`软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(4):1543−1569 [doi: 10.13328/j.cnki.jos.006682]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

资源独立工作流可满足性的最小增量模式回溯*

翟治年[1], 卢亚辉[2], 刘关俊[3], 雷景生[1], 向 坚[1], 吴茗蔚[1]

[1](浙江科技学院 信息与电子工程学院, 浙江 杭州 310023)

[2](深圳大学 计算机与软件学院, 广东 深圳 518060)

[3](同济大学 计算机科学系, 上海 201804)

通信作者: 翟治年, E-mail: zhaizhinian@zust.edu.cn

摘 要: 工作流可满足性是业务安全规划的基本问题, 正在面临高资源配比(资源数 n 显著大于步骤数 k)造成的性能挑战. 在资源独立约束下, 其最高效求解途径是模式空间上的增量回溯法 IPB. 为克服结点真实性验证的性能瓶颈, 它增量计算模式 k 指派(二部)图及其(左完备)匹配, 分别需要 $O(kn)$和 $O(k^2)$时间. 利用父子模式的原子差异增量计算完全指派图, 只需 $O(n)$时间, 特别是其实际性能, 将随模式块规模增长迅速提高. 但该图的 $O(kn)$规模导致了同样的增量匹配时间. 进而引入完备 k 核心匹配概念, 证明其存在性等价于左完备匹配, 且其增量计算时间为 $O(k^2)$. 由此, 建立了时间复杂度更低的最小增量模式回溯法. 在含互斥和两种全局值势约束而授权比例约为 1/4 的扩展公开实例集上进行实验, 结果表明: 当 n/k=10(及 n/k=100), 而 k 变化时, 该方法较 IPB 有平均超过 2(及 5)倍、最低 1.5(及 2.9)倍的性能优势; 当 k=18(及 k=36), 而 n/k=2~4096(及 n/k=2~2048)时, 该方法有平均超过 2.6(及 3.6)倍优势; 而较 2021 年 Minizinc 挑战赛的冠军求解器 Google OR-Tools CP-SAT, 该方法最低有超过 3 倍优势.

关键词: 工作流; 授权; 约束; 资源独立; 资源分配; 可满足

中图法分类号: TP311

中文引用格式: 翟治年, 卢亚辉, 刘关俊, 雷景生, 向坚, 吴茗蔚. 资源独立工作流可满足性的最小增量模式回溯. 软件学报, 2023, 34(4): 1543−1569. http://www.jos.org.cn/1000-9825/6682.htm

英文引用格式: Zhai ZN, Lu YH, Liu GJ, Lei JS, Xiang J, Wu MW. Minimum Incremental Pattern Backtracking for Resource-Independent Workflow Satisfiability Problem. Ruan Jian Xue Bao/Journal of Software, 2023, 34(4): 1543−1569 (in Chinese). http://www.jos.org.cn/1000-9825/6682.htm

Minimum Incremental Pattern Backtracking for Resource-independent Workflow Satisfiability Problem

ZHAI Zhi-Nian[1], LU Ya-Hui[2], LIU Guan-Jun[3], LEI Jing-Sheng[1], XIANG Jian[1], WU Ming-Wei[1]

[1](School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou 310023, China)

[2](College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)

[3](Department of Computer Science and Technology, Tongji University, Shanghai 201804, China)

Abstract: Workflow satisfiability problem is an elemental issue in the security planning of business process, and it is facing the performance challenge caused by high resource ratio (the number n of resources is significantly greater than the number k of steps). Under resource-independent constraints, its most efficient approach is incremental pattern backtracking (IPB) in the pattern space. To overcome the performance bottleneck of verifying whether a node is authentic, IPB computes the k-assignment (bipartite) graph of a pattern and its (left complete) matching in an incremental manner, which requires $O(kn)$ and $O(k^2)$ time respectively. This study computes a full-assignment graph incrementally with only $O(n)$ time by exploiting the atomic difference between a sub-pattern and its super one, and in particular its actual performance will increase rapidly with the size of a block in pattern. However, the size $O(kn)$ of such a graph will result in the same incremental matching time. Further, this study introduces the concept of complete k core matching and shows that its

* 基金项目: 国家自然科学基金(62172299, 61972357); 浙江省教育厅一般科研项目(Y201737476)

收稿时间: 2021-10-07; 修改时间: 2021-11-30, 2022-02-12; 采用时间: 2022-05-09

existence is equivalent to a left complete matching and its incremental computation only costs $O(k^2)$ time. Therefore, this study proposes an algorithm of minimum incremental pattern backtracking (MIPB) that is superior to IPB in time complexity. Experiments are conducted on an extended public instance set with constraints of two global value-cardinality types and of the mutual exclusion, and with an authorization ratio of about 1/4. The results show that: when k varies at n/k=10 (n/k=100, respectively), MIPB achieves averagely more than 2 (5, respectively) times and at least 1.5 (2.9, respectively) times advantage of performance compared to IPB; when k=18 (k=36, respectively), and n/k belongs to 2~4096 (2~2048, respectively), MIPB achieves averagely more than 2.6 (3.6, respectively) times advantage. While compared to the champion solver OR-Tools CP-SAT in the 2018~2021 Minizinc Challenges, MIPB achieves at least more than 3 times advantage.

Key words: workflow; authorization; constraint; resource-independent; resource allocation; satisfaction

云制造和众包等平台将大量分散资源汇集起来, 以服务化方式运营, 带来了动态适应、并发管理、访问控制与数据保护等一系列技术与安全问题[1−5]. 就单个工作流的资源分配而言, 资源服务化扩大了供给来源, 但也造成了安全隐患. 例如: 在云制造平台上执行客户的工作流时, 各步骤的资源可获取执行所需的业务数据, 而这些以虚拟服务形式接入的资源来自不同地域和组织[6], 很容易隐匿恶意, 威胁数据安全. 为防止恶意资源提供者获取关联的重要数据, 需对有关步骤施加职责分离等约束[7]. 而此类约束可能破坏资源分配的可行性, 需解决有关的冲突判定与正确决策问题, 也就是基于步骤和资源的授权关系, 求满足安全约束的任意资源分配(或确定其无解), 称为工作流可满足性问题(workflow satisfiability problem, WSP)[8,9]. 它是多种安全规划的基础[10−12], 并有相应的计数问题[13,14]. 平台日常处理大量的业务实例, 对 WSP 算法的性能要求很高, 但该问题是 NP 完全的[15−17], 缺乏高效求解的保证. WSP 有步骤数(k)和资源数(n)两个核心参数: 前者通常不超过 100[18]; 而在资源服务化趋势下, 后者可能很大[15,16,19,20]. 这种高资源配比特征, 给 WSP 求解造成了很大的性能压力.

WSP 的完备性要求制约了局部和智能搜索方法的应用. 不过, 其较低的步骤数有利于确定性求解(或结合启发式优化). 对符合特定代数规范的约束网络, 存在多项式时间的 WSP 算法[21], 但其可用性较为受限. 大部分研究均针对一定约束配置而不限网络结构, 并可归纳为解空间搜索和动态规划两条路线[22].

- 前者通用性较强, 但解空间“组合爆炸”严重, 给搜索优化造成了困难. 例如: 树分解回溯法[23,24]对约束网络进行簇集分解, 有助于提高稀疏约束 WSP 的求解性能[25], 但其时间复杂度高达 $O^*(d^{W+1})=O(n^k)$ (其中, W+1 是最大簇集的基数, 而 d 为各步骤授权资源数的最大值), 为依赖 n 的幂指函数, 难以应对高资源配比压力;
- 后一路线有利于结合约束特征进行优化, 实现固定 k 的参数化时间复杂度, 但其通常需要指数级空间代价, 导致性能严重失衡. 例如: 文献[19]应用基于赋权集容斥原理和快速 zeta 变换的动态规划方法, 对互斥约束 WSP 取得了目前最好的 $O^*(2^k(c+n^2))$时间复杂度(c 为约束数); 对其他若干约束配置, 也将时间的指数项降至以 2 或 3 为底. 但其空间代价至少为 $O^*(2^k)$, 极易耗尽内存, 严重制约了求解规模和时间性能表现[26].

近年来, 一个显著的研究动态是模式技术的出现[27]. 模式可看作步骤子集的划分. 根据常见于 WSP 约束的资源独立性(resource-independent, RI)特征, 对资源分配部分解进行模式抽象, 可以合并计算状态, 消除大量资源造成的状态膨胀, 从而设计固定 k 的参数化算法, 消解高资源配比压力. 该技术推动了前述两条路线的发展, 产生了模式回溯和模式动态规划两种新型途径. 特别地, 前者在规模与 n 无关的模式空间(如图 1 左端的示例)上开展回溯搜索, 可显著压缩搜索规模, 降低时间复杂度. 同时, 保留了回溯法通常为多项式空间复杂度的优势, 并可利用其丰富的技术积累. 该途径的最新进展称为增量模式回溯法(incremental pattern backtracking, IPB)[28], 对高资源配比的资源独立约束 WSP, 有目前最好的实际性能和突出的技术优势[29,30].

模式回溯[31]在模式空间上搜索, 对其结点进行合格性(即满足约束)和真实性(即满足授权)验证: 前者与传统回溯差别不大, 且对搜索性能影响较小, 特别是 RI 约束具有验证代价与 n 无关的优点, 而常见的值势约束验证代价不超过 $O(k)$($O(k)$条的验证代价为 $O(k^2)$[28]); 后者是模式回溯的特征性工作, 也是其为压缩解空间而付出的主要代价, 包括指派和匹配两个环节. IPB 以增量方式进行真实性验证, 即利用父子模式的差异来计

算授权指派二部图(其左侧顶点是模式划分块, 右侧顶点是资源, 每个资源对其相邻块中每个步骤都有授权关系, 如图 1 所示)及其左完备匹配(例如图 1 指派图的虚线边集), 分别取得了 O(kn)和 O(k^2)的时间复杂度. 这一综合结构的算法设计问题对 IPB 的性能优化至关重要, 而本文将从新的角度对其进行研究, 建立最小增量的模式回溯法(minimum incremental pattern backtracking, MIPB), 主要贡献如下.

1. 基于模式的划分表示, 定义了模式之间的增元和增集关系, 据此刻划动态模式空间的结构特征, 并证明其搜索完备性, 初步建立了相应的形式化理论, 为本文及进一步研究奠定了基础;
2. 充分利用子模式增块与其父模式原块的差别, 以最小增量方式计算增块的完全指派图邻域, 使子模式的完全指派图计算时间从 O(kn)降为 O(n); 且在此量级内, 其实际性能可随增块规模迅速提高. 通过对完全指派图使用全局存储, 在搜索深入、切换和回溯时进行恰当的备份、更新和恢复, 可在整个模式搜索中保持上述性能. 同时, 针对高资源配比条件, 证明在真实性验证等效前提下, 对邻点数超标的增块无须匹配. 由此将增量匹配实际使用的子图控制在 O(k^2)规模, 使该环节可在 O(k^2)时间内完成. 综上, 可将子结点真实性验证时间从 O(k^2+kn)降至 O(k^2+n), 从而使 MIPB 的时间复杂度达到 O(($f(c)$+k^2+n)$B(k)$)(其中,$B(k)$为第 k 个贝尔数, $f(c)$为单结点约束验证时间), 优于 IPB 的 O(($f(c)$+k^2+kn)$B(k)$)时间. 在公开和扩充实例集上的验证表明, 较现有 IPB 和 CSP 竞赛的冠军求解器, MIPB 有突出的实际性能优势.

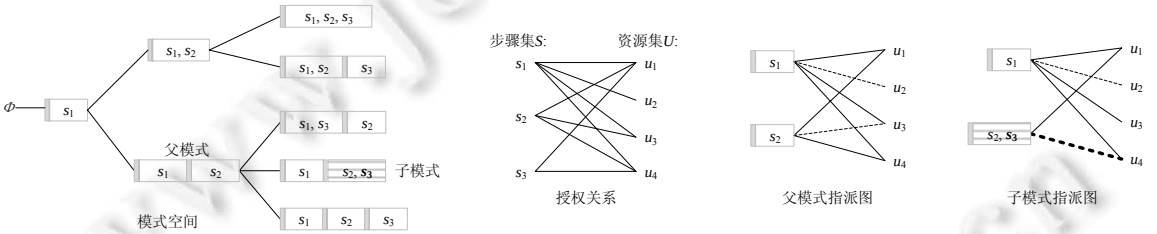

图 1 模式空间和授权指派图示例

本文第 1 节介绍模式技术和二部图匹配等相关工作. 第 2 节介绍 WSP 和模式回溯法相关概念. 第 3 节形式化描述动态模式空间理论. 第 4 节建立本文的 MIPB 算法, 其中, 第 4.1 节和第 4.2 节分别对模式真实性验证的指派和匹配两个环节, 给出相应的最小增量计算技术; 第 4.3 节给出算法描述与分析; 第 4.4 节进一步介绍算法实现方式. 第 5 节将本文算法与相关方法进行实验比较. 第 6 节总结全文并指出下一步研究方向.

1 相关工作

1.1 打破对称和模式技术

WSP 是约束满足性问题(constraint satisfaction problem, CSP)[32]在工作流应用特征和约束配置下的特殊形式. 而模式回溯等 WSP 求解技术属于 CSP 打破对称的范畴, 并代表了值对称打破的一项新进展.

所谓对称是赋值空间上的置换群, 而打破对称旨在减少对称不可行解的重复搜索. 2006 年, Cohen 在文献[33]中建立了严格的对称性定义, 从理论上证明了约束对称(也称问题对称)和解对称的深刻差别[34]. 他指出: 解对称群只能根据所有可行解确定, 难以完全打破; 约束对称群可根据问题定义确定, 有可能完全识别和打破. 但对于一般的赋值对称, 识别效率很低. 即使对更为单纯的值对称, 也存在一个长期困扰人们的问题: 若所有变量具有统一的值域, 则可能存在完全值对称, 可通过群等价树等技术高效打破[35]; 但若变量值域存在差异, 则会破坏值对称的完全性, 使其难以高效识别和打破. 有很多相关的研究, 如值可交换性[36]、逐片值可交换性[37]、逐片值-变量可交换性[38]等等. 它们都是从包括值域和约束的整个问题定义中, 在一定条件下寻找受限的对称性, 而未解决问题深层次的矛盾.

在此背景下, Cohen 等人通过分析常见于 WSP 约束的资源独立性特征, 提出了资源分配模式概念及其编码表示, 并设计了一种利用模式编码来压缩缓存的动态规划算法[27]. 模式概念将完全值对称现象归因于约束

的值独立性, 而不再捆绑到整个问题定义上. 而模式动态规划表明: 借助模式抽象, 有可能实现该对称性因素的单独利用或打破, 这是非常有意义的. 该算法的时间复杂度为 $O^*(3^k w\log w)$, w 是资源的分散度. 实验研究表明, 其性能优于文献[15]等早期代表性工作[39]. 该算法的主要缺陷是动态规划缓存需要指数级空间, 大大限制了求解规模和时间性能. 2016 年, Cohen 等人又在 at-most-r/at-least-r 等值势约束下, 通过约束传播、消除无用值和预处理, 优化了模式动态规划的实际性能[40], 但其空间开销仍为指数级, 且实际占用相当高.

2015 年, Karapetyan 等人开始将模式技术用于回溯搜索. 他们解决了模式真实性验证的问题, 从而可以利用回溯在模式空间上搜索可行解, 其时间复杂度为 $O^*(B(k))$[31]. 模式回溯首次实现了多项式空间复杂度的 WSP 固定参数求解, 相对文献[27]的模式动态规划和早期 WSP 算法有显著的实际性能优势. 2016 年, Crampton 等人识别了 WSP 约束中的类别独立性特征, 并建立了相应的模式回溯法, 以求解层次化组织等嵌套约束配置下的 WSP[41]. 2018 年, 翟治年等人指出: 文献[31]的模式回溯仅对完全模式进行(完整或充分的)真实性验证, 未能有效平衡验证代价与剪枝性能[22], 而对每个模式进行这一验证, 可显著提高较难实例的处理性能. 但其仍采用文献[31]的真实性验证方式, 对每个模式从头计算指派图及其匹配. 2019 年, Karapetyan 等人又提出了增量模式回溯法 IPB, 通过利用上下文信息计算指派图及其匹配, 有效降低了授权验证代价[29]. 在高资源配比的资源独立 WSP 上, IPB 的实际性能显著超过了多种代表性方法, 充分证实了模式回溯的先进性[30]. 最近, 文献[42]又利用模式块中各步骤授权邻点的分布间隙, 设计了一种边界收缩的加速方法, 以克服文献[28]邻点搜索范围过大的缺陷, 为 IPB 提供了一种优化实现. 此外, 近期的文献[30]借助模式和所谓上下文依赖的授权关系来建模更多类型的约束, 并基于 SAT4J 求解, 对非资源独立 WSP 也达到了固定 k 的参数化性能,进一步扩大了模式技术的意义. 特别地, 模式回溯提出了模式真实性验证这一新型任务, 下面单独介绍其问题和进展:

回溯法在根据授权关系(即各步骤变量的值域)构造的原始解空间上搜索, 只需验证结点是否满足约束. 在高资源配比条件下, 模式空间较原始解空间精简得多, 而资源独立性约束可在模式上验证[27], 故有必要和可能研究模式空间上的搜索算法. 但模式是一组部分解的抽象, 不一定真实(即满足授权). 文献[31]解决了孤立模式的真实性验证问题, 将其分解为求模式指派(二部)图及其(左完备)匹配这两个环节, 分别需要 $O(k^2n)$和 $O(k^3)$时间(其中, 匹配环节可用 Hopcroft-Karp 算法改进至 $O(k^{2.5})$时间). 文献[28]的 IPB 将真实性验证与模式空间结构联系起来, 利用搜索上下文, 增量计算指派图及其匹配. 其增量指派只需计算增块(例如图 1 中子模式的阴影块)的邻域, 而增量匹配只求一条从增块出发的增广路. 因指派图规模影响其计算代价和匹配效率, 该文献对增量指派与匹配进行了整体优化, 方法是: 计算不完全的 k 指派图(每块至多有 k 个邻点), 使边数 e 从 $O(kn)$降到 $O(k^2)$, 而在其上求匹配对真实性验证等效. 因 k 指派图的增量计算时间为 $O(kn)$, 其上的增量匹配时间为 $O(k^2)$, 故该文献的(结点)真实性验证代价为 $O(kn+k^2)$.

1.2 二部图最大匹配与匹配增广

模式真实性的匹配验证可归结为求二部图最大匹配(要求匹配边数 s 等于图左侧顶点数 L, 相当于求左完备匹配). 而结合搜索上下文, 可通过一次匹配增广完成. 文献[28]在指派图计算阶段对边集进行约简, 将图规模控制在 $O(k^2)$, 从而匹配增广时间复杂度为 $O(k^2)$. 该结果完全符合二部图匹配增广的标准时间复杂度 $O(e)=O(v^2)$, 其中, v 是二部图顶点数. 但其解除了与资源数 n 的耦合(注意, 指派图的 $v=O(k)+n$), 有利于高资源配比下的性能优化. 而本文提出了最小增量匹配技术, 事实上可直接对边数 $e=O(kn)$的完全指派图, 在 $O(k^2)$时间内完成匹配增广(见第 4.2 节末). 此结论是普适的, 即: 对左右侧顶点数分别为 L 和 R(并设 $R/L\geqslant 1$)的任意二部图, 均可将匹配增广时间精化至 $O(L^2)$(相应用匈牙利算法求最大匹配的时间为 $O(L^3)$). 下面回顾相关文献, 表明无类似结果可引用.

二部图最大匹配和匹配增广是多项式时间可解的经典问题, 其研究主要集中在早期. 1957 年, Berge 揭示了最大匹配与可增广性的关系[43], 表明一般图中的匹配最大, 当且仅当其不存在增广路. 1965 年, Edmonds 设计了求一般图最大匹配的匈牙利算法[44]. 它将一个初始匹配逐次增广至最大, 时间复杂度为 $O(v^4)$, 后改进至 $O(v^3)$[45]和 $O(ve)$[46]. 该算法包括 $O(s)=O(v)$次匹配增广. 在二部图上, 一次增广可通过广度或深度优先搜索完

成[47], 时间复杂度为 $O(e)=O(v+e)=O(v^2)$. 而在一般图上, 需以更复杂的方式标记和跟踪搜索过程, 才能得到同样的结果[45,46]. 1972 年, Hopcroft 和 Karp 将求二部图最大匹配的逐次增广过程归并为若干阶段, 每阶段基于一个原匹配进行广度优先搜索, 再对所得多层子图进行深度优先搜索, 找到原匹配的顶点不交的最短增广路的极大集, 由此得到比原匹配大 $l\geqslant 1$ 的新匹配, 作为下一阶段的基础. 其时间复杂度为 $O((v+e)s^{1/2})=O(v^{5/2})$,其中, $s^{1/2}=v^{1/2}$ 对应阶段数. 基于阶段归并思想, Even 和 Vazirani 对一般图匹配分别取得了 $O(v^{5/2})$[48]和 $O(ev^{1/2})$的结果[47,49]. 1991 年, Alt 通过融合相邻前/后阶段的深/广度优先搜索, 将 Hopcroft-Karp 方法的时间复杂度降至 $O(v^{3/2}(e/\log v)^{1/2})=O(v^{5/2}/(\log v)^{1/2})$[50], 这是至今最好的结果. 该算法包含 $O((e/v\log v)^{1/2})$个阶段的匹配增广, 每阶段需要 $O(v^2)$时间. 上述二部图算法主要是多次匹配增广的整体优化, 并未改进一次增广的时间, 也均未针对 R/L(或 $v/s=1+(v-s)/s\geqslant 1+R/L$)显著的情况, 给出只用参数 L 或 s 精化的结果(即完全用 L 或 s 代替 v, 或者用 L^2 或 s^2 代替 e).

通过从不同角度将图和匹配泛化, 二部图最大匹配问题有多种推广: 首先是带权图上的最大权匹配, 但其时间复杂度均未突破所谓 $O(ev^{1/2})$屏障[51,52]; 其次, 在带权或无权图上, 还可将求解目标推广为最大半匹配和最大 f-g 半匹配等, 而时间复杂度可达到 $O(ev^{1/2}\log v)$等[53,54], 变化不大. 在这些推广问题的研究中, 也均未见到针对显著 R/L 的时间复杂度精化.

特别地, 由于在线计算环境的兴起, 动态图结构上的匹配问题受到了重视. 其关注点是: 如何根据图结构的变化, 通过快速更新来维持最大匹配或某种泛化的匹配目标. 主要的图变化方式分为在线设置和完全动态设置: 前者可能每次增加一个顶点及其关联边[55,56], 或每次增加一条边[57]; 而后者允许边的增加和删除, 每次更新一条边[58,59]. 在相应更新最大匹配时, 至多计算一条匹配增广路[59], 并通过快速的近似或随机算法实现, 其时间复杂度可低至 $O(\log v)$[58]. 对此类算法的性能, 需要结合近似比、是否随机等指标进行综合度量. 本文研究的增量匹配计算也是一种动态匹配问题, 但其二部图的变化规律(搜索深入时, 有一个左侧顶点的邻域可能会缩小, 或者会增加一个左侧顶点及相应的邻域)不同于前述设置, 而且求匹配时不允许近似.

最后, 在凸二部图、平面图、间隔图等等特殊条件下求最大匹配, 可以优化其时间复杂度[60]. 但在二部图相关的优化结果中, 未见考虑 R/L 显著的情况.

2 预备知识

本节介绍 WSP 和模式回溯相关的基本概念.

定义 1. 工作流为一四元组$\langle S,U,A,C\rangle$, 其中,

- S, U 为步骤集和资源集, 记 $k=|S|$, $n=|U|$以及 $c=|C|$;
- A 为授权,定义为$\{A(u)\subseteq S|u\in U\}$或$\{N(s)\subseteq U|s\in S\}$, 其中: $A(u)$是资源 u 有权执行的步骤集, 而 $N(s)$是步骤 s 的授权资源集;
- C 为约束集, 每个 $c\in C$ 是一个二元组 $c=\langle T_c,\Theta_c\rangle$, 其中: $T_c\subseteq S$ 是约束的作用域, 而 Θ_c 表示所有满足约束 c 的对 T_c 中各步骤的资源分配.

定义 2. 资源分配$\pi.T\to U$ 为 $T\subseteq S$ 中每个步骤分配唯一的执行资源. $T=S$ 时, 称π是完全的. 若任取 $u\in U$ 都有$\pi^{-1}(u)\subseteq A(u)$, 则称$\pi$是授权的. 若任取 $x\in C$ 都有 $T_x\subseteq T$ 且π在 T_x 上的投影 $\pi|_{T_x}\in\Theta$, 则称π是合格的. 称授权且合格的资源分配是有效的, 完全且有效的资源分配是可行的.

定义 3. 模式 P 是 $T\subseteq S$的划分. $T=S$时, 称 P 是完全的. $T=\varnothing$时, 必有 $P=\varnothing$, 称 P 是空的. 称 $T=\cup P=\bigcup_{b\in P}b$ 为模式 P 的定义域. 任取 $s\in\cup P$, 将 s 在 P 中所属划分块记为 $b_P(s)$. 任何资源分配π都有模式 $P(\pi)=\{\pi^{-1}(u)\neq\varnothing|u\in U\}$. 若 $P(\pi)=P(\pi')$, 则称资源分配π和π'模式等价, 记为$\pi\sim\pi'$.

定义 4. 约束 $c=\langle T_c,\Theta_c\rangle$具有资源独立性, 是指若 $\pi|_{T_c}\in\Theta_c$ 且$\pi\sim\pi'$, 则 $\pi'|_{T_c}\in\Theta_c$. 可见: c 只要求资源分配π在 T_c 上的投影符合某种模式, 而不限制π为 T_c 中每个步骤具体分配哪个资源. 称一个模式满足 c, 当且仅当该模式的资源分配均满足 c.

设 C 中只含资源独立约束, 则模式 P 是合格的, 即其满足 C, 当且仅当任何 $P(\pi)=P$ 的资源分配π是合格

的[27]. 这为通过模式搜索 WSP 可行解提供了可能, 但还必须进行模式的授权验证.

定义 5. 给定模式 P, 其(授权)指派图 $G=G(P)$是一个二部图$\langle P,U,E\rangle$. 约定以 P, U 为左、右侧, 而 E 是从左到右的指派边集. 任取 $b\in P$ 和 $u\in U$, 若$\langle b,u\rangle\in E$, 则 $b\subseteq A(u)$. 任取 $b\in P$, 其在 G 中的邻域 $N_G(b)=\{u\in U|\langle b,u\rangle\in E\}$, 而其授权资源集 $N(b)=\{u\in N|b\subseteq A(u)\}$. 若 $N_G(b)=N(b)$, 称 b 是完全指派的. 若任取 $b\in P$ 都是完全指派的, 称 G 是完全指派的. 若任取 $b\in P$, 当 $N(b)\geqslant k$ 时, $|N_G(b)|=k$; 否则, $N_G(b)=N(b)$. 则称 b 是 k 指派的. 若任取 $b\in P$ 都是 k 指派的, 称 G 是 k 指派的.

文献[28]表明: 模式 P 是真实或授权的, 即存在使得 $P(\pi)=P$ 的授权资源分配π, 当且仅当 P 的完全或 k 指派图存在左(到右)完备匹配 $M:P\to U$. 对每个 $b\in P$, 将资源 $M(b)$分配至 b 中每个步骤, 即为授权资源分配.

指派图表示模块划分块和资源之间的多对多授权关系, 而左完备匹配为每个块分配唯一的执行资源. 故模式真实性验证的基本思路可归纳为先建立多对多指派, 再寻找一对一匹配.

3 动态模式空间理论

本节将给出动态模式空间的定义及有关性质, 为本文的最小增量模式回溯法奠定理论基础. 而在文献[28]中, 忽视了模式空间理论性质的研究, 并未给出搜索完备性等保证.

定义 6. 称模式 P'和 P 具有增元关系, 记作 $P'\succ P$, 当且仅当存在 $s\in b_{P'}\in P'$, 使$\cup P'\oplus\cup P=\{s\}$且:

$$P'\oplus P=\begin{cases}\{b_{P'},b_{P'}-\{s\}\}, & |b_{P'}|>1\\ \{b_{P'}\}=\{\{s\}\}, & |b_{P'}|=1\end{cases}.$$

将 P'和 P 分别称为增元模式和原模式. 将$\delta(P',P)=s$ 和 $b_{P'}$分别称为 P'(对 P)的增元和增块. 称 $b_{P'}-\{s\}$是 $b_{P'}$(在 P 中)的原块, 而 $P\cup\{\varnothing\}$是 P 的原块集.

以上用对称差来描述增元模式和原模式的差异, 而分别从两个方向来看, 有如下结论.

定理 1. 设模式 $P'\succ P$, $\delta(P',P)=s\in b_{P'}$, 则:

(i) 存在 $b\in P\cup\{\varnothing\}$, 使得 $b=b_{P'}-\{s\}$;

(ii) $P'-P=\{b_{P'}\}$;

(iii) $P-P'=\begin{cases}\{b\}, & |b_{P'}|>1\\ \varnothing, & |b_{P'}|=1\end{cases}$.

证明:

(i) 由 $s\in b_{P'}$知$|b_{P'}|\geqslant 1$:

若$|b_{P'}|>1$, 则 $P'\oplus P=\{b_{P'},b_{P'}-\{s\}\}$(定义 6). 由于 $b_{P'}\in P'$而 P'为划分, 必有$\varnothing\neq b_{P'}-\{s\}\notin P'$, 从而 $b_{P'}-\{s\}\in P-P'$. 于是, 存在 $b=b_{P'}-\{s\}\in P\subseteq P\cup\{\varnothing\}$;

若$|b_{P'}|=1$, 则 $P'\oplus P=\{\{s\}\}$(定义 6). 由于 $s\in b_{P'}\in P'$, 必有$\{s\}\in P'$(而若$\{s\}\in P$, 将与$\cup P'\oplus\cup P=\{s\}$矛盾). 又由于 P'为划分, 只能有 $b_{P'}=\{s\}$,从而 $b_{P'}-\{s\}=\varnothing\in P\cup\{\varnothing\}$;

(ii) 由 $b_{P'}\in P'\oplus P$ 且 $b_{P'}\in P'$可知 $b_{P'}\in P'-P$. 若存在 $b''\neq b_{P'}$使得 $b''\in P'-P\subseteq P'\oplus P$, 由定义 6, 只能是 $b''=b_{P'}-\{s\}$. 由于 $b_{P'},b''\in P'$而 P'为划分, $b_{P'}\cap(b_{P'}-\{s\})=\varnothing$, 从而 $b''=b_{P'}-\{s\}=\varnothing$, 与 b''为划分块矛盾, 于是 $P-P'=\{b_{P'}\}$;

(iii) 若$|b_{P'}|>1$, 由结论(i)的证明可知 $b_{P'}-\{s\}\in P'\oplus P$ 而 $b_{P'}-\{s\}\in P$, 故 $b_{P'}-\{s\}\in P-P'$, 且由 $b_{P'}\in P'$和 $b_{P'}\in P'\oplus P$ 知 $b_{P'}\notin P-P'$. 而 $P'\oplus P=\{b_{P'},b_{P'}-\{s\}\}$, 于是 $P-P'=\{b_{P'}-\{s\}\}=\{b\}$. 若$|b_{P'}|=1$, $P'\oplus P=\{\{s\}\}$, 由结论(i)的证明可知$\{s\}=b_{P'}\in P'$, 因此 $P'-P=\{\{s\}\}$, 而 $P-P'=\varnothing$.

证毕. □

定理 1(i)说明了增元模式相对原模式的最小差异, 即增块相对其原块, 只相差一个增元. 例如图 1 中, 增模式$\{\{s_1\},\{s_2,s_3\}\}$相对原模式$\{\{s_1\},\{s_2\}\}$的增块是$\{s_2,s_3\}$, 其原块是$\{s_2\}$, 两者只相差增元 s_3. 定理 1(ii)说明了增块是增元模式相对原模式的唯一增块. 定理 1(iii)则说明多元增块的原块是原模式中非空的划分块, 而一元增块的原块是为原模式附加的空块. 基于模式增元关系, 可描述进一步的增集关系, 并表明其对增长路径

的独立性.

定义 7. 增元关系的传递闭包称为增集关系, 记为>>. 若 $P'>>P$, 分别称 P'和 P 为增集模式和原模式. 称 $\Delta(P',P)=\cup P'-\cup P$ 为增集, 其任意排列为从 P 到 P'的增长路径.

定理 2. 设模式 $P'>>P$, 而 $s(t_1),s(t_2),\ldots,s(t_m)$是 P 到 P'的任意增长路径, 其中, $m=|\Delta(P',P)|$, 则必存在模式序列$(P_0=P),P_1,\ldots,(P_m=P')$, 使得 $P_i\succ P_{i-1}(1\leqslant i\leqslant m)$, 且$\delta(P_i,P_{i-1})=s(t_i)$.

证明: 从 P'中按 $s(t_m),\ldots,s(t_2),s(t_1)$顺序删除步骤及可能导致的空块, 得到模式序列$(P_m=P'),\ldots,P_1,P_0$. 对任意的 $1\leqslant i\leqslant m$, 存在 b'使得 $s(t_i)\in b'\in P_i$, $\cup P_i\oplus\cup P_{i-1}=\{s(t_i)\}$. 若$|b'|=1$, 即 $b'=\{s(t_i)\}$, 从 P_i中删除 $s(t_i)$将导致 b'变空被删除, 从而 $P_{i-1}=P_i-\{b'\}$, $P_i\oplus P_{i-1}=\{b'\}$. 若$|b'|>1$, 从 P_i中删除 $s(t_i)$将导致 b'变为 $b'-\{s(t_i)\}$, 从而 $P_i\oplus P_{i-1}=\{b',b'-\{s(t_i)\}\}$. 由定义 6 可知, $P_i\succ P_{i-1}$, 且$\delta(P_i,P_{i-1})=s(t_i)$. 删除完成后, $\cup P_0=\cup P$, $\Delta(P',P)=\Delta(P',P_0)$. 若能证明 $P_0=P$, 则 $P_0,P_1,\ldots,P_m$ 就是符合定理要求的序列.

反设 $P_0\neq P$, 必有 $s\in b\in P$ 且 $s\in b_0\in P_0$ 使 $b\neq b_0$. 不妨设 $s_0\in b_0$ 而 $s_0\notin b$. 又设 $s\in b'\in P'$, 则 $b_0\subseteq b'$和 $b\subseteq b'$, 且 $b'-b\subseteq\Delta(P',P)$, $b'-b_0\subseteq\Delta(P',P)$. 因 $s_0\in b_0\subseteq b'$但 $s_0\notin b$, 有 $s_0\in b'-b\subseteq\cup P'-\cup P$, 与 $s_0\in b_0\subseteq\cup P_0-\cup P$ 矛盾. 证毕. □

文献[28]直观描述了模式空间的概念. 例如图 1 为 $S=\{s_1,s_2,s_3\}$的模式空间, 其 $B(3)=5$ 片叶子对应于 S 的 5 种划分. 基于增元关系, 可将模式空间定义为如下树结构.

定义 8. 模式增长树Σ是以模式为结点、空模式$\varnothing$为根的树, 且满足以下条件.

(i) (单调性): 每个子结点与其父结点有增元关系;

(ii) (共域性): 任何兄弟结点(对父结点)的增元相同;

(iii) (正则性): 每个非叶结点 P 有$|P|+1$ 个子结点;

(iv) (完全性): 叶结点的层数为 $k=|S|$.

其中, 层数是结点到根的距离. 将Σ中所有结点的集合记为Ω, 第 $0\leqslant t\leqslant k$ 层结点记为Ω_t, 并将 P 的父结点记为 $p(P)$, 所有子结点记为 $Subs(P)$.

由于子结点对父结点的增元并不固定, 可在搜索过程中灵活选择, 故模式增长树是一个动态的模式空间. 而以下定理保证了其搜索完备性.

定理 3. 设 $P\in\Omega_t(0\leqslant t\leqslant k-1)$, $P'>>P$, $\cup P'=S$, 则 $P'\in\Omega_k$.

证明: 对 P 所在层数作归纳.

- 基始.

若 $P\in\Omega_{k-1}$, 则存在Ω中的模式序列$(P_0=\varnothing),P_1,\ldots,(P_{k-1}=P)$, 使得 $P_i\succ P_{i-1}(1\leqslant i\leqslant k-1)$. 由于每个$\delta(P_i,P_{i-1})\notin P_{i-1}$, 必有$\cup P=k-1$, 从而可设 $S-\cup P=\{s\}$. 任取 $Q\in Subs(P)$, 必有$\delta(Q,P)=s$. 由定义 8, $|Subs(P)|=|P|+1$. 又由 $P'>>P$ 和$\Delta(P',P)=S-\cup P$ 知 $P'\succ P$ 且$\delta(P',P)=s$. 由定理 1 知: 对应于 s 被加入 P 的哪个原块, P'也有$|P\cup\{\varnothing\}|=|P|+1$ 种可能, 从而必有 $P'\in Subs(P)\subseteq\Omega_k$.

- 归纳步骤.

假设$P\in\Omega_t(1\leqslant t\leqslant k-1)$时命题成立, 证明$P\in\Omega_{t-1}$时命题也成立. 此时, 由于 $P'>>P$, 根据定理 2, 必存在模式序列$(P_0=P),P_1,\ldots,(P_m=P')$使得 $P_i\succ P_{i-1}(1\leqslant i\leqslant m)$. 由于 $P_1\succ P$, 由定理 1 知, P_1 共有$|P|+1=|Subs(P)|$种可能. 而任取 $Q\in Subs(P)$, 有 $Q\succ P$. 从而只能有 $P_1\in Subs(P)\subseteq\Omega_t$. 由于也有 $P'>>P_1$, 由归纳假设知, $P'\in\Omega_k$.

综合基始与归纳步骤, 命题得证. 证毕. □

所有完全模式都从空模式增长而来. 根据定理 3, 它们都是模式增长树Σ的叶子. 因此, 模式增长树对完全模式搜索是完备的.

4 最小增量模式回溯法

文献[28]捕获了子模式具有唯一增块的特征, 而增块形成于父子模式的原子性差异, 即两者只相差一个增元(参见定义 6 和定理 1). 本文将利用这种最小差异表明: 在计算模式完全指派图时, 增块的候选邻点验证代价可降至 O(1), 而邻点搜索范围的实际规模可在 O(n)量级内极大收缩. 由此可将完全指派图的增量计算时

间从 O(kn)降至 O(n), 且实际性能极大提高. 不过, 由于完全指派图的边数为 O(kn), 相应的增量匹配代价为 O(kn), 弱于文献[28]的结果, 似乎会破坏整体优势的形成. 本文进而表明: 对授权资源数(即完全指派邻点数)至少为 k 的块, 允许其匹配资源空缺, 即有本文的 k 核心匹配概念(后文定义 9), 而完备 k 核(心匹配)的存在性与左完备匹配等价(后文推论 1). 当增块的邻点数至少为 k 时, 由搜索上下文可知完备 k 核存在, 无须增广. 而当增块邻点数小于 k 时, 由其出发对有关 k 核进行增广, 将只用到边数为 O(k^2)的子图, 故基于 k 核的增量匹配只需 O(k^2)时间. 找到一个模式可行解后, 先以 O(k^2)时间补全核心匹配, 再转换为真实解即可. 补全是一次性的, 不影响算法时间复杂度, 且实际代价极其微小. 下面将分指派和匹配两个阶段来介绍模式真实性验证的最小增量计算技术.

4.1 最小增量完全指派

本文不用文献[28]的 k 指派图, 而是面向完全指派图, 建立最小增量的计算方式. 其基本思路非常简明: 子模式的增块是将增元加入父模式的某原块(可能为空)而得, 故该原块的邻域(空原块的邻域可视为资源集 U)必包含增块邻域, 可从中搜索增块的所有邻点. 而此搜索范围的更替, 将导致比较明显的理论优势: 因为增块较其原块只多一个增元, 对每个原块邻点验证关于增元的授权, 便可判断是否增块邻点, 只需 O(1)时间, 相对于文献[28]每次验证候选点对增块中所有元素的授权, 降低了 O(k)倍. 特别地, 该方法的实际性能潜力很大: 原块邻域为块中各步骤授权资源集的交集, 很可能随着块的增大而迅速缩减, 导致增块邻点急剧富集, 以其代替文献[28]所用的 U 作为搜索范围, 有利于从少得多的候选找到一个邻点. 而由于单个邻点的搜索效率提高, 该方法相对文献[28]的优势也将随邻域规模而增长, 至少当完全邻域规模达到 k 时, 才可能出现转折(此时, k 邻域和完全邻域规模相同; 此后, 当完全邻域扩大时, k 邻域将不再扩大). 在具体描述和分析之前, 先给出其基本依据.

定理 4. 设 $P'\in Subs(P)$, $\delta(P',P)=s\in b_{P'}$, $b=b_{P'}-\{s\}$, $G'=G(P')$和 $G=G(P)$完全指派.

(i) 若$|b_{P'}|>1$, 则任取 $u\in U$, 有 $u\in N_{G'}(b_{P'})$当且仅当 $u\in N_G(b)$且 $u\in N(s)$;

(ii) 若$|b_{P'}|=1$, 则任取 $u\in U$, 有 $u\in N_{G'}(b_{P'})$当且仅当 $u\in N(s)$.

证明:

(i) 当$|b_{P'}|>1$ 时, 由定理 1(i)知 $b\in P$ 且非空. 任取 $u\in U$, $u\in N_{G'}(b_{P'})$当且仅当 $u\in N(b_{P'})$(完全指派), 当且仅当 $b_{P'}\subseteq A(u)$(定义 5), 当且仅当 $b\subseteq A(u)$且 $s\in A(u)$, 当且仅当 $u\in N(b)$且 $u\in N(s)$(定义 5), 当且仅当 $u\in N_G(b)$且 $u\in N(s)$ (G 完全指派);

(ii) 当$|b_{P'}|=1$ 时, 由于 $s\in b_{P'}$, $b_{P'}=\{s\}$. 再由 G'完全指派和定义 5, 结论显然.

证毕. □

设父子模式 P 和 P', 增元为 s, $b=b_{P'}-\{s\}$. 且 P 的完全指派图 $G=G(P)$已知. 为计算 P'的完全指派图 $G'=G(P')$, 由定理 1(ii)、定理 1(iii), 只需在 $G-\{b\}$基础上(从图中删除顶点, $b\neq\varnothing$时, 会将块 b 及其关联边从 G 中删除; 而 $b=\varnothing$时, 由于 b 不是 G 中顶点, 该操作对 G 没有影响), 补充计算增块 $b_{P'}$的邻域 $N_{G'}(b_{P'})$, 生成相应的关联边. 根据 $b_{P'}$的大小, 分两种情况计算.

- 情况 1. 若$|b_{P'}|=1$, 即 $b_{P'}=\{s\}$, 由定理 4(ii), 置 $N_{G'}(b_{P'})\leftarrow N(s)$即可, $N(s)=\{u\in U|s\in A(u)\}$;
- 情况 2. 若$|b_{P'}|>1$, 即 $b=b_{P'}-\{s\}\in P$, 则由定理 4(i), 先置 $N_{G'}(b_{P'})\leftarrow\varnothing$, 然后对每个 $u\in N_G(b)$循环, 若 $u\in N(s)$, 置 $N_{G'}(b_{P'})\leftarrow N_{G'}(b_{P'})\cup\{u\}$, 循环结束即得完整的 $N_{G'}(b_{P'})$.

由于 $u\in N(s)$的判断可根据预计算的授权关系矩阵以 O(1)时间完成, 上述 $N_{G'}(b_{P'})$计算的时间复杂度为 O(n), 较文献[28]的 O(kn)时间降低了一个因子. 下面表明此结果可以扩展到整个模式搜索中.

模式回溯有 3 种搜索操作: 从父到子深入(发生条件: 父结点模式验证通过)、从兄到弟切换(发生条件: 兄结点验证失败或上一步回溯到兄结点, 这里按搜索先后区分兄弟关系)以及从子到父回溯(发生条件: 子结点验证失败或上一步回溯到子结点, 且其无弟结点). 根据当前结点的验证结果、相对位置(是否存在子结点、弟结点)和上一步操作(是否被回溯), 将唯一决定下一步操作. 当某个叶结点通过验证时, 即可根据对应的完全模式和左完备匹配求出一个可满足解, 停止搜索. 或当根结点被回溯时, 搜索无法继续而停止, 此时必无解

(非有解停止).

在整个搜索中, 用全局唯一变量 B 来保存真实性验证使用的(完全)指派图, 通过恰当的备份和恢复, 可在每个搜索结点处, 以 $O(n)$时间将 B 更新为该结点的指派图. 具体方法是:

(1) 从父到子深入时, 若 B 为父(结点)指派图, 按前述分析, 可在 $O(n)$时间内将其更新为子指派图. 更新只对子结点增块的邻域进行, 更新前需将其原块的邻域备份, 也只需 $O(n)$时间, 不影响更新的整体时间复杂度;

(2) 从兄到弟切换时, 若 B 为兄指派图, 则先利用从父深入到兄时的备份, 将其恢复为父结点完全指派图并释放备份, 只需 $O(n)$时间; 然后将 B 更新为弟指派图, 仍为一从父到子的深入过程, 如前述, 只需 $O(n)$时间;

(3) 从子到父回溯时, 若 B 为子结点指派图, 相对于父结点指派图, 只在增块邻域上不同, 利用深入时的备份, 可在 $O(n)$时间内将增块邻域恢复为原块邻域, 并释放备份, 由此将 B 恢复为父结点指派图.

搜索从对应于空模式的根结点开始, 此时 B 被设为空指派图. 然后, 根据上述 3 种情况的分析, 整个搜索可以正确衔接, 连续进行, 从而对每个搜索结点, 保持 $O(n)$的完全指派图更新时间.

在资源服务化环境中, 资源配比μ很大. 给定步骤数 k, 文献[28]从大小为 $n=\mu k$ 的 U 中搜索邻点, 会严重影响指派计算和模式搜索性能. 设每个步骤平均有 n/Δ个授权资源, 当增块大小为β时, 本文从原块邻域中搜索增块邻点, 其范围大小平均为 $n/\Delta^{\beta-1}$, 可达到$\Delta^{\beta-1}$倍的收缩. 即使底数Δ取值不太大, 该幂指函数也会产生很强的收缩作用, 对提高增量指派计算的性能极为有利, 但其也伴随着如下限制.

(1) 若增块存在邻点, 则Δ越大, 即授权比例 $1/\Delta$越小, 从原块邻域中找出增块的邻点越快, 对本文方法越有利. 然而给定 k, 即使对较高的资源配比 n/k, 极小的授权比例也可能使各步骤的授权资源集大为缩减, 相应得到很小的解空间, 用回溯法+约束传播可高效搜索, 此时模式空间规模反而可能更大. 故作为模式回溯法的优势条件, 高资源配比隐含着授权比例不太低的要求;

(2) 设增块大小为β, 为找到它的一个邻点, 本文较文献[28]在搜索范围上收缩了$\Delta^{\beta-1}$ 倍, 且每个候选邻点的验证代价降低 $O(k)$倍, 从而在共同计算邻点集上具有 $O(k\Delta^{\beta-1})$倍的优势. 但是, 本文要对每个块进行完全指派, 相对于文献[28]的k指派, 可能会计算更多的邻点, 每个额外的邻点需要$O(\Delta)$的代价(在原块邻域中搜索, 为找到一个增块邻点平均要检查 $O(\Delta)$个候选点, 每次的检查代价为 $O(1)$). 为防止代价抵消优势, 额外计算邻点集不应超过共同计算邻点集的 $O(k\Delta^{\beta-1})$–1 倍. 本文指派计算方法较文献[28]的性能优势不是绝对的, 然而对稍大的块, 除非资源配比极大, 该倍数要求已经很难突破.

4.2 最小增量完备匹配

本文的最小增量匹配计算可在边数为 $O(kn)$的完全指派图上取得 $O(k^2)$的匹配时间复杂度, 其核心思想是: 将匹配简化为所谓 t 核($|P'|\leqslant t\leqslant k$), 并以完备 k 核为计算目标. 相关概念定义如下.

定义 9. 设完全指派图 $G'=\langle P',U,E'\rangle$, $|P'|\leqslant t\leqslant k$. 任取 $b\in P'$, 称其邻点数超标, 当且仅当$|N(b)|=|N_{G'}(b)|\geqslant t$. 称 G'上的匹配 m 为 t 核心匹配, 简称 t 核, 当且仅当 m 只饱和邻点数未超标的块, 或不存在$\langle b,u\rangle\in m$ 使得 b 是邻点数超标的. 设 m 为 t 核, 则称其是最大或完备核, 当且仅当$|m|=|\{b\in P'||N_{G'}(b)|<t\}|$, 即 m 饱和所有邻点数不超标的块.

在完全指派图 $G'=\langle P',U,E'\rangle$上, 给定一个 t 核 m, 若存在邻点数超标的块 b, 即$|N_{G'}(b)|\geqslant t$, 则其未被 m 所饱和, 但在其邻域 $N_{G'}(b)$中检查前 t 个资源, 必能找到空闲资源(m 至多占用 $N_{G'}(b)\subseteq U$ 中的$|m|$个资源, 由于$t\geqslant|P'|>|m|$, 故 $N_{G'}(b)$中任意 t 个资源都有未被 m 匹配者), 从而将 m 扩展为一个非核的匹配. 若存在更多的未饱和超标块, 重复上述操作, 可得更大的非核匹配. 反之, 给定一个非核的匹配, 不论其饱和了多少个超标块, 总可将相关匹配边都删除而得一 t 核. 于是, 在核与一般匹配之间有着某种等价性, 有可能用 t 核概念来简化匹配, 降低计算时间. 基于这一思想, 本文得到如下结论.

定理 5. 完全指派图 $G'=\langle P',U,E'\rangle$存在左完备匹配, 当且仅当其存在完备 t 核.

证明:

- 必要性.

设 M_f 为 G'的左完备匹配, 并记 $M=M_f-\{\langle b,u\rangle\in M_f|b\in P'\wedge|N_{G'}(b)|\geqslant t\}$, 则 M 是 G'的 t 核, 且$|P'|-|M|=|M_f|-|M|=\{\langle b,u\rangle\in M_f|b\in P'\wedge|N_{G'}(b)|\geqslant t\}=|\{b\in P'||N_{G'}(b)|\geqslant t\}|$, 从而 M 是完备 t 核.

- 充分性.

设 M 为 G'的完备 t 核. 由二部图匹配的 Hall 定理, 要证 G'存在左完备匹配, 可证任取 $Q\subseteq P'$, $|N_{G'}(Q)|\geqslant|Q|$, 其中, $N_{G'}(Q)=\bigcup_{b\in Q}N_{G'}(b)$. 令 $P'_c=\{b\in P'\mid\exists u\in U,\langle b,u\rangle\in M\}$, 表示被 M 匹配了资源的块集, 则任意 $Q\subseteq P'$必为如下情况之一.

(1) $Q\subseteq P'_c$, 此时$|N_{G'}(Q)|=|\{u\in U|\exists b\in Q,u\in N_{G'}(b)\}|\geqslant|\{u\in U|\exists b\in Q,\langle b,u\rangle\in M\}|=|Q|$;

(2) $Q\cap(P'-P'_c)\neq\varnothing$, 即存在 $b\in Q\cap(P'-P'_c)$. 由定义 9 可知, 任取 $b\in P'-P'_c$, 均有$|N_{G'}(b)|\geqslant t$. 于是, 存在 $b\in Q$, 使得$|N_{G'}(b)|\geqslant t$, 从而$|N_{G'}(Q)|\geqslant|N_{G'}(\{b\})|\geqslant t\geqslant|P'|\geqslant|Q|$(注意定义 9 中 t 的取值范围).

两种情况下均有$|N_{G'}(Q)|\geqslant|Q|$. 充分性得证.

证毕. □

在定义 9 中取 $t=k$, 可保证对任何 P'均有 $t\geqslant|P'|$, 有助于利用父子模式的关系进行增量计算. 相应地, 由定理 5 易得如下推论.

推论 1. 设完全指派图 $G'=\langle P',U,E'\rangle$, 则 G'存在左完备匹配当且仅当其存在完备 k 核.

为便于描述和分析完备 k 核的增量计算方式, 先给出 t 核增广的如下性质.

定理 6. 完全指派图 $G'=\langle P',U,E'\rangle$, m 是 G'的 t 核, 且存在未被 m 匹配资源的块 $b\in P'$, 使得$|N_{G'}(b)|<|P'|\leqslant t$, 则从 b 出发对 m 进行增广, 仍得一 t 核, 而增广过程的时间复杂度为 $O(t^2)$.

证明: m 是二部图 G'的左到右不完备匹配, 求其增广路具有标准的搜索算法与代价. 无论深度还是广度优先搜索, 均可归纳为两个方向的操作: 在一个左侧顶点的邻域中, 搜索到一个未被匹配 m 所饱和(且之前搜索未到达过)的右侧顶点; 或者从一个已被饱和的右侧顶点跳到它的左侧匹配点. 从右到左操作总代价为 $O(|m|)$; 而从左到右操作总代价为 $O\left(\sum_{d\in P'}|N_{G'}(d)|\right)=O(|E'|)=O(kn)$, 这里, $|E'|=\sum_{d\in P'}|N_{G'}(d)|$ 是总边数, 也是所有左侧顶点的关联边数(即邻域大小)之和. 可见: 决定整体代价的主要是从左到右操作, 而相应的左侧顶点要么是出发点, 要么是初始匹配所饱和的点.

在本定理条件下, 增广出发点 b 的邻域小于 t, 由于初始匹配 m 为 t 核, 故其饱和的左侧顶点的邻域也一定小于 t, 从 b 出发搜索到的 m 增广路不可能经过邻点数超标的左侧顶点, 于是对增广路进行匹配边和非匹配边翻转后, 所得匹配的左侧顶点也都是授权资源数不超标的, 即增广后仍为 t 核. 而增广过程中, 从左到右操作的总代价为 $O\left(\sum_{d=b\vee(\exists u\in U,\langle d,u\rangle\in m)}|N_{G'}(d)|\right)=O(|P'|t)=O(t^2)$. 证毕. □

对定理 6, 还可换一角度理解, 即在完全指派图 G'中, 从邻点数不超标的块出发求 t 核的增广路, 实际上只用到了其 $O(t^2)$规模的子图 $G'-\{d\in P'||N_{G'}(d)|\geqslant t\}$. 而$\{d\in P'||N_{G'}(d)|\geqslant t\}$中的左侧顶点既非出发点, 也不被 t 核所饱和, 将不会被搜索到, 连同其关联边都可以忽略. 例如, 图 2 左端给出的 G'有一个 $t=4=|P'|$核 $m=\{\langle b_2,u_2\rangle,\langle b_3,u_3\rangle\}$(虚线标出), 但 m 没有给邻点数未超标的块 b_1 匹配资源, 故非完备核, 可从 b_1 出发对 m 进行增广. 求增广路的广度优先搜索过程如图 2 中间所示, 由此得到一条 m 增广路 $b_1u_3b_3u_6$, 通过翻转其中的匹配与非匹配边, 可将 m 扩展为完备 t 核$\{\langle b_1,u_3\rangle,\langle b_2,u_2\rangle,\langle b_3,u_6\rangle\}$, 如图 2 右端图的虚线边所示. 不难看出: 求增广路的过程与邻点数超标的块 b_4 及其关联边没有关系, 实际上是在子图 $G'-\{b_4\}$中进行的. 该子图中, 所有块的邻域都小于 t, 而其块数不超过$|P'|=4=t$, 故其总边数为 $O(t^2)$, 而求增广路的代价也被控制在同一量级. 由于该子图也是 k 指派图的子图, 故其上匹配增广的实际性能也不弱于文献[28].

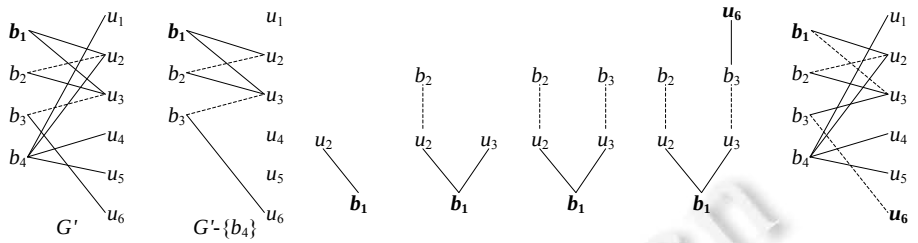

图 2 t 核心匹配的增广示例

下面给出完备 k 核的增量计算方法及其代价分析. 设父子模式 P 和 P'的完全指派图分别为 G 和 G', 增元为 s, 增块为 $b_{P'}$, 其原块为 $b=b_{P'}-\{s\}$, 且 G 存在完备 k 核 M, 则由于 G 和 G'的差异仅由增元 s 引起, 可增量计算子模式 P'的完备 k 核 M', 并判断其完备与否. 其主要工作是求增块 $b_{P'}$的匹配资源, 有如下两种情况.

- 情况 1. 若$|N_{G'}(b_{P'})|\geqslant k$, 则 M'不为 $b_{P'}$匹配资源. 此时, 若 $b=\varnothing$, 即 $b_{P'}=\{s\}$, 则 $G=G'-\{b_{P'}\}$, 故 G 的完备 k 核 M 也是 G'的完备 k 核. 而若$\varnothing\neq b\in P$, 则 $G'-\{b_{P'}\}=G-\{b\}$, 且由于$|N_G(b)|=|N(b_{P'}-\{s\})|\geqslant|N(b_{P'})|=|N_{G'}(b_{P'})|\geqslant k$, 故 M 并没有给 G'较 G 缺少的左侧顶点 b 匹配资源, 从而 M 也是 G'中的匹配, 且为 k 核. 又 G'较 G 新增的 $b_{P'}$邻域大于 k, 故 M 是 G'的完备核. 两种情况下, 均只需取 $M'\leftarrow M$, 即得 G'的完备 k 核, 其计算时间复杂度为 O(|M|)=O(k);
- 情况 2. 若$|N_{G'}(b_{P'})|<k$, 则 M'需要为 $b_{P'}$匹配资源. 此时, 由 M 是 G 的完备 k 核, 不难得出:

$$m=\begin{cases}M, & b_{P'}=\{s\}\\ M-\{\langle b,M(b)\rangle\}, & \varnothing\neq b\in P\end{cases}\text{是 }G'-\{b_{P'}\}=\begin{cases}G, & b_{P'}=\{s\}\\ G-\{b\}, & \varnothing\neq b\in P\end{cases}\text{的完备 }k\text{ 核.}$$

其计算时间复杂度为 O(|M|)=O(k). 在 G'中, 以 m 为初始匹配, 从 $b_{P'}$出发进行一次增广, 其成功与否, 即表明了 G'是否存在完备核心匹配. 特别地, 由于 m 是 k 核, 其增广可以 O(k^2)时间完成. 此外, 当$\varnothing\neq b\in P$ 时, 如果 $M(b)=M(b_{P'}-\{s\})\in N(s)$, 则因 $M(b)\in N_G(b)=N(b)$, 也有 $M(b)\in N(b_{P'})=N_{G'}(b_{P'})$. 此时置$m\leftarrow m\cup\langle b_{P'},M(b)\rangle$, 即为 G'的完备 k 核, 只须 O(1)时间. 总体上, 情况 2 仍需要 O(k^2)时间.

当找到一个模式可行解时, 必然伴随得到一个完全指派图及其完备 k 核. 对该核进行一次性补全, 可得左完备匹配. 此时, 有 O(k)个块需要补充匹配资源, 而每个这样的块都有不少于 k 个授权资源(也就是完全指派图中的邻点), 而其他块至多匹配其中 k–1 个, 故至多搜索 k 个即可找到空闲. 从而, 匹配补全的时间复杂度为 O(k^2).

上述最小增量匹配以(搜索上下文直接给出或略加修改的)k 核为初始匹配, 且增广时, 仅从邻点数未超标的左侧顶点出发. 实际上, 给定边数为 O(kn)的完全指派图 G 及任意初始匹配 m, 从任意 m 未饱和的左侧顶点 b 出发, 均可以 O(k^2)时间完成一次增广, 具体分为 3 步.

(1) 先将 m 中邻点数超标($\geqslant k$)的左侧顶点全部解除匹配, 得到一个 k 核, 需 O(k)时间;

(2) 若 b 邻点数超标, 跳过本步; 否则由 b 出发, 对上一步所得 k 核做一次增广, 由定理 6 知, 其需 O(k^2)时间;

(3) 若存在第(1)步解除匹配的左侧顶点或第(2)步跳过的出发点, 设其集合为 $L\subseteq L(G)$, 对 L 中的左侧顶点进行匹配补全. 任取 $d\in L$, 其邻域至少包含 k 个资源, 而第(2)步得到的 k 核至多占用$|L(G)|-|L|\leqslant k-1$ 个资源, 故 d 的补全至多需要 O(k)时间, 而整个 L 的补全需要 O(k^2)时间(注意: 当 L 有多个顶点时, 前一个顶点的补全会扩大匹配, 但只要有待补全的后一个顶点, 现有匹配占用的资源就不可能达到 k 个).

最终, 所得匹配将比 m 大 1, 且饱和了 b. 因这一增广方式未利用搜索上下文, 实际性能偏弱, 故不将其用于本文算法的正式描述.

4.3 算法描述与分析

本节将对最小增量模式回溯算法进行形式化描述和时间、空间复杂度分析. 如下算法 1 给出了 MIPB 的

框架:

算法 1. *MIPB*(P,&B,M) //递归过程.

输入: 模式 P; 共享的二部图变量 B=G(P), 为完全指派图, 对 $b\notin P$, $N_B(b)=\varnothing$; 匹配变量 M=M_B, 其中, M_B:$P\to U$ 是 B 的完备 k 核, 对 $b\notin P$ 或未被 M_B 匹配的 $b\in P$, $M_B(b)$=*nil*;

输出: 返回*WS=⟨S,U,A,C⟩的一个其模式增长(扩展)于 P 的可行解; 若不存在, 返回 *nil*.

```
1.  if (|∪P|=|S|) {   //P 已经是所有步骤的划分
2.      M←maximize(M);   //用第 4.2 节所述方法将完备 k 核补全为左完备匹配
3.      return M∘P;   //完全模式 P 为 S→P 映射, 可将任意步骤映射到其分块, 再由 M 将块映射为资源
4.  } else {
5.      select s∈S−∪P;   //为增强剪枝, 按一定变量排序规则选择下一个步骤
6.      计算将 s 加入 P 形成的合格模式集 EligibleSubs(P,s);
7.      while (EligibleSubs(P,s).hasNext(·)) {   //P 有未搜索的子模式
8.          P′←EligibleSubs(P,s).getNext(·);   //跳到下一个未搜索子模式
9.          update(B,P′);   //按最小增量方式将 B 更新为完全指派图 G(P′), 并备份更新前的邻域
10.         if (|N_B(b_P′)|<k) {   //第 4.2 节情况 2: 增块邻点数未超标, 需求 G(P′)初始匹配 m 并增广
11.             if (|b_P′|>1) {   //增块 b_P′非全新, 其原块 b_P′−{s}非空
12.                 m←M;
13.                 if (m(b_P′-{s})∈N(s))   //借助预计算的授权矩阵, 此条件判断只需 O(1)时间
14.                     m(b_P′)←m(b_P′−{s});   //沿用原块在 G(P)中的匹配资源
15.                 m(b_P′−{s})←nil;   //解除原块的匹配
16.                 if (|m|=|M|) M′←m;   //沿用成功时, 无须再对初始匹配进行增广
17.             } else {m←M;}   //增块全新时, 原块为空, 无须解除匹配
18.             if ((|b_P′|>1∧|m|<|M|)∨|b_P′|=1)   //增块非全新, 且不可沿用原块匹配点, 或增块全新
19.                 M′←augment(B,m,b_P′);   //在 G(P′)中, 从 b_P′出发求 m 的增广路, 以求完备 k 核 M′
20.             if (|M′|=|m|) continue;   //增广失败则真实性验证失败; 否则, M′已是 G(P′)的完备 k 核
21.         } else {   //第 4.2 节情况 1: 增块邻点数超标, 真实性验证必通过, 完备 k 核不变, 无须增广
22.             M′←M;   //全新增块, 或原块也因邻点数超标而省略匹配, 均有 M(b_P′)=nil
23.         }
24.         π←nil;
25.         π←MIPB(P′,B,M′);   //M′是 B=G(P′)的完备 k 核, 符合调用输入要求
26.         if (π≠nil) return π;
27.         restore(B);   //利用备份, 将 B 恢复为 G(P)
28.     }   //while
29.     return nil;
30. }
```

算法 1 用一个全局唯一变量 B 来存放每个搜索模式的完全指派图, 由第 9 行的 *update*(·)负责更新和备份, 可展开为如下代码.

```
9.1.  assert(N_B(b_P′)=∅);
9.2.  if (|b_P′|>1) {   //将完全指派图 G=G(P)更新为完全指派图 G′=G(P′)
9.3.      for (u∈N_B(b_P′−{s})∧u∈N(s)) N_B(b_P′)←N_B(b_P′)∪{u};   //由预计算矩阵判断 u∈N(s), 耗时 O(1)
9.4.      N_back←N_B(b_P′−{s});   //备份 b_P′原块的邻域, 以便回溯时恢复
9.5.      N_G(b_P′−{s})←∅;   //b_P′−{s}∉P′, 故置其邻域为空
```

9.6. } **else** { //$|b_{P'}|$=1, 为全新块, 且只含 s 一个步骤

9.7. $N_B(b_{P'})\leftarrow N(s)$; //$N(s)$在搜索之前预计算, 直接复制即可

9.8. }

由于第 9.3 行的 $N_G(b_{P'}-\{s\})$和第 9.7 行的 $N(s)$均为 O(n)规模, 在最坏情况下, 上述代码可能检查处理 O(n)个值, 但每次只需常数时间, 故总代价为 O(n).

指派图更新后, 算法 1 直接在其上求匹配, 以验证 P'的真实性. 若验证失败, 要切换到弟结点; 或虽通过, 但因子树搜索无解又回溯到本结点, 则需将 B=G'恢复为 B=G. 由第 27 行的 *restore*(·)负责, 可展开为如下代码:

27.1. $N_B(b_{P'})\leftarrow nil$; //清空增块的邻域

27.2. $N_B(b_{P'}-\{s\})\leftarrow N_{back}$; //利用备份, 恢复相应原块的邻域

易知, 其时间代价为 O(N_{back})=O(n).

对算法 1 的初始调用置 $P\leftarrow\varnothing$, $B\leftarrow\varnothing$, $M\leftarrow\varnothing$. 在最坏情况下, 算法 1 可能遍历模式空间, 故其时间复杂度仍为 O*($B(k)$), 其中, $B(k)$为第个 k 贝尔数, 反映模式空间的规模. 第 9 行和第 27 行对应于单结点处的最小增量指派代价, 共需 O(n)时间. 第 10 行−第 23 行对应于单结点处的最小增量匹配代价, 共需 O(k^2)时间, 其中: 第 11 −17 行和第 20 行共需 O(k)时间, 第 19 行需 O(k^2)时间, 第 22 行需要 O(k^2)时间. 从而更准确的时间复杂度为 O(($f(c)+k^2+n)B(k)$), 其中, $f(c)$是单结点处的约束验证代价, 取决于约束类型及其验证实现. 相对于文献[28]中 IPB 的 O(($f(c)+k^2+kn)B(k)$)时间复杂度, 本文的 MIPB 算法具有理论优势.

算法 1 的空间代价主要来自问题实例和部分关键数据结构的全局存储. 其中, 授权关系和全局唯一指派图实例 B 占用 O(kn)空间. 约束关系的存储代价与约束类型有关, 对 c, 不妨设其占用的空间为 $g(c)$, 通常是约束数量 c 的多项式函数. 而邻域备份和匹配关系在搜索深入时分配, 搜索切换或回溯时释放, 因递归深度不超过 k, 相应的空间复杂度为 O(kn). 于是, 算法 1 的整体空间复杂度为 O($kn+g(c)$).

4.4 算法实现方式

算法 1 的第 9−23 行的模式真实性验证包括指派和匹配两个环节, 其实现要点是:

(1) 在计算块的指派邻点时, 需要判断资源是否有权执行步骤, 可借助预计算的 $k\times n$ 授权关系矩阵完成, 每次判断只需 O(1)时间. 对于新增的一元块, 直接从其唯一步骤的授权资源集中取邻点即可;

(2) 采用匈牙利算法在二部图上进行匹配增广, 若原(或初始)匹配 m 为空, 将从头求出整个左完备匹配. 本文采用该算法的深度优先形式, 但对于增块(增广的出发点), 先对其邻域进行一次局部广度搜索. 对初始匹配 m, 用块号(左侧顶点)为索引、资源号(右侧顶点)为元素的数组来存放. 为保证从右侧饱和点访问其匹配点的操作能以 O(1)时间完成, 为 m 建立反向索引, 用资源号为索引、块号为元素的数组 rm 来存放, 块 $rm[i]\geqslant 0$ 被 m 匹配到资源 i, 而 $rm[i]<0$ 表示 i 不是饱和点. 由于 rm 大小为 n, 在高资源配比下可能大于 k^2, 为了实现第 4.2 节所述的 O(k^2)匹配增广时间, 反向索引的创建时间不能与 n 相关. 为此, 注意到不同搜索结点处的匹配验证彼此独立, 将 rm 数组设置为全局或静态存储, 一次性完成分配, 并初始化为全−1. 每次匹配验证时, 遍历 m 一趟完成 rm 的设置, 并将 $rm[i]\geqslant 0$ 的 i 值备份到一个规模不超过 O(k)数组中. 本次匹配结束后, 遍历备份数组一次, 将 rm 恢复为全−1, 等待下一次使用. 如此循环. 备份数组也可设置为全局存储, 并初始化为空, 在每次恢复 rm 后清空, 等待下次备份使用. 上述方式保证了本文最小增量匹配的 O(k^2)时间复杂度, 且当 n 非常大时, 有利于优化实际性能.

算法 1 的第 6 行计算合格子模式集合时, 需进行模式上的约束验证, 其主要思路是:

(1) 每个子模式由增元 s 进入父模式中不同的块(或另外的空块)形成. 采用文献[31]的编码方式将模式向量化, 每个块对应一个整数编码, 而当前模式 P'各块的编码必为一段连续整数 1~$|P'|$, 最大可取 k. 用一个大小为 k 的 bool 数组表示 s 可进入的每个块, 初始化为全 true;

(2) 第 6 行只检查增元 s 参与的约束即可, 且只检查一趟, 若一条约束禁止 s 进入某个块, 将该块对应的数组元素置为 false.

约束检查的方式依赖于约束类型. 常见的互斥约束很容易验证, 只需 O(1)时间. 在工作流应用中, 另一类常见的约束是值势, 又分为 at-most-r 和 at-least-r 这两种. $|T_x|\geqslant r$ 元 at-most-r 和 at-least-r 分别要求 T_x 中步骤最多/最少由 r 个不同资源执行. 下面给出其快速验证方法: 为每条约束 x 关联一个全局整型数组 $ref(x,1\ldots k)$, 在搜索前初始化为全 0 值, 其中, $ref(x,i)$表示当前 x 的约束变量有几个被赋予(块编码)值 i, 并为每条约束 x 附加一个初始化为 0 的全局计数变量$\#diff(x)$, 表示 T_x 当前被赋不同值的个数, 也就是 T_x 中的步骤当前分配的不同资源的个数(注意: 每个编码值对应一个模式划分块, 而不同块中的步骤由不同资源执行). 在搜索深入或(从兄弟结点)切入到当前模式时, 会把增元 s 加入编码为 i 的块中(变量 s 此前未被赋值). 此时, 对 s 所参与的每条值势约束 x_s, 置 $ref(x_s,i)\leftarrow ref(x_s,i)+1$, 表示赋值为 i 的约束变量数增 1. 之后, 若 $ref(x_s,i)=1$, 置$\#diff(x_s)\leftarrow \#diff(x_s)+1$, 表示各约束变量所赋的不同值个数增 1. 而在搜索切出(到兄弟结点)或回溯前进行反向维护, 即对每个 x_s, 置 $ref(x_s,i)\leftarrow ref(x_s,i)-1$. 之后, 若 $ref(x_s,i)=0$, 置$\#diff(x_s)\leftarrow \#diff(x_s)-1$. 在每个搜索结点处, 每条约束的维护耗时 O(1). 而验证 x_s 时, 可直接比较$\#diff(x_s)$与 r, 只需 O(1)时间. 总体上, 单条值势约束的验证时间为 O(1).

算法 1 第 5 行的变量排序规则也依赖于约束配置. 例如, 文献[28]对以上 3 种约束的混合配置设计了启发式排序规则(对每个候选步骤计算一个优先级并取最大者), 可有效增强搜索剪枝, 且不影响整体时间复杂度.

5 实验研究

5.1 数据生成与测试方法

文献[28]配置互斥及 5 元 at-most/least-3 约束, 研究了资源独立 WSP 随机相变实例的生成模型. 互斥是最常见的工作流约束, 而值势是常见和主要的资源独立性约束[10,28,31,42], 故上述约束配置有很好的代表性. 该模型的实例规模由步骤数 k、资源数 n、互斥约束数 e 和 at-most-3/at-least-3 约束数γ共 4 个参数衡量. 由于 WSP 研究关注以 k 为小参数的固定参数性能, 应进一步选择(k,n)空间中 k 较小的区域. 该模型从变化 k 和变化 n 两个维度对参数空间切片: 前者保持 n/k 而改变 k; 后者固定 k, 令 $n\geqslant k$ 并极度增大. 由于 at-most-3/at-least-3 等复杂约束相对少见, 故仅取$\gamma=k$, 随 k 线性变化. 给定(k,n,γ)后, 取 $e=e_{50}$, 该点处有/无解实例各占约 50%, 处于欠/过约束的临界状态, 称为相变点. 相变实例难以求解, 有利于衡量 WSP 算法的能力, 但相变点测量非常耗时. 对变化 k 维度, 文献[28]在 $n=10k$ 和 $n=100k$ 的高资源配比下, 分别取 $18\leqslant k\leqslant 58$ 和 $18\leqslant k\leqslant 55$, 测定了一系列 e_{50} 值. 而其变化 n 维度取 $k=18$, 将资源配比从 2 推高至 524 288, 相应测定了 e_{50}, 本文又增加了 $k=36$ 的系列. 由参数(k,n,e,γ)生成单个实例的方法是: 取 $S=\{0,1,\ldots,k-1\}$, $U=\{0,1,\ldots,n-1\}$. 为反映各资源掌握业务技能种类的差异, 对每个 $u\in U$, 从$\{1,2,\ldots,\lfloor k/2\rfloor\}$中随机取值作为$|A(u)|$, 然后从 S 中随机不重复地选择$|A(u)|$个元素, 得到 $A(u)$. 这将导致约 1/4 的授权比例, 即每个步骤平均有 $n/4$ 个授权资源. 若从$\{1,2,\ldots,k\}$中取$|A(u)|$, 将导致约 1/2 的授权比例, 为其最大可能值. 但工作流每个步骤都有其专业技能要求, 较少出现每个资源都能执行大部分步骤的情况. 又如第 4.1 节所指出: 高资源配比是模式回溯法的优势条件, 但授权比例不能太低. 本模型 1/4 的授权比例无论在应用意义还是技术条件上都有其代表性. 随机不重复地选取 e 对步骤, 即得互斥约束集. 随机不重复地选取γ个 5 步骤子集, 即得 at-most-3 约束集, 同样可生成 at-least-3 约束集. 对每组的 4 个参数, 按上述方法生成 100 个实例.

文献[28]的 3 个实例集可由 http://researchdata.essex.ac.uk/114/下载. 本文扩充实例集的 e_{50} 值见表 1.

表 1 “变化 n” (k=36): e_{50} 值

n	72	144	288	576	1 152	2 304	4 608	9 216	18 432	36 864
e_{50}	20	40	55	71	82	91	98	103	105	107
n	73 728	146 456	294 912	589 824	1 179 648	2 359 296	4 718 592	9 437 184	18 874 368	/
e_{50}	108	109	110	110	109	109	110	109	109	/

测试时, 本文限制单个实例求解(不含数据文件读取)不超过 1 小时. 对每组 100 个实例, 分有/无解两种情况统计平均执行时间, 以削弱随机因素, 研究算法性能随实例参数的变化趋势. 当某组参数出现超时实例后,

不统计该组的结果, 也不再测试更大的参数.

5.2 对比算法选择与实现

文献[28]的 IPB 充分确立了模式回溯技术路线在资源独立约束和高资源配比条件下的优势, 是本文首先要比较的方法. 该文献提供了 C#源代码(http://researchdata.essex.ac.uk/114/). 该实现的候选邻点验证和值势约束验证均基于块的位图表示(该实现用 64 位整数存储)及其比较操作, 利用了位运算的高效性[61]. 为验证一个候选邻点, 预计算该资源的授权步骤集并当作一个虚拟的块, 判断其是否覆盖增块即可, 时间复杂度为 O(k), 但实际性能相当于常数. 对约束验证来说, 由于增块和原块具有同一块编号, 而该编号是搜索过程中对增元的尝试赋值, 故可将增元参与的值势约束 x 的变量集 T_x 作为一个虚拟的块, 借助位运算与原块比较, 根据其不相交与否, 判断 T_x 是否被赋了一个新值. 以此方式维护第 4.4 节所述的#*diff*(x), 将形成 O(k)的单约束验证时间复杂度. 但其主要开销是两个 64 位整数的位运算和比较, 实际性能相当于常数时间. 位运算相关实现依赖于额外硬件特性, 并伴有一定使用限制(按照 C++标准库相关类型的用法, 块位图大小可扩展至 64 位以上, 但必须在编译期固定, 不能根据实例参数动态设置). 但文献[28]的指派邻点验证方法较 MIPB 更适于位运算加速(MIPB 只需验证每个候选邻点与 1 个增元的授权, 用位运算加速空间不大), 采用此类实现与 MIPB 对比, 可以充分发挥其本身的优点. 该实现还使用了块邻域缓存技术, 即以块位图为关键字、块指派邻域为值进行散列存储, 以降低重复计算. 该技术的加速效果非常好, 但存在前述块位图表示、缓存容量选取、缓存清理策略等值得深入研究的问题, 宜作为下一步研究, 在 MIPB 中暂未采用. 为公平起见, 将该实现按关闭缓存与否, 分为 K19IPB 和 C-K19IPB 两个版本, 同时加入比较, 并以前者为准来报告 MIPB 的性能提升. 该实现的匹配增广搜索采用广度优先形式, 故无需加入第 4.4 节所述的局部广度搜索. 当资源数 $n>k^2$ 时, 该实现采用散列表来建立初始匹配的反向索引(尽管 k 指派图的边数和右侧顶点数为 O(k^2), 但其右侧顶点仍使用[0,n)中的资源编号, 上述处理是为了避免反向索引的规模依赖于 n). 总体上, 除了真实性验证方式的差异, 该实现其他方面与本文第 4.4 节描述的方式相同, 或有对等的优化处理.

近年的其他相关研究中, 文献[22]将早期的模式回溯法[31]改进为充分验证模式回溯法 FPB(full pattern backtracking). 它对每个搜索结点从头计算模式完全指派图, 原理上显然落后于增量式计算, 实测性能也偏弱, 故不再纳入对比. 文献[42]给出了 IPB 的变体算法 IVIPB. 本实验在匹配增广、约束验证和变量排序规则上采用第 4.4 节所述方式, 重新实现了该算法并加入比较. 文献[40]提出了模式动态规划的优化实现方式, 而文献[30]的推广方法可局限于资源独立约束. 对这两者, 文献[28]都进行过实验并显著胜出, 本文将不再重复. 目前, 与文献[28]方法性能最接近的是 Google OR-Tools 的约束求解器 CP-SAT, 它基于回溯+约束传播技术路线, 在 2021 年 10 月的 MiniZinc 约束规划竞赛中夺得了冠军, 是 CSP 通用求解器的优异代表. 本文选择其最新稳定版 v9.2 (对 at-most-3/at-least-3 约束, 按文献[28]所述方式建模)进行比较. 由于 CP-SAT 的技术路线不同且实现复杂, 后文将主要报告其黑箱性能, 而不作过多分析.

MIPB 的最小增量匹配是为控制完全指派图上的匹配时间复杂度而设计的. 为了解其对实际性能的影响, 将该特性取消(但采用第 4.4 节的方式实现匹配增广), 得到简化算法 sMIPB (simplified MIPB), 一同加入比较.

5.3 实验结果与分析

本节将 CP-SAT, K19IPB, C-K19IPB, IVIPB, MIPB 和 sMIPB 这 6 种算法进行对比实验, 其中, CP-SAT 使用安装版 OR-Tools 库文件, 建模处理代码用 C++实现; K19IPB/C-K19IPB 为文献[28]提供的 C#代码; 而 MIPB、sMIPB 和 IVIPB 用 C++实现. 编译执行环境为: Win10、Visual Studio 2022 专业版+Release x64 编译模式+默认优化选项(C++为-O2, C#为打开代码优化, C#目标框架为.Net4.7.2)、主/睿频为 3.6/4.2 GHz 的 Intel Core i3-9100 CPU、DDR4 2400 8 G*2 双通道内存、NVME M.2 512 G 固态硬盘.

- 实验 1. 变化 k 维度: n=10k

用 n=10k 系列的实例集测试 6 种算法. CP-SAT/K19IPB/C-K19IPB/IVIPB/MIPB/sMIPB 分别在 k=52/52/53/52/55/55 处出现超时. 对每个算法的无超时实例组, 分有/无解两种情形统计平均执行时间, 绘制其随 k 变化的

对数坐标曲线, 如图 3 所示. 可以看到, 6 种算法通常在有解情形下取得更好的性能. 因为此时有机会提前找到一个解, 而不必遍历解空间. 12 条曲线的主体均接近直线走势, 表明 6 种算法的渐进上界至少是指数函数. 对 5 种 IPB 来说, 这符合它们 $O^*(B(k))$的时间复杂度(模式空间结点数为 $B(k) \leqslant \sum_{i=0}^{k} B(i) \leqslant B(k+1)$, 其中, $B(k)$ 为超指数函数). 尽管最坏情况下可能遍历模式空间, 但验证剪枝大大削减了实际搜索的结点数量, 使 5 种 IPB 的执行时间未表现出明显的超指数增长. K19IPB 和 C-K19IPB 初始走势不同于其他 3 种 IPB, 这与其编写语言和程序结构有关. CP-SAT 的曲线走势明显不同, 则主要与其底层技术路线有关.

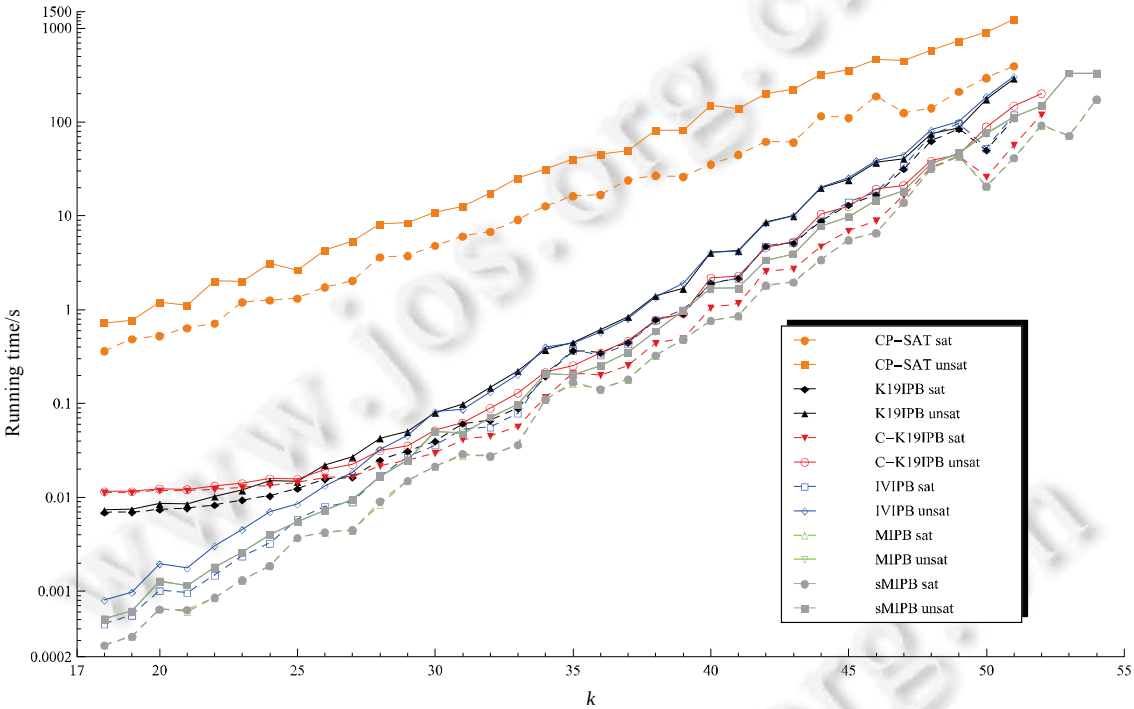

图 3 变化 k (n=10k): 6 种算法执行时间对比

先将 K19IPB 与 C-K19IPB 比较, 以了解块邻域缓存的加速效果. 统计作图数据可知: 当 k=18~51 时, 有/无解情形下, C-K19IPB 的时间性能为 K19IPB 的 0.613~1.958/0.637~1.966 倍, 平均 1.456/1.516 倍. 有/无解情形下, C-K19IPB 分别从 k=28/k=26 处开始取得优势, 并保持至终. 在此范围内, C-K19IPB 的时间性能平均为 K19IPB 的 1.738/1.737 倍, 表明块邻域缓存极大提高了搜索验证效率, 而指派计算代价是模式验证性能的重要瓶颈. k 较小时, K19IPB 占据优势, 是因为此时缓存的性能增益不大, 未抵消散列表的启动和维护代价.

接着, 将 MIPB 与 sMIPB 比较, 以了解最小增量匹配的作用. 统计作图数据可知: 当 k=18~54 时, 有/无解情形下, MIPB 的时间性能为 sMIPB 的 0.968~1.080/0.964~1.038 倍, 平均 1.006/1.001 倍, 几乎没有多大差异. 这既是因为最小增量方式的性能增益不大, 也是因为匹配计算在结点验证代价中所占比例不高. 首先, 尽管 sMIPB 的增量匹配在 O(kn)规模的完全指派图上进行, 而 MIPB 的最小增量匹配在 O(k^2)规模的子图上进行, 但对实际性能影响很小. 其原因在于: 最小增量匹配主要是在增块邻域超过 k 时起作用, 可避免一次匹配增广; 而 sMIPB 在深度搜索之前会检查增块邻域一趟, 在上述条件下, 至多检查 k 个邻点即可, 增广时间复杂度为 O(k), 故采用最小增量匹配并未免除多少开销. 另一方面, 匹配计算是结点验证的最后一个环节, 在剪枝搜索树的相当一部分叶结点处(主要是约束验证失败, 也有增块邻域为空的情形, 注意, 变量排序规则强化了约束剪枝)不执行. 而在每个内结点处都成功执行, 故往往有机会(在高资源配比和授权比例不低条件下, 指派图总边数通常较多, 有利于匹配增广搜索)快速完成. 故总体上, 匹配代价在结点验证中所占比例较低, 匹配环节

的优化对整体性能影响不大.

在无块邻域缓存的两种 IPB 实现中, K19IPB 的时间性能略优于 IVIPB. 统计作图数据可知: 当 k=18~51 时, 有/无解情形下, K19IPB 的时间性能为 IVIPB 的 0.065~1.145/0.109~1.158 倍, 平均 0.779/0.814 倍; 有/无解情形下, K19IPB 均从 k=34 处开始取得优势, 此后只偶尔出现很弱的劣势. 在此范围内, K19IPB 的性能倍数平均为 1.043/1.034, 优势较弱. 两者变量排序规则相同, 约束验证实际性能均为常数时间, 且匹配增广都基于 k 指派图, 主要是在指派计算方式上差异较大. IVIPB 在搜索块的指派邻点时, 综合利用了块中各变量的邻域信息, 相当于从某个(平均化的)变量邻域中搜索块的邻点. 按照本实验的实例生成模型, 每个步骤平均对 $1/\Delta$=1/4 的资源授权, 故其搜索范围大约是 K19IPB 的 1/4. 但其检查增块中 O(β)个步骤的授权关系(每次检查可以常数时间完成), 这里, β为增块大小, 才能确认或排除一个邻点, 故其候选邻点的验证代价约为 O(β). 而 K19IPB 采用 64 位整数位运算验证候选邻点, 实际性能相当于常数时间. 因此, 若计算同一增块的 k 邻域, K19IPB 可能达到 O(β/Δ)倍于 IVIPB 的性能. 但只有形成足够大的增块, 即$\beta>\Delta$=4, K19IPB 的上述优势才能凸显出来. 粗略地, 可以只分析 $n/\Delta^{\beta}\geqslant 1$ 的情况(此时, 增块通常存在邻点, 而其原块至少有Δ个邻点, 不会在父结点处导致剪枝). 本实验两者最大可求解 k=51 的实例, 其 n=510, 故β最大取到 4, 故 K19IPB 难以形成明显的优势.

现在将 MIPB 和 K19IPB 比较. 由作图数据可知: 在 k=18~51 时, 有/无解情形下, MIPB 的时间性能达到 K19IPB 的 1.776~25.546/1.602~14.375 倍, 平均为 4.708/3.453 倍; 有/无解情形下, 分别从 k=28/k=27 开始, MIPB 的优势稳定至 3 倍以内, 且下降趋势很快消失, 平均倍数为 2.337/2.284. 上述性能差异同样是由指派计算方式主导的. MIPB 和 K19IPB 检查一个候选邻点的实际性能均为常数时间, 只不过 MIPB 的检查范围是原块的邻域, 其实际规模相对资源集 U 通常大为收缩, 而 K19IPB 可能检查整个 U, 但找到 k 个邻点即可停止. 设搜索结点的增块大小为β, 而每个步骤约有 n/Δ个授权资源, 仍不妨假设 $n/\Delta^{\beta}\geqslant 1$, 则 MIPB 与 K19IPB 找到一个邻点的实际代价之比约为 $1/\Delta^{\beta-1}$, 所求邻点数之比约为 $\begin{cases} n/(k\Delta^{\beta}), & (n/k>\Delta^{\beta}) \\ 1, & (n/k\leqslant\Delta^{\beta}) \end{cases}$, 从而 MIPB 的性能倍数为

$$\begin{cases} k\Delta^{2\beta-1}/n=k4^{2\beta-1}/n, & (n/k>\Delta^{\beta}=4^{\beta}) \\ \Delta^{\beta-1}=4^{\beta-1}, & (n/k\leqslant\Delta^{\beta}=4^{\beta}) \end{cases} \tag{1}$$

本实验中, n/k=10, 故若$\beta\geqslant 2$, 邻点数之比总为 1, 此时, MIPB 便有 $4^{\beta-1}\geqslant 4$ 倍的指派性能优势. 不过, 给定 n, 当β较大时, 不满足 $n/\Delta^{\beta}\geqslant 1$ 条件, 上述 $4^{\beta-1}$ 倍的性能优势不会出现. 两个算法的共同解出规模最大为 k=51, n=510, 要保证 $4^{\beta-1}$ 倍性能优势, β最大可取 4, 这就限制了 MIPB 指派性能的优势倍数. 而在整体上, 由于每个结点处先验证约束, 失败后不计算指派图, 有指派代价时, 一定存在约束验证代价, 还可能存在匹配代价, 内结点处存在变量排序代价, 搜索前存在初始代价等等因素, MIPB 的上述性能优势也会被冲淡.

将 MIPB 和 IVIPB 比较. 由作图数据可知: 当 k=18~51 时, 在有/无解情形下, MIPB 的时间性能达到 IVIPB 的 1.542~2.888/1.515~2.710 倍, 平均为 2.136/2.080 倍; 当 k=34~51 时, MIPB 的平均性能倍数达到 2.437/2.378. 类似于上一段的分析, 当 $n/\Delta^{\beta}\geqslant 1$ 时, MIPB 与 IVIPB 找到一个邻点的实际代价之比约为$\Delta/(\beta\Delta^{\beta-1})=1/(\beta\Delta^{\beta-2})$, 所求的邻点数之比约为 $\begin{cases} n/(k\Delta^{\beta}), & (n/k>\Delta^{\beta}) \\ 1, & (n/k\leqslant\Delta^{\beta}) \end{cases}$, 从而 MIPB 的性能倍数约为

$$\begin{cases} k\beta\Delta^{2\beta-2}/n=k\beta 4^{2\beta-2}/n, & (n/k>\Delta^{\beta}=4^{\beta}) \\ \beta\Delta^{\beta-2}=\beta 4^{\beta-1}, & (n/k\leqslant\Delta^{\beta}=4^{\beta}) \end{cases} \tag{2}$$

因本实验中$\beta\leqslant 4$, 所以 MIPB 的整体性能倍数同样受到了限制.

由以上两段的分析, 不难理解 MIPB 相对 K19IPB 和 IVIPB 的性能优势, 其优势与 n/k 的取值有关. 后文将固定 k=18 和 k=36, 对极大的 n 做进一步研究.

再将 MIPB 与另外两种算法比较, 统计作图数据可知.

(1) 当 k=18~52 时, 有/无解情形下, MIPB 的时间性能达到 C-K19IPB 的 1.017~41.706/0.928~22.577 倍, 平均为 5.450/3.376 倍; 有无/解情形下, 从 k=28/k=25 开始, MIPB 的优势落入 3 倍以内, 且下降趋势

较快消失, 平均倍数为 1.377/1.421;

(2) 当 k=18~51 时, 在有/无解情形下, MIPB 的时间性能达到 CP-SAT 的 4.424~1469.963/11.055~1408.857 倍, 平均 319.501/353.940 倍, 总体呈下降趋势; 有无/解情形下, 从 k=47/k=50 开始, MIPB 的优势落入 15 倍以内, 也不再有下降趋势, 平均倍数为 8.479/11.584, 平均绝对差距为 205.1/986.7 s. 尽管 CP-SAT 是较大型通用软件, 统计、日志等辅助开销偏高, 当 k 较小时影响较大, 但 MIPB 在 k 很大时仍有上述优势, 且绝对优势突出, 这是仅靠简化辅助开销无法实现的.

对不依赖.Net 框架的进程, 通过 Windows 接口函数较难得到峰值虚拟内存, 故本文用便于采集的峰值工作集大小(PeakWorkingSetSize, 为峰值物理内存, 记为 PRAM)和峰值页面文件使用量(PeakPagefileUsage, 记为 PPage)来衡量空间性能. 本实验 6 种算法的空间代价见表 2. 由该表及原始数据可知: 各算法的空间占用均随 k 增加呈增长趋势; 对同样的 k, MIPB/sMIPB 的空间占用基本相同, 略低于 IVIPB, 显著低于 K19IPB; K19IPB 又略低于 C-K19IPB; 而 CP-SAT 的空间占用最大. 当 k=18~51 时, 相对于 K19IPB, C-K19IPB 在 PRAM 均值上平均高出 3.217 MB, 在 PPage 均值上平均高出 2.879 MB. 这表明块邻域缓存占用的空间并不高. 实际上, C-K19IPB 使用的块邻域缓存以 16 384 项为限, 每项有不超过 $k\leqslant$52 个 32 位整数, 至多也只有 3.25 MB. 对同样的 k, MIPB 的两个指标均略低于 IVIPB. 主要是因为 IVIPB 多出了若干索引数组, 抵消了其指派图规模上的优势. K19IPB 没有这些索引, 故从指派图上规模判断, 其空间性能更应该优于 MIPB. 不过原始数据表明: 对同样的 k, MIPB 两个指标均明显低于 K19IPB. 这可能与程序结构等因素有关. 另外, C#程序由.Net 框架管理内存, 相对 C++程序可能有较大的基础差异. sMIPB 的两个指标与 MIPB 基本相同, 尽管两者在匹配增广时所用初始匹配大小可能不同, 但本文采用以块号为索引、匹配资源为元素的数组来存储, 实际空间占用并没有差异.

表 2 变化 k (n/k=10): 6 种算法的空间代价(MB)

空间代价\算法、k		CP-SAT	K19IPB	C-K19IPB	IVIPB	MIPB	sMIPB
		18–51	18–51	18–52	18–51	18–54	18–54
PRAM 均值	最小	10.736	15.751	16.104	3.334	3.485	3.485
	最大	86.878	18.619	28.644	4.181	3.723	3.725
	平均	35.696	17.873	21.239	3.755	3.587	3.587
PPage 均值	最小	7.306	17.383	17.410	0.741	0.659	0.657
	最大	86.899	19.658	29.318	1.365	0.901	0.904
	平均	33.640	19.041	22.069	1.023	0.762	0.763

- 实验 2. 变化 k 维度: n=100k

用 n=100k 系列的实例集测试 6 种算法. CP-SAT/K19IPB/C-K19IPB/IVIPB 分别在 k=54/52/53/49 处出现超时, 而 MIPB 和 sMIPB 按时解出了所有实例. 对每个算法的无超时实例组, 分有/无解两种情形统计平均执行时间, 绘制随 k 变化的对数坐标曲线, 如图 4 所示. 相对于图 3 主要的变化是: MIPB(以及 sMIPB)的曲线位置与其他算法明显拉开了差距; 另外, CP-SAT 相对其他非 MIPB 算法, 都在更大的 k 处出现超时.

对 K19IPB 与 C-K19IPB, 统计作图数据可知: 当 k=18~51 时, 有/无解情形下, C-K19IPB 的时间性能为 K19IPB 的 0.763~2.921/0.783~2.867 倍, 平均为 1.904/1.924 倍. 这说明块邻域缓存仍然发挥了很大作用.

对 MIPB 与 sMIPB, 统计作图数据可知: 当 k=18~55 时, 有/无解情形下, MIPB 的时间性能为 sMIPB 的 0.974~1.046/0.961~1.037 倍, 平均 1.004/1.002 倍, 差异仍很微弱. 其原因与实验 1 类似.

对 K19IPB 和 IVIPB, 统计作图数据可知: 当 k=18~48 时, 有/无解情形下, K19IPB 的时间性能为 IVIPB 的 0.075~1.092/0.100~1.085 倍, 平均 0.723/0.731 倍. 这仍然是由于 IVIPB 在 k 较小时优势突出导致. 有/无解情形下, K19IPB 从 k=37/k=39 处开始取得优势, 并大体保持至终. 在此范围内, K19IPB 的性能倍数平均为 1.047/1.030, 只能认为其与 IVIPB 的性能大体相同. 本实验两者共同可求解的最大规模为 k=48, n=4800, 此时β最大取到 6. 但有/无解情形下, K19IPB 也只达到了 1.092/1.085 倍的性能优势. 这主要是因为 K19IPB 的 O(β/Δ)=O(β/4)倍指派性能优势是一个上限. 实际上, IVIPB 确认一个邻点时才会检查β次授权, 更多时候只是排除候选, 其检查次数往往少于β, 故β达到 6 仍然不足以凸显 K19IPB 的优势.

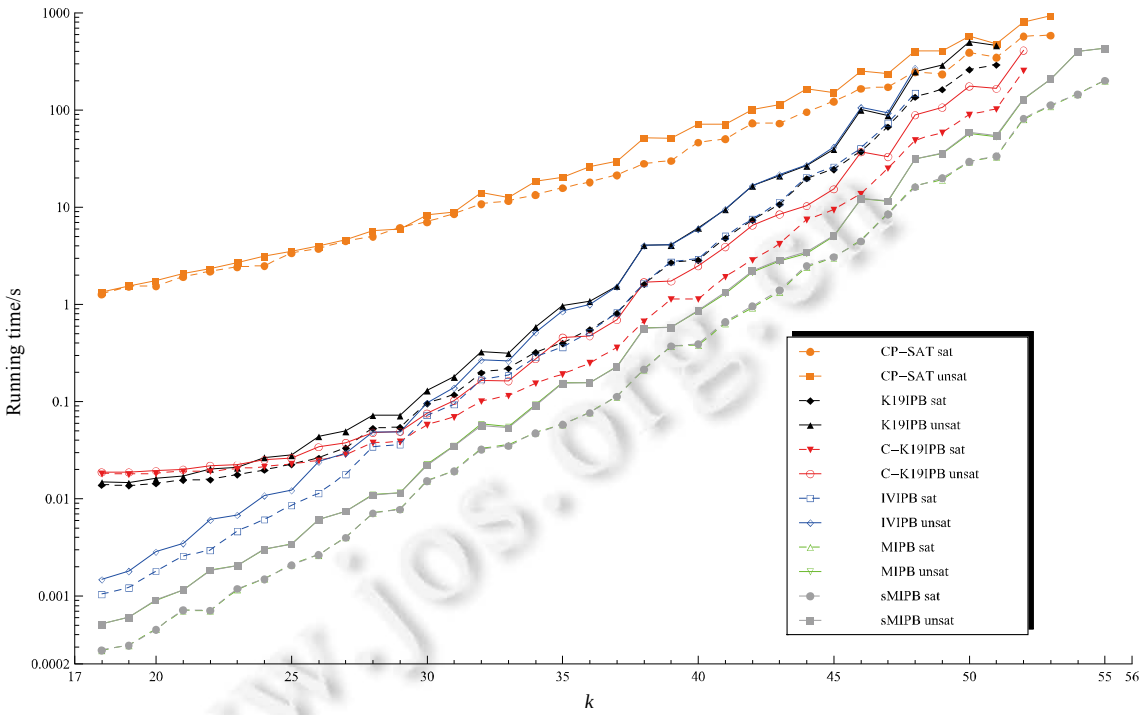

图 4 变化 k (n=100k): 6 种算法执行时间对比

现在将 MIPB 与 K19IPB 比较. 当 k=18~51 时, 在有/无解情形下, MIPB 的时间性能达到 K19IPB 的 6.038~50.325/5.132~28.695 倍, 平均为 11.977/9.067 倍. 在有/无解情形下, 分别从 k=26/k=24 开始, MIPB 的优势稳定至 10 倍以内, 且下降趋势很快消失, 平均倍数为 7.606/7.184, 稳定之后的优势比例达到实验 1 的 3 倍以上. 如实验 1 对应的分析, MIPB 较 K19IPB 的指派性能倍数以增块大小β为指数. 但该实验的解出实例$\beta\leqslant 4$, 而本实验对相同的 k, n 值扩大到 10 倍, 从而满足 $n/\Delta^{\beta}=n/4^{\beta}\geqslant 1$ 的β值可增加 1 或 2. 且对增大后的β值, 从 k 个步骤中取β个的组合方式很多, 有利于相应增块的形成. 在这些增块相应的结点处, MIPB 较 K19IPB 的指派性能会显著提高(指数部分增加了 1 或 2), 而指派计算是结点验证的主要开销, 故整体搜索性能也会随之提升.

将 MIPB 与 IVIPB 比较. 当 k=18~48 时, 在有/无解情形下, MIPB 的时间性能达到 IVIPB 的 3.637~8.992/2.880~8.705 倍, 平均为 6.018/5.475 倍. 其优势比例也扩大到实验 1 的近 3 倍. 如实验 1 的对应分析, MIPB 较 IVIPB 的指派性能倍数以增块大小β为指数, 而本实验条件导致β增加 1 或 2, 将会使 MIPB 明显获益.

再将 MIPB 与另外两种算法比较, 由作图数据可知.

(1) 当 k=18~52 时, 在有/无解情形下, MIPB 的时间性能达到 C-K19IPB 的 2.894~65.958/2.781~36.322 倍, 平均为 10.262/6.673 倍. 在有/无解情形下, 均从 k=30 开始, MIPB 的优势稳定至 4 倍以内, 且不再有明显的变化趋势, 而平均倍数达到 3.141/3.005;

(2) 当 k=18~53 时, 在有/无解情形下, MIPB 的时间性能达到 CP-SAT 的 5.344~4909.639/4.518~2613.183 倍, 平均为 871.900/504.950 倍, 且随着 k 的增大呈下降趋势. 本实验中, CP-SAT 较其他非 MIPB 算法更晚出现超时, 表明其性能更稳定. 尽管在每组的平均性能通常偏弱, 但 CP-SAT 在 100 个实例间的性能波动较小, 有时能够按时解出其他算法的超时实例.

统计各算法的空间代价见表 3. 由该表及原始数据可知, 各算法空间占用随 k 的变化趋势以及 k 相同时的对比关系都与实验 1 类似. CP-SAT 的解出规模较实验 1 增加, 但空间占用却有所下降, 这应该与其算法机制有关. K19IPB 的指标较实验 1 略有增长, 这是因为输入实例的存储规模增加; 另外, 每个步骤的授权资源数扩大 10 倍, 也会使 k 指派图的规模扩大(达到 k 个邻点的块数增加). C-K19IPB 的指标仍较 K19IPB 略高, 也较实

验 1 略有增长, 原因均类似于之前的分析. IVIPB 的求解规模较实验 1 下降, 但空间占用却明显增加. 主要是因为数组索引的规模随 n/k 而扩大 10 倍, 同时, k 指派图的规模有所增加. MIPB 的空间占用较实验 1 有所增长, 这与输入实例的存储规模增加有关, 也与完全指派图的规模增长有关(同样大小的块, 其邻域规模较实验 1 扩大 10 倍).

表 3 变化 k (n/k=100): 6 种算法的空间代价(MB)

空间代价\算法、k		CP-SAT	K19IPB	C-K19IPB	IVIPB	MIPB	sMIPB
		18–53	18–51	18–52	18–48	18–55	18–55
PRAM 均值	最小	10.323	18.452	18.796	4.194	3.639	3.639
	最大	71.730	21.942	31.720	8.008	4.996	5.001
	平均	30.717	20.349	22.730	5.838	4.263	4.173
PPage 均值	最小	6.835	19.967	19.986	1.371	0.802	0.801
	最大	70.904	23.341	32.397	5.244	2.209	2.212
	平均	28.235	21.770	23.602	3.209	1.452	1.452

- 实验 3. 变化 n 维度: k=18

用 k=18 系列的实例集测试 6 种算法, 均可按时完成. 对每个算法, 分有/无解两种情形统计各组实例的平均执行时间, 绘制其随 n 变化的双对数曲线, 如图 5 所示. 本实验中主要的变化是: K19IPB/C-K19IPB 在 n 足够大时显著超过了 MIPB/sMIPB 和 IVIPB. 下面仍按之前的顺序进行比较和分析.

对 K19IPB 与 C-K19IPB, 统计作图数据可知: 有/无解情形下, C-K19IPB 的性能达到 K19IPB 的 0.604~0.853/0.626~0.876 倍, 平均 0.760/0.778 倍. K19IPB 占据明显优势, 与前两个实验在 k=18 时的结果大体一致. 这是因为块邻域缓存用散列表实现, 有一定的管理和维护开销. 当 k 较小时, 模式空间不大, 其结点间增块重复的情况较少, 缓存的性能增益不能抵消其代价. 而当 n 增大时, 虽有利于剪枝搜索树的扩大(根据前述 $n/\Delta^{\beta}\geqslant 1$ 的粗略要求, 大 n 值有利于形成较大的增块), 增加重复, 但其以完整模式空间为限度, 对 k=18 的情况作用不大. 故本实验中, 块邻域缓存的效果对大 n 值并不敏感.

对 MIPB 与 sMIPB, 统计作图数据可知: 有/无解情形下, MIPB 的性能达到 sMIPB 的 0.981~1.025/0.966~1.009 倍, 平均 0.999/0.997 倍, 差异非常微弱. 其基本原因见实验 1 的分析. 同时, 本文的匹配增广实现针对大 n 值进行了优化, 故匹配计算部分对整体性能的影响进一步缩小, 有关差异也更难以凸显.

接着比较 K19IPB 与 IVIPB. 有/无解情形下, n=2k~2048k 时, IVIPB 有 1.029~21.088/1.058~16.201 倍的优势, 平均 11.094/7.863 倍; 而 n=4096k~524288k 时, K19IPB 有 1.791~146.544/1.832~138.567 倍的优势,平均为 37.839/36.385 倍. 这种优劣的转换, 主要是因为 IVIPB 预计算了多个大小为 n 的数组索引. 当 n 较小时, 可以快速计算这些索引, 且 K19IPB 的 $O(\beta/\Delta)$=$O(\beta/4)$倍指派性能优势难以凸显(此时β不会很大, 且该优势为偏高的估计), 故整体上 IVIPB 优势. 而当 n 很大时, 按 n/Δ^{β}=$n/4^{\beta}\geqslant 1$ 估计, 有利于出现大的β. 例如: n=524288k 时, β可取到 11; 而 n=4096k 时, β也可取到 8. 但是由于存在互斥约束, β/k 必须很小, 增块才有可能形成. 本实验中, k=18, e_{50} 通常大于 10, 故β=8 的增块有 90%以上概率包含互斥的步骤. 于是, 大体从 n=4096k 开始, β值便很难真正地增加, 故 K19IPB 后半段的优势不能由此解释. 对 IVIPB 进行分段计时, 发现对特大的 n 值, 初始化开销远超过搜索开销, 因而正是这一点主导了 IVIPB 后半段的劣势.

将 MIPB 与 K19IPB 比较. 有/无解情形下, n=2k~8192k/n=2k~4096k 时, MIPB 有 1.020~53.025/1.721~35.367 倍的优势, 平均 23.647/16.228 倍; 而当 n=16384k~524288k/n=8192k~524288k 时, K19IPB 有 2.251~109.218/1.541~162.531 倍的优势, 平均为 35.066/44.071 倍. 由实验 1 分析给出的公式(1), 给定 k 和 n, 增块大小β越大, 对 MIPB 越有利. 但如上一段的分析, 大体从 n=4096k 开始, β值便很难真正地增加, MIPB 的优势也不会再扩大. 相反, 由于 n 的增加, $n/k>\Delta^{\beta}=4^{\beta}$的情况更容易出现. 此时, MIPB 的性能倍数为 $k\Delta^{2\beta-1}/n=k4^{2\beta-1}/n$, 会出现缩小的趋势. 这就解释了 MIPB 相对 K19IPB 的性能变化. 不过, 优劣转换点与若干β值及其频度有关, 很难进一步分析.

将 MIPB 与 IVIPB 比较. 在有无解情形下, MIPB 分别有 1.159~6.134/0.853~3.776 倍的优势, 平均为 2.969/2.086 倍, 大体上全程处于优势. 在 n 较小的前半段, 由于 IVIPB 的初始开销很小, MIPB 的优势主要来自于搜

索阶段, 特别是搜索结点上的指派计算性能. 随着 n 的增加, β值却难以增加, 由实验 1 分析给出的公式(2)可知, MIPB 的优势会快速缩小(n 以指数速度增长)并转为劣势. 但是, IVIPB 的初始开销也快速增长, 使其相对 MIPB 仍然处于劣势. 若对照与 K19IPB 的比较, 限定 n=2k~8192k/n=2k~4096k, 则 MIPB 较 IVIPB 有 1.159~6.134/1.107~3.776 倍的优势, 平均可达 3.593/2.618 倍.

再将 MIPB 与另外两种算法比较. 统计作图数据可知.

(1) 在有/无解情形下, 当 n=2k~8192k/n=2k~4096k 时, MIPB 相对 C-K19IPB 有 1.271~69.165/2.138~45.194 倍的优势, 平均 33.091/22.067 倍; 而当 n=16384k~524288k/n=8192k~524288k 时, C-K19IPB 有 1.822~92.870/1.249~142.300 倍的优势, 平均为 29.504/37.891 倍;

(2) 在有/无解情形下, MIPB 的性能达到 CP-SAT 的 22.072~7613.784/23.050~4747.871 倍, 随着 k 的增大呈下降趋势, 平均倍数为 2741.766/1744.197.

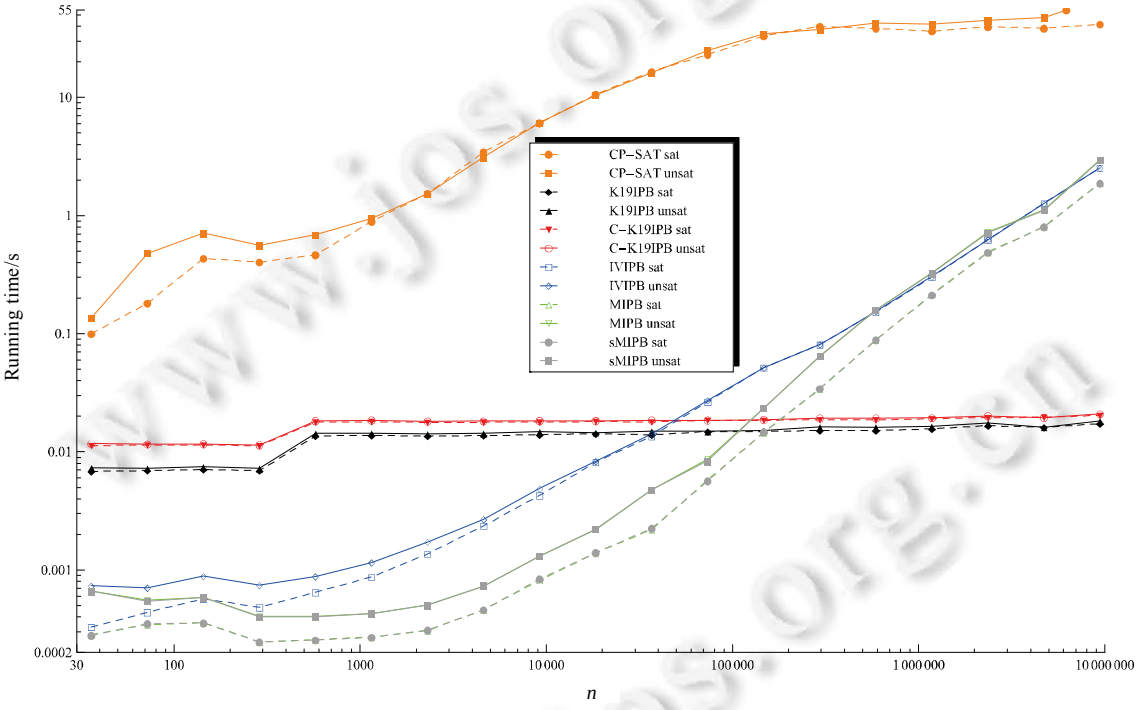

图 5 变化 n (k=18): 6 种算法执行时间对比

当 n=2k~524288k 时, 各算法的空间代价见表 4.

表 4 变化 n (k=18): 6 种算法的空间代价(MB)

空间代价\算法		CP-SAT	K19IPB	C-K19IPB	IVIPB	MIPB	sMIPB
PRAM 均值	最小	8.103	15.427	15.817	3.312	3.337	3.318
	最大	2443.410	95.366	95.600	3386.220	760.209	760.191
	平均	266.952	25.927	26.242	359.876	83.206	83.186
PPage 均值	最小	5.269	17.063	17.080	0.625	0.636	0.638
	最大	2470.466	787.128	786.368	3408.905	783.054	783.065
	平均	272.469	100.377	100.370	362.088	85.373	85.373

由该表及原始数据可知: 随着 n 值的快速增加, 各算法的空间占用也快速增长; 当 n 很大时, MIPB/sMIPB 的空间占用显著高于 K19IPB/C-K19IPB, 但明显低于 CP-SAT; 而 IVIPB 的空间占用是最高的. 表中 K19IPB 的两个指标与 K19IPB 大体相同, 这是因为本实验中实例存储规模随 n 值快速增长, 而 k=18 较小, 块邻域缓存能够占据的空间非常有限. 当 n 足够大时, IVIPB 的空间占用不但显著超过了 K19IPB, 也超过了 CP-SAT,

这是因为其附加的索引数组大小均为 n. 当 n 足够大时, MIPB 的两个指标特别是 PRAM 已经显著超过 K19IPB, 这是因为完全指派图的规模显著超过 k 指派图. 但由于 IVIPB 的大型索引数组, MIPB 的空间占用仍然显著优于 IVIPB.

- 实验 4. 变化 n 维度: k=36

用 k=36 系列的实例集测试 6 种算法, 除 CP-SAT 在 n=262144k 处发生超时外, 其他算法均按时解出了所有实例. 对每个算法, 分有/无解两种情形统计各组实例的平均执行时间, 绘制其随 n 变化的双对数曲线, 如图 6 所示. 本实验主要的变化是, IVIPB 对极大的 n 值也超过了 MIPB.

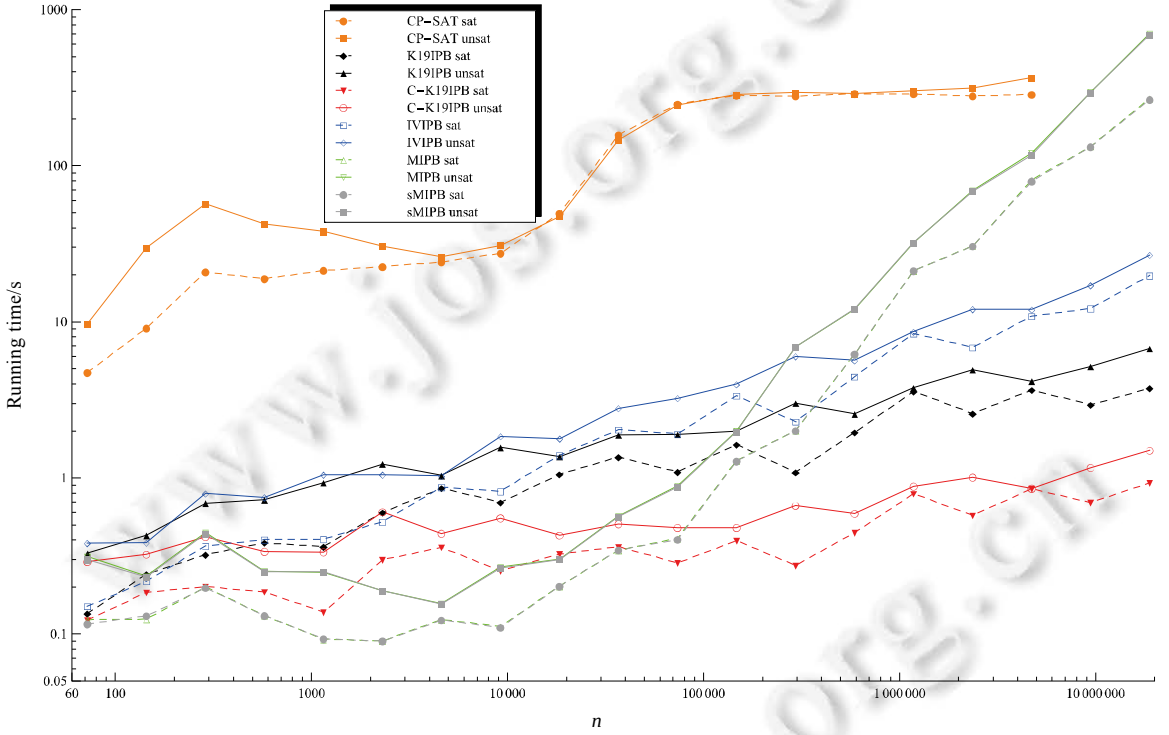

图 6 变化 n (k=36): 6 种算法执行时间对比

下面仍按顺序进行比较分析.

对 K19IPB 与 C-K19IPB, 统计作图数据可知: 有/无解情形下, C-K19IPB 的性能达到 K19IPB 的 1.100~4.539/1.134~4.895 倍, 平均为 3.197/3.327 倍, 且随 n 的增加呈明显的增长趋势. 相对于实验 3, 因 k 值翻倍, 模式搜索空间极大扩张, 增块重复的情况也更为显著, 故块邻域缓存的性能增益随之增加, 而其管理开销变为次要的因素. 而且随着 n 值的增大, 同样的增块会有更多的授权资源(因授权比例不变, 在同样的 n 下, 同一增块的邻域大小约为实验 3 的 2 倍), 这使更多增块的邻域大小从不足 k 变为等于 k, 其直接计算代价增加, 而读取缓存将更为有利. 故本实验中, C-K19IPB 的优势会随着 n 值的增大而增加.

对 MIPB 与 sMIPB, 统计作图数据可知: 有/无解情形下, MIPB 的性能分别为 sMIPB 的 0.940~1.050/0.953~1.003 倍, 平均 0.994/0.989 倍, 反而呈现微弱劣势. 该现象不全是偶然的. 步骤的授权资源集扩大 2 倍, 将有更多增块的邻域大小超过 k, 可在一趟广度搜索中完成匹配增广, 而不进入更耗时的深度搜索(见本文第 3.4 节的匹配增广方式), 这就削弱了 MIPB 相对 sMIPB 的优势. 又由于 MIPB 为简化匹配而做了更多判断, 反而可能造成其劣势.

接着比较 K19IPB 和 IVIPB. 有/无解情形下, 在 n=2k~256k 一段, K19IPB 相对 IVIPB 具有 0.871~1.179/0.852~1.175 倍的优势, 平均 1.048/1.049 倍. 考虑到编写语言和程序结构不同, 很难区分两者的性能优劣. 而

在 n=512k~524288k 一段, K19IPB 的优势扩大到 1.326~5.281/1.305~3.793 倍, 平均 2.583/2.324 倍, 且随 n 增大呈明显增长趋势. 由于 k=36 时模式搜索空间显著扩张, IVIPB 的线性初始代价已难以主导整体性能, 而 K19IPB 在后半段的优势主要是由其指派性能优势导致. 尽管由于互斥约束的限制, 当 n 值增大时, 峰值β很难总是增大, 但是步骤的授权资源数会随之增大, 使更多较大增块所计算的邻点数从少于 k 个变为刚好 k 个. 而 K19IPB 对每个邻点时有 O(β/Δ)=O(β/4)倍的优势, 其指派优势将会随之扩大, 进而导致整体优势的扩大.

现在比较 MIPB 与 K19IPB. 有/无解情形下, n=2k~4096k/n=2k~2048k 时, MIPB 有 1.095~6.958/1.052~6.619 倍的优势, 平均 3.703/3.639 倍; 而当 n=8192k~524288k/n=4096k~524288k 时, K19IPB 有 1.853~71.830/1.006~104.410 倍的优势, 平均为 23.151/27.631 倍. 随着 n 的增大, MIPB 仍然发生了优劣转换, 转换点的 n/k 较实验 3 变小了约 2 倍, 但其 n 值大体不变, 仍在 100 000 附近. 这是因为在同样的授权比例下, 能够形成的增块大小直接依赖于 n 值, 本例中 k 值翻倍, 故转换点的 n/k 相应变小. 在图 5 和图 6 中, 随着 n/k 的增大, 都可以观察到, MIPB 相对 K19IPB 的优势先扩大再缩小的过程. 一层原因是: 随着 n/k 的增大, 峰值β先有所扩张, 后因互斥约束难以扩张, 使得 MIPB 计算 1 个邻点的最大性能优势先迅速提升, 后难以提升. 另一层原因在于: 随着 n/k 的增大, MIPB 和 K19IPB 对同一增块共同计算的邻点数将首先增加, 而 MIPB 计算每个邻点的性能都有优势, 故其优势将随之扩大. 而当 n/k 继续增大时, 更大的增块难以形成, 而两个算法对存量增块共同计算的邻点数增加到 k 为止, 上述有利因素将逐渐失去作用, MIPB 计算邻点较多的弱点将逐渐成为主导因素.

将 MIPB 与 IVIPB 比较. 有/无解情形下, n=2k~8192k/n=2k~4096k 时, MIPB 有 1.139~7.339/1.222~6.903 倍的优势, 平均 4.130/3.943 倍; 而当 n=16384k~524288k/n=8192k~524288k 时, IVIPB 有 1.397~13.602/1.147~26.283 倍的优势, 平均为 6.729/9.483 倍. 随着 n 的增大, MIPB 也发生了优劣转换. 原因仍是由于互斥约束, β 值随 n 值而增大时会受到限制, 而 n 值随 n/k 值持续增大, 对 MIPB 的性能优势不利. 若参照与 K19IPB 的比较而限定 n=2k~4096k/n=2k~2048k, 则 MIPB 相对 IVIPB 有 1.221~7.339/1.222~6.903 倍的优势, 平均为 4.380/4.120 倍.

再将 MIPB 与另外两种算法比较, 统计作图数据可知.

(1) 在有/无解情形下, 当 n=2k~1024k/n=2k~512k 时, MIPB 相对 C-K19IPB 有 0.995~3.314/0.928~3.193 倍的优势, 平均为 1.754/1.713 倍; 而当 n=2048k~524288k/n=1024k~524288k 时, C-K19IPB 有 1.433~289.959/1.119~470.500 倍的优势, 平均为 75.876/100.958 倍;

(2) n=2k~131072k 时, 在有/无解情形下, MIPB 的性能达到 CP-SAT 的 3.530~607.277/3.051~274.439 倍, 平均为 177.834/115.505 倍.

6 种算法的空间代价见表 5.

表 5 变化 n (k=36): 6 种算法的空间代价(MB)

空间代价\算法、n/k		CP-SAT	K19IPB	C-K19IPB	IVIPB	MIPB	sMIPB
		2~131072	2~524288	2~524288	2~524288	2~524288	2~524288
PRAM 均值	最小	22.400	17.614	18.821	3.048	3.364	3.350
	最大	2 358.888	174.192	175.617	12 543.560	1 946.341	1 946.313
	平均	298.854	35.193	36.064	1 323.932	208.375	208.356
PPage 均值	最小	19.851	19.369	20.110	0.716	0.679	0.682
	最大	2 484.112	2 927.193	2 928.991	12 684.731	2 099.576	2 099.575
	平均	309.444	326.836	327.191	1 336.850	222.093	222.086

由该表及原始数据可知, 各算法空间占用随 n 的变化趋势以及 n 足够大时的对比关系都与实验 3 类似. 相对于表 4, 各算法的对应指标基本都出现了比较明显的增长. 这主要是因为 k 和 n 的倍增导致了实例本身以及指派图规模的增大.

6 结论与下一步工作

本文对资源独立性约束 WSP 的增量模式回溯法 IPB 进行改进, 通过建立基于完全指派图和完备 k 核心匹配的最小增量计算方法, 优化了模式真实性验证的性能, 将整体搜索时间复杂度由 O(($f(c)+k^2+kn$)$B(k)$)降至

$O((f(c)+k^2+n)B(k))$, 由此建立了最小增量模式回溯法 MIPB. 在包含互斥和 at-most-r/at-least-r 约束、资源授权比例约为 1/4 的相变随机实例集上进行实验, 结果表明:

相对于 IPB 的原始实现 K19IPB(关闭块邻域缓存)和另一种优化实现 IVIPB, 若固定资源配比 n/k=10MIPB 的(实例组内)平均性能最低, 具有 1.6 和 1.5 倍优势, (组间)平均 2 倍以上; 而固定 n/k=100, MIPB 的平均性能最低具有 5.1 和 2.9 倍的优势, 趋势稳定后平均大于 7 倍和 5 倍; 固定 k=18, 在 n=2k~4096k 区间, MIPB 对 K19IPB 有平均 16 倍以上的优势, 对 IVIPB 有平均 2.6 倍以上的优势; 固定 k=36, 在 n=2k~2048k 区间, MIPB 对 K19IPB 有平均 3.6 倍以上的优势, 对 IVIPB 有平均 4.1 倍以上的优势. 大体上, 在 n 值为 10 万左右, MIPB 较 IPB 出现劣势, 此后其相对性能显著下降; 但在此前更有实际意义的 n 值范围内, MIPB 的优势比较显著, 并呈先增大再减小的趋势, 在一段范围内格外突出.

相对于 IPB 的原始实现 C-K19IPB(打开块邻域缓存), MIPB 在固定 n/k=10 的切片上, 趋势稳定后的平均优势为 1.4 倍左右; 在固定 n/k=100 的切片上, 趋势稳定后的平均优势大于 3 倍; 在固定 k=18 的切片上, 在 n=2k~4096k 区间, 平均有 22 倍以上的优势; 在固定 k=36 的切片上, 在 n=2k~512k 区间, 平均有 1.7 倍以上的优势. 由于块邻域缓存技术同样可以加速 MIPB, 上述比较是技术栈不对等的, 但也更加凸显了 MIPB 的优势.

相对于 2021 CSP 竞赛的冠军 CP-SAT, MIPB 在上述固定 n/k 的两个切片上, 平均性能最低具有 4.4 倍的优势; 在上述固定 k 的两个切片上, 平均性能最低具有 3 倍以上的优势. 在资源独立性约束和高资源配比条件下, MIPB 强化了 IPB 技术路线相对回溯+约束传播的优势.

另外, 现有 IPB 针对 WSP 只求一个可满足解的目标, 使用了简化的 k 指派图, 这会丢失大量可满足解的信息. 而 MIPB 建立了完全指派图的快速计算方法, 不仅在前述相当大范围内具有性能优势, 而且对可满足解的计数、枚举或优选问题更具参考价值.

下一步将研究 MIPB 与块邻域缓存、约束传播、约束图分解等优化技术的结合. 由于 WSP 面向单工作流的安全规划与资源分配, 而实际场景中的资源往往为多工作流所共享, 将 WSP 相关问题推广到并发环境中, 也有很多工作可以开展.

References:

[1] Zhang ZJ, Zhang YM, Xu XS, *et al*. Manufacturing service composition self-adaptive approach based on dynamic matching network. Ruan Jian Xue Bao/Journal of Software, 2018, 29(11): 3355−3373 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5475.htm [doi: 10.13328/j.cnki.jos.005475]

[2] Li ZW, Zhou MC. Modeling, Analysis and Deadlock Control of Automated Manufacturing Systems. Beijing: Science Press, 2009 (in Chinese).

[3] Liu GJ. Complexity of the deadlock problem for Petri nets modelling resource allocation systems. Information Sciences, 2016, 363(10): 190−197.

[4] Wen YP, Liu JX, Dou WC, *et al*. Privacy-aware multi-tenant access control for cloud workflow. Computer Integrated Manufactoring Systems, 2019, 25(4): 894−900 (in Chinese with English abstract).

[5] Wang J, Han Y. Cloud workflow scheduling method with data privacy protection. Computer Integrated Manufactoring Systems, 2013, 19(8): 1859−1867 (in Chinese with English abstract).

[6] Zhong RY, Xu X, Klotz E, *et al*. Intelligent manufacturing in the context of industry 4.0: A review. Engineering, 2017, 3(5): 96−127 (in Chinese with English abstract).

[7] Bertino E, Ferrari E, Atluri V. The specification and enforcement of authorization constraints in workflow management systems. ACM Trans. on Information System Security, 1999, 2(2): 65−104.

[8] Crampton J, Khambhammettu H. Delegation and satisfiability in workflow systems. In: Proc. of the 13th ACM Symp. on Access Control Models and Technologies. New York: Association for Computing Machinery, 2008. 31−40.

[9] Basin D, Burri JS, Karjoth G. Obstruction-Free authorization enforcement: Aligning security and business objectives. Journal of Compter Security, 2014, 22(5): 661−698.

[10] Zavatteri M, Vigano L. Last man standing: Static, decremental and dynamic resiliency via controller synthesis. Journal of Computer Security, 2019, 27(3): 343−373.

[11] Berge P, Crampton J, Gutin G, *et al*. The authorization policy existence problem. IEEE Trans. on Dependable and Secure Computing, 2020, 17(6): 1333−1344.

[12] Crampton J, Gutin G, Karapetyan D, *et al*. The bi-objective workflow satisfiability problem and workflow resiliency. Journal of Computer Security, 2017, 25(1): 83−115.

[13] Zhai ZN, Lu YH, Yu FH, *et al*. Counting workflow satisfiability (≠,=) and its #P completeness. Acta Electronica Sinica, 2017, 45(3): 605−611 (in Chinese with English abstract).

[14] Zhai ZN, Lu YH, Yu J, *et al*. Quick bounding algorithm for #ws(≠) based on pattern backtracking. Journal of Chinese Computer Systems, 2020, 41(12): 2494−2499 (in Chinese with English abstract).

[15] Wang QH, Li NH. Satisfiability and resiliency in workflow authorization systems. ACM Trans. on Information and System Security, 2010, 13(4): 40:1−35.

[16] Yang P, Xie X, Ray I, *et al*. Satisfiability analysis of workflows with control-flow patterns and authorization constraints. IEEE Trans. on Services Computing, 2014, 7(2): 237−251.

[17] Gutin G, Wahlstrom M. Tight lower bounds for the workflow satisfiability problem based on the strong exponential time hypothesis. Information Processing Letters, 2016, 116(3): 223−226.

[18] Aalst W, Hee K. Workflow Management: Models, Methods and Systems. Cambridge: MIT Press, 2002. 1−368.

[19] Crampton J, Gutin G, Yeo A. On the parameterized complexity and kernelization of the workflow satisfiability problem. ACM Trans. on Information and System Security, 2013, 16(1): 1−31.

[20] Gutin G, Kratsch S, Wahlstrom M. Polynomial kernels and user reductions for the workflow satisfiability problem. Algorithmica. 2016, 75(7): 383−402.

[21] Yang B, Xia CH, Zhang ZG, *et al*. On the satisfiability of authorization requirements in business process. Frontiers of Computer Science, 2017, 11(3): 528−540.

[22] Zhai ZN, Lu YH, Wan J, *et al*. Match-Pruning pattern backtracking algorithm for exclusion constrained workflow satisfiability decision. China Mechanical Engineering, 2018, 29(24): 2988−2998 (in Chinese with English abstract).

[23] Jegou P, Terrioux C. Combining restarts, nogoods and bagconnected decompositions for solving CSPs. Constraints, 2017, 22(2): 191−229.

[24] Jegou P, Terrioux C. Hybrid backtracking bounded by treedecomposition of constraint networks. Artificial Intelligence, 2003, 146(1): 43−75.

[25] Zhai ZN, Lu YH, Yu FH, *et al*. Backtracking tree-decomposition method with complete independence for workflow satisfiability decision (≠). Journal of Frontiers of Computer Science and Technology, 2018, 12(12): 2021−2032 (in Chinese with English abstract).

[26] Zhai ZN, Lu YH, Zhou WJ, *et al*. Fixed parameter linear algorithm for counting workflow satisfiability (≠). Chinese Journal of Computers, 2016, 39(11): 2291−2306 (in Chinese with English abstract).

[27] Cohen D, Crampton J, Gagarin A, *et al*. Iterative plan construction for the workflow satisfiability problem. Journal of Artificial Intelligence Research, 2014, 51(11): 555−577.

[28] Karapetyan D, Parkes AJ, Gutin G, *et al*. Pattern-based approach to the workflow satisfiability problem with user- independent constraints. Journal of Artificial Intelligence Research, 2019, 66(9): 85−122.

[29] Gutin G. The workflow satisfiability problem with user-independent constraints. In: Proc. of the 1st Int'l Conf. on Graph Computing. New York: IEEE, 2019. 1−4.

[30] Gutin G, Karapetyan D. Constraint branching in workflow satisfiability problem. In: Proc. of the 25th ACM Symp. on Access Control Models and Technologies. New York: ACM, 2020. 93−103.

[31] Karapetyan D, Gagarin A, Gutin G. Pattern backtracking algorithm for the workflow satisfiability problem with user-independent constraints. In: Proc. of the 2015 Int'l Workshop on Frontiers in Algorithmics. Berlin: Springer, 2015. 138−149.

[32] Chen JN, Li Z, Li ZS. Research on parallel propagation mode of table constraint based on multi-core CPU. Ruan Jian Xue Bao/ Journal of Software, 2021, 32(9): 2769−2782 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5989.htm [doi: 10.13328/j.cnki.jos.005989]

[33] Cohen D, Jeavons P, Jefferson C, *et al*. Symmetry definitions for constraint satisfaction problems. Constraints, 2006, 11(7): 115−137.

[34] Rossi F, Beek VP, Walsh T. Handbook of Constraint Programming. Amsterdam: Elsevier, 2006. 1−955.

[35] Roney-Dougal MC, Gent PI, Kelsey T, *et al*. Tractable symmetry breaking using restricted search trees. In: Proc. of the 16th European Conf. on Artificial Intelligence. 2004. 211−215.

[36] Freuder CE. Eliminating interchangeable values in constraint satisfaction problems. In: Proc. of the 9th National Conf. on Artificial Intelligence, Palo Alto: AAAI, 1991. 227−233.

[37] Hentenryck VP, Flener P, Pearson J, *et al*. Tractable symmetry breaking for CSPs with interchangeable values. In: Proc. of the 18th Int'l Joint Conf. on Artificial Intelligence. Califonia: IJCAI, 2003. 277−282.

[38] Flener P, Pearson J, Sellmann M, *et al*. Dynamic structural symmetry breaking for constraint satisfaction problems. Constraints, 2009, 14(4): 506−538.

[39] Cohen D, Crampton J, Gagarin A, *et al*. Engineering algorithms for workflow satisfiability problem with user- independent constraints. In: Proc. of the 2014 Int'l Workshop on Frontiers in Algorithmics. Berlin: Springer, 2014. 48−59.

[40] Cohen D, Crampton J, Gagarin A, *et al*. Algorithms for the workflow satisfiability problem engineered for counting constraints. Journal of Combinatorial Optimization, 2016, 32(7): 3−24.

[41] Crampton J, Gagarin A, Gutin G, *et al*. On the workflow satisfiability problem with class-independent constraints for hierarchical organizations. ACM Trans. on Privacy & Security, 2016, 19(3): No.8.

[42] Zhai ZN, Lu YH, Zhou WJ, *et al*. Acceleration of boundary shrinking in incremental pattern backtracking for *WS-RI. Computer Engineering and Application, 2020, 56(24): 236−241 (in Chinese with English abstract).

[43] Berge C. Two theorems in graph theory. Proc. of the National Academy of Sciences, 1957, 43(9): 842−844.

[44] Edmonds J. Paths, trees and flowers. Canadian Journal of Mathematics, 1965, 17(3): 449−467.

[45] Gabow NH. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. Journal of the ACM, 1976, 23(2): 221−234.

[46] Kameda T, Munro I. A O($|V|\cdot|E|$) algorithm for maximum matching of graphs. Computing, 1974, 12(3): 91−98.

[47] Vazirani VV. A theory of alternating paths and blossoms for proving correctness of the O($V^{1/2}E$) general graph maximum matching algorithm. Combinatorica, 1994, 14(1): 71−109.

[48] Even S, Kariv O. An O($N^{2.5}$) algorithm for maximum matching in general graphs. In: Proc. of the 16th Annual IEEE Symp. on Foundations of Computer Science. New York: IEEE, 1975. 100−112.

[49] Micali S, Vazirani VV. An O($|V|^{1/2}\cdot|E|$) algoithm for finding maximum matching in general graphs. In: Proc. of the 21st Annual Symp. on Foundations of Computer Science. New York: IEEE, 1980. 17−27.

[50] Alt H, Blum N, Mehlhorn K, *et al*. Computing a maximum cardinality matching in a bipartite graph in time O($n^{1.5}(m/\log n)^{0.5}$). Information Processing Letters, 1991, 37(2): 237−240.

[51] Duan R, Pettie S. Linear-Time approximation for maximum weight matching. Journal of the ACM, 2014, 61(1): No.1.

[52] Harvey AJN. Algebraic algorithms for matching and matroid problems. SIAM Journal on Computing, 2009, 39(2): 679−702.

[53] Fakcharoenphol J, Laekhanukit B, Nanongkai D. Faster algorithms for semi-matching problems. ACM Trans. on Algorithms, 2014, 10(3): No.14.

[54] Katrenic J, Semanisin G. Maximum semi-matching problem in bipartite graphs. Discussiones Mathematicae: Graph Theory, 2013, 33(3): 559−569.

[55] Bosek B, Leniowski D, Sankowski P, *et al*. Online bipartite matching in offline time. In: Proc. of the 2014 Symp. on Foundations of Computer Science. New York: IEEE, 2014. 384−393.

[56] Chaudhuri K, Daskalakis C, Kleinberg DR, *et al*. Online bipartite perfect matching with augmentations. In: Proc. of the 2009 Int'l Conf. on Computer Communications. New York: IEEE, 2009. 1044−1052.

[57] Gupta A, Kumar A, Stein C. Maintaining assignments online: Matching, scheduling and flows. In: Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms. New York: ACM, 2014. 468−479.

[58] Baswana S, Gupta M, Sandeep S. Fully dynamic maximal matching in O(log$N$) update time. SIAM Journal on Computing, 2018, 47(3): 617−650.

[59] Bernstein A, Stein C. Faster fully dynamic matchings with small approximation ratios. In: Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms. New York: ACM, 2016.

[60] Mertzios BG, Nichterlein A, Niedermeier R. A linear-time algorithm for maximum-cardinality matching on cocomparability graphs. SIAM Journal on Discrete Mathematics, 2018, 32(4): 2820−2835.

[61] Wang HY, Guo JS, Ouyang DT, *et al*. Adaptive propogation algorithm based on bitwise operation. Journal of Jilin University (Engineering and Technology Edition), 2012, 42(5): 1219−1224 (in Chinese with English abstract).

附中文参考文献:

[1] 章振杰, 张元鸣, 徐雪松, 等. 基于动态匹配网络的制造服务组合自适应方法. 软件学报, 2018, 29(11): 3355−3373. http://www.jos.org.cn/1000-9825/5475.htm [doi: 10.13328/j.cnki.jos.005475]

[2] 李志武, 周孟初. 自动制造系统建模、分析与死锁控制. 北京: 科学出版社, 2000.

[4] 文一凭, 刘建勋, 窦万春, 等. 云工作流环境下隐私感知的多租户访问控制模型. 计算机集成制造系统, 2019, 25(4): 894−900.

[5] 王菁, 韩燕波. 保护私有数据的云工作流调度方法. 计算机集成制造系统, 2013, 19(8): 1859−1867.

[6] 钟润阳, 徐旬, Klotz E, Newman TS. 对工业 4.0 背景下的智能制造的回顾. 工程, 2017, 3(5): 96−127.

[13] 翟治年, 卢亚辉, 余法红, 等. 工作流可满足性(≠,=)计数及其#P 完全性. 电子学报, 2017, 45(3): 605−611.

[14] 翟治年, 卢亚辉, 俞坚, 等. 基于模式回溯的#WS(≠)快速定界算法. 小型微型计算机系统, 2020, 41(12): 2494−2499.

[22] 翟治年, 卢亚辉, 万健, 等. 互斥约束工作流可满足性决策的匹配剪枝模式回溯法. 中国机械工程, 2018, 29(24): 2988−2998.

[25] 翟治年, 卢亚辉, 余法红, 等. 工作流可满足决策(≠)的完备独立树分解回溯法. 计算机科学与探索, 2018, 12(12): 2021−2032.

[26] 翟治年, 卢亚辉, 周武杰, 等. 工作流可满足性(≠)计数的固定参数线性算法. 计算机学报, 2016, 39(11): 2291−2306.

[32] 陈佳楠, 李哲, 李占山. 基于多核 CPU 的表约束并行传播模式研究. 软件学报, 2021, 32(9): 2769−2782. http://www.jos.org.cn/1000-9825/5989.htm [doi: 10.13328/j.cnki.jos.005989]

[42] 翟治年, 卢亚辉, 周武杰, 等. *WS-RI 增量模式回溯的边界收缩加速. 计算机工程与应用, 2020, 56(24): 236−241.

[61] 王海燕, 郭劲松, 欧阳丹彤, 等. 基于比特位操作的自适应约束传播算法. 吉林大学学报(工学版), 2012, 42(5): 1219−1224.

翟治年(1977－), 男, 博士, 讲师, 主要研究领域为约束求解, 组合优化, 访问控制, 工作流.

雷景生(1966－), 男, 博士, 教授,主要研究领域为数据科学与大数据, 机器学习, 人工智能应用.

卢亚辉(1976－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为数据挖掘, 金融大数据, 博弈论, 业务过程建模与分析.

向坚(1976－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为人工智能, 机器学习, 多媒体分析与检索, 计算机动画.

刘关俊(1978－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为 Petri 网理论与应用, 模型检测, 机器学习, 人机物系统, 工作流系统, 无人机协同系统.

吴茗蔚(1977－), 女, 博士, 教授, 主要研究领域为智能感知, 通信技术, 机器学习.