

On Quantified Weighted MAX-SAT

Amol Dattatraya Mali

Dept. of Electrical Engg. & Computer Science,

University of Wisconsin, Milwaukee, WI 53211,

mali@miller.cs.uwm.edu, Phone: 1-414-229-6762,

Fax: 1-414-229-2769, <http://www.cs.uwm.edu/faculty/mali>

Abstract

In this paper we introduce quantified weighted MAX-SAT (Q-W-MAX-SAT) as the problem of assigning values to existentially quantified variables permitted by order of quantification, so that the sum of the weights of satisfied clauses equals or exceeds a given threshold, for all combinations of all values of universally quantified variables. Q-W-MAX-SAT serves as a prototypical conditional decision making problem in which satisfying all constraints is either impossible or unnecessary or satisfying some constraints is more important than satisfying some other constraints. We report on two branching heuristics and four simplification rules to solve Q-W-MAX-SAT efficiently, along with an empirical evaluation.

Keywords: DSS foundations, Conditional decision making, Approximate decision making, First-order logic, Combinatorial problems, Agent preferences.

1 Introduction

There has been a significant interest in decision making under uncertainty in artificial intelligence. There may be uncertainty due to unknown states of an agent's environment or imperfect perception or dynamic environment or inaccessible environment or unknown agent preferences. Uncertainty is common in problems in the fields of game playing, planning and autonomous multi-agent systems. One approach to handling uncertainty due to unknown initial state of an agent's environment is to generate a conditional solution which works for

multiple initial states. One such problem is conditional planning, where the initial state of an agent's environment is unknown. To solve this problem, a plan that achieves the goal from several initial states needs to be generated. Conditional planning can be solved by casting it as the problem of solving a quantified boolean formula (QBF) and then solving the QBF and decoding the solution to the QBF [8]. The combinations of values of universally quantified variables in the QBF model multiple environmental states. The combinations of values of universally quantified variables can also be used to express varied and unpredictable boolean preferences of intelligent agents.

Solving a QBF is the problem of assigning values to existentially quantified variables to satisfy all given clauses for all combinations of all values of universally quantified variables, permitted by the order of quantification. Since a QBF is an augmentation of SAT with quantifiers, it is also referred to as Q-SAT (Quantified SAT). For example, the QBF $\forall x \exists y ((x \vee y) \wedge (\neg x \vee \neg y))$ contains two clauses and two variables (x and y) such that x is universally quantified and y is existentially quantified. \forall and \exists denote a universal quantifier and an existential quantifier respectively. \vee and \wedge respectively denote logical OR and logical AND. Solving a QBF can be viewed as solving a conditional decision making problem, since the combinations of values of universally quantified variables can be considered as various conditions and the values assigned to the existentially quantified variables can be viewed as decisions taken under these conditions. The SAT part of a QBF is obtained by removing quantifiers (or making all quantifiers existential). We assume that SAT part of a QBF is specified in conjunctive normal form. In this form, a SAT instance is a conjunction

of clauses. A clause is a disjunction of literals. The SAT part of the QBF in the example is $(x \vee y) \wedge (\neg x \vee \neg y)$. The QBF is satisfiable since there exists a value of y for all values of x to satisfy all given clauses. If $x = \text{true}$, assigning false to y satisfies all clauses. If $x = \text{false}$, assigning true to y satisfies all clauses. The QBF $\exists y \forall x ((x \vee y) \wedge (\neg x \vee \neg y))$ is unsatisfiable (evaluates to false) since there is no single value that can be assigned to y to satisfy all clauses given, for all values of x . A quantifier may quantify multiple variables. Consecutive quantifiers of same type can be replaced by one such quantifier. For example, the quantification $\forall x \forall y \forall z \exists u \exists e$ can be rewritten as $\forall x, y, z \exists u, e$.

Many reasoning tasks involving abduction, reasoning about knowledge using modal logics, temporal and description logics and non-monotonic reasoning are PSPACE-complete problems. These can be reduced in polynomial time to the problem of evaluation/satisfaction of a QBF [1], [4]. Classical planning is the problem of synthesizing a sequence of actions to reach partially specified goal state starting from a completely specified initial state, under restrictions of static and completely observable environment, perfect perception and deterministic actions. Conditional planning [8] is the problem of reaching a partially specified goal state from multiple initial states, under the restrictions of static environment and deterministic actions. This problem can be stated as the problem of constructing a plan (there exists a plan) such that for each initial state there exists an execution that achieves the goal [8]. This alternation of quantifiers $\exists \forall \exists$ makes it easier to see why conditional planning can be cast as a QBF [8].

Consider the job shop scheduling (JSS) problem [5] in which integer starting times and

resources are to be assigned to all operations in given jobs so that the jobs are finished by a given deadline without any resource conflicts and violation of the orderings over operations specified in the JSS problem. Each job has a release date before which no operation from the job can be started. This problem has been solved as propositional satisfiability [2]. A scheduling problem may be over-constrained or it may be more important to satisfy only some of the constraints. Such a problem can be cast as a weighted maximum SAT (W-MAX-SAT) and solved to satisfy clauses so that the sum of weights of the satisfied clauses is either maximized or exceeds or equals a threshold. If the release dates of jobs are uncertain, but are going to be from a known set of dates, one may want to generate a conditional schedule which contains a schedule for every combination of the release dates of all jobs. Similarly, if the resource availability is uncertain but the quantities of resource are going to be from a known set of values, one may want to generate a conditional schedule which contains a schedule for every condition defined by the combinations of values of different quantities of various resources. Such problems can be cast as quantified SAT instances or a quantified constraint satisfaction problems. In such a case, if a schedule meeting all constraints is impossible or unnecessary, one can cast the problem of generating an approximate or acceptance conditional schedule as a Q-W-MAX-SAT problem. We define Q-W-MAX-SAT as the problem of assigning values to existentially quantified variables for all combinations of all values of universally quantified variables permitted by the order of quantification, so that the sum of weights of satisfied clauses exceeds or equals a given threshold θ . W-MAX-SAT, MAX-SAT and SAT are special cases of Q-W-MAX-SAT. These relationships are shown in

Fig. 1. Q-MAX-SAT is a special case of Q-W-MAX-SAT where all clauses have unit weight.

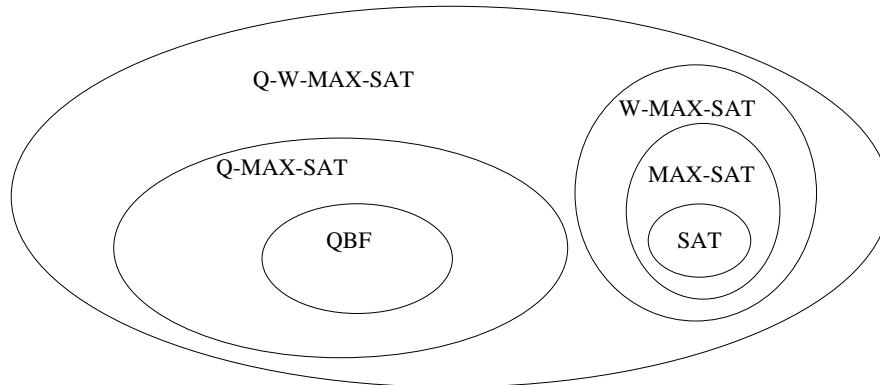


Figure 1: Relationship between Q-W-MAX-SAT and variants of SAT and QBF

Consider a small academic department with three professors. The professors may not be willing to teach in any semester due to reasons like intended sabbatical leave or the need to devote more time to research. If the professors do not want to teach, qualified students must be hired to teach their courses. It is not desirable to hire lots of students due to limited budget and it is not desirable to force the professors to teach either. The professors' wishes are not known in advance. p_i is a boolean variable denoting "professor i wants to teach his/her courses". s_i is a boolean variable denoting "student i teaching professor i 's courses is hired". The constraint $(\neg p_i \longrightarrow s_i)$ which is equivalent to $(p_i \vee s_i)$ denotes that if professor i does not want to teach, a student teaching i 's courses is hired. The constraint $(p_1 \vee p_2 \vee p_3)$ denotes that some professors are willing to teach. The constraint $(\neg s_1 \vee \neg s_2)$ which is equivalent to $\neg(s_1 \wedge s_2)$ denotes that students are not hired to teach both professor 1 and 2's courses (because it is hoped that at least one of these two professors can be convinced to teach, if it is necessary). One needs to find if there is a way to satisfy these constraints

to an acceptable degree, irrespective of the wishes of the three professors. One can create a Q-W-MAX-SAT instance and solve it in this case. Consider the following Q-W-MAX-SAT instance where weights of clauses are specified in the parentheses following them. For example, the weight of clause $(p_1 \vee s_1)$ is 10 and that of $(p_1 \vee p_2 \vee p_3)$ is 5. There are three universally quantified variables, three existentially quantified variables, two quantifiers and five clauses.

$$\forall p_1, p_2, p_3 \exists s_1, s_2, s_3 ((p_1 \vee s_1)(10) \wedge (p_2 \vee s_2)(10) \wedge (p_3 \vee s_3)(10) \wedge (p_1 \vee p_2 \vee p_3)(5) \wedge (\neg s_1 \vee \neg s_2)(5)).$$

Clearly, a QBF instance obtained by dropping the clause weights from this Q-W-MAX-SAT instance has no solution. This is because the clause $(p_1 \vee p_2 \vee p_3)$ is not satisfiable when p_1, p_2 and p_3 are all false. The sum of the weights of clauses in the instance is 40. The tree of truth assignments in Fig. 2 shows that there is a solution to this Q-W-MAX-SAT instance if the threshold θ is 30 or less. The sums of the weights of clauses satisfied by assignments along eight paths for eight combinations of values of the three universally quantified variables are shown at the ends of the paths in Fig. 2.

Over-constrained SAT problems involving a single or multiple agents and problems of generating approximate conditional plans for one or more agents can be cast as Q-W-MAX-SAT. A Q-W-MAX-SAT is a very compact representation of an exponentially large number of W-MAX-SAT problems defined by various combinations of the values of the universally quantified variables. Recent work on conditional planning [8] has shown that solving a large number of unconditional problems is infeasible and that solving a single conditional

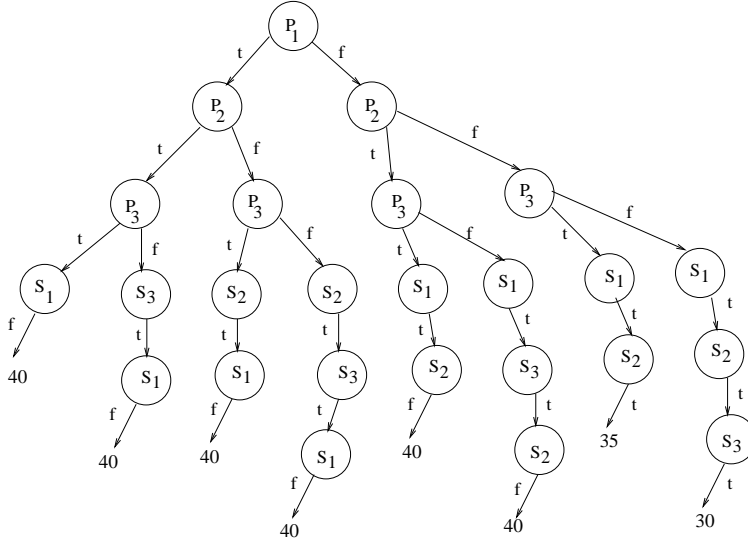


Figure 2: Solution to the Q-W-MAX-SAT instance about academic hiring

problem like a QBF can be feasible. Q-W-MAX-SAT serves as a prototypical conditional approximate decision making problem. Since this is a combinatorially hard problem, it is important to have good heuristics to solve it efficiently. We propose two such heuristics and four soundness and completeness preserving simplification rules in this paper. We show how additional heuristics and simplification rules can be developed. We introduce two variants of Q-W-MAX-SAT called variable-based and state-based variants. A combination of various values of n universally quantified variables can be considered as an uncertainty in the space of uncertainties of size 2^n . Variable-based variant is useful for handling all uncertainties of certain types. State-based variant is useful when handling a specific number of uncertainties is important.

The rest of the paper is organized as follows. In section 2, we report on branching heuristics we developed to solve a Q-W-MAX-SAT efficiently. In section 3, we report on several

rules for simplifying the process of solving a Q-W-MAX-SAT. These rules can be applied fast to find solution or detect absence of solution, for certain Q-W-MAX-SAT instances. Empirical evaluation on 40 problems from four benchmark domains is reported in section 4. Possible extensions of our work are discussed in section 5. The conclusions are reported in section 6.

2 Solving Q-W-MAX-SAT

Several branching heuristics have been developed to solve SAT efficiently [6],[5],[7]. Since the key choice in solving a SAT, QBF, MAX-SAT, W-MAX-SAT, Q-MAX-SAT or Q-W-MAX-SAT is the choice of variable to be assigned a value next, both heuristics we report on in this section are variable ordering heuristics. These decide the order in which to assign values to variables. We assume that there are n clauses in the SAT part of a Q-W-MAX-SAT specified in conjunctive normal form. We denote the clauses by $c_i, i \in [1, n]$. We denote the weight of clause c_i by w_i . W denotes the sum of weights of all n clauses. We denote the number of variables in a Q-W-MAX-SAT by m . We denote the variables by v_1, v_2, \dots, v_m . We use S_i to denote the sum of weights of clauses in which variable v_i occurs. We assume that a Q-W-MAX-SAT does not contain tautologous clauses. Tautologous clauses are always satisfiable. If a Q-W-MAX-SAT contains tautologous clauses, we remove them before solving it and reduce the threshold θ by the sum of the weights of these clauses. We also assume that pure literals containing existentially quantified variables are removed from given Q-W-MAX-SAT and that the threshold θ is reduced by the sum of the weights of clauses containing these

literals. Pure literals occur if there are variables with only positive or only negated occurrence everywhere in the SAT instance. Consider the clauses $(x \vee y) \wedge (x \vee \neg y) \wedge (x \vee z \vee \neg p) \wedge (\neg p \vee r \vee s)$. Here $z, x, \neg p, r$ and s are all pure literals. Before explaining our heuristics, we explain the notions of constraint graph, difference between two variable orderings, node width and ordering width in the following subsection on background.

2.1 Background

Consider the Q-W-MAX-SAT $\forall x, y, z \exists p, q \forall r, t ((x \vee y \vee p) \wedge (\neg y \vee \neg p) \wedge (p \vee r \vee \neg t) \wedge (z \vee q) \wedge (\neg z \vee q) \wedge (\neg t \vee q) \wedge (r \vee \neg q))$ where $c_1, c_2, c_3, c_4, c_5, c_6$ and c_7 are respectively $(x \vee y \vee p)$, $(\neg y \vee \neg p)$, $(p \vee r \vee \neg t)$, $(z \vee q)$, $(\neg z \vee q)$, $(\neg t \vee q)$ and $(r \vee \neg q)$ and $w_1, w_2, w_3, w_4, w_5, w_6$ and w_7 are respectively 10, 0, 10, 2, 2, 1 and 1. Variables that occur in the same clause are said to constrain each other. A partial assignment is an assignment of values to some of the variables in the Q-W-MAX-SAT. A constraint graph is constructed by creating a separate vertex for each variable in the clauses in the SAT part of the Q-W-MAX-SAT and connecting vertices if their variables appear in same clause in the SAT part of the Q-W-MAX-SAT. The constraint graph for the Q-W-MAX-SAT above is shown in Figure 3.

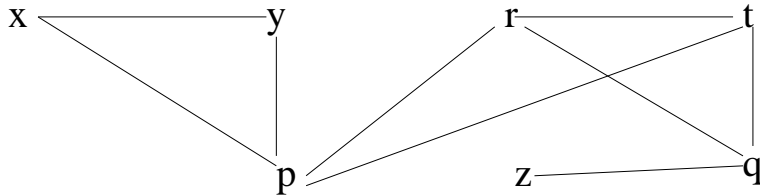


Figure 3: Constraint graph for the clauses in the QBF from Section 2.1

If variables v_1, v_3, v_2 and v_4 are assigned values in this order, the variable ordering $v_1 \prec$

$v_3 \prec v_2 \prec v_4$ is said to be used while making the assignment. \prec denotes precedence. Given a variable ordering, its width and node widths are found as follows. Different variables are considered to be different nodes. The width of a node v_i is same as the number of nodes preceding v_i in the given variable ordering such that v_i constrains variables in these nodes. The width of a variable ordering is maximum of the widths of its nodes. Two variable orderings along with their widths and corresponding node widths are shown in Fig. 4. One variable ordering heuristic used in solving constraint satisfaction problems is minimum width ordering heuristic (MWO) [9]. This heuristic assigns values to variables in an order which has minimum width. One variable ordering with minimum width is shown in Fig. 4. Any process of solving a QBF can be viewed as generating an AND-OR tree in which nodes are same as the variables in the QBF such that the universally quantified variables are AND nodes and the existentially quantified variables are OR nodes. We define an ordering in which variables with higher value of S_i are placed before variables with lower value of S_i as “**sum of clause weights**” ordering. For example, if $S_1 > S_4 > S_5 > S_2 > S_3$, then the sum of clause weights ordering is $v_1 \prec v_4 \prec v_5 \prec v_2 \prec v_3$. If variables v_i and v_j are such that $S_i = S_j$, then v_i is placed before v_j in the sum of clause weights ordering if $i < j$.

We define the difference between two variable orderings as the sum of absolute values of the differences in the positions of the variables in these two orderings. For example, the difference between the variable orderings $v_1 \prec v_3 \prec v_2 \prec v_5 \prec v_4$ and $v_5 \prec v_4 \prec v_1 \prec v_2 \prec v_3$ is $|1 - 3| + |2 - 5| + |3 - 4| + |4 - 1| + |5 - 2|$, which is 12.

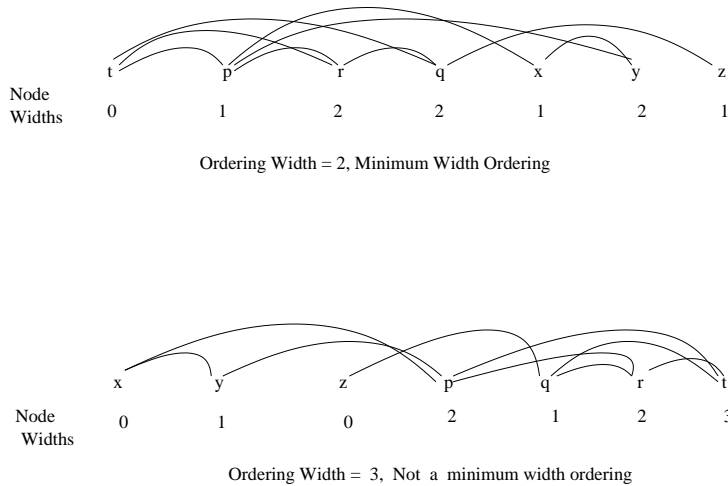


Figure 4: Two variable orderings along with their widths

2.2 Branching Heuristics

The search tree for solving the unsatisfiable QBF from section 2.1 using the minimum width ordering from Fig. 4 is shown in Fig. 5. The search tree for solving this QBF following the variable ordering from Fig. 4 which does not have minimum width is shown in Fig. 6. It is clear from Fig. 5 and Fig. 6 that the minimum width ordering does not allow detection of unsatisfiability of the QBF earlier than the variable ordering with higher width. It is important to take into account the nature of quantifiers of variables and this is something that the MWO heuristic does not consider. In the rest of the paper we assume that a Q-W-MAX-SAT solver keeps a record of the sum of weights of clauses violated by assignments along various paths in its search tree for the purpose of fast detecting failing assignments. In QBF solving, a partial assignment need not be extended once some clause is found to be violated by it or is certainly going to be violated by it. In Q-W-MAX-SAT solving, a partial assignment need not be extended once it is found that the sum of weights of clauses already

need not be assigned and this reduces the size of the search tree. The variables quantified by a quantifier are assigned values in an order which has minimum width such that the width of node v_i is found by considering only the variables quantified by its quantifier that v_i constrains. Thus orderings of variables quantified by different quantifiers are found separately by separately applying the MWO heuristic to these sets of variables. For example, if the quantification part of a Q-W-MAX-SAT is $\forall v_1, v_2, v_3 \exists v_4, v_5 \forall v_6, v_7$, then variables v_1, v_2 and v_3 are assigned values before v_4 and v_5 . v_4 and v_5 are both assigned values before v_6 and v_7 . MWO heuristic is used to compute an order over v_1, v_2 and v_3 by considering only variables constrained by each of these from the set $\{v_1, v_2, v_3\}$ when computing the node widths. Similarly, MWO heuristic is used by MMWO-1 to compute orderings over the variables from the sets $\{v_4, v_5\}$ and $\{v_6, v_7\}$ separately. In case there are multiple minimum width orderings over the variables quantified by a quantifier, any k of these are isolated and the ordering out of these k orderings which differs the least from the sum of clause weights ordering is returned by MMWO-1. k is a predefined constant. For example, if both $v_1 \prec v_3 \prec v_2$ and $v_1 \prec v_2 \prec v_3$ are both minimum width orderings, then $v_1 \prec v_3 \prec v_2$ is chosen if $v_1 \prec v_3 \prec v_2$ is sum of clause weights ordering. The variable ordering returned by MMWO-1 heuristic for the Q-W-MAX-SAT in section 2.1 is shown in Fig. 7. Assuming that x, y, z, p, q, t and r are denoted by $v_1, v_2, v_3, v_4, v_5, v_6$ and v_7 , it is clear that $S_1, S_2, S_3, S_4, S_5, S_6$ and S_7 are respectively 10, 10, 4, 20, 6, 11 and 11. Widths of orderings over variables quantified by different quantifiers are “individual ordering widths” or “local ordering widths”. The MMWO-1 heuristic orders variables so that the individual ordering widths are minimized. The width of an ordering

over all variables in the Q-W-MAX-SAT found by computing node widths using all given clauses is the “overall ordering width” or “global ordering width”. Overall ordering width is relevant to MMWO-2 described next.

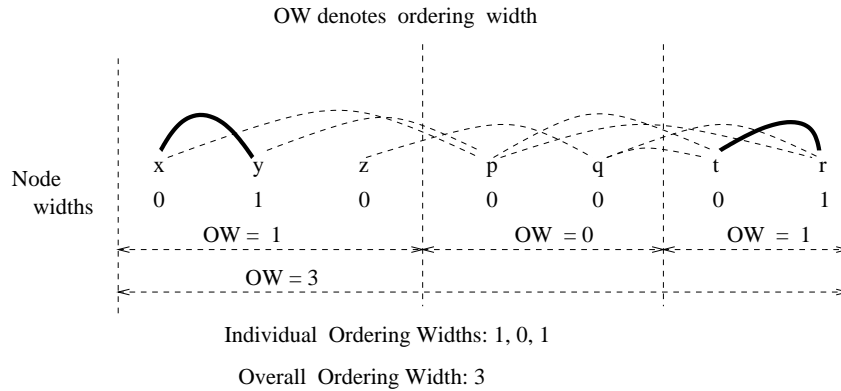


Figure 7: A variable ordering returned by MMWO-1 heuristic

Modified Minimum Width Ordering 2 heuristic (MMWO-2)

Out of all variable orderings which (i) place variables quantified by outer quantifiers earlier than the ones quantified by inner quantifiers and (ii) minimize individual ordering widths, MMWO-2 returns variable ordering with minimum overall width. Note that this minimum overall width could be higher than the width of an ordering over all variables in the Q-W-MAX-SAT returned by MWO. The set of variable orderings MMWO-2 can return is a subset of the variable orderings MMWO-1 can return.

3 Simplification Rules

In this section, we report on four soundness and completeness preserving rules to simplify the process of solving a Q-W-MAX-SAT. The rules can be applied to fast detect the absence of solution or fast find solution for certain Q-W-MAX-SAT instances. In case the rules

do not detect absence of solution or find solution, search must be conducted to solve the Q-W-MAX-SAT instance. Some rules are computationally cheaper than other rules. The cheaper ones can be applied more often than the more expensive ones, in order to control the computational cost of applying the rules. The rules can be applied either statically (before solving process begins) or dynamically (during the solving process). If the rules are applied dynamically, we assume that W and θ are updated continuously as follows. W then represents the sum of weights of clauses that are yet to be satisfied by the current assignment. θ represents threshold to be exceeded or equaled by the sum of weights of clauses to be satisfied by extending the current assignment. The rules are presented so that they can be understood independently. Some of the rules share some computation, allowing an efficient application of multiple rules.

Rule 1. This rule tries to find solution to Q-W-MAX-SAT by solving a SAT instance. Solve the SAT problem obtained by removing quantifiers and universally quantified variables from the Q-W-MAX-SAT. This SAT problem does not contain clauses containing only universally quantified variables. Also, if the Q-W-MAX-SAT instance contains a clause $(x \vee \neg y \vee z)$ where z is universally quantified, the SAT instance will contain a shrunk version of this clause which is $(x \vee \neg y)$. The shrunk clauses in the SAT instance are obtained by removing universally quantified variables from the clauses in the Q-W-MAX-SAT instance. If the SAT problem is satisfiable and the sum of the weights of its clauses is greater than or equal to θ , then return the satisfying assignment as the solution of the Q-W-MAX-SAT problem. This rule is sound because universally quantified variables do not affect the correctness of the

solution to the SAT instance, if the SAT instance is satisfiable. If the SAT instance is not satisfiable, Q-W-MAX-SAT is solved by search. So rule 1 does not prevent a Q-W-MAX-SAT solver from finding a solution. Thus rule 1 does not affect completeness of Q-W-MAX-SAT solver.

Rule 2. This rule tries to find solution to Q-W-MAX-SAT by solving a W-MAX-SAT instance. Solve the W-MAX-SAT problem obtained by removing quantifiers and universally quantified variables from given Q-W-MAX-SAT problem, to find a truth assignment for which the sum of the weights of the satisfied clauses exceeds or equals θ . If such an assignment is found, return it as a solution of the Q-W-MAX-SAT. The rule is sound because if there is a solution to the W-MAX-SAT instance, then inclusion of quantifiers and universally quantified variables does not affect the correctness of the solution. The rule does not affect completeness of a Q-W-MAX-SAT solver if the solver conducts search, when the rule does not find solution.

Rule 3. This rule tries to quickly detect the absence of solution to a Q-W-MAX-SAT instance. Find the maximum of the weights of the non-tautologous clauses which contain only universally quantified variables. Let the maximum be α . W is the sum of weights of all clauses in the given Q-W-MAX-SAT instance. If $(W - \alpha) < \theta$, then the Q-W-MAX-SAT does not have a solution. A non-tautologous clause containing only universally quantified variables cannot be satisfied for all combinations of all values of universally quantified variables. There is one combination that falsifies the clause. So the weight of such a clause places an upper bound on the sum of weights of clauses satisfied for all combinations of all values of universally

quantified variables. So the sum of weights of satisfied clauses cannot exceed $(W - \alpha)$. If $\theta > (W - \alpha)$, then there is no solution to the Q-W-MAX-SAT instance. Hence the rule is sound. If $\theta \leq (W - \alpha)$, the Q-W-MAX-SAT solver can do search to find solution. So the rule does not affect completeness of the solver.

Rule 4. This rule tries to find solution to a Q-W-MAX-SAT instance by solving a SAT instance. Isolate the clauses from Q-W-MAX-SAT which contain only existentially quantified variables. Let the set of these clauses be C . Let the sum of their weights be γ . If $\gamma \geq \theta$, solve the SAT instance formed by taking a conjunction of the clauses from C . If this instance is satisfiable, return the satisfying assignment as solution of the Q-W-MAX-SAT. The solution to the SAT instance will be a solution to the Q-W-MAX-SAT instance as well, since the presence of quantifiers and clauses containing universally quantified variables cannot violate $\gamma \geq \theta$. Hence the rule is sound. If the rule does not yield solution to Q-W-MAX-SAT, a solver can conduct search to find solution. So the rule can be used without affecting the completeness of a Q-W-MAX-SAT solver.

Eleven additional sound and completeness preserving rules have been reported in our technical report [10]. Some of these rules can be applied in low order polynomial time. Some other rules involve solving SAT or W-MAX-SAT instances. Some rules involve partitioning the set of clauses.

4 Empirical Evaluation

In this section, we report on the empirical evaluation of blind search and searches using heuristics H1 and H2 and simplification rule 3. H1 is same as MMWO-1. H2 is same as MMWO-2. All current QBF solvers are modifications of the SAT solving Davis-Logemann-Loveland (DLL) procedure [3]. The blind search of our Q-W-MAX-SAT solver is an extension of the DLL type global search in QBF solvers which deals with clause weights. Searches using H1 and H2 are also DLL type global searches using a statically found variable ordering.

The information about problems used and the results obtained is reported in tables 1,2,3 and 4. The evaluation was conducted on a Dell PC with 512 K cache, 256 M RAM, p3 800 MHz Windows NT 4. We used 40 problems from 4 benchmark domains. Since no Q-W-MAX-SAT problems are available, the 40 problems were obtained from the benchmark QBF problems available at <http://www.informatik.uni-freiburg.de/~rintanen/qbf.html>, by assigning weights to clauses and removal or introduction of clauses from/into some problems. Some Q-W-MAX-SAT problems were derived from the same QBF by changing the threshold θ . Problems with names of the form x-y, x-z, x-q are problems derived from problem x by changing the threshold. For example, problems impl02-1, impl02-2 and impl02-3 in Table 2 differ because of different values of the threshold. The ratio $\frac{\theta}{W}$ was varied. Five out of the 40 problems had $0.95 < \frac{\theta}{W} \leq 1$. 5 out of the 40 problems had $0.8 \leq \frac{\theta}{W} < 0.95$. Two of the forty problems had $0.7 \leq \frac{\theta}{W} < 0.8$. Twelve of the forty problems had $0.4 \leq \frac{\theta}{W} < 0.7$. Two of the forty problems had $\frac{\theta}{W} < 0.1$. Three of the forty problems had $\frac{\theta}{W} > 1$ and obviously, they had no solution. Two other problems had no solution due to presence of

some clauses containing only universally quantified variables (which made it impossible to have a solution).

The 40 problems include 17 problems from the chain of implications domain, 11 problems from bomb in the toilet domain, 3 problems from blocks world and 9 problems from 3-CNF domain. The format for Q-W-MAX-SAT encodings is an augmentation of DIMACS format for SAT encodings, with a notation to handle quantifiers and clause weights. In these formats, variables are denoted by positive integers and negated variables are denoted by negative integers. A problem from chain of implications domain contains implications like $1 \rightarrow (4 \vee 5)$, $5 \rightarrow (4 \vee 1)$, $3 \rightarrow (4 \vee 6)$, $6 \rightarrow (3 \vee 4)$, $4 \rightarrow (-3 \vee 5)$, specified in conjunctive normal form (CNF) as $(-1 \vee 4 \vee 5)$, $(-5 \vee 4 \vee 1)$, $(-3 \vee 4 \vee 6)$, $(-6 \vee 3 \vee 4)$, and $(-4 \vee -3 \vee 5)$ respectively. In blocks world, a goal is a tower/towers of blocks. A QBF encoding with a solution can be solved to get a conditional plan which achieves the goal irrespective of the initial state of blocks. Disarmament is the goal in the problems in the bomb in toilet domain. There are packages which may contain a bomb and there is a toilet which may be clogged. All clauses from problems in 3-CNF domain have exactly three literals.

“Problem” in tables 1,2,3 and 4 denotes problem names. Problems with names of the form “impl-x” are from the chain of implications domain. Problems with names of the form “toil-x” are from the bomb in toilet domain. Problems with names of the form “R3CNF-x” are from the 3-CNF domain, and problems with names of the form “BLOCKS-x” are from blocks world. #UQ in table 1 denotes the number of universal quantifiers in the Q-W-MAX-SAT encoding of the problem. #EQ in table 1 denotes the number of existential quantifiers

in the Q-W-MAX-SAT encoding of the problem. # UV in table 1 denotes the number of universally quantified variables in the Q-W-MAX-SAT encoding of the problem. # V, # C and BS in tables 2,3, and 4 respectively denote the number of variables, number of clauses and the solving times with blind search. H1 and H2 in tables 2 and 3 denote the solving times with MMWO-1 and MMWO-2 respectively. BS + Rule 3 in table 4 denotes the solving time obtained with blind search applying simplification rule 3. The solving times obtained are reported in cpu seconds. (-) in tables 2 and 4 indicates that the problems did not have solution, either because $\frac{\theta}{W} > 1$ or because of introduction of some clauses containing only universally quantified variables.

The following statements apply to the set of 40 problems we used in experiments. The number of universal quantifiers varied from 1 to 10. The number of existential quantifiers varied from 2 to 11. The number of universally quantified variables varied from 1 to 15. The number of existentially quantified variables varied from 8 to 198. The total number of variables varied from 10 to 202. The number of clauses varied from 18 to 1433.

Blind search, H1 and H2 were evaluated on 36 problems. Blind search and blind search with rule 3 were evaluated on 4 problems. So blind search, H1 and H2 were respectively evaluated on 40, 36 and 36 problems. The searches were allowed to continue without any time cutoff.

The performance of blind search, H1 and H2 is reported in tables 2 and 3. Blind search, H1-driven search, and H2-driven search all solved 33 out of the 36 problems which had solution. Blind search, H1-driven search and H2-driven search terminated without finding

solution on the remaining 3 problems. The speedup obtained by one search over other can be found by computing the ratio of their solving times. The speedup obtained with H1-driven search over blind search varied from 1.0007 to 133.36. The speedup obtained with H2-driven search over blind search varied from 1.33 to 7.93.

The performances of blind search and blind search using rule 3 are reported in table 4. Two of the four problems in this table had no solution due to presence of clauses that contained only universally quantified variables and the weights of these clauses which made it impossible to achieve θ . The absence of solution on these problems was detected very fast by blind search with rule 3. In fact blind search with rule 3 was 381047 times faster than blind search on problem impl06-r3-2. Blind search with rule 3 was 24776 times faster than blind search on problem toil01-r3-2. Since the solving time of blind search with rule 3 is slightly more than the solving time of blind search on problems which have solution, it is clear that applying rule 3 takes a negligible amount of time.

5 Discussion

We introduced Q-W-MAX-SAT problem in this paper. It is a generalization of SAT, W-MAX-SAT, MAX-SAT and QBF problems. Q-W-MAX-SAT encodings can be solved in order to get approximate conditional plans, approximate conditional schedules and even approximate multi-agent plans or schedules. We reported on two variable ordering heuristics to solve Q-W-MAX-SAT instances efficiently. We also reported on four sound simplification rules for solving Q-W-MAX-SAT. We showed how the rules can be used without losing

completeness. We reported on an empirical evaluation of the heuristics and the rules on several problems from four domains. In this section, we discuss possible extensions of our work.

Our variable ordering heuristics are based on ordering widths. Additional heuristics can be developed, using the number of occurrences of variables in the clauses, the weights of these clauses and the lengths of the clauses. In SAT solving, assigning values to variables with maximum occurrences in clauses of minimum length can be effective, since this can lead to more simplification. In Q-W-MAX-SAT solving, a solver can assign values to variables with maximum occurrences in clauses with minimum length and high/highest weight first.

Variants of Q-W-MAX-SAT can be more appropriate for certain practical situations than Q-W-MAX-SAT. For example, unpredictable abilities/wishes of agents to do tasks can be expressed using universally quantified boolean variables. If it is important to consider unpredictable abilities/wishes of only any m out of n agents, one can try to find a truth assignment for which the sum of weights of satisfied clauses exceeds or equals a given threshold for all combinations of all values of any m universally quantified variables. Once a solution is found, a solver need not branch on the remaining universally quantified variables. We refer to this as **variable-based variant of Q-W-MAX-SAT**. There are 2^n combinations of all values of n universally quantified variables. If there are n universally quantified variables in a quantified SAT/W-MAX-SAT problem, 2^n can be used as a measure of the amount of uncertainty in the task encoded. This is because universally quantified variables can be viewed as encoding contingencies. So to generate a solution to handle a large number of contingencies, one may

want to satisfy clauses so that the sum of weights of satisfied clauses exceeds or equals θ for at least p combinations of all values of universally quantified variables, where $p \leq 2^n$. We refer to this as **state-based variant of Q-W-MAX-SAT**.

Both variable-based and state-based variants of Q-W-MAX-SAT are useful when achieving θ for 2^n combinations of values of n universally quantified variables is either unnecessary or impossible. In many domains, the universally quantified variables can be also viewed as encoding preferences of agents in a boolean fashion, e.g. an agent wanting to do a task or not wanting to do it. Variable-based variant of Q-W-MAX-SAT is useful when it is important to respect all preferences of some agents. State-based variant of Q-W-MAX-SAT is useful when the total number of preferences respected is important. The worst-case solving times of variable-based and state-based variants of Q-W-MAX-SAT do not exceed the worst-case solving time of Q-W-MAX-SAT, if the number of universally and existentially variables, quantifiers, types of quantifiers for variables and the order of quantification, clauses, and clause weights are same in these three problems.

Eleven additional simplification rules have been reported in [10]. All of these involve solving simpler problems whose worst-case solving times are lower than that of Q-W-MAX-SAT. Some of the rules involve solving SAT/W-MAX-SAT instances and can take significant time to apply in practice. Techniques to estimate the potential of various rules for a given problem can be developed. Such techniques can be based on solving simpler instances than the ones in the rules.

6 Conclusion

We introduced Q-W-MAX-SAT as the problem of assigning values to existentially quantified variables for all combinations of all values of the universally quantified variables permitted by the order of quantification, so that the sum of the weights of the satisfied clauses exceeds or equals a given threshold θ . The universally quantified variables are useful for modeling contingencies or agents' preferences in a boolean fashion. Q-W-MAX-SAT serves as a prototypical conditional approximate decision making problem. Problems of generation of (i) conditional approximate plan/schedule for a single or multiple agents or (ii) conditional approximate solution to a multi-agent constraint satisfaction problem can be cast as instances of Q-W-MAX-SAT. We reported on two variable ordering heuristics and four soundness and completeness preserving simplification rules for solving Q-W-MAX-SAT instances efficiently. We reported on an empirical evaluation of the heuristics on several problems from four domains. We showed how additional heuristics and simplification rules can be developed. We reported on variable-based and state-based variants of Q-W-MAX-SAT. Variable-based variant is useful for dealing with all contingencies of certain types or dealing with all preferences of certain number of agents. State-based variant of Q-W-MAX-SAT is useful for dealing with larger total number of contingencies or preferences of agents. Since the SAT, MAX-SAT, W-MAX-SAT and QBF problems are special cases of Q-W-MAX-SAT, the advances in solving these problems efficiently can also be used to solve Q-W-MAX-SAT and its variants efficiently.

Acknowledgement: This work is funded by NSF grant IIS-0119630 to the author.

References

- [1] Marco Cadoli, Marco Schaerf, Andrea Giovanardi and Massimo Giovanardi, An algorithm to evaluate quantified boolean formulae and its experimental evaluation, *Journal of Automated Reasoning*, Special issue SAT 2000 on satisfiability at the start of the year 2000.
- [2] James M. Crawford and Andrew B. Baker, Experimental results on the application of satisfiability algorithms to scheduling problems, *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI)*, 1994.
- [3] M. Davis, G. Logemann and D. Loveland, A machine program for theorem proving, *Journal of the ACM*, 5(7), 1962.
- [4] Enrico Giunchiglia, Massimo Narizzano and Armando Tacchella, QUBE: A system for deciding quantified boolean formulas satisfiability, *Proceedings of International Joint Conference on Automated Reasoning (IJCAR)*, 2001.
- [5] John Hooker, *Logic-based methods for optimization: Combining optimization and constraint satisfaction*, John Wiley and Sons, Chapter 15 “Branching Rules”, pp. 292-299, 2000.
- [6] Michail G. Lagoudakis and Michael L. Littman, Learning to select branching rules in DPLL procedure for satisfiability, *Workshop on theory and applications of satisfiability*, June 2001, Boston, MA.
- [7] Chu Min Li and Anbulagan, Heuristics based on unit propagation for satisfiability problems, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

- [8] Jussi Rintanen, Constructing conditional plans by a theorem prover, *Journal of Artificial Intelligence Research*, 10, 1999, pp. 323-352.
- [9] Edward Tsang, *Foundations of constraint satisfaction*, Academic press, 1993.
- [10] Amol D. Mali, Simplification rules for Q-W-MAX-SAT, Technical report, Computer Science, University of Wisconsin, Milwaukee, Dec. 2003.

Biographical Sketch:

Amol Dattatraya Mali was born in Pune, India in 1970. He got Ph.D in computer science from Arizona state university in May 1999, in the area of hierarchical planning as satisfiability. Prior to this, he received B.E and M.Tech from VJTI and IIT Kanpur in India respectively. He worked in industry in India for 1.5 years. He joined the electrical engineering and computer science department at University of Wisconsin, Milwaukee as an assistant professor in August 1999. His areas of research interest are artificial intelligence (AI), planning, constraint satisfaction and autonomous agents. He has published 46 papers which include 7 journal papers, 1 book chapter and 25 conference papers. He has published in the AAAI, AIPS, ECP, ICTAI and ICRA conferences. He has been a reviewer for several journals and conferences. He has served on program committees of several conferences.

Figure Legends

Figure 1: Relationship between Q-W-MAX-SAT and variants of SAT and QBF

Figure 2: Solution to the Q-W-MAX-SAT instance about academic hiring

Figure 3: Constraint graph for the clauses in the QBF from Section 2.1

Figure 4: Two variable orderings along with their widths

Figure 5: Search tree for QBF solving using minimum width ordering

Figure 6: Search tree for QBF solving using variable ordering which does not have minimum width

Figure 7: A variable ordering returned by MMWO-1 heuristic

Problem	#UQ	#EQ	#UV
toil03-1	1	2	1
toil01-1	1	2	2
toil02-1	1	2	3
R3CNF01-1	1	2	15
BLOCKS01-1	1	2	4
impl02-1	2	3	2
R3CNF02-1	3	4	15
R3CNF03-1	3	4	15
impl04-1	4	5	4
impl06-1	6	7	6
impl08-1	8	9	8
impl10-1	10	11	10

Table 1: Number of universally quantified variables, universal quantifiers and existential quantifiers in various problems.

Problem	#V	#C	BS	H1	H2
impl02-1	10	18	0.02	0.01	0.01
impl02-2	10	18	0.02	0.01	0.01
impl02-3 (-)	10	18	0.03	0.03	0.03
impl04-2	18	34	9.914	4.967	4.987
impl06-1	26	50	229.039	174.05	172.197
impl06-3	26	50	3659.141	1843.59	1837.402

Table 2: Solving times with blind search and the two heuristics.

Problem	#V	#C	BS	H1	H2
toil01-1	22	52	118.91	118.95	120.242
toil01-2	22	52	237.942	118.941	120.212
toil02-1	33	90	199.917	200.267	25.196
toil02-2	33	90	800.911	801.863	556.339
toil02-3	33	90	4012.549	3215.825	2432.197
R3CNF02-1	150	390	136.195	8.392	137.237
R3CNF02-2	150	390	552.394	8.392	275.536
R3CNF02-3	150	390	2248.112	16.794	558.873

Table 3: Solving times with blind search and the two heuristics.

Problem	# V	# C	BS	(BS + Rule 3)
impl06-r3-1	26	52	118.87	118.89
impl06-r3-2 (-)	26	52	3810.469	0.01
toil01-r3-1	22	54	15.472	15.482
toil01-r3-2 (-)	22	54	247.756	0.01

Table 4: Solving times for blind search with and without Rule 3.