Branching on-the-fly for solving satisfaction problems

香港科技大学（广州）
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

功能枢纽
FUNCTION HUB
先进材料学域
Advanced Materials Thrust

Yi-Jia Wang†, Xuan-Zhao Gao* (co-first author), Pan Zhang† and Jin-Guo Liu*

*Advanced Materials Thrust, Function Hub, Hong Kong University of Science and Technology (Guangzhou)

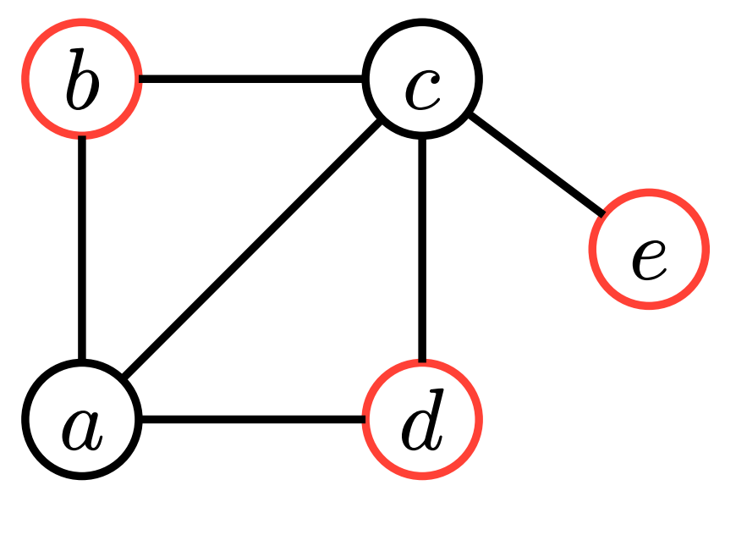
†CAS Key Laboratory for Theoretical Physics, Institute of Theoretical Physics, Chinese Academy of Sciences

Abstract

The branching algorithm is a fundamental technique for designing fast exponential-time algorithms to solve combinatorial constraint satisfaction problems (CSPs) exactly. It divides the entire solution space into independent search branches using predetermined branching rules, and ignores the search on suboptimal branches to significantly reduce the exponential time complexity. The complexity of a branching algorithm is primarily influenced by the rules it employs. However, existing branching algorithms are often limited by their predetermined finite rule set which can not cover all possible situations, so that can not achieve optimality in most cases. In this paper, we focus on the maximum independent set problem, and propose to automatically generate optimal branching rules for a given sub-graph with existing numerical tools including the generic tensor network and the integer/linear programming solvers. Through experimental evaluation, we demonstrate that our algorithm's ability to discover branching rules with the lowest future complexity. Furthermore, it is shown that by integrating our approach with existing methods, we achieve an average complexity of $O(1.0455^n)$ on 3-regular graphs, representing a significant enhancement in performance over previous methods that depend on expert-designed branching rules.

Maximum independent set (MIS) problem

An independent set is a set of vertices in a graph, no two of which are adjacent.



$G = (V, E)$

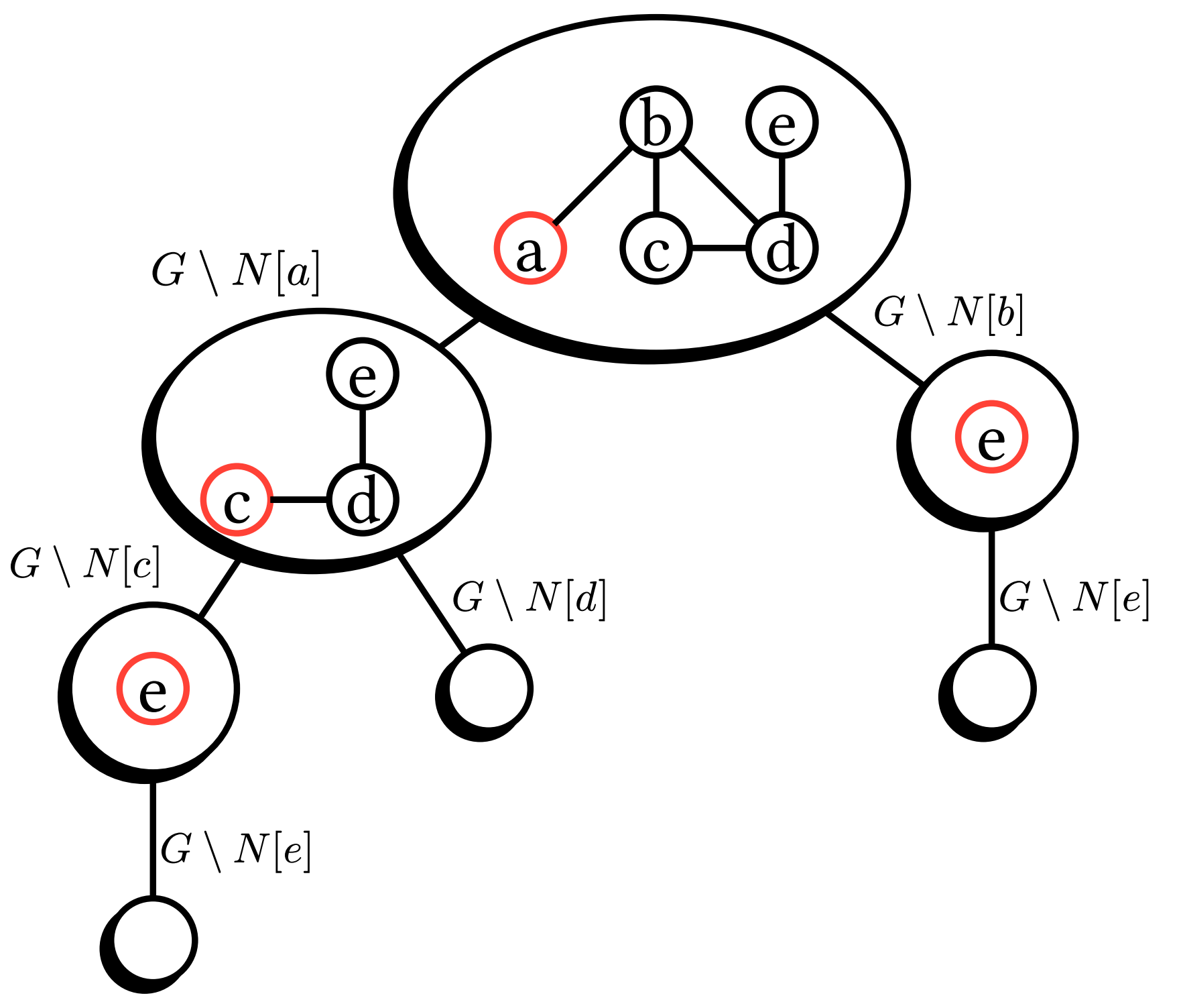
- Independent sets: $\{\}, \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{a, e\}, \{b, d\}, \{b, e\}, \{d, e\}, \{b, d, e\}$
- MIS: $\{b, d, e\}$, size $\alpha(G) = 3$.
- Independence polynomial: $I(x) = 1 + 5x + 4x^2 + x^3$

4 independent sets with 2 vertices

Finding the maximum independent set for a given graph is in NP-complete - a class of problems that unlikely to be solved in time polynomial in the input size.

Why branching?

While the **generic tensor network based algorithm**[1] can solve the independent set problems efficiently on graphs with small tree width, e.g. tree graph (tree width 1), geometric graphs (tree width $O\left(n^{\frac{d-1}{d}}\right)$ such as the grid graph), **branching** works much better in high dimensional graphs, e.g. it can solve a fully connected graph in linear time. The **border line** is a 3-regular graph, high dimensional, but sparse enough. Its tree width is $\approx \frac{n}{6}$, rendering a tensor network algorithm with complexity $O(1.1225^n)$. The best branching algorithm gives the lowest complexity of $O(1.0836^n)$ [2]. Theorem (or branching rule) plays a central role in branching. Better rule gives lower complexity. The following example is from the book **Exact Exponential Algorithms** [3].



1. Given a n-vertex graph G, select a vertex, e.g. a
   - Case 1: a is in the independent set, then create a branch $G \setminus N[a]$
   - Case 2: a is not in the independent set, then create a branch $G \setminus N[b]$
2. Solve two subproblems with sizes $(n-2)$ and $(n-4)$ recursively.

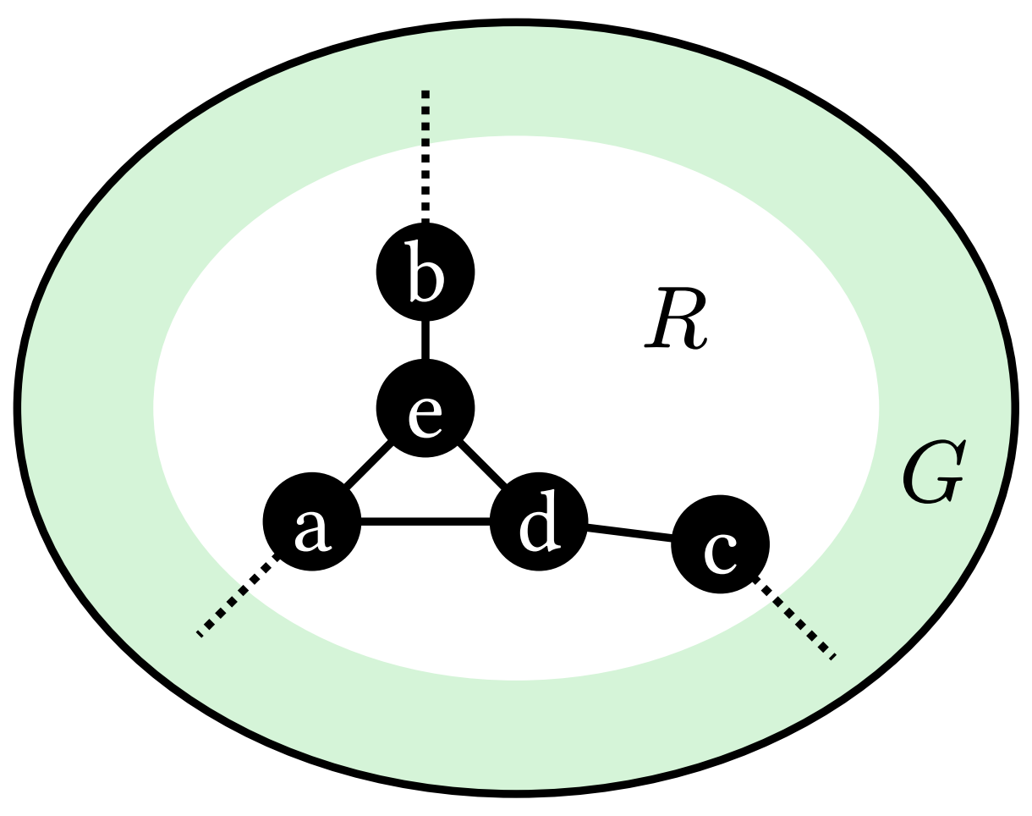
Let the **computational complexity** be γ^n, we have

$$\gamma^n = \gamma^{n-2} + \gamma^{n-4}$$

The rule used here is: *If a vertex v is not in the MIS, then at least one of its neighbors must be in the MIS*, which gives the complexity of $\gamma = 1.4423$. Finding better branching rules to reduce γ is the main theme in this field. Well known rules include *domination rule*, *sattelite rule*, *mirror rule*, etc.

Discover optimal branching rules automatically!

We show computer programs can automatically discover the optimal branching rules on sub-graphs up to size tens of vertices, which is way beyond the capability of human experts. For example, we consider a sub-graph R with boundary vertices $\partial R = \{a, b, c\}$, where the dashed lines indicate their connections to vertices in the environment G without R. Due to the independence constrains on this local graph, most local configurations are *irrelevant*. The remaining configurations are listed below, grouped by the boundary configurations $s_{\partial R} = \{000, 001, 010, 111\}$.



s_{abcde}	clauses in $\mathcal{D}$
$S_{000} = \{00001, 00010\}$	$\neg a \wedge \neg b \wedge \neg d \wedge e$
$S_{001} = \{00101\}$	
$S_{010} = \{01010\}$	$\neg a \wedge b \wedge \neg c \wedge d \wedge \neg e$
$S_{111} = \{11100\}$	$a \wedge b \wedge c \wedge \neg d \wedge \neg e$

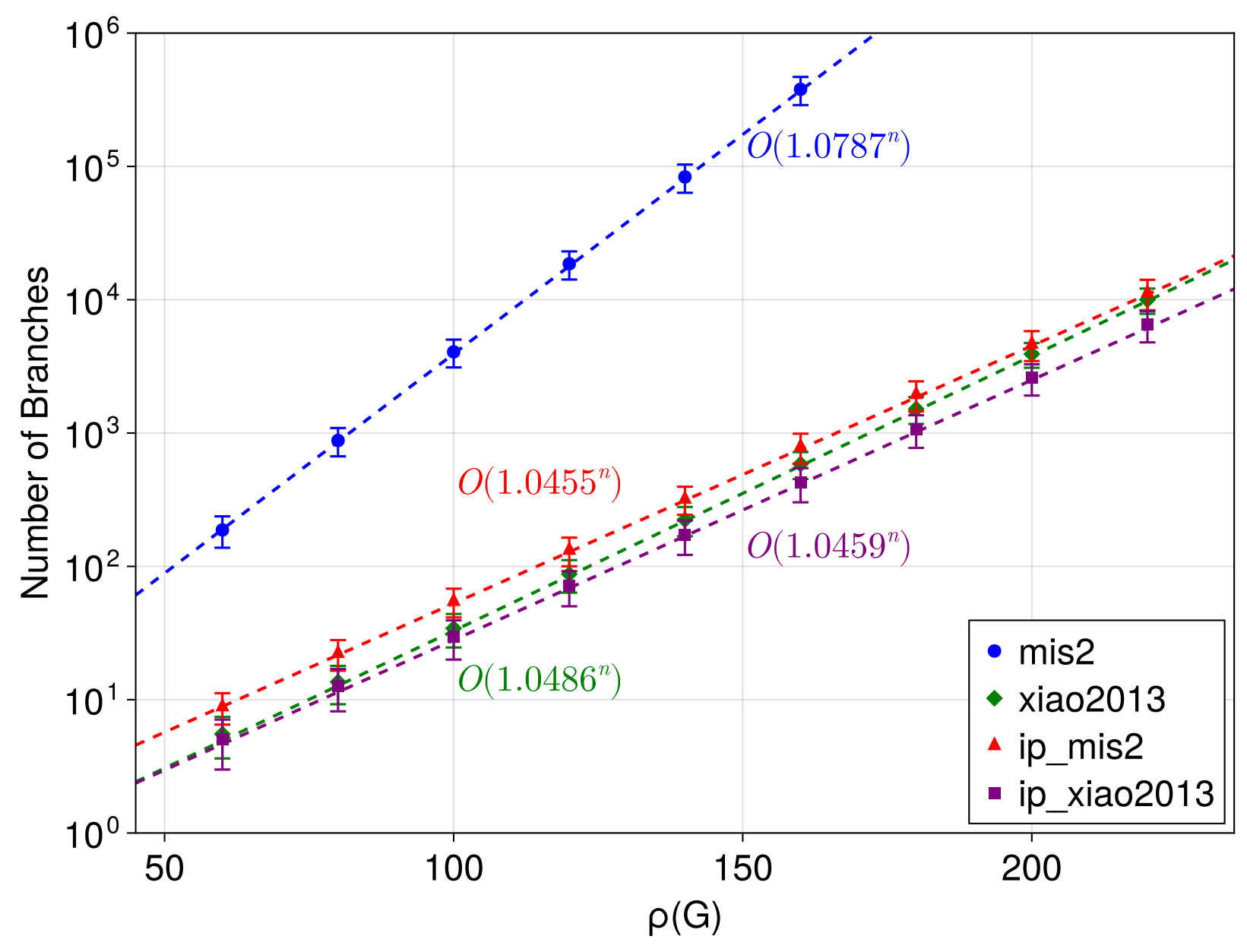
The optimal branching rule is about how to capture the patterns of the relevant configurations. Formally,

Definition: *A branching strategy for bitstring searching* denoted as δ, is a function that maps a subgraph R to a boolean formula in disjunctive normal form (DNF) $\mathcal{D} = c_1 \vee c_2 \vee l ... \vee c_{|\mathcal{D}|}$, where each clause $c_k = l_1 \wedge l_2 \wedge ... \wedge l_{|V(c_k)|}$ is associated with a branch, in which a positive or negative literal $l_i = v$ or $l_i = \neg v$ represents the vertex $v \in V(R)$ is or isn't in the set. Here, $V(c_k)$ is the set of vertices involved in the k-th clause c_k.

For a n-vertex sub-graph, the number of possible branching rules (DNF) scales as 2^{3^n}, for $n = 6$, this number is: 282401395870821749694910884220462786335135391185157752468340193086269383036119849990587392099522999697089786549828399657812329686587839094762655308848694610643079609148271612057263207249270352772375735947883453036573 4912.

In our work[4], we developed an algorithm to find the best (with the lowest γ) branching rule from the super-exponential number of branching rules (page too short, algorithm omitted, talk to Xuan-Zhao Gao or Jin-Guo Liu).

Benchmark on 3-regular graphs



- Better (practical) performance than state of the art methods, with computer generated branching rules!
- Source code on GitHub: ArrogantGao/OptimalBranching.jl

References

[1] J.-G. Liu, X. Gao, M. Cain, M. D. Lukin, and S.-T. Wang, “Computing Solution Space Properties of Combinatorial Optimization Problems Via Generic Tensor Networks,” *SIAM Journal on Scientific Computing*, vol. 45, no. 3, pp. A1239–A1270, 2023.

[2] M. Xiao and H. Nagamochi, “Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs,” *Theoretical Computer Science*, vol. 469, pp. 92–104, 2013.

[3] F. V. Fomin and P. Kaski, “Exact exponential algorithms,” *Communications of the ACM*, vol. 56, no. 3, pp. 80–88, 2013.

[4] Y.-J. Wang, X.-Z. Gao, P. Zhang, and J.-G. Liu, “Automated Discovery of Optimal Branching Rules for the Maximum Independent Set Problem,” *to be released*, 2024.

GiggleLiu jinguoliu@hkust-gz.edu.cn