Generating the Top K Solutions to Weighted CSPs: A Comparison of Different Approaches

Ang Li
Department of Computer Science
University of Southern California
ali355@usc.edu

Yuling Guan
Department of Physics and Astronomy
University of Southern California
yulinggu@usc.edu

Sven Koenig
Department of Computer Science
University of Southern California
skoenig@usc.edu

Stephan Haas
Department of Physics and Astronomy
University of Southern California
shaas@usc.edu

T. K. Satish Kumar
Departments of Computer Science and Industrial and Systems Engineering
University of Southern California
tkskwork@gmail.com

***Abstract*—The weighted constraint satisfaction problem (WCSP) is a general and very useful combinatorial optimization tool. Despite its importance, the task of generating the top K solutions to it is understudied. One benefit of generating the top K solutions is in creating a framework for "human-in-the-loop AI". Most real-world problems cannot be modeled accurately/completely up front and, hence, generating the top K solutions gives users a chance to exercise preferences that are not explicitly included in the modeling phase. In this paper, we first discuss the importance of generating the top K solutions to WCSPs in various contexts. We then propose various approaches to do so and empirically compare them. We include approaches based on quadratization, pseudo-Boolean optimization, constraint propagation, and integer linear programming. Together, they cover all major algorithmic ingredients derived from constraint programming (CP), artificial intelligence (AI), and operations research (OR).**

***Index Terms*—Weighted CSP; Top K Solutions.**

I. Introduction

The weighted constraint satisfaction problem (WCSP) is a combinatorial optimization problem and a generalization of the constraint satisfaction problem (CSP). Each tuple in a constraint—i.e., an assignment of values to all variables in that constraint—is associated with a non-negative weight (sometimes referred to as "cost"). The goal is to find an assignment of values to all variables from their respective domains such that the total weight is minimized [1].

More formally, the WCSP is defined by a triplet $\mathcal{B} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{X_1, X_2, \dots, X_N\}$ is a set of N variables, $\mathcal{D} = \{D_1, D_2, \dots, D_N\}$ is a set of N domains with discrete values, and $\mathcal{C} = \{C_1, C_2, \dots, C_M\}$ is a set of M weighted constraints. Each variable $X_i \in \mathcal{X}$ can be assigned a value in its associated domain $D_i \in \mathcal{D}$. Each constraint $C_i \in \mathcal{C}$ is defined over a certain subset of the variables $S_i \subseteq \mathcal{X}$, called the scope of C_i. C_i associates a non-negative weight with each possible assignment of values to the variables in S_i. (For notational convenience, we use S_i and C_i interchangeably throughout this paper when referring to the variables participating in a weighted constraint, e.g., $X_k \in C_i \equiv X_k \in S_i$.) The goal is to find an assignment of values to all variables in $\mathcal{X}$ from their respective domains that minimizes the sum of the weights specified by each weighted constraint in $\mathcal{C}$ [1]. This combinatorial task can equivalently be characterized by having to compute

$$\underset{a \in \mathcal{A}(\mathcal{X})}{\arg\min} \sum_{C_i \in \mathcal{C}} E_{C_i}(a|C_i), \tag{1}$$

where $\mathcal{A}(\mathcal{X})$ represents the set of all $|D_1| \times |D_2| \times \ldots \times |D_N|$ complete assignments to all variables in $\mathcal{X}$. $a|C_i$ represents the projection of a complete assignment a onto the subset of variables in C_i. E_{C_i} is a function that maps each $a|C_i$ to its associated weight in C_i.

The Boolean WCSP is the WCSP in which each domain $D_i \in \mathcal{D}$ has its cardinality restricted to be 2. Despite this restriction, the Boolean WCSP is representationally as powerful as the WCSP, and it is also NP-hard to solve in general. The (Boolean) WCSP can be used to model a wide range of useful combinatorial problems. For example, in artificial intelligence (AI), it can be used to model user preferences [2] and combinatorial auctions. In bioinformatics, it can be used to locate RNA motifs [3]. In statistical physics, the energy minimization problem on the Potts model is equivalent to that on its corresponding pairwise Markov random field [4], which in turn can be modeled as the WCSP. In computer vision, it can be used for image restoration and panoramic image stitching [5], [6].

Despite the importance of the WCSP, the problem of generating the top K solutions to it has not been studied much. An important benefit of generating the top K solutions is in creating a framework for "human-in-the-loop AI". Most real-world problems cannot be modeled accurately/completely up front and, hence, generating the top K solutions gives users a chance to exercise preferences that are not explicitly included in the modeling phase. It also facilitates knowledge elicitation since users can choose viable solutions and reject others, declaring reasons for doing so that can then be incorporated for further reasoning.

One example domain is in hypothesis selection over knowledge graphs (KGs). A KG is an effective representation of knowledge. It consists of a collection of knowledge elements, each of which in turn is extracted from the web or other sources. Information extractors that use natural language processing techniques or other complex algorithms are usually noisy. That is, the vast number of knowledge elements extracted from the web may not only be associated with different confidence values but may also be inconsistent with each other. Moreover, there might be additional domain knowledge available in the form of ontological constraints. Many applications such as question-answering systems that are built on top of large-scale KGs are required to generate the top K hypotheses, i.e., coherent subgraphs of a KG that are consistent with the ontological constraints and that are of high confidence values. This problem can be reformulated as the problem of generating the top K solutions to a WCSP [7].

A second related benefit is in quickly adapting to a dynamically changing environment. Suppose a timetabling problem is formulated and solved as a WCSP. If the top solution to it becomes unviable due to an unexpected change in one of the timetabling constraints or preferences, the next best viable solution can be sought within the available list of top K solutions before attempting to solve a new WCSP. Therefore, generating the top K solutions serves as a caching of viable solutions that can be used to adapt to unforeseen changes in the environment. Such a framework is also very useful in planning domains where new information gathered from the real world at execution time can make certain plans unviable.

A third benefit is in computational physics. At the microscopic level, a material is composed of particles with associated spins. The interactions between spins create ferromagnetic, anti-ferromagnetic or other interaction potentials between them similar to those between variables in a WCSP. However, the macroscopic observables of the material, like its total magnetization, are not merely properties of a single spin configuration but are instead properties of all possible spin configurations summed via the Boltzmann equation of statistical mechanics [8]. Since the Boltzmann equation involves exponentials, under certain weak assumptions, it can be approximated well using the top K solutions of the interaction potentials. Generating the top K solutions of the interaction potentials can be used to study macroscopic properties such as magnetic hysteresis [9].

In this paper, we propose various approaches to generate the top K solutions to WCSPs and empirically compare them. We include approaches based on quadratization, pseudo-Boolean optimization, constraint propagation, and integer linear programming (ILP). Together, they cover all major algorithmic ingredients derived from constraint programming (CP), AI, and operations research (OR).

II. Top K Solutions to WCSPs

The task of generating the top K solutions $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$ needs a more formal specification since it can be conceived in many ways. The following are some conceivable methods.

- **Method A** is to request the solutions $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$ such that $\mathcal{S}_1$ is the top solution, $\mathcal{S}_2$ is the second best solution and differs from $\mathcal{S}_1$ in the assignment of values to at least one of the N variables, $\mathcal{S}_3$ is the third best solution and differs from both $\mathcal{S}_1$ and $\mathcal{S}_2$ in the assignment of values to at least one of the N variables, and so forth.
- **Method B** is to request the solutions $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$ such that $\mathcal{S}_1$ is the top solution, $\mathcal{S}_2$ is the second best solution and differs from $\mathcal{S}_1$ in the assignment of values to at least d of the N variables, $\mathcal{S}_3$ is the third best solution and differs from both $\mathcal{S}_1$ and $\mathcal{S}_2$ in the assignment of values to at least d of the N variables, and so forth.
- **Method C** is to request the solutions $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$ such that the sum of their respective costs $c_1, c_2, \dots, c_K$ is minimized and any two distinct $\mathcal{S}_i$, $\mathcal{S}_j$ differ in the assignment of values to at least one of the N variables.
- **Method D** is to request the solutions $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_K$ such that the sum of their respective costs $c_1, c_2, \dots, c_K$ is minimized and any two distinct $\mathcal{S}_i$, $\mathcal{S}_j$ differ in the assignment of values to at least d of the N variables.

We first note that Methods A and C are equivalent but Methods B and D are not equivalent for fixed K.[1] Therefore, in the rest of this paper, we focus on Methods A ($\equiv$ C) and B, with Method B being more general. We omit discussion on Method D noting that it requires a fixed value of K and is equivalent to solving a larger WCSP with KN variables.

III. Methodologies

Methods to generate only the top solution to WCSPs have been studied in a number of previous works. Many of these successful methods, applicable to both Boolean and non-Boolean variables, have been incorporated in Toulbar2 [10], a state-of-the-art WCSP solver. Finding the top solution to WCSPs can also be reformulated as the minimum weighted vertex cover problem [11]–[13]. Moreover, for the special case of Boolean variables, finding the top solution can also be done via pseudo-Boolean optimization [14].

Despite the existence of many works for finding the top solution, the problem of finding the top K solutions to WCSPs is understudied.[2] [7] uses heuristic methods to generate the top K solutions but does not provide any theoretical guarantees. Moreover, many methods, such as those based on analyzing the variable-interaction graphs [16], are known to work specifically for generating the top solution but are not applicable to generating the top K solutions. This is because generating the top K solutions involves global constraints even if the original WCSP does not.

In this section, we provide methodologies to make the top solution techniques for WCSPs applicable for generating the top K solutions as well. We first note that in order to generate

[1]Suppose we have only two Boolean variables X_1 and X_2 with costs $c(X_1 = 0, X_2 = 0) = 1$, $c(X_1 = 0, X_2 = 1) = 2$, $c(X_1 = 1, X_2 = 0) = 2$ and $c(X_1 = 1, X_2 = 1) = 4$. For $K = 2$ and $d = 2$, Methods B and D produce different results.

[2]For CSPs, the equivalent problem of generating K solutions is relatively well studied [15].

the k^{th} solution for $1 \leq k \leq K$, prohibitive constraints are added in the k^{th} iteration to prevent the top $k-1$ solutions found thus far. These prohibitive constraints are naturally global constraints. They can be dealt with in different ways in different frameworks.

A. Quadratization

A prohibitive constraint is required to enforce a difference between the k^{th} solution and each of the top $k-1$ solutions found thus far in the values assigned to at least d variables. Stated directly in the language of weighted constraints, a prohibitive constraint is a global constraint that involves all variables. Therefore, its tabular representation is exponential in N, rendering its explicit encoding for WCSP solvers completely unviable.

To circumvent this problem, we propose the use of *quadratization*. In essence, quadratization refers to the idea of decomposing higher-arity interactions between variables to only binary interactions between them but at the cost of introducing auxiliary variables [17]. While quadratization is hard to study for general functions, quadratization of functions on Boolean variables has received more attention. Recent progress in this field suggests that a *symmetric* function on Boolean variables can be decomposed to a sum of quadratic functions on the same Boolean variables plus a logarithmic number of auxiliary Boolean variables [18].

This theory can be applied to WCSPs with only Boolean variables. In such cases, the prohibitive global constraints are in fact symmetric Boolean functions since they only specify *how many* variables, as opposed to *which* variables, should have differing assignments compared to each of the top $k-1$ solutions found thus far. Therefore, these global constraints can be decomposed to binary weighted constraints easily.

For a WCSP solver such as Toulbar2, the binary weighted constraints coming from the decomposition of the prohibitive global constraints have simple tabular representations. They can be added to the set of original weighted constraints before invoking the solver in the k^{th} iteration to obtain the k^{th} solution. For WCSPs that have only unary and binary weighted constraints, quadratic pseudo-Boolean optimization (QPBO) solvers can be invoked [14]. Moreover, since the prohibitive global constraints can also be decomposed to binary constraints, they can be added to the objective function in the k^{th} iteration to obtain the k^{th} solution without compromising the quadratic form.

B. ILP

Without loss of generality, we first assume that every variable has a unique unary weighted constraint associated with it. If there are multiple unary weighted constraints associated with a variable, they can be combined into one; and if there are no unary weighted constraints associated with a variable, one with all weights set to zero can be introduced artificially.

Suppose we denote the top ℓ^{th} solution as follows: $\mathcal{S}_\ell \equiv (X_1 = v^\ell_{X_1}, X_2 = v^\ell_{X_2}, \ldots, X_N = v^\ell_{X_N}\}$.[3] Extending on our previous work in [19], the problem of generating the k^{th} solution, for $1 \leq k \leq K$, can be formulated as an ILP with only Boolean variables as follows.

$$\begin{aligned}
\underset{q^C_a : q^C_a \in \boldsymbol{q}}{\text{minimize}} \quad & \sum_{C \in \mathcal{C}} \sum_{a \in A(S(C))} w^C_a q^C_a \\
\text{s.t.} \quad q^C_a & \in \{0,1\} \quad \forall q^C_a \in \boldsymbol{q} \\
\sum_{a \in A(S(C))} q^C_a & = 1 \quad \forall C \in \mathcal{C} \\
\sum_{a \in A(S(C)) : a|S(C') = a'} q^C_a & = q^{C'}_{a'} \\
\forall C, C' \in \mathcal{C} : |S(C')| = 1 \wedge & S(C') \subset S(C), \forall a' \in A(S(C')) \\
\sum_{C \in \mathcal{C} : |S(C)| = 1} (1 - q^C_{v^\ell_{S(C)}}) & \geq d \quad \forall 1 \leq \ell \leq k-1,
\end{aligned}$$

where $\boldsymbol{q} = \{q^C_a \mid C \in \mathcal{C} \wedge a \in A(S(C))\}$, and w^C_a denotes the weight of assignment a specified by constraint C. The cardinality of $\boldsymbol{q}$ is $\sum_{C \in \mathcal{C}} \prod_{X \in S(C)} |D(X)|$. The first line represents the minimization of the sum of weights. The second line represents the ILP constraints that enforce the Boolean property for all q^C_a's. It consists of $\sum_{C \in \mathcal{C}} \prod_{X \in S(C)} |D(X)| = \mathcal{O}\left(|\mathcal{C}|\hat{D}^{\hat{C}}\right)$ ILP constraints, where $\hat{C} = \max_{C \in \mathcal{C}} |S(C)|$ and $\hat{D} = \max_{X \in \mathcal{X}} |D(X)|$. The third line represents the ILP constraints that enforce a unique assignment of values to variables in each WCSP constraint. It consists of $|\mathcal{C}|$ ILP constraints, each of which has $|A(S(C))| = \prod_{X \in S(C)} |D(X)| = \mathcal{O}\left(\hat{D}^{\hat{C}}\right)$ variables. The fourth line represents the ILP constraints which enforce that every two assignments in two WCSP constraints must be consistent on their shared variables. It consists of $\mathcal{O}\left(|\mathcal{C}| \cdot \hat{C} \cdot \hat{D}\right)$ ILP constraints. Each of these ILP constraints has $\mathcal{O}\left(\hat{D}^{\hat{C}-1}\right)$ variables. The last line represents the prohibitive global constraints that enforce a difference from each of the top $k-1$ solutions found thus far in the values assigned to at least d variables. It consists of $k-1$ ILP constraints, each of which has N variables.

IV. Experimental Results

We now provide experimental results that compare the various methods for generating the top K solutions to WCSPs. All experiments were run on a laptop with a 3.1GHz quad-core Intel Core i7 processor and 16GB 2133MHz LPDDR3 memory. All running times are measured in seconds. We used three datasets for the experiments: the UAI dataset, the Ising model dataset, and the Erdös-Rényi dataset.

The UAI dataset is made available by the University of California, Irvine via the link: http://sli.ics.uci.edu/~ihler/uai-data/. We used 17 available WCSP instances. These instances have maximum domain size ≤ 10 with only unary and binary weighted constraints.

The WCSP instances in the Ising model dataset were generated as follows. We used a 40×40 2-dimensional lattice structure of Ising spin variables. We considered nearest-neighbor interactions without an external magnetic field. An

[3] For notational convenience, $v^\ell_{X_N}$ will also be written as $v^\ell_{\{X_N\}}$.

Name \ K	1	2	3	4	5	6
29	0.045406	0.099680	0.159102	0.216908	0.272228	0.324488
DSJC125	0.106832	0.217126	0.349660	0.458314	13.261575	37.407089
GEOM30a_3	0.010978	0.023778	0.034906	0.105019	0.196941	0.269005
GEOM30a_4	0.014780	0.031078	0.046270	0.059286	0.157992	0.227041
GEOM30a_5	0.016725	0.034106	0.054135	0.078917	0.096374	0.223302
driverlog01ac	0.097509	0.196127	0.285424	0.374978	0.469777	0.563200
driverlog02ac	876.335637	1777.000926	2523.380823	4066.095331	4951.544352	6052.300541
le450_5a_2	0.298482	0.592172	59.360335	166.525465	210.161505	265.589398
le450_5a_3	0.726636	1.443531	2.160011	408.277814	6252.195417	7797.841997
le450_5a_4	1.050851	2.168919	17.118558	18.248981	8031.845313	53194.151083
myciel5g_3	0.022384	0.047081	0.069788	0.723032	1.478215	2.262733
myciel5g_4	0.031199	0.063706	0.101349	0.132265	1.803422	4.224769
myciel5g_5	0.049457	0.096153	0.151467	0.206150	0.255206	7.421422
queen5_5_3	0.015450	0.030404	0.043798	0.622870	1.108807	1.609571
queen5_5_4	0.020579	0.042416	0.068729	0.090424	1.629419	2.786443
satellite01ac	50.114745	101.713977	153.408043	205.403035	258.982060	313.314846
satellite02ac	211.403276	437.370921	684.208177	958.824607	1207.126499	1487.134897

TABLE I: Gurobi on the UAI dataset with $d = 1$.

$p \setminus K$	1	2	3	4	5	6
0.0	0.638326	0.856472	7.123784	8.919827	16.910342	18.909787
0.1	75.948327	152.059466	236.236200	319.718994	431.744436	526.693602
0.2	88.022129	177.322234	281.253682	386.028938	616.844662	651.965991
0.3	90.149931	231.156042	338.193184	451.910688	605.158208	725.752342
0.4	96.800210	227.634564	339.146524	452.696057	600.612583	721.518038
0.5	91.156919	184.369873	290.984962	395.283319	531.946094	639.167596
0.6	92.009285	226.944586	324.766628	427.980900	599.313912	657.587034
0.7	94.771781	188.455620	297.191212	391.415626	652.865360	646.465575
0.8	93.545905	211.445277	323.273459	435.189458	582.965451	702.248596
0.9	84.854042	161.436855	253.349112	336.670010	465.082251	551.220683
1.0	0.716885	0.948226	12.168897	16.772620	19.766996	42.548012

TABLE II: Gurobi on the Ising model dataset with $d = 1$.

interaction between two nearest-neighbor spins can either be ferromagnetic or anti-ferromagnetic. The control parameter p determined the fraction of anti-ferromagnetic spin interactions. We varied p from 0 to 1, with step size 0.1. For each of the 11 possible values of p, we averaged our results over 10 instances.

The WCSP instances in the Erdös-Rényi dataset were generated as follows. We first generated Erdös-Rényi graphs [20] with 60 nodes each. Each node represents a Boolean variable, and the probability parameter p determines the presence of an edge between any two distinct nodes. An edge represents a binary weighed constraint between the two variables representing its endpoint nodes. Each weight in a weighted constraint was randomly chosen to be an integer in the interval $[0, 4]$. We varied p from 0.1 to 0.9 with step size 0.1.[4]

In our experiments, we compared the following solvers: (a) the Gurobi Optimizer [21], using ILP formulations, (b) Toulbar2 [10], a state-of-the-art WCSP solver, and (c) qpboMex [14], a state-of-the-art QPBO solver. To generate the k^{th} solution for $1 \leq k \leq K$, prohibitive constraints are added in the k^{th} iteration to prevent the top $k - 1$ solutions. These prohibitive constraints are global constraints. They were encoded as linear inequality constraints suitable for Gurobi or as quadratic symmetric function constraints suitable for Toulbar2 and qpboMex. However, the quadratic symmetric function constraints are applicable only for Boolean variables [18]. In addition, qpboMex also allows only for Boolean variables.

In the first subsection, we compare different methods and observe that Gurobi significantly outperforms other methods. In the second subsection, we study the scaling behavior of Gurobi with respect to increasing values of K and d.

$p \setminus K$	1	2	3	4	5	6
0.1	0.016447	0.031001	0.046962	0.062126	0.083085	0.103564
0.2	0.644113	1.207603	1.706096	2.155048	2.635259	3.104022
0.3	0.898389	1.613020	2.381056	3.180436	3.974385	4.785877
0.4	22.256275	43.734139	64.237934	81.844331	107.726265	130.647845
0.5	113.138839	217.982236	362.823347	512.906342	639.578730	757.373041
0.6	113.771927	286.496653	418.216053	533.634755	691.852802	817.851893
0.7	204.130670	384.311431	590.988404	781.153543	978.822693	1197.676624
0.8	203.865396	447.951235	726.605473	952.589703	1206.942405	1470.229164
0.9	1058.537818	2250.422945	3190.207245	4220.987575	5415.125955	6747.523805

TABLE III: Gurobi on the Erdös-Rényi dataset with $d = 1$.

p	$K = 1$
0.1	0.012589
0.2	0.028718
0.3	0.038424
0.4	1.917937
0.5	26.899478
0.6	21.694742
0.7	70.020841
0.8	148.281038
0.9	1168.726508

TABLE IV: Toulbar2 on the Erdös-Rényi dataset with $d = 1$.

A. Comparison of Different Methods

Table I shows the performance of Gurobi on the UAI dataset for different values of K with $d = 1$. The entries indicate the cumulative time required to generate the top K solutions. On all these instances, Toulbar2 failed to generate even the top solution since the problem sizes were deemed to be too large. qpboMex was applicable to only 1 instance 'le450_5a_2'. It took $0.075s$ for qpboMex to generate the optimal solution for this case; but it failed to generate other suboptimal solutions.[5]

Table II shows the performance of Gurobi on the Ising model dataset for different values of p and K with $d = 1$. The entries indicate the cumulative time required to generate the top K solutions; and a time limit of $300s$ was given to each of the K iterations. On these instances, Toulbar2 was able to generate only the top solution and only when $p = 0$ or $p = 1$. Its average running time on the successful instances for $p = 0$ and $p = 1$ was $0.286s$ and $0.285s$, respectively. In all other cases, it timed out. The performance of qpboMex was very similar to that of Toulbar2. It was able to generate only the top solutions and only when $p = 0$ or $p = 1$. Its average running time on the successful instances for $p = 0$ and $p = 1$ was $0.042s$ and $0.041s$, respectively.

Table III shows the performance of Gurobi on the Erdös-Rényi dataset for different values of p and K with $d = 1$. The entries indicate the cumulative time required to generate the top K solutions; and a time limit of $1200s$ was given to each of the K iterations. Toulbar2 was able to generate only the top solution for each instance, with running times shown in Table IV. In all other cases, it timed out. qpboMex could only generate the top solution for one case, i.e., for $p = 0.1$. It took $0.004s$ for qpboMex to generate the optimal solution for this case; but it failed to generate other suboptimal solutions.

From these results, it is easy to conclude that Gurobi is currently the only viable method among the existing off-the-shelf solvers for generating the top K solutions to WCSPs

[4]Without averaging over 10 instances for each possible value of p, we report on 9 individual instances since they are indicative of the general trends.

[5]qpboMex returns a specific exit code to indicate that it cannot solve a problem instance.

Name \ K	1	2	3	4	5	6
29	0.044889	0.093653	0.143490	0.187253	0.234077	0.279256
DSJC125	0.090875	0.183604	0.294834	0.389074	38.485259	73.381663
GEOM30a_3	0.009502	0.020100	0.029540	0.092820	0.157183	0.222529
GEOM30a_4	0.015141	0.029802	0.044493	0.056798	0.128169	0.194509
GEOM30a_5	0.016150	0.033638	0.053905	0.078138	0.094940	0.227355
driverlog01ac	0.103221	0.207878	0.306139	0.403821	0.498249	0.612439
driverlog02ac	867.613675	1907.417619	3108.379250	3704.813593	4463.120142	5483.314061
le450_5a_2	0.297118	0.599144	176.928288	435.679361	543.542576	879.377756
le450_5a_3	0.612869	1.246977	1.869716	Time Out	Time Out	Time Out
le450_5a_4	1.308174	2.558808	15.769929	16.902670	Time Out	Time Out
myciel5g_3	0.022216	0.045431	0.067313	0.992660	2.797885	3.936749
myciel5g_4	0.032702	0.065108	0.103696	0.136518	4.206785	7.676406
myciel5g_5	0.048541	0.097438	0.153795	0.209105	0.254415	5.146981
queen5_5_3	0.015447	0.029199	0.043584	1.860184	4.377596	6.042508
queen5_5_4	0.030955	0.057675	0.085025	0.107589	3.161552	12.332942
satellite01ac	51.777251	101.707292	153.824613	204.074684	256.636463	310.895090
satellite02ac	208.870650	434.627622	655.586530	878.524620	1099.588159	1325.111695

TABLE V: Gurobi on the UAI dataset with $d = 2$.

Name \ K	1	2	3	4	5	6
29	0.044668	0.097963	0.155840	0.205683	0.255335	0.304775
DSJC125	0.091101	0.182812	0.297171	0.410300	26.060292	141.772509
GEOM30a_3	0.010676	0.022573	0.033179	0.091357	0.144641	0.199868
GEOM30a_4	0.015035	0.029429	0.043880	0.056533	0.159064	0.248232
GEOM30a_5	0.019166	0.042792	0.067256	0.096821	0.116154	0.247865
driverlog01ac	0.106663	0.216099	0.321633	0.412149	0.501990	0.617877
driverlog02ac	868.608740	1891.746099	2970.052148	3927.390976	4882.853929	5987.147603
le450_5a_2	0.301343	0.592013	968.398253	2198.892417	3315.096333	5003.164408
le450_5a_3	0.634132	1.274331	1.910956	Time Out	Time Out	Time Out
le450_5a_4	1.088893	2.215917	20.257083	21.391217	Time Out	Time Out
myciel5g_3	0.022054	0.045229	0.067041	1.571629	2.455481	5.577489
myciel5g_4	0.036597	0.076109	0.124124	0.162250	5.096919	13.061579
myciel5g_5	0.057012	0.112605	0.178010	0.247078	0.294847	8.660527
queen5_5_3	0.015492	0.031395	0.046358	1.696383	7.339393	10.740527
queen5_5_4	0.021755	0.043642	0.069763	0.091862	4.289730	19.510938
satellite01ac	51.002272	99.963361	153.121990	204.057705	256.304444	307.728991
satellite02ac	257.838993	532.110179	806.995655	1083.737633	1359.451652	1621.945288

TABLE VI: Gurobi on the UAI dataset with $d = 3$.

even with $d = 1$. This might be in part due to the fact that matrix manipulations help OR methods reason about global constraints much more efficiently than other methods.

B. Further Experiments with Gurobi

Because Gurobi seems to be the only viable method for generating the top K solutions, we were able to conduct further exclusive experiments with it. In this subsection, we report on two such kinds of experiments. First, we wanted to understand how the running time of Gurobi scales with K, retaining $d = 1$. Second, we wanted to understand how it performs for higher values of d.

Figure 1 shows the scaling behavior of Gurobi on some selected problem instances for increasing values of K and $d = 1$. For many instances, the scaling is linear, as in UAI 29 and Erdös-Rényi $p = 0.5$. This is very encouraging since the added complexity of generating the top K solutions leads to only a linear increase in the running time of Gurobi, making it viable for a human-in-the-loop AI framework. Of course, there are some interesting exceptions, as in UAI le450_5a_3 and Ising model $p = 0$. Here, the problem instances become harder—not after the first but—after the third or fourth introduction of prohibitive global constraints.

Tables V,VI&VII show the performance of Gurobi on the UAI dataset for different values of K with $d = 2$, $d = 3$ and $d = 4$, respectively. The entries indicate the cumulative time required to generate the top K solutions; and a time limit of $3600s$ was given to each of the K iterations. Gurobi's ability to solve most of these problem instances is also very encouraging from the perspective of human-in-the-loop AI since users can control the desired "difference" between solutions (hypotheses).

Name \ K	1	2	3	4	5	6
29	0.046694	0.105542	0.164010	0.216135	0.269785	0.322676
DSJC125	0.098438	0.193016	0.307094	0.398736	93.707280	277.215449
GEOM30a_3	0.010637	0.022767	0.033709	0.085857	0.135265	0.187612
GEOM30a_4	0.015244	0.029354	0.044004	0.056564	0.146394	0.213167
GEOM30a_5	0.017284	0.039570	0.061497	0.086970	0.103684	0.207135
driverlog01ac	0.106813	0.213720	0.331723	0.426196	0.522932	0.647410
driverlog02ac	873.423437	1747.764080	2735.096566	3689.474908	4672.599442	5461.077691
le450_5a_2	0.306906	0.606134	2182.485199	5146.494628	Time Out	Time Out
le450_5a_3	0.610946	1.278551	1.905660	Time Out	Time Out	Time Out
le450_5a_4	1.080703	2.196142	18.394387	19.517824	Time Out	Time Out
myciel5g_3	0.023438	0.048679	0.072588	3.167216	7.341393	11.570325
myciel5g_4	0.031082	0.066118	0.108446	0.140856	7.670225	17.986415
myciel5g_5	0.056299	0.109280	0.168188	0.223375	0.271757	10.809051
queen5_5_3	0.015610	0.030363	0.044272	3.047800	9.151658	15.738343
queen5_5_4	0.021841	0.044804	0.073067	0.095571	5.327037	26.360986
satellite01ac	50.773345	99.318204	152.345519	202.980463	256.221921	306.026177
satellite02ac	245.764104	519.915170	790.435468	1062.916955	1333.381003	1567.425389

TABLE VII: Gurobi on the UAI dataset with $d = 4$.

V. Conclusions and Future Work

In this paper, we introduced the problem of generating the top K solutions to WCSPs. While WCSPs themselves are used to model a wide range of combinatorial optimization problems, generating the top K solutions to them is important from the perspective of "human-in-the-loop AI" and in computational physics. Despite the significance of generating the top K solutions, the problem is largely understudied in AI, theoretical computer science and computational physics. In this paper, we used various off-the-shelf methods and empirically compared them on a variety of WCSP instances. We included methods based on quadratization, pseudo-Boolean optimization, constraint propagation, and ILP. Together, they covered all major algorithmic ingredients derived from CP, AI and OR. We found that Gurobi alone is viable in producing the top K solutions to WCSPs using an ILP formulation.

There are many avenues for future work. In terms of techniques, we will develop new methods based on propagating global constraints that encode symmetric functions. In terms of applications, we will apply them to various problems in AI and computational physics.

References

[1] S. Bistarelli, U. Montanari, F. Rossi, T. Schiex, G. Verfaillie, and H. Fargier, "Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison," *Constraints*, vol. 4, no. 3, pp. 199–240, 1999.

[2] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements," *Journal of Artificial Intelligence Research*, vol. 21, pp. 135–191, 2004.

[3] M. Zytnicki, C. Gaspin, and T. Schiex, "DARN! A weighted constraint solver for RNA motif localization," *Constraints*, vol. 13, no. 1, pp. 91–109, 2008.

[4] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Exploring Artificial Intelligence in the New Millennium*, vol. 8, pp. 236–239, 2003.

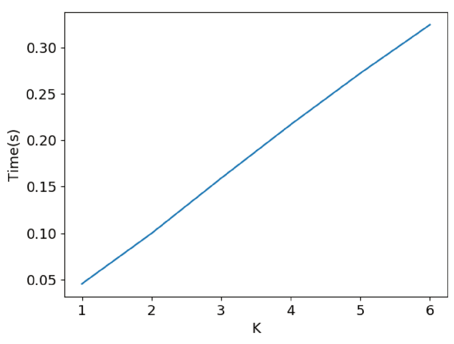
[5] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[6] V. Kolmogorov, "Primal-dual algorithm for convex Markov random fields," Microsoft Research, Tech. Rep. MSR-TR-2005-117, 2005.

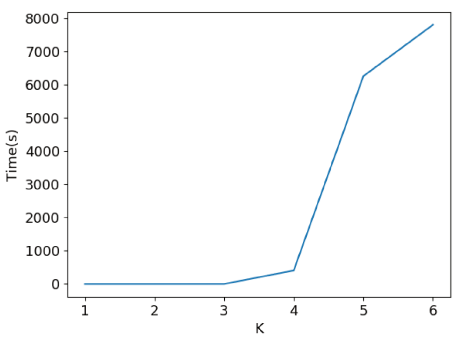
[7] K. Sun, K. Maddali, S. Salian, and T. K. S. Kumar, "Top K hypotheses selection on a knowledge graph," in *Proceedings of the Thirty-Second International FLAIRS Conference*, 2019.

[8] M. Kardar, *Statistical Physics of Particles*. Cambridge University Press, 2007.

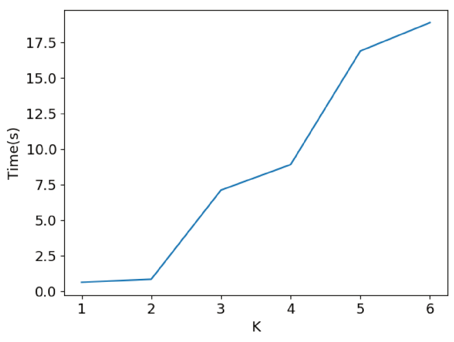
[9] S. L. Whittenburg, N. Dao, and C. A. Ross, "Micromagnetic studies of hysteresis in nickel pillars," *Physica B: Condensed Matter*, vol. 306, no. 1, pp. 44–46, 2001.



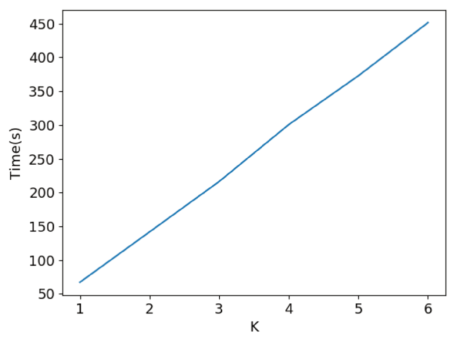
(a) UAI 29



(b) UAI le450_5a_3



(c) Ising model $p = 0$



(d) Erdös-Rényi $p = 0.5$

Fig. 1: Shows the scaling behavior of Gurobi for increasing values of K on four problem instances: (a) UAI 29; (b) UAI le450_5a_3; (c) Ising model $p = 0$; and (d) Erdös-Rényi $p = 0.5$. The X-axis represents K, and the Y-axis represents the cumulative time required to obtain the top K solutions.

[10] B. Hurley, B. O'sullivan, D. Allouche, G. Katsirelos, T. Schiex, M. Zytnicki, and S. D. Givry, "Multi-language evaluation of exact solvers in graphical model discrete optimization," *Constraints*, vol. 21, no. 3, pp. 413–434, 2016.

[11] T. K. S. Kumar, "A framework for hybrid tractability results in Boolean weighted constraint satisfaction problems," in *Proceedings of the Fourteenth International Conference on Principles and Practice of Constraint Programming*, 2008, pp. 282–297.

[12] H. Xu, T. K. S. Kumar, and S. Koenig, "The Nemhauser-Trotter reduction and lifted message passing for the weighted CSP," in *Proceedings of the Fourteenth International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming*, 2017, pp. 387–402.

[13] H. Xu, S. Koenig, and T. K. S. Kumar, "A constraint composite graph-based ilp encoding of the boolean weighted csp," in *Proceedings of the Twenty-Third International Conference on Principles and Practice of Constraint Programming*, 2017, pp. 630–638.

[14] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts - a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 7, pp. 1274–1279, 2007.

[15] R. Dechter, K. Kask, E. Bin, and R. Emek, "Generating random solutions for constraint satisfaction problems," in *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, 2002, pp. 15–21.

[16] R. Dechter, "Constraint networks," *Artificial Intelligence*, vol. 49, pp. 61–95, 1992.

[17] K. W. Yip, H. Xu, S. Koenig, and T. K. S. Kumar, "Quadratic reformulation of nonlinear pseudo-boolean functions via the constraint composite graph," in *Proceedings of the Sixteenth International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming*, 2019, pp. 643–660.

[18] E. Boros, Y. Crama, and E. Rodríguez-Heck, "Quadratizations of symmetric pseudo-boolean functions: Sub-linear bounds on the number of auxiliary variables," in *Proceedings of the Fifteenth International Symposium on Artificial Intelligence and Mathematics*, 2018.

[19] H. Xu, K. Sun, S. Koenig, I. Hen, and T. K. S. Kumar, "Hybrid quantum-classical algorithms for solving the weighted csp," in *Proceedings of the Sixteenth International Symposium on Artificial Intelligence and Mathematics*, 2020.

[20] E. P. and R. A., "On random graphs I," *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959.

[21] Gurobi optimizer reference manual (2020). [Online]. Available: http://www.gurobi.com