多元约束混合搜索算法研究

孙吉贵 [1, 2] 张居阳 [1, 2] 陈尚伟 [1, 2]

摘 要 解空间搜索是约束求解的关键环节. 目前较为常用的搜索算法一般是基于二元约束或单一搜索策略设计的. 本文设计了六个基于多元约束的混合搜索算法 (BM_GASBJ, BM_GBJ, BM_CBJ, FC_GASBJ, FC_GBJ, FC_CBJ), 它们分别混合同一类搜索策略中不同算法或不同类搜索策略; 分析并给出了不同混合算法的性能差异. 系统测试结果表明混合搜索算法明显提高了解搜索效率和约束求解系统的性能.

关键词 搜索, 约束满足问题, 约束求解, 多元约束
中图分类号 TP301.6

Non-binary Constraint Hybrid Search Algorithm

SUN Ji-Gui[1, 2] ZHANG Ju-Yang[1, 2] CHEN Shang-Wei[1, 2]

Abstract Searching is the key step of constraint solving. Most of search algorithms are based on binary constraint or designed upon a single search strategy. Six hybrid search algorithms (BM_GASBJ, BM_GBJ, BM_CBJ, FC_GASBJ, FC_GBJ, FC_CBJ) based on non-binary constraint are illustrated. The hybrid algorithms are blended by various search algorithms or strategies. The differences of these hybrid algorithms are pointed out and analyzed . The constraint solving system we designed implements all of the hybrid algorithms. The results indicate that the new algorithms have a higher searching efficiency and improve the performance of the constraint solving system.

Key words Search, constraint satisfaction problem, constraint solving, non-binary constraint

1 引言

约束满足问题 (Constraint satisfaction problem, CSP) 是约束求解研究领域的热点问题. 问题解空间搜索是约束求解中的关键环节, 目前较为成熟且常用的系统搜索策略主要有两种[1]: 回顾策略和展望策略. 回顾策略针对已经进行过的搜索过程, 从中找出产生不相容的关键因素或者减少不必要的搜索节点访问开销; 而展望策略则侧重于对尚未搜索的解空间进行削减. 常用的回顾算法有 BackMarking (BM)[2]、BackJumping (BJ)[2, 3]. BM 算法的缺点是不能找到产生冲突的真正原因且具有很大的空间复杂度; 而 BJ 算法的缺点是无法减少重复的访问节点的开销. 对于展望策略, 比较常用的算法是 Forward Checking (FC)[4~6], 它的缺点和 BM 算法类似, 也是无法找到产生冲突的真正原因. 基于以上分析, 人们尝试将两类基本搜索策略或者不同算法混合起来进行联合求解, 互相弥补缺点, 从而提高搜索效率. Nadel[7] 首先提出将 BM 和 BJ 算法相融合的思想. Prosser[8] 设计了名为 BMJ 和 FC-CBJ 的两类混合算法. 虽然在一些特例 (如皇后问题) 的测试中 BMJ 表现得还不如单一的 BM 算法, 但是在其他 CSP 问题测试中都表现出其优越性; FC-CBJ 算法在求解大规模问题中的表现显然比单一算法优越得多. 文献 [9] 提供了文献 [8] 中的混合算法的 C 语言实现.

混合算法在搜索效率上的优势得到了充分的证实, 但是以上提到的这些混合搜索算法都是基于二元约束建模和表示的, 即约束条件中只含有两个约束变量, 本文将基于不同策略的混合搜索算法进行扩展, 用 JAVA 语言实现了两类混合搜索算法. 第一类是基于回顾策略的 BM 算法和 BJ 三种改进算法的混合; 第二类是基于展望策略的 FC 算法和基于回顾策略的 BJ 三种改进算法的混合. 混合搜索算法对文献 [8] 中算法的扩展体现在它们是基于多元约束[10, 11] 的, 多元约束更能体现现实世界中事物之间的相互关系, 同时又可以保留很多有用的关键性信息, 这是二元约束所不具备的特性. 通过混合基于单一策略的不同算法, 新算法在约束传播过程中尽量减少不必要的约束检查, 减小访问节点所需要的开销, 而且在约束传播失败的情况下有效地回跳, 尤其是当使用了多元约束的表示形式以后, 它能考虑到约束中多个变量之间的相容性关系, 使得回跳因素不只局限于两个变量, 这样有利于在发生连续回跳时相容性信息的更新, 以及回跳变量信息的提取. 测试结果证明混合算法有效地提高了搜索求解效率和约束求解系统的性能.

2 CSP 约束搜索策略

CSP 解空间的搜索过程是约束求解的基本步骤. 搜索策略大致可分为系统搜索策略和随机搜索策略两类. 系统搜索策略也称为穷尽搜索策略, 是完备的, 通用性较强, 但系统开销大; 随机搜索策略是不完备的, 但是对于一些大规模问题求解效率较高. 我们所设计的通用约束求解系统[12, 13] 融和了这两类搜索策略, 使得问题解的搜索具备一定的智能性. 本文着重讨论系统中多元约束表示下的系统搜索策略及其相应算法, 我们在随机搜索算法方面的工作可参见文献 [14,15].

系统搜索策略主要分为: 普通搜索策略和约束传播策略. 早期求解 CSP 的方法是系统化搜索变量所有的可能赋值, 通常称之为普通搜索方法. 体现这一思想的最主要的算法就是基本回溯 (BackTrack, BT) 算法. BT 算法保证在有解的情况下找到解, 或者证明问题无解. BT 算法的缺点是在搜索解时没有用到任何约束关系信息来削减解空间, 因而其效率较低. 普通搜索策略和相容性技术结合起来产生了约束传播方法, 其策略主要分为两类: 回顾策略和展望策略. 这两类策略都是在系统化搜索过程中对问题的搜索空间进行一定程度的约减, 而它们的区别在于约减过程发生在搜索过程中的不同阶段.

展望策略的主要思想是: 通过对还没有实例化的变量进行相容性检查来避免未来的冲突, 该操作通常发生在准备对下一个变量实例化的时候, 这样可以通过有限的约束传播发现当前变量及其实例化的值对于后续搜索的影响. 回顾策略的主要思想: 当一个新值被选择来扩充不完全解时, 通过 "look back" 不完全解的先前选择的值来测试, 看新值是否与先前选择的值相容, 通过记录更多的关于搜索过程的路线, 这些变化减少了冗余的检查.

收稿日期 2006-1-10 收修改稿日期 2006-11-10
Received January 10, 2006; in revised form November 10, 2006
国家自然科学基金 (60473003), 教育部 "新世纪优秀人才支持计划", 吉林省杰出青年基金 (20030107) 资助
Supported by National Natural Science Foundation of China (60473003), Program for New Century Talents in University, the Outstanding Youth Foundation of Jilin Province (20030107)
1. 吉林大学计算机科学与技术学院 长春 130012 2. 符号计算与知识工程教育部重点实验室 长春 130012
1. College of Computer Science and Technology, Jilin University, Changchun 130012 2. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun 130012
DOI: 10.1360/aas-007-0974

最常用的展望算法是 FC 算法, FC 算法不利用值与值的比较来实现对未实例化变量的论域进行约减, 而是采用边界相容的方式进行值与论域的比较. 算法在每次对当前变量进行实例化的时候都要检查当前变量和未实例化变量之间的相容性, 但是它并不考虑未实例化变量之间的相容性. 如果发现当前变量论域中的所有值都不可取, 则算法会回退到上一个被实例化的变量处, 并且重新实例化该变量为一个新的论域值.

具体体现回顾策略的算法都是在 BT 算法上的改进, 比如 BM 算法、BJ 及其改进算法等. BM 算法是一种缓存查找信息的算法, 它可以通过减少访问节点所需要的开销来提高算法的求解效率. 算法的目的是要通过减少生成节点所需要的开销, 而不是尽量对查找空间进行剪枝来提高算法本身的效率. 通过记录过去失败的相容性检查, BM 算法可以避免重复那些以前执行过的, 没有必要的检查. BJ 算法是一种可以避免重复产生相同冲突点的算法, 当算法在约束传播的过程中发现冲突时, 可以直接回跳到产生冲突的真正变量处对其重新进行实例化, 而不是仅仅回跳到当前变量的前一个变量, 并对其反复赋予不同的值. BJ 算法主要有三种改进算法: GASBJ (Gaschnig′s Backjumping)、GBJ (Graph-based Backjumping)、CBJ (Conflict-directed Backjumping). GASBJ 算法实现了由叶终结状态回跳到冲突变量的思想; GBJ 算法则只是从约束图中提取出无关回溯点的信息, 它引入了在中间终结状态和叶终结状态回跳的思想, 但是这并不能保证就一定会回跳到冲突变量; CBJ 算法将在叶终结状态回跳到最远变量和在中间终结状态回跳到最远变量的思想结合到一起.

不论是基于哪类传播策略的搜索算法, 其共同目的都是为了提高算法的解搜索效率. 对于这些算法, 不能简单地说一种算法比另一种算法好, 面对具体问题还有一个选择适合算法的考虑, 一方面要根据经验, 另一方面要根据算法本身的特性.

3 基于多元约束的混合搜索算法

约束求解系统设计了六个基于回顾和展望策略的混合算法. 系统所定义的约束类型基本上能够表示大多数约束满足问题中的约束, 且大部分约束是三元甚至多元约束. 对于约束传播算法, 系统没有简单地采用弧相容技术来进行约束传播, 而是采用了边界相容技术, 为不同的约束设计了不同的相容性传播规则. 尽管有很多文献指出, 可以通过扩展弧相容的定义来得到对多元约束的弧相容, 同时提出了实现这种弧相容的算法. 但是, 事实上这种算法的复杂度取决于约束中的变量个数以及满足约束的变量元组数. 在解决实际问题时, 它可能是指数级增长的. 同时实际问题中的变量论域通常是一个连续的取值空间, 而且在问题求解的初期, 变量论域中不会有太多的值由于约束不相容而被删除. 因而, 如果此时使用弧相容技术进行约束传播, 则会对变量论域中的大部分值进行相容性检测, 这样的检测可能是不必要的, 因为这些值有可能会在以后的约束传播中通过边界相容以区间的形式被删除, 从而提高了问题的求解效率. 由于篇幅关系, 关于多元约束相容性算法的设计这里不予介绍. 下面仅对六种混合搜索算法分别进行讨论.

3.1 BM 算法和 BJ 算法混和

系统实现了 BM 算法和 BJ 改进算法 (GASBJ, GBJ, CBJ) 相结合的混合算法. 这三种混合算法的搜索过程基本遵循 BM 算法的实现过程, 而在约束传播过程中遇到不相容时, 利用 BJ 算法的特点引导算法找到一个合适的回跳位置.

1) **BM_GASBJ** 算法

BM_GASBJ 算法是 BM 算法和 GASBJ 算法的混合. 其实现过程基本上遵循 BM 算法的实现过程, 但是它要为每个已经实例化的变量 x 保存一个最远回跳位置, 这样在遇到叶终结状态时, 就可以回跳到它所指定的最远位置, 然后再对相应的变量重新进行实例化测试. 对于相容性检查, BM_GASBJ 算法和 BM 算法的不同之处在于如果一个变量 x_j 在当前变量 x_i 遇到冲突并回跳到 x_k 时被跳过, 即 x_j 在 x_k 的后面, x_i 在 x_j 的后面, 则当算法再次回到这个被越过的变量 x_j 并将其实例化成 a_{jl} 时, 如果 a_{jl} 之前已经进行过相容性检查, 则不用对其进行相容性检查就可以发现当 x_j 取值 a_{jl} 时冲突仍然存在, 此时只需要对 x_j 的最远回跳位置进行更新并测试 x_j 的其他值即可, 只有在 a_{jl} 没有进行过相容性检查时, 才作必要的相容性检查, 如果此时发现不相容则更新 x_j 的最远回跳位置, 然后再测试 x_j 的其他值.

2) **BM_GBJ** 算法

BM_GBJ 算法是 BM 算法和 GBJ 算法的混合. 其结合方式与 BM_GASBJ 算法相同, 同时对于相容性检查也进行了改进. BM_GBJ 算法中的每个变量 x_i 的回跳位置完全由约束图中的祖先节点关系以及相关终结变量集来决定.

3) **BM_CBJ** 算法

BM_CBJ 算法是 BM 算法和 CBJ 算法的混合算法. 其结合方式与 BM_BJ 算法完全相同, 同时对于相容性检查也进行了改进. 由于采用了 BM_BJ 算法对相容性检查的改进方式, 为了保证在算法重新实例化之前由于回退而跳过的变量时不用检查那些已经进行过约束检查且又对于此次回跳仍会导致不相容的值, 引入了一个约束数组, 它对每个变量 x_i 的每个论域值 a_{i1} 都保存一个约束, 这个约束就是当前变量 x_i 取值 a_{i1} 时, 与之相冲突的最早约束. 这样, 假定变量 x_i 是由于回跳而被跳过的一个变量, 则当算法再次对 x_i 进行实例化时, 如果发现变量 x_i 取值 a_{i1} 时仍会引起不相容, 则直接把相应的约束中在 x_i 之前实例化的变量加入到冲突集中即可.

这三个 BM 混和算法基本步骤相似, 其伪代码描述见文献 [12]. BM 算法和 BJ 算法的结合不仅使得算法可以在约束传播过程中尽量减少不必要的约束检查, 减小访问节点所需要的开销, 而且可以在约束传播失败的情况下有效地回跳, 尤其是当使用了多元约束的表示形式以后, 它能考虑到约束中多个变量之间的相容性关系, 使得回跳因素不只局限于两个变量, 这样就有利于在发生连续回跳时相容性信息的更新, 以及回跳变量信息的提取, 有效的提高了混合算法的求解效率.

3.2 FC 算法和 BJ 算法混和

同 BM 算法和 BJ 算法的混合算法一样, FC 算法和 BJ 算法的混合算法也有 3 种实现方式. 但是它们并不是通过减少访问节点的开销来提高算法的求解效率, 而更多的是通过相容性检查来对搜索树进行剪枝, 以达到削减搜索空间的目的. 这三种混合算法都是基于多元约束的, 它们的核心就在于约束传播过程的实现. 由于约束中存在多个变量, 因此约束传播过程中变量之间的相互影响就更加强烈, 这时不仅需要考虑到已实例化变量对未实例化变量的约减关系, 而且还需要考虑到未实例化变量之间的间接约减关系, 而对于这种约减关系是否会传递到下一个准备实例化的变量也是需要考

虑的因素之一. 虽然算法的约束传播过程变得更为复杂, 访问节点的开销会变得更大, 但是它的剪枝力度也很大, 而且这些传播过程中所记录的相容性信息还可以用来指导冲突产生时的回跳, 因此足以平衡掉访问的开销. 下面具体介绍这三种混合算法的关键思想和步骤.

1) **FC_GASBJ** 算法

FC_GASBJ 算法是 FC 算法和 GASBJ 算法的混合算法. FC 算法是检查当前变量与未实例化变量的相容性, 而 GASBJ 算法是检查当前变量与已实例化变量的相容性. 因此, 混合算法对当前变量的相容性处理比较特殊. 算法在检查当前变量的相容性时所考虑到的是那些未实例化的变量, 即算法检查当前变量与未实例化变量的相容性, 它把相容性检查时遇到的情况分为如下两种:

情况 1. 如果在某次相容性检查中使得一些未实例化变量的论域发生了约减, 则记录该约束所涉及的所有已实例化变量, 并表明他们对约束中哪些变量的论域产生了约减, 且若约束中存在不只一个未实例化变量, 则对每个未实例化变量产生了约减的已实例化变量均标记为对这个约束中论域发生了变化的变量的约减变量.

情况 2. 如果在某次相容性检查中发现了不相容, 则在该约束的所有已实例化变量, 以及对约束的所有未实例化变量产生了约减的已实例化变量中寻找离当前变量最近的变量, 看这个变量是否比当前变量原有的回跳变量离当前变量更近, 如果是, 则令它作为当前变量的新回退变量.

如果一个变量的所有值都是不相容的, 即算法遇到了一个叶终结变量, 则它会在这个变量的回跳变量, 以及对它产生了约减的变量中选取一个离其最近的变量作为回跳变量.

在这个算法中利用一个数组来记录哪些变量对任意一个变量产生了约减, 但是对上面所说的相容性检查过程, 只有当前变量对未实例化变量产生的约减才可以传递到对下一个变量的访问, 而其他的间接约减关系只是临时生成的, 并不会带到下一个变量. 如果一个已实例化的变量由于算法的回退而被恢复了论域, 使之不再是实例化的了, 则对它所约减的变量记录将被清空. 这样做不会影响问题求解的正确性, 因为在约减关系中我们所记录下来的是当前变量对未实例化变量的约减, 而它是离该未实例化变量最近的约减变量, 同时又由于 FC_GASBJ 算法只会在遇到叶终结变量时回跳, 而在遇到中间终结变量时并不回跳, 所以可以保证求得问题的解.

2) **FC_GBJ** 算法

FC_GBJ 算法是 FC 算法和 GBJ 算法的混合算法. 该算法完全不考虑变量之间存在的约减关系, 这些信息都将被约束图中的连接关系所替代. 算法本身与 FC_GASBJ 算法相同, 对相容性检查, 具有其特殊性, 是检查当前变量同未实例化变量的相容性. 它把相容性检查中遇到的情况分为如下两种:

情况 1. 如果在某次相容性检查中使得一些未实例化变量的论域发生了约减, 且约束中存在不只一个未实例化变量, 则令每一个未实例化变量的, 在当前变量之前实例化的祖先都成为论域发生了约减的变量的祖先.

情况 2. 如果在某次相容性检查中发现了不相容, 则令该约束中所有未实例化变量的, 在当前变量之前实例化的祖先变量均成为当前变量的祖先变量.

如果算法发现当前变量的所有值都是不相容的, 则它会从当前变量的所有祖先变量中选取一个离其最近的变量作为回跳变量, 同时把当前变量的所有祖先变量都合并到回跳变量的祖先变量中去.

需要注意的是, 该算法不同于 FC_GASBJ 算法, 它在相容性检查阶段对未实例化变量的祖先节点集的更新可以一直延续到下一个要访问的变量, 即在访问下一个变量时, 并不恢复前一个变量对未实例化变量的祖先节点集的更新. 如果算法中的某个已实例化变量因为回跳而导致它的论域得以恢复, 使之不再是实例化的了, 则对于那些尚未实例化的变量的祖先变量集合都要恢复成这个变量实例化之前的状态.

3) **FC_CBJ** 算法

FC_CBJ 算法是 FC 算法和 CBJ 算法的混合算法. 同 FC_GASBJ 算法一样具有特殊的相容性检查过程. 它把相容性检查过程中遇到的情况分为两种:

情况 1. 如果在某次相容性检查中使得一些未实例化变量的论域发生了约减, 则记录该约束所涉及的所有已实例化变量, 并表明他们对约束中哪些变量的论域产生了约减, 且若约束中存在不只一个未实例化变量, 则对每个未实例化变量产生了约减的已实例化变量都要标记为对这个约束中论域发生了变化的变量的约减变量.

情况 2. 如果在某次相容性检查中发现了不相容, 则把该约束中的所有已实例化变量都加入到当前变量的冲突变量集中, 同时把对约束中的未实例化变量产生了约减的已实例化变量也加入到冲突变量集中.

如果当前变量的所有值都是不相容的, 则在它的冲突变量集和对它产生了约减的变量中选取一个离其最近的变量作为它的回跳变量, 同时对这个回跳变量的冲突变量集进行更新, 即把在回跳变量之前的, 且位于当前变量冲突变量集中或者对当前变量产生了约减的变量都加入到回跳变量冲突变量集中.

该算法同 FC_GBJ 算法一样, 都会把在相容性检查过程中所生成的信息带到对下一个变量的访问中去, 所谓的信息指的是变量的约减信息, 即哪些变量对哪些变量产生了约减 (直接或间接约减). 同时在算法回跳时, 这种约减信息也要进行恢复, 它需要恢复成回跳变量被实例化之前的状态, 以免做出错误或者不必要的指导.

这三个 FC 混合算法基本步骤相似, 其伪代码描述见文献 [12].

4 算法测试与分析

约束求解系统分别实现了以上六种基于多元约束的混合搜索算法和文献 [8] 中基于二元约束的两类混合搜索算法, 并对经典的 N 皇后问题和标准约束问题库 (CSPLib)[16] 中实例进行了测试. 对 N 皇后问题的测试旨在对比分析单一算法与混合算法, 基于二元约束与基于多元约束的混合算法在性能效率上的差别: 对标准约束库问题的测试侧重分析不同混合算法在搜索效率上的差别. 测试主要分为两方面: 算法运行时间和搜索效率. 测试参数包括: 1) CPU 运行时间 $TIME$ (ms); 2) 访问搜索树的节点数 $NODE$; 3) 相容性检查数目 $CHECK$. 测试环境: IBM Netvista P4 2.0G CPU/256M, RAM Windows2000 Server SP4/Sun j2sdk1.4.0.

4.1 N 皇后问题测试

N 皇后问题是约束满足问题中的一个经典理论问题. 建模 N 皇后问题的一种方法是为每一行 $i \in 1, \cdots, N$ 引入一个整型变量 $X_i \in [1, N]$, X_i 表示在第 i 行的第 X_i 列上放有一个皇后. 这样, N 皇后问题中的约束有如下形式:

对于任意的 $1 \le i < j \le N$, X_i, $X_j \in 1, \cdots, N$, 都有下面的三个不等式成立

$$x_i \neq x_j \tag{1}$$

$$x_i - x_j \neq i - j \tag{2}$$

$$x_i - x_j \neq j - i \tag{3}$$

式 (1) 表示任意两个皇后都不在同一列上, 式 (2) 和式 (3) 表示任意两个皇后不在同一条对角线上. 但以上约束建模方式是基于二元约束的, 无法直观体现皇后问题中的全局多元约束 Alldifferent, 即每一行、列及对角线上只能出现一个皇后. 由式 (2) 和式 (3), 有 $X_i - i \neq X_j - j, X_i + i \neq X_j + j$, 引入我们定义的多元约束 Alldifferent 就可以将这个模型转化成如下形式

$$\text{Alldifferent}(X_i, \cdots, X_N), \quad X_i \in 1, \cdots, N \tag{4}$$

$$\text{Alldifferent}(Y_i, \cdots, Y_N), \quad Y_i \in -N+1, \cdots, N-1 \tag{5}$$

$$\text{Alldifferent}(X_i, \cdots, X_N), \quad Z_i \in 2, \cdots, 2*N \tag{6}$$

系统对 N 皇后问题用六种混合算法进行了测试. 系统测试 N 皇后问题数目为 4 个到 500 个, 测试结果数据量很大, 我们发现当 $N \ge 12$, 这六种混合算法在性能表现上开始出现较大差异, 所以这里我们选取了 $N = 12$ 到 $N = 18$ 的五个皇后问题. 图 1、图 2、图 3 分别给出单一 BJ 算法的三种改进算法 (GASBJ、GBJ、CBJ) 和混合算法的运行时间比较 (图中坐标为对数坐标).

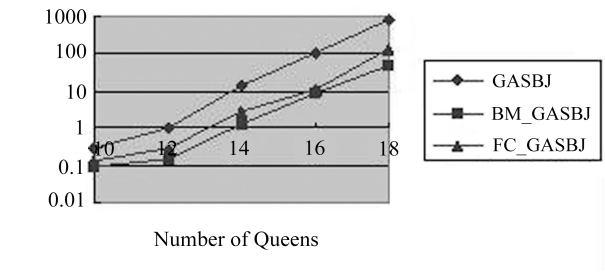

图 1　GASBJ、BM_GASBJ、FC_GASBJ 运行时间比较

Fig. 1　Comparison of run times of algorithms of GASBJ, BM_GASBJ, and FC_GASBJ

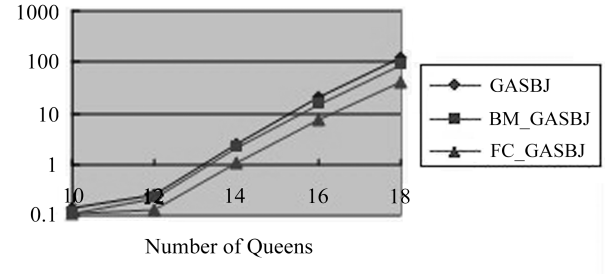

图 2　GBJ、BM_GBJ、FC_GBJ 运行时间比较

Fig. 2　Comparison of run times of algorithms of GBJ, BM_GBJ, and FC_GBJ

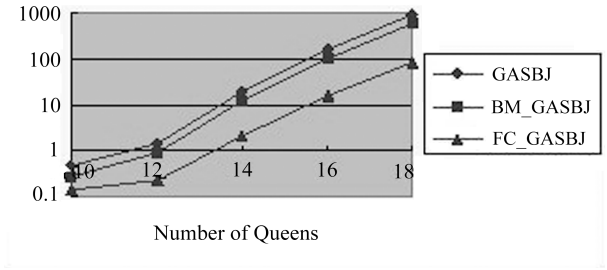

图 3　CBJ、BM_CBJ、FC_CBJ 运行时间比较

Fig. 3　Comparison of run times of algorithms of CBJ, BM_CBJ, and FC_CBJ

由于文献 [8] 中 BMJ 混合算法在皇后问题中的搜索效率还不及一般单一算法, 所以我们选择文献 [8] 中基于二元约束 FC-CBJ 算法和本文基于多元约束 FC_CBJ 算法对 30 皇后问题进行测试. 图 4、图 5 分别为两个算法的求解结果. 文献 [8] 算法的 $TIME = 1\,551\,550$ ms, 本文算法 $TIME = 634\,719$ ms.

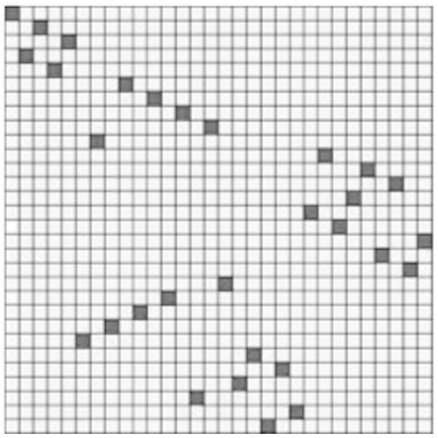

图 4　文献 [8] FC-CBJ 算法

Fig. 4　The result of algorithm of FC-CBJ in [8]

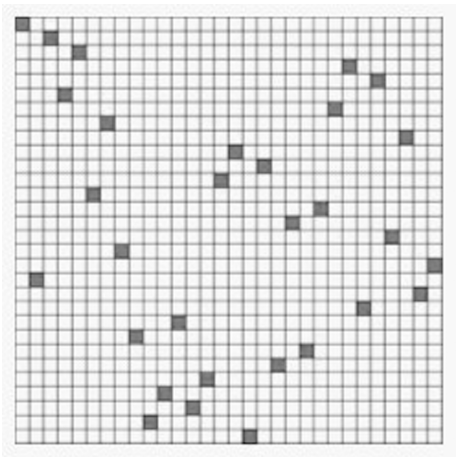

图 5　本文 FC_CBJ 算法

Fig. 5　The result of algorithm of FC_CBJ

分析测试结果, 我们可以得出以下结论:

1) 无论是基于 BM 算法的三种混合算法, 还是基于 FC 算法的三种混合算法, 本文混合算法都有效地在约束传播过程中减少了不必要的约束检查, 减小了访问节点所需要的开销, 从而提高了搜索效率.

2) BM_GASBJ 算法、BM_GBJ 算法和 BM_CBJ 算法这三种混合算法既具有 BM 算法跳过已经进行过的相容性检查的特性, 又有 BJ 算法在发生冲突时回跳的特性. 它们的结合在某种程度上会改善单纯使用 BM 算法或 BJ 算法的性能, 但是它也继承了 BM 算法维护相容性信息的空间巨大的问题, 因此在求解大规模 CSP 问题方面会相对困难.

3) FC_GASBJ 算法、FC_GBJ 算法和 FC_CBJ 算法均是回跳算法和前向检查算法的混合算法. 它们兼有了两种算法的优点于一身, 但是回跳算法对于相容性信息维护的价值并没有很好地体现出来. 不过它已明显优于一般回跳算法.

4) 从图 4、图 5 中的运行结果可以看出, 当使用了多元约束的表示形式以后, 相容性检查涉及了多个约束变量, 当发生连续回跳时相容性信息的更新速度更快, 搜索效率更高.

4.2 CSPLib 问题测试

为了证明多元约束混合算法的可行性和通用性, 我们从标准 CSPLib 库全部 20 个实例中挑选出 15 个实例, 用基于多元约束的 FC 混合算法对其进行了测试和比较, 测试结果见文献 [12]. 实例名称为标准库中的问题序号. 问题描述和其他系统测试结果可参见文献 [16]. 这些测试实例为含有复杂多元约束的 CSP 问题, 对于这些约束, 如果用二元约束建模和求解, 如文献 [16] 中介绍的其他系统的求解实例, 则问题的表达清晰性较差, 代码量较大. 而用我们系统所提供的多元约束[12], 如 AtLeast 、Minimize 、Sum 、Alldifferent 、Sort 等约束, 来建模和求解, 很大程度上提高了问题的表示能力. 通过实验测试, 我们可以得出以下一般性结论:

1) 对于小规模 CSP 问题, 算法在单个节点上避免的约束检查数量越多, 则它的执行效率越高. FC_CBJ 算法的求解效率高于 FC_GASBJ 算法和 FC_GBJ 算法.

2) 对于大规模 CSP 问题, 算法在单个节点上进行的约束检查数量越多, 则它的执行效率越高. FC_GASBJ 算法的求解效率高于 FC_CBJ 算法和 FC_GBJ 算法.

3) 过度强调多元约束建模势必增大系统的开销. 这之间需要具体分析.

综合分析测试结果, 可以看出这六种基于多元约束的混合算法各具特点, 它们以不同的方式实现了问题解空间的削减, 当然这是以对当前变量实例化后进行额外的约束传播为代价的. 因此, 我们不能笼统地说哪个算法就一定优于另一个算法, 或者说哪个算法一定是最优的, 而是需要具体问题具体分析.

5 结论

问题解空间搜索是约束求解领域的核心问题. 全局、系统的搜索策略可以提高算法求解的通用性, 但是很多实际约束满足问题的约束关系是多元的, 对于这类约束的表示和处理是关键, 更是难点. 本文所述的各种混合搜索算法均已在我们所设计的通用约束求解器中实现, 且这些搜索算法都基于多元约束, 即充分利用了多元约束的强大表示能力和约束传播能力, 从而提高了搜索效率, 改善了系统的总体性能, 适合于解决一般的多元约束满足问题.

References

1 Dechter R. *Encyclopedia of Artificial Intelligence, 2nd ed.*. Canada: John Wiley & Sons, 1992. 276∼285

2 Dechter R. *Constraint Processing*. USA: Morgan Kaufmann, 2003. 67∼72

3 Dechter R, Frost D. Backjump-based backtracking for constraint satisfaction problems. *Artificial Intelligence*, 2002, **136**(2): 147∼188

4 Bacchus F, Grove A. Looking Forward in Constraint Satisfaction Algorithms. Technical Report, University of Waterloo, 1999

5 Bessiere C, Meseguer P, Freuder E C, Larrosa J. On forward checking for non-binary constraint satisfaction. In: Proceedings of Principles and Practice of Constraint Programming. 1999. 88∼102

6 Bacchus F, Grove A. *Principles and Practice of Constraint Programming*. Berlin: Springer-Verlag, 1995. 292∼309

7 Nadel B A. Constraint satisfaction algorithm. *Computational Intelligence*, 1989, **5**(3): 188∼224

8 Prosser P. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 1993, **9**(3): 268∼299

9 Beek P V. Networks [Online], available: http://ai.uwaterloo.ca/vanbeek/software/software.html, November 1, 2006

10 Stergiou K. Representation and Reasoning with Non-binary Constraints. The University of Strathclyde, 2001

11 Sun Ji-Gui, Jing Shen-Yan. Solving non-binary constraint satisfaction problem. *Chinese Journal of Computers*, 2003, **26**(12): 1746∼1752
(孙吉贵, 景沈艳. 非二元约束满足问题求解. 计算机学报, 2003, **26**(12): 1746∼1752)

12 Chen Shang-Wei. The Design and the Implementation of the Constraint Solver based on Java [Master dissertstion], Jilin University, 2005
(陈尚伟. 基于 Java 的约束求解器的设计与实现 [硕士学位论文], 吉林大学, 2005)

13 Jiang Ying-Xin, Sun Ji-Gui. Solving constraint satisfaction problem and ILOG SOLVER introduction. *Journal of Jilin University (Science Edition)*, 2002, **20**(1): 53∼60
(姜英新, 孙吉贵. 约束满足问题求解及 ILOG SOLVER 系统简介. 吉林大学学报 (理学版), 2002, **20**(1): 53∼60)

14 Jing Shen-Yan, Sun Ji-Gui, Zhang Yong-Gang. Solving scheduling problems with genetic algorithms. *Journal of Jilin University (Science Edition)*, 2002, **20**(3): 263∼267
(景沈艳, 孙吉贵, 张永刚. 用遗传算法求解调度问题. 吉林大学学报 (理学版), 2002, **20**(3): 263∼267)

15 Yang Qing-Yun, Sun Ji-Gui, Zhang Ju-Yang. Improvements of particle swarm in binary CSPs with maximal degree variables ordering. *Journal of Computer Research and Development*, 2006, **43**(3): 436∼441
(杨轻云, 孙吉贵, 张居阳. 最大度二元约束满足问题粒子群算法. 计算机研究与发展, 2006, **43**(3): 436∼441)

16 Gent I P, Walsh T. CSPLib home page [Online], available: http://www.4c.ucc.ie/tw/csplib/, 1999

孙吉贵 吉林大学教授. 主要研究方向为人工智能, 约束程序, 智能信息处理. 本文通信作者. E-mail: jgsun@jlu.edu.cn
(**SUN Ji-Gui** Professor at Jilin University. His research interest covers artificial intelligence, constraint programming, and intelligence information processing. Corresponding author of this paper.)

张居阳 吉林大学博士研究生. 主要研究方向为约束程序, 智能调度, 离散优化. E-mail: zjy2003@vip.sina.com
(**ZHANG Ju-Yang** Ph. D. candidate at Jilin University. His research interest covers constraint programming, intelligent scheduling, and discrete optimization.)

陈尚伟 吉林大学硕士研究生. 主要研究方向为约束程序.
E-mail: kassy2002cc@yahoo.com.cn
(**CHEN Shang-Wei** Master student at Jilin University. His main research interest is constraint programming.)