает с определением 2-совместности.

Имеет место следующая теорема, свидетельствующая о полезности сведения задачи к эквивалентной задаче, совместной по дугам.

Теорема 1. Поиск в ЗУО может выполняться без использования возвратов, если граф задачи является деревом и задача является совместной в узлах и по дугам.

Определение 2.26. Путь $(x_0,x_1,...,x_m)$ в графе ограничений для некоторой ЗУО называется *совместным по путям* (PC- path consistency), если и только если для любой 2-составной метки $(<x_0,v_0>,<x_m,v_m>)$, которая удовлетворяет все ограничения на x_0 и x_m, существует метка для каждой из переменных от x_1 до x_{m-1}, так что удовлетворяется каждое бинарное ограничение на соседние переменные в пути.

Заметим, что определение совместности по путям не требует, чтобы значения $v_0,v_1,...,v_m$ удовлетворяли все ограничения в выражении из ограничений $CE(\{x_0,x_1,...,x_m\},(X,D,C))$. Например, если переменные x_i и x_j не являются соседними в пути, то 2-составная метка $(<x_i,v_i>,<x_j,v_j>)$ не обязана удовлетворять ограничение $C_{x_i x_j}$.

Определение 2.27. ЗУО называется *совместной по путям*, если и только если каждый путь в ее графе ограничений является совместным.

Другими словами, ЗУО является совместной по путям, если и только если выполняется следующее условие: если для всех пар переменных *x* и *y* составная метка *(<x,a>,<y,b>)* удовлетворяет ограничения на *x* и *y*, то для каждой переменной *z* существует метка <z,c>, такая что 3-составная метка *(<x,a>,<y,b>,<z,c>)* удовлетворяет все ограничения на *x, y, z*.

Существует теорема, сводящая установление совместности по путям к совместности путей длины 2.

Теорема 2. ЗУО является совместной по путям тогда и только тогда, когда все пути длины 2 совместны.

Заметим, что совместность в узле совпадает с 1-совместностью, совместность по дугам — с 2-совместностью, а совместность по путям эквивалентна 3-совместности в случае бинарных ЗУО.

2.3 Соотношения между совместностью и удовлетворяемостью

Возникает вопрос: как совместность соотносится с удовлетворяемостью, в частности, является ли *k*-совместность необходимым или достаточным условием для *k*-удовлетворяемости или удовлетворяемости задачи вообще? Как доказано в [17], *k*-совместность не является ни необходимым, ни достаточным условием для удовлетворяемости задач удовлетворения ограничений. В частности в примере выше было показано, что ЗУО, которые 1-совместны, не обязаны быть 1-удовлетворяемы. Вместе с тем, существуют расширения понятия совместности, в частности сильная совместность, достижение которой гарантирует удовлетворяемость ЗУО, если она является 1-удовлетворяемой, что ведет к практическим результатам по решению ЗУО.

Таким образом, *k*-совместность не является ни необходимым, ни достаточным условием для удовлетворяемости ЗУО. В частности выше было показано, что ЗУО, которые 1-совместны, не обязаны быть 1-удовлетворяемы. Вместе с тем, существуют расширения понятия совместности, в частности сильная совместность, достижение которой гарантирует удовлетворяемость ЗУО, если она является 1-удовлетворяемой.

Таким образом, рассмотренные виды совместности являются *локальными совместностями* — они обеспечивают выполнение каждого из ограничений по отдельности, но не обеспечивают решения задачи в целом. Получение решения задачи гарантирует глобальная совместность.

Определение 2.28. ЗУО *P=(X, D, C)* называется *глобально совместной* тогда и только тогда, когда каждое значение любой переменной задачи принадлежит решению задачи, т.е. для каждого значения из области каждой переменной существует составная метка, удовлетворяющая все ограничения задачи.

Тем самым, если нам дана некоторая ЗУО, то сведя ее к эквивалентной глобально совместной задаче, мы решим исходную проблему. Однако нахождение глобально совместной задачи в общем случае является NP-полной проблемой и поэтому практически не применимо. Тем не менее, сведение задачи к эквивалентной локально совместной задаче путем определения лишних значений и/или составных меток и их последующего удаления приводит к задаче, имеющей те же решения, что и исходная, но которая может быть решена легче.

Хотя одно сведение задач редко приводит к решениям, оно может разными путями помочь решать ЗУО. Такое сведение может быть использовано как препроцессирование перед применением других техник для нахождения решений. Оно также может быть использовано во время поиска для уменьшения пространства поиска после каждого означивания переменных. Иногда сведение может помочь сделать поиск безвозвратным. Суммируя сказанное, из комбинирования сведения задач с поиском можно получить следующие выгоды:

1. Уменьшение пространства поиска. Так как размер пространства поиска измеряется произведением размеров всех областей в задаче, то сведение задач может помочь уменьшить пространство поиска за счет уменьшения размеров областей отдельных переменных.
2. Избежание повторного поиска в "бесполезных" поддеревьях Лишние значения и составные метки представляют ветви и пути, которые ведут к поддеревьям, не содержащим решений. Если лишние значения и составные метки могут быть удалены в процессе сведения задачи, то это поможет избежать повторного поиска в этих пустых поддеревьях.
3. Определение неразрешимых задач. Если правильный алгоритм сведения задач выдает в качестве результата ЗУО, в которой по крайней мере одна область сведена к пустому множеству, то можно сделать вывод, что задача неразрешима. В этом случае не требуется дальнейших усилий для нахождения решений.

Сведение задач часто называется достижением совместности. Под достижением свойств определенной совместности данной ЗУО мы понимаем сведение задачи посредством удаления лишних значений из областей и лишних составных меток в ограничениях таким образом, что это свойство совместности выполняется в сведенной задаче. Например, процедура, которая "достигает совместность по дугам" в ЗУО — это процедура, которая берет ЗУО *P* и возвращает ЗУО *P'*, такую что *P* и *P'* эквивалентны и *AC(P')* истинно. Свойства совместности определены таким образом, что полученные ЗУО эквивалентны исходным задачам.

2.4. Алгоритмы достижения совместности в узлах и по дугам

В этом разделе мы рассмотрим базовые алгоритмы достижения двух видов совместностей. Алгоритм достижения совместности по путям является достаточно громоздким, поэтому мы его рассматривать не будем.

Достижение совместности в узлах. Этот алгоритм позволяет удалить из областей определения всех переменных задачи значения, которые не удовлетворяют унарным ограничениям. Достижение совместности в узлах тривиально. Все, что требуется сделать — это просмотреть каждый элемент в каждой области и проверить удовлетворяет ли это значение унарным ограничениям на эту переменную. Все значения, которые нарушают унарные ограничения, удаляются из этих областей. Алгоритм, обеспечивающий совместность в узлах и называющийся **NC-1**, может быть записан следующим образом:

Алгоритм *NC-1(X,D,C)*
/* $D_x = D_x \cap \{x : C_x\}$
Для каждой переменной $x \in X$ выполнить:
Для каждого значения $v \in D_x$ выполнить:
Если метка <x,v> не удовлетворяет ограничение C_x, то D_x =- D_x — {v};
Возврат с *(X,D,C)*;

После завершения алгоритма исходная задача будет сведена к задаче. которая является совместной в узлах. Пусть a — максимальный размер областей и n — число переменных в задаче. Так как каждое значение проверяется один раз, то временная и емкостная сложности алгоритма равны $O(a \cdot n)$.

Алгоритм достижения совместности по дугам. В отличие от алгоритмов достижения совместности в узлах, этот алгоритм более сложен и в настоящее время существует ряд алгоритмов достижения совместности по дугам [5,11-15,19]. Самый простой алгоритм называется AC-1 [11,12], а самый эффективный AC-7 [5]. Наиболее используемым из этого множества является алгоритм AC-3 [12], который также применяется и в алгоритмах установления совместности для непрерывных областей. Поэтому мы здесь рассмотрим только этот алгоритм.

Алгоритмы достижения совместности по дугам в общем случае удаляют больше лишних значений из областей, чем алгоритм **NC-1**. Самым первым алгоритмом достижения совместности по дугам был алгоритм фильтрации Вольца. Основное действие в алгоритме достижения совместности по дугам — это установление совместности для каждой дуги. Очевидно, что дугу (x_i, x_j) можно сделать совместной, удаляя из D_{x_i} все значения, для которых не существует значений в D_{x_j}, при которых удовлетворяется соответствующее ограничение. При удалении таких значений заведомо не выкидывает ни одного решения исходной задачи. Процедура установления совместности для одной дуги может быть сформулирована следующим образом:

Процедура *REVISE_DOMAIN(x, y)*
DELETED <- ложь;
Для каждого значения $a \in D_x$ выполнить:
Если в области D_y не существует значения b, такого что метка (<x,a>,<y,b>) удовлетворяет ограничение $C_{xy,,}$ то
$D_x = D_x$ — {a};
DELETED <- истина
Возврат с DELETED

Для того, чтобы сделать граф ограничений совместным по дугам, недостаточно выполнить процедуру REVISE для каждой дуги однажды. Если эта процедура уменьшила область определения для некоторой переменной x, то каждая дуга (x, y), проверенная на предыдущем шаге, должна быть подвергнута этой процедуре еще раз, т.к. некоторые значения из области D_y могут оказаться уже несовместными с оставшимися значениями в полученной области D_x. Поэтому мы должны к ней снова применить процедуру REVISE_DOMAIN. Приводимый ниже алгоритм **AC-3** решает эту проблему наиболее эффективным образом.

Алгоритм *AC-3(X,D,C)*
Выполнить алгоритм *NC-1(X,D,C)*;
Q <- $\{(x,y) \mid C_{xy} \in C\}$
Пока $Q \neq \{\}$ выполнять:
Выбираем какую-либо дугу (x,y) из Q и удаляем ее из Q;
Если *REVISE_DOMAIN(x,y)* = истина, то $Q = Q \cup \{(z,x) \mid C_{zx} \in C, x \neq z, z \neq y\}$
Возврат с *(X,D,C)*;

Алгоритм **AC-3** выполняет повторную проверку только тех дуг, на которые могло повлиять предыдущее

применение процедуры REVISE_DOMAIN. Условие $y \neq z$ при формировании множества Q следует из того, что ни для одного из удаленных процедурой REVISE_DOMAIN(x,y) значений области D_x не существует значения в D_y, обеспечивающего совместность по этой дуге. Временная сложность этого алгоритма равна $O(a^3n^2)$, то есть является полиномиальной по мощности областей переменных и числу переменных.

Более совершенные алгоритмы, начиная с **AC-4** [13], используют более сложные структуры данных, чтобы уменьшить объем проверок удовлетворяемости ограничений. Для достижения совместности по путям также существует класс алгоритмов, называемых **PC** алгоритмами (от **PC-1** до **PC-4**) [6,12,13], но они являются достаточно трудоемкими и результаты сведения, полученные с помощью этих алгоритмов, не являются достаточно удовлетворительными, чтобы компенсировать расходы на работу алгоритма. Поэтому алгоритмы достижения совместности по путям применяются весьма редко на практике и имеют в основном теоретический интерес.

3 Методы распространения ограничений над непрерывными областями

Задачи распространения ограничений над непрерывными областями обычно называются численными задачами удовлетворения ограничений и обозначают ЧЗУО или NCSP. Задачи этого класса очень часто встречаются в реальных приложениях, например, при моделировании физических явлений, при описании химических процессов, в системах моделирования и проектирования. Как правило, эти задачи представляют собой совокупность уравнений и неравенств, связывающих некоторые непрерывные переменные, хотя иногда ограничения могут задаваться в виде таблиц, а также включать целочисленные переменные. Очень часто реальные системы являются недоопределенными или переопределенными. Поэтому применение классических вычислительных методов или методов компьютерной алгебры для решения таких задач оказывается неудачным и приходится прибегать к упрощению проблемы. Методы, основанные на применении техники распространения ограничений, могут оказаться единственным подходом к решению реальных задач. В случае задач с непрерывными переменными таким аппаратом является интервальное распространение ограничений.

3.1 Особенности задач распространения ограничений над непрерывными областями

Задачи распространения ограничений над непрерывными областями определяются аналогично случаю дискретных областей (определение 2.10). Однако для непрерывных областей есть особенности в определении понятия значение переменной. Ниже этот вопрос будет обсужден подробнее. Кроме того, мы будем считать, что ограничения для числовых ЗУО задаются в виде алгебраических уравнений и неравенств (хотя в общем случае они могут быть заданы таблицами, функциями и т.д.).

Так как мы рассматриваем вещественные (непрерывные) переменные, то следует уточнить понятие "значение" в определении 2.10 для числовых ЗУО и в дальнейшем изложении. Когда x_i является целочисленной переменой, то соответствующая область D_i является подмножеством целых чисел и значением переменной является некоторый элемент этого подмножества. Если же x_i является непрерывной переменной, то соответствующая область содержит бесконечное число элементов, являющихся вещественными числами, большинство из которых не представимо в компьютере. Так как на компьютере можно представить только конечное множество вещественных чисел (это множество называется множеством чисел с плавающей запятой и в общем случае для разных видов компьютеров оно разное), то тем самым в практических приложениях бесконечная область значений непрерывной переменной заменяется конечной. Мы будем обозначать множество чисел с плавающей запятой через **FP**. Таким образом, для вещественных чисел, не входящих в множество **FP**, мы используем аппроксимацию элементами из **FP**. Обычно принимается, что некоторое вещественное число a представлено либо числом a^+, либо числом a^-, где a^- — максимальный элемент из **FP**, такой что $a^- \leq a$, а a^+ — соответственно минимальный элемент из **FP**, такой что $a^+ \geq a$.

Пусть мы имеем ограничение $C(x_1,...,x_n)$, заданное в виде отношения равенства

$$f(x_1,...,x_n) = 0.$$

Очевидно, что определенная аппроксимация значений переменных приводит к появлению ошибок вычислений значения левой части отношения и, тем самым, к вопросу, когда ограничение C справедливо. Одно из решений проблемы состоит в ведении понятия точности вычислений ε, которая задается пользователем.

Определение 3.1. Будем говорить, что *отношение C на множестве непрерывных переменных справедливо* для множества значений $(a_1,...,a_n)$, если $abs(f(a_1,...,a_n)) < \varepsilon$, где ε- некоторое заданное малое число.

Аналогичным образом можно определить справедливость ограничений для отношений больше и меньше. Но такое определение справедливости отношений не позволяет адекватно использовать понятия совместности, введенные в предыдущей главе, так как мы должны будем связывать их с точностью вычислений. Более точным и надежным с вычислительной точки зрения является представление вещественного числа a интервалом

$[a^-, a^+]$. Таким образом, мы аппроксимируем бесконечное множество значений конечным, заменяя одним элементом бесконечное множество значений, лежащих в данном интервале и не являющихся элементами из **FP**. Таким образом, мы используем описанные в предыдущей главе методы интервальной математики для определения значений переменных и нахождения значений ограничений, то есть в этом случае результатом вычисления значения левой части отношения будет некоторый интервал. Для интервального представления вещественных чисел изменим определение 3.1 следующим образом.

Определение 3.2. Будем говорить, что отношение C на множестве непрерывных переменных справедливо для множества значений $(a_1,...,a_n)$, где каждое a_i есть интервал, если $f(a_1,...,a_n) \supset 0$.

Таким образом, мы заменили бесконечную область значений переменной x_i на конечную, элементами которой являются определенные выше интервалы. Поэтому далее под значением непрерывной переменной будем понимать, как и в случае целочисленных переменных, некоторый элемент конечной области. Однако, говоря об удовлетворении ограничений, мы будем иметь в виду определение 3.2.

Определение 3.3. Решением численной ЗУО $P=(X,D,C)$ называется множество $(a_1,...,a_n)$ значений переменных $x_1,...,x_n$, такое что $a_i \in D_i$ и все ограничения из C удовлетворяются.

Использование интервалов для представления значений переменных и методов интервальной математики для установления совместностей привело к тому, что распространение ограничений над непрерывными областями называется *интервальным распространением ограничений*.

3.2 Совместность численных задач удовлетворения ограничений

В предыдущем разделе было рассмотрено несколько видов совместностей, в частности, совместность в узлах, совместность по дугам и совместность по путям. На практике наиболее эффективным оказалось понятие совместности по дугам, различные алгоритмы достижения которого широко используются в разных системах. . Для интервальных методов распространения ограничений существует несколько обобщений понятия совместности по дугам, в частности, 2B-совместность [4,7], совместность на брусе (box consistency) [3], совместность по границам (bound consistency) [18] и ряд других. Ниже будут даны определения наиболее используемых видов совместностей для численных ЗУО и дано их сравнение.

Определение 3.4. Пусть даны две ЧЗУО $P = (X,D,C)$ и $P' = (X, D', C)$, то есть задачи с одним и тем же множеством переменных и одним и тем же множеством ограничений. Будем говорить, что *задача P меньше задачи P'*, если $D \subseteq D'$ (то есть для всех i $D_i \subseteq D_i'$).

Определение 3.5. Будем говорить, что *совместность C_1 слабее совместности C_2*, если задача P_2, сведенная из задачи P к совместности C_2, меньше, чем задача P_1, сведенная из P к совместности C_1.

Определение 3.6. Пусть имеется некоторое подмножество S множества $\boldsymbol{R}$. *Интервальной оболочкой* множества S (обозначаемой □) называется минимальный интервал I, такой что $S \subseteq I$.

Определение 3.7. Пусть дана ЧЗУО $P = (X, D, C)$ и $c \in C$ есть k-местное ограничение над переменными $(x_1,...,x_k)$. Будем говорить, что *ограничение с является 2B-совместным*, если и только если

$$\forall i,\ \mathbf{x}_i = □\{x_i \mid \exists x_1 \in \mathbf{x}_1,\ldots, \exists x_{i-1} \in \mathbf{x}_{i-1}, \exists x_{i+1} \in \mathbf{x}_{i+1},\ldots, \exists x_k \in \mathbf{x}_k \text{ такие что } c(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_k) \text{ справедливы}\}$$

Определение 3.8. Пусть дана ЧЗУО $P = (X, D, C)$. Будем говорить, что *задача P является 2B-совместной*, если и только если все ее ограничения являются 2B-совместными.

Можно показать, что 2B-совместность слабее совместности по дугам. Рассмотрим численную ЗУО:

$P = (\{x_1,x_2\}, \{[1,4],[-2,2]\}, \{x_1 = x_2 * x_2\})$. Задача P является 2B-совместной, но не является совместной по дугам, так как в области переменной x_1 не существует значения, удовлетворяющего ограничение при $x_2 = 0$.

Алгоритм достижения 2B-совместности практически совпадает с приведенным выше алгоритмом ***AC-3*** достижения совместности по дугам. Он может быть представлен следующим образом:

Алгоритм *IC(X,D,C)*
  Выполнить алгоритм *NC-1(X,D,C)*;
  Q <- C;
  Пока $Q \neq \{\}$ выполнять:
    Выбираем какое-либо ограничение из *Q* и удаляем его из *Q*;
    D' <- narrow(c, D);
    Если $D' \neq D$, то $D = D'$, $Q = Q \cup \{c' \in C \mid Var(c) \cap Var(c') \neq \{\}\}$

Возврат с *(X,D',C)*;

Var(c) в алгоритме выше обозначает множество всех переменных в ограничении *c*. Самым важным шагом в алгоритме является изменение областей переменных с помощью оператора *narrow*. Одним из способов реализации этого оператора является разбиение множества ограничений на примитивные ограничения, которые представляют собой бинарные и тернарные ограничения, и вычисление интервальных значений одних аргументов этих ограничений на основе интервальных значений других аргументов. В [1] показано, что такое разбиение не приводит к потере решений или появлению новых решений исходной системы ограничений. Рассмотрим пример построения системы примитивных ограничений. Пусть мы имеем численную задачу удовлетворения ограничений $P = (\{x, y\}, D = (-\infty,+\infty)\times(-\infty,+\infty), C)$, где

C: $e^{\sin(x)} + x^y = 0$

Записав это уравнение в виде примитивных отношений, мы получим систему *C'*:

$t_1 = \sin(x) \quad x = \arcsin(t_1)$

$t_2 = e^{t_1} \quad t_1 = \ln(t_2)$

$t_3 = x^y \quad x = e^{\ln(t_3)/y} \quad y = \dfrac{\ln(t_3)}{\ln(x)} \qquad t_1 = -t_3$

Таким образом, мы построили численную ЗУО, эквивалентную исходной:

$P' = (\{x, y, t_1, t_2, t_3\}, D\times[-\infty,+\infty]\times[-\infty,+\infty]\times[-\infty,+\infty], C')$

Таким образом, все ограничения новой системы имеют вид *x = f(y,z)* или *x = f(y)*. Оператор *narrow* для каждого ограничения *c'* данной системы записывается следующим образом:

x' = x ∩ f(y)

где значениями всех переменных, входящих в ограничения, являются интервалы, и эти интервалы фактически задают области соответствующих переменных. Например, в результате выполнения первого ограничения область переменной t_1 будет сужена от интервала *[-∞, +∞]* до интервала *[-1, +1],* а в результате .выполнения шестого ограничения область переменной *x* станет *[0, +∞]*.

Определение 3.9. Пусть дана ЧЗУО *P = (X, D, C)* и *c* ∈ *C* есть *k*-местное ограничение над переменными $(x_1,\ldots,x_k)$. Будем говорить, что *ограничение с является совместным на брусе*, если для всех переменных x_i выполняются следующие соотношения:

$c(x_1,\ldots,x_{i-1},[\underline{x}_i, \underline{x}_i^{+}),x_{i+1},\ldots,x_k)$

$c(x_1,\ldots,x_{i-1},(\overline{x}_i^{-}, \overline{x}_i],x_{i+1},\ldots,x_k)$

Определение 3.10. Пусть дана ЧЗУО *P = (X, D, C)*. Будем говорить, что *задача P является совместной на брусе*, если и только если все ее ограничения являются совместными на брусе.

В работе [8] показано, что *совместность на брусе является более слабой, чем 2B-совместность*, то есть задача, которая 2B-совместна является также и совместной на брусе, но задача, которая совместна на брусе, может не быть 2B-совместной. Рассмотрим численную ЗУО:

P = ({x₁,x₂}, {[-1,1],[0,1]}, {x₁ + x₂ + x₂ = 0}). Задача *P* не является 2B-совместной по x_2 так как не существует значения x_1 такого, что ограничение удовлетворяется при $x_2 = 1$., но задача является совместной в брусе для x_2, так как для обоих граничных значений этой переменной ограничение задачи удовлетворяется.

Также можно показать, что 2B-совместность примитивной системы ограничений слабее, чем совместность в брусе исходной системы ограничений.

Достижение совместности на брусе может быть выполнено с помощью описанного выше алгоритма **IC**, в котором оператор сужения narrow реализован с помощью интервального одноперменного алгоритма Ньютона, описанного в предыдущей главе, и бисекции получаемых областей до достижения требуемой точности [18].

Аналогично определению совместностей более высокого порядка для ЗУО над непрерывными областями (например, совместности по путям), для численных ЗУО также можно ввести понятия k-совместностей для определенных выше совместностей. Наиболее практичными являются понятия 3B-совместности и совместности по границам, которые позволяют отбросить большие области лишних значений по сравнению с 2B-совместностью и совместностью на брусе. Однако достижение совместностей более высоких порядков требует существенного увеличения объема работ, но не всегда трудозатраты окупаются качеством полученных результатов.

Определение 3.11 Пусть дана ЧЗУО *P = (X, D, C)* и пусть $x \in X$. Обозначим:

$P_{[\underline{x},\underline{x}^+)}$ — задача, полученная из *P* заменой *x* на $[\underline{x}, \underline{x}^+)$

$P_{(\overline{x}^-,\overline{x}]}$ — задача, полученная из *P* заменой *x* на $(\overline{x}^-, \overline{x}]$

Область D_x называется 3B-совместной, если и только если сведение задач $P_{[\underline{x},\underline{x}^+)}$ и $P_{(\overline{x}^-,\overline{x}]}$, к 2B-совместным задачам не порождает задач с пустыми областями значений.

Определение 3.12 Численная ЗУО *P = (X, D, C)* называется 3B-совместной, если и только если все ее области являются 3B-совместными.

Доказано, что совместность в боксе слабее, чем 3B-совместность примитивной системы ограничений, то есть достижение 3B-совместности для примитивной системы ведет к совместности в брусе исходной системы.

Определение 3.13. Пусть дана ЧЗУО *P = (X, D, C)* и $c \in C$ есть *k*-местное ограничение над переменными $(x_1,\ldots,x_k)$. Будем говорить, что *ограничение c является совместным по границам*, если для всех переменных x_i сведение ограничения *c* с подставленными значениями границ переменной x_i, то есть ограничений

$$c(x_1,\ldots,x_{i-1},[\underline{x}_i,\underline{x}_i^{+}),x_{i+1},\ldots,x_k)$$
$$c(x_1,\ldots,x_{i-1},(\overline{x}_i^{-},\overline{x}_i],x_{i+1},\ldots,x_k)$$

к совместным на брусе ограничениям не порождает пустых областей переменных задачи *P*.

Определение 3.14. Пусть дана ЧЗУО *P = (X, D, C)*. Будем говорить, что *задача P является совместной по границам*, если и только если все ее ограничения являются совместными по границам.

В [8] показано, что 3B-совместность примитивной системы ограничений слабее совместности по границам исходной задачи. Таким образом, можно выписать следующие соотношения совместностей для интервальных задач распространения ограничений ($C_1 \le C_2$ означает, что совместность C_2 слабее совместности C_1):

по-границам $\le$ 3B(примитивной) $\le$на-брусе

2B $\le$на-брусе $\le$2B(примитивной)

Данные соотношения показывают, что 2B-совместность для примитивной системы ограничений является самой слабой, а самой сильной является совместность по границам. Вместе с тем, установление 2B-совместности для примитивной системы ограничений может быть выполнено легче всего и, к тому же, оно применимо для любых систем ограничений, например, содержащих разрывные функции, целочисленные переменные (при наличии операций целочисленной интервальной арифметики) и другие типы данных. Для достижения совместности на брусе требуется применение метода Ньютона, что предполагает дифференцируемость функций или необходимости наклонов вместо производных. Для достижения 3B-совместности и совместности по границам требуется трудоемкое исследование областей переменных, выполняемое с помощью алгоритмов бисекции и соответствующих алгоритмов установления совместностей более низких порядков.

Следует отметить, что рассмотренные алгоритмы достижения совместностей для задач интервального распространения ограничений основаны на алгоритме **IC** — версии алгоритма **AC-3**. Вместе с тем, в литературе существуют ссылки на расширение алгоритма **AC-5** [14,15], которое может использоваться для непрерывных областей. Этот алгоритм используется в коммерческой системе ILOG [20] и его описание пока не доступно.

4. Системы, основанные на методах распространения ограничений

4.1 ILOG Solver [20]

ILOG является одной из крупнейших в мире фирм, разрабатывающих продукты с использованием методов распространения ограничений. Их основным продуктом в этой области является ILOG Solver — библиотека Си++ классов для решения комбинаторных задач и нахождения оптимального решения. Он основан на методах распространения ограничений и может также служить средством для интеграции с другими оптимизационными технологиями, включая линейное программирование, локальный поиск, определяемые пользователем эвристики и генетические алгоритмы.

ILOG Solver поддерживает работу с целыми, вещественными, логическими и множественными переменными и позволяет использовать линейные, нелинейные и логические ограничения. В ILOG Solver реализованы

несколько так называемых глобальных ограничений, которые позволяют связывать в одном ограничении несколько переменных, и эти ограничения могут эффективно обрабатываться. При этом существенно уменьшается общее число ограничений по сравнению с использованием отдельных ограничений с той же семантикой. Глобальными ограничениями являются alldiff, distribute, table constraint, sequence.

Новые типы и ограничения на них, а также новые ограничения для существующих типов могут быть расширены разработкой новых C++ классов. Существуют метаограничения — конъюнкция и дизъюнкция существующих ограничений, и допускается определение ограничений с весами и упорядочивание ограничений (это также может рассматриваться как метаограничения). Решатель содержит возможность динамического добавления и удаления ограничений.

В качестве алгоритма распространения ограничений применяется модифицированный алгоритм совместности по дугам AC-5, который в отличие от базового алгоритма, работающего только с целыми числами, может применяться также и с вещественными, логическими и множественными переменными. По мнению авторов, этот алгоритм реализован очень тщательно, и они утверждают, что их алгоритм превосходит другие аналогичные реализации более чем в 2000 раз. Следует также отметить наличие параллельной версии ILOG Solver, что позволяет получать хороший выигрыш во времени при решении больших прикладных задач.

В ILOG Solver реализовано несколько наиболее эффективных алгоритмов поиска, включающих как стандартные стратегии (поиск в глубину, поиск сначала наилучшего), так и специализированные алгоритмы для задач удовлетворения ограничения над конечными областями (Limited Discrepancy Search, Depth Bounded Discrepancy Search). Пользователь может управлять процессом поиска, устанавливая параметры поиска, добавляя различные эвристики для сокращения областей, а также может создавать комбинации алгоритмов поиска. Пользователь может также выбирать требуемое решение (все, первое, лучшее и т.д.), изменять имеющееся решение с учетом новой информации, проверять имеющееся решение на оптимальность и т.д.

4.2 Numerica [18]

Numerica представляет собой одну из наиболее известных систем, разработанных на основе интервального распространения ограничений. Ее автор — Паскаль Ван Хентенрик, является автором ряда базовых концепций в этой области и им были разработаны основные алгоритмы достижения совместности на брусе и совместности по границам. Эти, а также другие специальные методы, и лежат в основе системы Numerica.

Numerica представляет собой систему с графическим пользовательским интерфейсом, доступную IBM PC совместимых на машинах. Система предоставляет входной язык, достаточно удобный для записи систем уравнений и неравенств. Numerica позволяет решать только системы с вещественными переменными, которые должны быть квадратными, то есть число переменных в них должно совпадать с числом уравнений. Все остальные уравнений, а также неравенства рассматриваются как дополнительные ограничения. При этом все уравнения решаемой системы должны быть дифференцируемы. Все это является следствием применения алгоритма достижения совместности в брусе, в котором используется интервальный алгоритм Ньютона.

Достоинствами Numerica является использование нескольких способов вычисления значений функций, включая разложение в ряд Тейлора, что позволяет получать достаточно точные границы при оценивании функций с интервальными параметрами. Также в Numerica используется специальная модификация алгоритма ветвей и границ, называемая методом ветвей и отсечений, которая позволяет весьма эффективно решать задачи глобальной оптимизации. Еще одним привлекательным свойством системы является доказательство существования корней, которое основано на использовании интервального метода Ньютона.

В [18] приводятся результаты численных экспериментов, показывающих, что Numerica может решать достаточно сложные задачи с хорошей эффективностью. В частности, она превосходит большинство численных пакетов для решения систем нелинейных уравнений, основанных как на классических, так и на интервальных методах. Наличие большого количества стандартных тестовых примеров, поставляемых с системой, наряду с приводимыми решениями и временами получения решений делает Numerica хорошим инструментом для тестирования эффективности пакетов для решения систем нелинейных уравнений.

4.3 UniCalc [2]

Решатель UniCalc представляет собой интегрированную среду с графическим пользовательским интерфейсом, доступную на большинстве платформ. Эта среда включает встроенный редактор, различные настройки, удобный входной язык для записи решаемых задач и средства графического отображения одномерных интервальных функций. UniCalc содержит несколько различных методов, в том числе некоторые виды символьных преобразований (упрощение, символьное дифференцирование, приведение подобных). Базовым вычислительным методом является один из вариантов методов распространения ограничений, называемый методом недоопределенных вычислений [1] и основанный на достижении 2B-совместности для примитивной системы ограничений, которая получена из исходной задачи. Ядро решателя, объединяющее все методы, называется вычислителем и

может быть использовано отдельно. В настоящее время на базе этого вычислителя создано несколько программных продуктов, и он встроен в ряд промышленных систем.

UniCalc предназначен для решения задач, представленных системами алгебраических уравнений, неравенств и логических выражений. Решаемая система может быть переопределенной или недоопределенной, а параметры уравнений и неравенств могут быть заданы в виде интервалов допустимых значений. UniCalc позволяет проводить вычисления как с целыми, так и с вещественными переменными, причем они могут входить в систему одновременно. В результате вычислений находится покоординатная внешняя оценка всех действительных решений системы (в общем случае неоптимальная). Если заданная система не имеет действительных решений, то выдается сообщение об ее несовместности. В решатель также встроен алгоритм локализации корней, позволяющий выделять отдельные решения из множества всех решений.

Для локализации корней используется алгоритм, основанный на рекурсивном делении интервалов переменных и применении метода распространения ограничений. В этом алгоритме выбирается переменная, по которой начинается поиск корня, и задается направление — слева или справа. Выбором последовательности переменных, по которым осуществляется бисекция, можно управлять. В этом алгоритме сужение интервалов происходит как за счет их деления, так и за счет применения метода распространения ограничений. Корень считается найденным, если максимальная ширина интервалов по всем переменным не превышает заданную точность и система остается совместной. Базовый алгоритм локализации корней является основой для нескольких дополнительных алгоритмов, в которых используются различные эвристики для динамического выбора переменных разбиения и их комбинаций. В частности, это позволяет сводить исходную систему к 2B- и 3B-совместным задачам.

Заключение

В работе были рассмотрены основные понятия и алгоритмы программирования в ограничениях — одной из наиболее интенсивно развивающихся областей искусственного интеллекта. Этот подход применяется как для задач с дискретными (целочисленными, перечислимыми) переменными, так и для задач с непрерывными (вещественными) переменными. Для обоих случаев в работе были даны определения основополагающего понятия методов распространения ограничений — совместности и представлены некоторые алгоритмы достижения совместности, а для непрерывных областей также были приведены сравнения четырех видов специальных совместностей. Однако понятие совместности не ограничивается вариантами, рассмотренными в работе. В частности, для дискретных областей тоже существует целый ряд различных видов совместности — направленная, сильная, адаптивная и другие, которые также используются для решения задач. Для достижения различных совместностей разработаны специальные алгоритмы (в том числе параллельные), реализованные в прикладных системах.

В настоящее время основные усилия исследователей, работающих в этой области, направлены на разработку эффективных алгоритмов поиска решений. Эти алгоритмы представляют собой комбинацию различных подходов — методов распространения ограничений, алгоритмов поиска и специальных алгоритмов из предметных областей. Одно из направлений работ в этих исследованиях — создание кооперативных решателей, которые объединяют несколько различных подходов и используют их совместно при решении одной задачи, включая параллельную работу различных методов с обменом полученными результатами. Данное направление кажется очень перспективным для решения действительно сложных задач. Примером такого кооперативного решателя является решатель SibCalc [16].

Список литературы

1. Кашеварова Т.П., Семенов А.Л. Некоторые вопросы сходимости метода недоопределенных вычислений. В: *Проблемы представления и обработки не полностью определенных знаний*. — Москва–Новосибирск: РосНИИ ИИ, 1996, с. 31-37.
2. Babichev A.B., Kadyrova O.B., Kashevarova T.P., Leshchenko A.S., Semenov A.L. UniCalc, A Novel Approach to Solving Systems of Algebraic Equations. *Interval Computations* **2** (1993), pp. 29-47.
3. Benhamou F., McAllester D., Van Hentenryck P. CLP(Intervals) Revisited. In: *Proc. of ILPS'94,* Ithaca, NY, USA, 1994, pp. 124–138.
4. Benhamou F., Older W. Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *Journal of Logic Programming* **32** (1997), pp.1-24.
5. Bessiere C. Using constraint metaknowledge to reduce arc consistency computation. *Artificial Intelligence* **65**, (1994), pp. 179-190.
6. Bessiere C., Freuder E.C., and Régin J.-R. Comments on Mohr and Henderson's path consistency algorithm. *Artificial Intelligence.* **107** (1999), pp. 125-148.
7. Cleary. J. Logical Arithmetic. *Future Comput. Syst*. **2** (2) (1987), pp. 125–149.
8. Collavizza H., Delobel F., Rueher M. A note on partial consistencies over continuous domains. In: *Proc. CP98*, LNCS 1520, Springer Verlag, 1998, pp. 147–161.

9. Freuder E. Synthesizing Constraint Expression. *CACM* **21** (11) (1978), pp. 958–966.
10. Lhomme O. Consistency Tecniques for Numeric CSP's. In: *Proc. of the 13th IJCAI, R. Bajcsy (Ed)*. — IEEE Computer Society Press, 1993, pp. 232-238.
11. Macworth A.K. Consistency in Networks of Relations. *Artificial Intelligence* **8** (1) (1977), pp.99-118.
12. Mackworth A.K. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artificial Intelligence* **8** (1997), pp. 99-118.
13. Mackworth A.K. and Freuder E.C.. Arc and path consistency revised. *Artificial Intelligence* **25** (1985), pp. 65-74.
14. Mohr R., Henderson T.C. Arc consistency for factorable relations. *Artificial Intelligence* **28** (1986), pp. 225-233.
15. Perlin M. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence* **53** (1992), pp. 329-342
16. A. Semenov, D. Petunin, A. Kleymenov. GMACS — the general-purpose module architecture for building cooperative solvers. In: *Proceedings of the 2000 ERCIM/Compulog Net Workshop on Constraints*. Padova, Italy, June, 2000.
17. Tsang E. *Foundation of Constraint Satisfaction,* London: Academic Press Ltd., 1993.
18. Van Hentenryck P., Michel L. and Deville Y. *Numerica: a modeling language for global optimization,* Cambridge: MIT Press, 1997.
19. Van Hentenryck P., Deville Y, and Teng C.-M. Arc-consistency and arc-consistency again. *Artificial Intelligence* **57** (1992), pp. 291-321.
20. http://www.ilog.com/products/solver/