Convergence Analysis of Tree-Width-Based Warning Propagation Algorithm (Postprint)

Authors: Xie Zhixin, Wang Xiaofeng, Yu Zhuo, Cao Zexuan, Wu Yuxiang, Mo Chunhui

Date: 2022-05-18T00:00:00+00:00

Abstract

Warning Propagation algorithm, as a fundamental message-passing algorithm, is highly effective for solving satisfiability problems when it converges. However, when the factor graph structure becomes relatively complex, the algorithm often fails to converge, resulting in solution failure. To provide a theoretical explanation for this phenomenon and to effectively analyze the convergence properties of Warning Propagation algorithm, we employ tree decomposition methods to construct a tree width measurement model for the factor graph corresponding to propositional formulas, and compute the tree width of satisfiable random instances. By establishing the relationship between tree width and the convergence of Warning Propagation algorithm, we propose tree-width-based convergence determination conditions for Warning Propagation algorithm. Through experimental analysis, the results demonstrate the effectiveness of this method, which holds significant importance for research on analyzing the convergence of other message-passing algorithms.

Full Text

Preamble

Vol. 39 No. 10
Application Research of Computers
ChinaXiv Partner Journal

Convergence Analysis of Warning Propagation Algorithm Based on Tree Width

Xie Zhixin[1], Wang Xiaofeng[1],[2]†, Yu Zhuo[1], Cao Zexuan[1], Wu Yuxiang[1], Mo Chunhui[1]

[1]School of Computer Science and Engineering, North Minzu University
[2]Key Laboratory of Images & Graphics Intelligent Processing of State Ethnic Affairs Commission, Yinchuan 750021, China

Abstract: The warning propagation algorithm serves as a fundamental information propagation algorithm that proves highly effective for solving satisfiability problems upon convergence. However, when the factor graph structure becomes complex, the algorithm often fails to converge, resulting in solution failure. To provide a theoretical explanation for this phenomenon and enable effective analysis of warning propagation algorithm convergence, this paper employs tree decomposition methods to construct a tree width measurement model for factor graphs corresponding to propositional formulas and calculates the tree width of satisfiable random instances. We establish a relationship between tree width and warning propagation algorithm convergence, presenting convergence determination conditions based on tree width. Experimental analysis demonstrates the effectiveness of this method, which holds significant importance for analyzing the convergence of other information propagation algorithms.

Keywords: warning propagation algorithm; convergence; tree width; propositional formula; satisfiability problem

0 Introduction

Constraint satisfaction problems (CSP) have profound influence in artificial intelligence research and have garnered widespread attention from domain experts. Many real-world problems can be reduced to CSP problems, such as graph coloring, traveling salesman problem (TSP), and scheduling problems. The satisfiability problem (SAT) represents a typical CSP problem that has been proven NP-complete, with its NP-completeness extensively applied to research on other NP-complete problems across various domains. The SAT problem investigates whether there exists a Boolean variable assignment that makes a given propositional formula true. Due to its broad applications and central importance, the SAT problem has attracted universal attention from researchers.

Physicists proposed the Warning Propagation (WP) algorithm based on statistical physics as a most fundamental information propagation algorithm that operates primarily through warning message transmission and iteration on factor graphs. When the algorithm converges, it fixes values for partial variables and simplifies formulas to achieve SAT problem solving. However, as factor graph structures become increasingly complex, information circulates infinitely within loops, preventing warning messages from converging to fixed values and ultimately causing WP algorithm solving failure—a phenomenon known as algorithm non-convergence. WP algorithm convergence guarantees effective problem solving, making the study of WP algorithm convergence crucial for its application.

Current research on WP algorithm convergence has achieved some results. In 2001, Weiss Y et al. concluded that when a factor graph contains only a single cycle, the algorithm converges and yields valid approximations of variable marginal distributions. In 2007, Maneva E et al. provided a convergence condition for random satisfiable instance generation models with planted assignments, though its application was limited to this specific model and lacked evaluation on 3-SAT instance generation models. In planted assignments, when probability p is sufficiently large, generated instances have a satisfying assignment with high probability. In 2013, Wang Xiaofeng et al. studied special SAT structures where two variables do not belong to the same clause, presenting necessary and sufficient conditions for algorithm convergence based on Gaussian models, but only proved for a small portion of planted assignment instances with limited application. In 2014, Su Qinliang et al. utilized semidefinite programming to examine convergence based on Gaussian models, analyzing algorithm convergence properties. The bottleneck for semidefinite programming lies in computation—when SAT problem scale increases (i.e., factor graph scale increases and becomes complex), semidefinite scale checking becomes inefficient and extremely difficult. In 2016, Wang Xiaofeng et al. analyzed convergence for algorithms with positive semidefinite message matrices, presenting necessary and sufficient conditions for convergence. In 2018, She Guangwei et al. studied modified WP algorithm convergence based on (3,4)-SAT problems, concluding that modified WP converges on regular problems, but required converting 3-SAT problems to (3,4)-SAT special structures, where WP algorithm fails on satisfiability determination for converted instances. In 2020, Cui Li et al. studied WP algorithm convergence on WP-solvable formulas, providing a sufficient condition for convergence. In 2021, Niu Jin et al. analyzed WP algorithm convergence using structural entropy concepts and provided a convergence threshold, though community partition accuracy lacked evaluation metrics, affecting accurate calculation of factor graph structural entropy. In 2022, Liang Chen et al. analyzed Survey Propagation (SP) algorithm convergence using K-dimensional structural entropy, presenting a sufficient condition for convergence, but the required structural information minimization conditions were difficult to achieve. The above WP algorithm convergence research is primarily based on special models or specific applications, with insufficient research on non-special structure factor graphs.

Therefore, substantial work remains to be completed for WP algorithm convergence analysis research. Factor graph structure significantly impacts WP algorithm convergence. As problem scale increases, the number of nodes and edges in factor graphs grows, making WP algorithm information transmission extremely complex. In 1991, Robertson N et al. proposed tree decomposition technology in their work, aiming to decompose graphs into trees and utilize tree structural information to reflect certain properties of original graphs. Graph tree width and tree decomposition provide a new approach for problem solving—by decomposing graphs and restricting intractable problem graph models within limited tree width, problems become solvable. From a computational perspective, tree width has important connections with tractability; restricting

tree width parameters can limit combinatorial explosion, with low tree width implying rapid computation completion. Research has found that when CNF formula factor graphs are trees, their satisfiability can be determined in polynomial time. Literature [**?**] restricted QCSP (Quantified Constraint Satisfaction) problems in constraint satisfaction problems using tree width parameters, obtaining a polynomial-time complexity solving algorithm. Literature [**?**] converted regular formula factor graphs into tree structures using a special tree structure and presented possible phase transition points for random regular (3,r)-SAT. Literature [**?**] solved constrained satisfiability problems using tree decomposition technology, and Lei Ying et al. analyzed SAT problems using tree decomposition, presenting the relationship between solving difficulty and tree width. WP algorithm convergence is closely related to its factor graph structure. Introducing tree width to measure factor graph structural characteristics can simultaneously reflect the scale of decomposed subproblems, better indicating problem complexity. Compared with previous information propagation algorithm convergence analysis, this approach has no conditional restrictions, wide applicability, and better reflects the connection between propositional formula structural features and convergence from factor graph structure analysis, thereby improving WP algorithm convergence property research on factor graph structural properties.

In summary, this paper analyzes and studies WP algorithm convergence when solving SAT problems. Using the concept of tree width, we generate SAT instances of different scales using the G(n,3,m) model, employ tree decomposition algorithms to obtain tree width, analyze propositional formula factor graph structural information, propose a tree width model for SAT instances, analyze the relationship between WP algorithm convergence and tree width through WP algorithm convergence conditions, and present tree width-based WP algorithm convergence determination conditions.

1.1 Factor Graph

A factor graph is a bipartite graph representing the factorization of a "global" function of many variables into a product of "local" functions [**?**]. The graph contains two types of nodes: variable nodes and clause nodes. Circles represent variable nodes, squares represent clause nodes, solid lines indicate variables appearing as positive literals in clauses, and dashed lines indicate variables appearing as negative literals in clauses. Example: For a CNF formula:

$$F = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee \neg x_5) \wedge (x_3 \vee \neg x_4 \vee x_5) \wedge (x_1 \vee x_2 \vee x_3 \vee x_4)$$

Its factor graph is shown in Figure 1.

[Figure 1: see original paper] Factor graph

The warning propagation algorithm iterates on factor graphs. It has been proven that when factor graphs have tree structures, information propagation algorithms converge effectively. For instances whose factor graphs contain multiple loops, information propagation algorithms are not always effective and often exhibit non-convergence [**?**].

1.2 Warning Propagation Algorithm

In the WP algorithm, messages passed from clause nodes to variable nodes are warning messages, denoted as $\vec{u}_{a\to i}$. A value of 1 indicates that clause a' s satisfiability depends on variable i's assignment, while a value of 0 indicates that variable i' s assignment does not play a decisive role in clause a' s satisfiability. WP algorithm information update iteration equations are generally written in warning message difference form as follows:

If clause a contains only variable i, indicating that clause a' s satisfiability is determined by variable i, then $\vec{u}_{a\to i}$ is recorded as 1. When the WP algorithm converges, variable i' s value can be fixed based on warning messages: when $\sum_a \vec{u}_{a\to i} < 0$, assign value 0; when $\sum_a \vec{u}_{a\to i} > 0$, assign value 1; otherwise, temporarily do not assign variable i. The cavity domain $H_{a\setminus i}$ is defined as the set of clauses containing variable i except clause a, with value 1 when variable i appears only in clause a.

Algorithm 1: WP Algorithm
Input: CNF formula, maximum iteration steps $t_{\max}$
Output: Warning messages at convergence or non-convergence status

a) Construct corresponding factor graph

b) Initialize iteration count $t = 0$, randomly assign values from $\{0, 1\}$ with equal probability to each edge pair $\vec{u}_{a\to i}$ on the factor graph

c) For $t = 1$ to $t_{\max}$:
   - Randomly permute edges of factor graph and update warning messages $\vec{u}_{a\to i}$ using equation (1) according to random permutation order

d) If $\vec{u}_{a\to i}(t) = \vec{u}_{a\to i}(t-1)$ for all edges, exit loop

e) If $t == t_{\max}$, return non-convergence

Based on the WP algorithm' s iteration process and equations, the value of $\vec{u}_{a\to i}$ is crucial for fixing variable i' s value. With fixed variable scale, as factor graph scale tends toward complexity, the probability of the same variable appearing in different clauses increases rapidly, and loops emerge with sharply increasing numbers. In equation (1), $\vec{u}_{a\to i}$ calculation is closely related to $H_{a\setminus i}$, which computes message values of variables appearing in clause a in other clauses. The more clauses containing the same variable, the more information needs to be calculated, making $\vec{u}_{a\to i}$ computation more complex, increasing solving

time, and reducing the probability that information values stabilize after each iteration, significantly affecting algorithm convergence probability. When the algorithm converges, fixed points of warning messages can be obtained, and partial variable assignments based on fixed-point information simplify CNF formulas to complete solving. Due to increasing problem scale, variable i appears in more clauses. According to equation (3), cavity domain computation grows rapidly with factor graph scale, increasing computation time, and information calculated in each iteration differs due to complex graph structures, ultimately leading to non-convergence and algorithm failure. When the algorithm does not converge, iterations reach the set maximum, solving time becomes maximal, but the algorithm has already failed—iteration count is not the main factor affecting solving time. Therefore, during WP algorithm iteration, factor graph structure significantly impacts WP algorithm performance.

1.3 Tree Decomposition and Tree Width

Tree width can measure graph structural complexity. Researchers have found that decomposing large-scale intractable problems into smaller, more tractable problems using tree decomposition ideas proves highly effective. For graph $G = (V, E)$, tree decomposition partitions its node set V to transform graph G into a tree structure as much as possible, where a node subset of the graph corresponds to a node in tree $T = (I, F)$, with I being decomposition tree nodes. Tree width describes the tree structure information, characterizing graph G' s structural properties on the tree structure. Using graph tree width as a parameter reflects problem solving complexity while enabling analysis of certain algorithm properties during problem solving, achieving convergence analysis of WP algorithms on factor graphs.

In tree decomposition, node subsets form a bag, denoted as $TG = (X, T)$ for a graph' s tree decomposition, where $X = \{X_i : i \in I\}$ is a non-empty subset of graph G' s node set, with:

a) The union of X_i equals graph G' s node set

b) If any two nodes in graph G have an edge between them, they must simultaneously belong to some X_i

c) If j, k, l are three nodes in the tree and j lies on the path from l to k, then if v belongs to both X_l and X_k, v also belongs to X_j

Tree decomposition width equals $\max(|X_i| - 1 : i \in I)$, i.e., the number of points in the largest bag after decomposition minus 1. Graph G' s tree width is the width of its minimum tree decomposition.

2.1 Factor Graph Processing

Tree decomposition algorithms apply to undirected graph decomposition, while factor graphs contain two types of nodes and two types of edges. During tree decomposition, constructing decomposition trees requires using node degree and edge relationship information between nodes. For graph G' s tree decomposition, the tree width $tw(G)$ has the following basic property:

Property 1 [**?**]: For graph G, $tw(G) \geq \gamma(G) \geq \sigma(G)$, where:

$$\sigma(G) = \min_{v \in V} \deg(v)$$

$$\gamma(G) = \min_{u,v \in V, uv \notin E} \max\{c(u), c(v)\}, \text{ where } c(v) = \begin{cases} |V| & \text{if } G \text{ is a clique} \\ \deg(v) & \text{otherwise} \end{cases}$$

From Property 1, tree decomposition-generated tree width is independent of node types and edge types. Variable nodes and clause nodes in factor graphs are treated as one type of node during tree decomposition. According to tree decomposition width definition, the two edge types in factor graphs representing variables appearing as positive/negative literals do not affect tree width calculation and can be treated as one edge type.

Definition 1 (Elimination Ordering): Let graph $G = (V, E)$, $n = |V|$, bijection $f : V \to [n]$ is called an elimination ordering, where set $[n] = \{1, 2, 3, \ldots, n\}$.

Definition 2 (Chordal Graph): For any cycle in graph G with size exceeding 4, if there exists a chord connecting non-adjacent nodes in the cycle such that the cycle size does not exceed 3, then graph G is a chordal graph.

Graph G' s tree width equals the size of the smallest maximum clique among all its chordal supergraphs minus 1. Generally, elimination ordering and cut sets are used to construct decomposition trees, while some strategies using intelligent algorithms also exist for tree decomposition construction. Here we present an algorithm strategy using elimination ordering to construct tree decomposition:

Algorithm 2: Tree Decomposition via Elimination Ordering
Input: Graph $G = (V, E)$
Output: A tree decomposition $T = (I, F)$ of graph G

a) Delete points from graph according to elimination ordering

b) Each time a point is deleted, add edges among its neighbor nodes to form a cycle and record

c) Repeat step b) until all nodes are deleted, then generate tree nodes in reverse order of elimination ordering, placing connected points in each bag

d) Merge bags to eliminate proper subset relationships between bags in the bag sequence

e) Connect bag sets to generate decomposition tree

According to the tree decomposition process, when performing tree decomposition algorithm processing, first convert variable and clause nodes in factor graphs into one node type representation, converting dashed edges representing negative literals into solid edges. The processing procedure is shown in Figure 3.

[Figure 3: see original paper] Factor graph tree decomposition

Figure 2 provides an example showing graph G and its tree decomposition result. For graph G, which contains a cycle exceeding size 4, when deleting node v_2 during tree decomposition, connect v_3 and v_5 to form a cycle. In the decomposed tree structure, place v_3, v_5, v_6 in one bag as a tree node. According to tree width definition, the decomposed tree width is $\max(|X_i| - 1 : i \in I)$, yielding a tree width of 2.

[Figure 2: see original paper] Graph and tree decomposition of graph

2.2 Tree Decomposition Algorithm and Tree Width Measurement

WP algorithms are highly effective for solving 3-SAT problems but often fail to converge in hard regions, rendering them ineffective. Therefore, studying WP algorithm convergence is crucial. Current systematic theoretical analysis of WP algorithm convergence on factor graphs remains insufficient. This paper uses tree width to measure factor graph structural complexity, establishing a factor graph tree width measurement model.

For given propositional formulas generated by $G(n, 3, m)$ model, we first convert them according to factor graph processing procedures, then use elimination ordering-based tree decomposition algorithms to generate corresponding tree decompositions. The tree decomposition algorithm used in experiments is an approximate decomposition algorithm, where the obtained tree width approximates the graph' s actual width.

In elimination ordering tree decomposition algorithms, elimination ordering affects tree decomposition solution quality—good elimination orderings yield tree widths closer to actual tree width and are mostly superior in solving time. Constructing a good elimination ordering is difficult. The LB algorithm performs well based on arbitrary elimination orderings for tree decomposition construction. Based on the LB algorithm, we add rules for randomly ordering nodes within clause nodes and factor nodes to generate elimination orderings, specifying the ordering sequence for the two node types. Under the elimination fill-in strategy, this ensures no edges are added between variable nodes and clause

nodes, thereby not affecting WP algorithm warning information propagation processes.

Elimination Ordering Rule: For each generated CNF formula, number variables and clause nodes—variable numbers are randomly generated within the range of variable count, clause node numbers are randomly generated after variable node numbers. According to tree decomposition rules, fill-in edges at this point do not affect WP algorithm operation.

The LB algorithm performs fill-in processing according to elimination ordering, achieving graph triangulation to obtain its chordal graph. For cycles existing in the graph, it uses its processing logic to perform fill-in operations, ensuring cycles exceeding size four contain chord edges. LB algorithm triangulation results are shown in Figure 4.

[Figure 4: see original paper] LB triangulation

For the LB algorithm requiring initialization sequences, here we provide the sequence $\{a, b, c, d, e, f\}$ according to node numbering in the graph. First process point a—its neighbor set $N_H[a]$ is $\{a, b, c\}$ (including itself), and the connected component using it as a separator is $\{d, e, f\}$. Add edges between d and f (dashed lines represent added edges). Repeat the process according to the sequence to achieve LB triangulation of graph G and generate a chordal graph.

Algorithm 3: LB Tree Decomposition Algorithm
Input: Graph $G = (V, E)$, sequence-restricted vertex sequence α
Output: Triangulated graph H and tree width

a) Initialize: $G \to H$

b) For x in α:
   - Compute neighbor set $N_H[x]$ to obtain connected components using it as a separator, add edges to form cycles among neighbor points of each connected component, record added edge set as F'
   - $F + F' \to F$; $(V, E \cup F) \to H$

c) For constructed graph H, construct bag sets in reverse elimination ordering, delete bags that are proper subsets of other bags, connect to generate decomposition tree

d) The number of nodes in the largest bag of decomposition tree minus 1 is the decomposition tree width

When solving k-SAT problems, WP algorithms can narrow hard regions. Using decomposition tree width parameters to classify NP-hard problems, since WP algorithms fail and exhibit non-convergence in hard regions, tree width-based convergence determination provides conditions for algorithm effectiveness while classifying k-SAT problems. This algorithm solves CNF formula tree

width, reflecting CNF formula structural complexity. The decomposition process transforms factor graphs into tree structures, which embody connections between different variables and clauses and correlations among original graph nodes within tree nodes. Using tree width parameters to utilize factor graph structural information for WP algorithm convergence analysis research, compared with other WP algorithm convergence studies that restrict propositional formula and graph model structures, this approach has unrestricted application scope and provides convenient determination conditions. It improves WP algorithm convergence analysis on factor graph structures, and as WP algorithm is the most basic among information propagation algorithms, it provides new ideas and directions for other information propagation algorithm convergence research.

3 Convergence Analysis

This paper uses model $G(n, 3, m)$ to generate random instances, where n is the number of variables, 3 represents clause length, and m is the number of clauses randomly and uniformly selected from all possible $\binom{n}{3}$ clauses. We select two experimental datasets with variable scales set to $n = 100$ and $n = 150$, respectively, with both experiments setting WP algorithm maximum iteration count to 1000.

Tree width is obtained using Algorithm 3, where m/n is represented by constraint density α. Research has shown that constraint density significantly impacts factor graph structure and WP algorithm convergence. Here α ranges from 2 to 5.5 with an increment gradient of 0.05. Since random SAT instances generated at the same constraint density may include 极少数 abnormal instances, to eliminate their impact, we take 100 groups of random instances at each constraint density. Experimental tree width values are the mean of tree widths obtained by Algorithm 3 for 100 instance groups, with non-integer means rounded. For each instance group, we statistically analyze WP algorithm convergence conditions and calculate convergence rates. SAT instance tree width variation with instance constraint density is shown in Figures 5 and 6.

Figures 5 and 6 describe the trend of SAT instance factor graph tree width changing with constraint density α growth for two variable scales of 100 and 150, with horizontal coordinates representing constraint density α. In two different SAT instance scales, as constraint density increases, tree width also increases, but its growth rate gradually slows, indicating that during tree decomposition, as factor graph scale increases and structure becomes complex, tree decomposition-generated tree node bag size growth slows, suggesting factor graph loop structure regions stabilize. Figures 7 and 8 compare WP algorithm convergence rates with SAT instance tree width, thereby presenting the relationship between tree width and convergence.

[Figure 5: see original paper] Tree Width distribution of SAT instance ($n = 100$)
[Figure 6: see original paper] Tree Width distribution of SAT instance ($n = 150$)

[Figure 7: see original paper] WP algorithm converges probability distribution ($n = 100$)
[Figure 8: see original paper] WP algorithm converges probability distribution ($n = 150$)

Figures 7 and 8 show WP algorithm convergence probability variation with tree width tw, counting WP algorithm convergence instances among 100 instance groups to calculate convergence probability. When tree width exceeds the threshold for current-scale instances, WP algorithm convergence probability drops sharply to non-convergence. For instance variable scale $n = 100$, when $tw \leq 62$, WP algorithm completely converges on random SAT instances; when $63 \leq tw \leq 67$, WP algorithm convergence probability on random SAT instances rapidly decreases from 1 to 0. For instance variable scale $n = 150$, when $tw \leq 91$, WP algorithm completely converges on all instances; when $92 \leq tw \leq 100$, WP algorithm convergence probability on random SAT instances rapidly drops to 0. Figures 7 and 8 demonstrate that WP algorithm converges with high probability on most instances, but once exceeding a certain threshold, convergence probability plummets, almost to non-convergence.

In random SAT instances, different tree widths are obtained for the same constraint density, corresponding to different loops in factor graphs and different variable node degrees. As constraint density increases, tree decomposition algorithm solving time also increases accordingly. In the tree decomposition algorithm described, edges are added within connected components formed by current nodes and their neighbors. As factor graph constraint density increases, loops increase correspondingly, increasing connected components and thus solving time. Meanwhile, increased constraint density means increased clause nodes, and tree decomposition width also relates to node count in the graph. When WP algorithm convergence probability suddenly drops, tree width still increases slowly—compared with random SAT instance scale, tree width magnitude and growth rate are far smaller than random instances. This provides a foundation for studying factor graph structural information and algorithm convergence for information propagation algorithms on hard instances.

Factor graph structure significantly impacts WP algorithm performance. Experiments provide sufficient conditions for WP algorithm convergence under two variable scales. This method can also obtain tree width thresholds for WP algorithm convergence in other-scale random SAT instances.

4 Conclusion

Information propagation algorithms are highly effective for solving 3-SAT problems. As a fundamental information propagation algorithm, studying WP algorithm properties significantly impacts other information propagation algorithms. This paper proposes an algorithm for measuring propositional formula structural features, using tree width to represent factor graph structural information. Experiments obtain tree width variations and WP algorithm convergence changes

for different-scale random instances, providing tree width sufficient conditions for algorithm convergence across different-scale random instances. Future work will improve tree decomposition algorithms to obtain decomposition tree widths closer to actual graph tree width, apply tree decomposition strategies to other information propagation algorithm convergence analysis, and conduct in-depth analysis and verification of the relationship between tree width and factor graph complexity.

References

[1] Dyer M, Frieze A, Molloy M. A probabilistic analysis of randomly generated binary constraint satisfaction problems [J]. Theoretical Computer Science, 2003, 290 (3): 1815-1828.

[2] Creignou N, Daudé H. The SAT–UNSAT transition for random constraint satisfaction problems [J]. Discrete mathematics, 2009, 309 (8): 2085-2098.

[3] Gaspers S, Papadimitriou C, Sæther S H, et al. On satisfiability problems with a linear structure [J]. arXiv preprint arXiv:1602.07876, 2016.

[4] Aiello A, Burattini E, Massarotti A. The Complexity of Theorem Proving Procedures [J]. Rairo Informat Théor, 1977, 11 (1): 75-82.

[5] Mézard M, Parisi G, Zecchina R. Analytic and algorithmic solution of random satisfiability problems [J]. Science, 2002, 297 (5582): 812-815.

[6] Weiss Y. Correctness of Local Probability Propagation in Graphical Models with Loops [J]. Neural computation, 2000, 12 (1): 1-41.

[7] Weiss Y, Freeman W T. Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology [J]. Neural Computation, 2001, 13 (10): 2173-2200.

[8] Maneva E, Mossel E, Wainwright M. A new look at survey propagation and its generalizations [J]. ACM, 2007, 54: 1089–1098.

[9] Wang Xiaofeng, Xu Daoyun, Wei Li. Convergence of warning propagation algorithms for random satisfiable instances [J]. Journal of Software, 2013, 24 (1): 1-11.

[10] Su Q, Wu Y C. Convergence analysis of the variance in Gaussian belief propagation [J]. IEEE Transactions on Signal Processing, 2014, 62 (19): 5119-5131.

[11] Su Q, Wu Y C. On convergence conditions of Gaussian belief propagation [J]. IEEE Transactions on Signal Processing, 2015, 63 (5): 1144-1155.

[12] Wang Xiaofeng, Xu Daoyun. Sufficient conditions for convergence of the warning propagation algorithm [J]. Journal of Software, 2016, 27 (12): 3003-3013.

[13] She Guangwei, Xu Daoyun. Modified Warning Propagation Algorithm for Solving Regular (3, 4)-SAT Instance Sets [J]. Computer Science, 2018, 45 (11): 312-317.

[14] Cui Li, Wang Xiaofeng, Niu Jin. Effective conditions for warning propagation algorithm convergence on WP solvable formula [J]. Application Research of Computers, 2020, 37 (05): 1406-1410.

[15] Niu Jin, Wang Xiaofeng, Lin Qingwen. Convergence analysis of warning propagation algorithm based on structural entropy [J]. Application Research of Computers, 2021, 38 (03): 760-763+776.

[16] Liang Chen, Wang Xiaofeng, Liu Zilin, et al. Convergence analysis of survey propagation algorithm based on K-dimensional structure entropy [J/OL]. Application Research of Computers, 1-6 [2022-01-15].

[17] Robertson N, Seymour P D. Graph minors. X. Obstructions to tree-decomposition [J]. Journal of Combinatorial Theory, Series B, 1991, 52: 153-190.

[18] Chen H. Quantified constraint satisfaction and bounded treewidth [C]//ECAI. 2004, 16: 161.

[19] Krishnamurthy S, Sahoo S. Balanced k-satisfiability and biased random k-satisfiability on trees [J]. Physical Review E, 2013, 87 (4): 042130.

[20] Lei Ying, Xu Daoyun. Algorithm of Graph and Its Application [J]. Computer Science, 2020, 47 (05): 51-58.

[21] Kschischang F R, Frey B J, Loeliger H A. Factor graphs and the sum-product algorithm [J]. IEEE Transactions on Information Theory, 2001, 47 (2): 498-519.

[22] Braunstein A, Mézard M, Zecchina R. Survey propagation: An algorithm for satisfiability [J]. Random Structures & Algorithms, 2005, 27 (2): 201-226.

[23] Hammerl T, Musliu N, Schafhauser W. Metaheuristic algorithms and tree decomposition [M]//Springer Handbook of Computational Intelligence. Springer, Berlin, Heidelberg, 2015: 1255-1270.

Note: Figure translations are in progress. See original paper for figures.