1 PRELIMINARY DEFINITIONS, EXAMPLES, AND LEMMATA

Definition 1. *A **branch-and-bound algebra** is a semiring $\mathcal{S} = (S, +, \times, 0, 1)$ equipped with orders $(\leq, \sqsubseteq)$ such that:*

(1) $(S, \leq)$ is a total order,
(2) $(S, \sqsubseteq)$ is a join–semilattice with join $\sqcup$ where:
  (a) $\sqsubseteq$ respects $+, \times$,
  (b) For all $s \in S$, $0 \sqsubseteq s$,
*(3) $\leq, \sqsubseteq$ are compatible in the sense that for all $a \sqsubseteq b$, we also have $a \leq b$. We will henceforth call this **compatibility**.*

Example 1. The nonnegative real numbers $\mathbb{R}^{\geq 0}$ forms a branch-and-bound algebra with the usual semiring structure of $\mathbb{R}^{\geq 0}$ and the usual order serving as both $\leq$ and $\sqsubseteq$, with join being the max function. The standard extension $\mathbb{R}^{\geq 0} \cup \{\infty\}$ is also a branch-and-bound algebra with the usual extended operations.

Example 2. The Boolean semiring $\mathbb{B} = \{\top, \bot\}$ with $+ = \vee, \times = \wedge, 0 = \bot, 1 = \top$ forms a branch-and-bound algebra with the order $\bot \leq \top$ with join $\wedge$.

Example 3. The expected utility semiring $\mathbb{R}^{\geq 0} \times \mathbb{R}$ with the usual semiring operations forms a branch-and-bound algebra with:

(1) $(p, u) \leq (q, v)$ iff $u \leq v$ or $u = v$ and $p \leq q$
(2) $(p, u) \sqsubseteq (q, v)$ iff $p \leq q$ and $u \leq v$, with join being a coordinatewise max.

It is straightforward to see that these are compatible.

Example 4. For any branch-and-bound algebra $\mathcal{B} = (\mathcal{B}, 0, 1, +, \times, \leq, \sqsubseteq)$ consider the collection of finite sets with elements in $\mathcal{B}$, $\mathcal{P}_{<\omega}(\mathcal{B})$. This forms a semiring with additive and multiplicative identities $\{0\}, \{1\}$ with:

(1) $A + B = \cup_{A,B}\{a + b\}$,
(2) $A \times B = \cup_{A,B}\{a \times b\}$.

Moreover it becomes a branch-and-bound algebra with:

(1) $A \leq B$ iff $\max A \leq \max B$, where max is the greatest in the set with respect to $\leq$,
(2) $A \sqsubseteq B$ iff for all $a \in A$ there exists $b \in B$ with $a \sqsubseteq b$, with join

$$A \sqcup B = \cup_{A,B} \begin{cases} a & a \sqcup b = a \\ b & a \sqcup b = b \\ \{a, b\} & else. \end{cases} \tag{1}$$

The intuition here is that $\leq$ is a total order that allows for a selection between "fully evaluated" values and $\sqsubseteq$ is a partial order that allows for comparisons between "partially evaluated" values. The compatibility condition is effectively saying that "comparable partially evaluated values will stay comparable once fully evaluated".

Definition 2. *Let X be a set and $Y(X)$ a set disjoint but possibly dependent on X. $\mathcal{B}$ a branch-and-bound algebra. Let $f : X \times Y \to \mathcal{B}$ be a function. Then the **max-sum problem (MSP)** associated to f is*

$$\max_{x \in X} \sum_{y \in Y(X)} f(x, y). \tag{2}$$

where max *is taken with respect to the total order* $\leq$ *of the branch and bound algebra. Relatedly, the* ***join-sum problem (JSP)*** *associated to* f *is*

$$\bigsqcup_{x \in X} \sum_{y \in Y(X)} f(x, y). \tag{3}$$

Example 5. The marginal MAP problem is the MSP problem with $X = inst(M)$ instantiations of MAP variables, $Y = inst(V)$ instantiations of the marginal variables, and

$$f(m, v) = \Pr(M = m, V = v \mid E = e).$$

Example 6. The maximum expected utility problem is the MSP problem with $X = \pi$ policies, $Y = E$ the event in which the policy $x \in X$ was taken, and

$$f(\pi, E) = \sum_{\omega \in E} \Pr(\omega) U(\omega)$$

where $U(\omega)$ is additive coordinatewise if ω if the probability distribution is a joint distribution.

Example 7. The weak weighted SAT problem asks for a boolean formula φ over variables V the maximum numbers of true variables in a satisfying assignment. This the MSP problem with $X = inst(V)$, $Y = V$, and

$$f(x, v) = \begin{cases} 1 & x(v) = \top, \\ 0 & x(v) = \bot \end{cases}$$

if $\varphi(x)$ is SAT and $f(x, \_) = 0$ if $\varphi(x)$ is UNSAT.

Lemma 1 (JSP is weak MSP[1]). *Let* X, Y *be disjoint and* $\mathcal{B}$ *a branch-and-bound algebra. Let* $f : X \times Y \to \mathcal{B}$ *be a function. Let MSP, JSP be the max-sum problem and the join-sum problem with respect to* f *as in 2. Then we have*

$$MSP \leq JSP. \tag{4}$$

Proof. It suffices to show for all $a, b \in \mathcal{B}$, $\max\{a, b\} \leq a \sqcup b$. Note that $\max\{a, b\} = a$ or $= b$; by definition of join we have $a, b \sqsubseteq a \sqcup b$; by compatibility we are done. □

For the following definition we write for V a set of Boolean variables $inst(V)$ the set of instantiations of V and $lits(V)$ the set of literals of V. We write $m \models \varphi$ to denote that m is an instantiation of the variables of φ such that the evaluation is true.

Definition 3. *Let* φ *be a Boolean formula over variables* V *and let* $X \subseteq V$ *a subset of variables. . Let* $\mathcal{B}$ *a branch-and-bound algebra. Let* $f : inst(V) \to \mathcal{B}$ *a function. Then the* ***Boolean max-sum problem*** *is*

$$MSP(\varphi) = \max_{x \in inst(X)} \sum_{m \models \varphi|_x} f(x, m) \tag{5}$$

where max *is taken with respect to the total order* $\leq$ *of the branch and bound algebra. Relatedly, the* ***join-sum problem (JSP)*** *associated to* f *is*

$$JSP(\varphi) = \bigsqcup_{x \in inst(X)} \sum_{m \models \varphi|_x} f(x, m). \tag{6}$$

We call the problems factorizable *if there exists a weight function* $w : lits(V) \to \mathcal{B}$ *such that for all instantiations* v *of* V *it is that*

$$f(v) = \prod_{\ell \in v} w(\ell). \tag{7}$$

[1]One may think that this is reminiscent of weak duality. Unfortunately weak duality is completely irrelevant to this Lemma and is only used for namesake purposes.

It is easy to see that Definition 3 is a special case of Definition 2, thus Lemma 1 specializes.

2 SOLVING FACTORIZABLE BOOLEAN MAX-SUM PROBLEMS EXACTLY THROUGH BRANCH-AND-BOUND SEARCH

2.1 Interlude on notation and related lemmata

We assume from this point forward that we are working with max-sum and join-sum problems strictly in the Boolean setting where the objective function (that is, f in Defintion 2) is factorizable into a weight function over literals w. When f and w are implicit, instead of writing

$$\sum_{m \models \varphi} \prod_{\ell \in m} w(\ell) \tag{8}$$

we write

$$\sum \prod \varphi. \tag{9}$$

We provide a few supporting lemmata that follow directly from the model theory of Boolean functions.

Lemma 2. *[Independent Conjunction] Suppose φ and ψ Boolean formulae with nonintersecting variables. Then*

$$\sum \prod \varphi \wedge \psi = \left(\sum \prod \varphi\right) \left(\sum \prod \psi\right). \tag{10}$$

Lemma 3. *[Inclusion-Exclusion] Suppose φ and ψ are Boolean formulae. Then*

$$\sum \prod \varphi \vee \psi = \sum \prod \varphi + \sum \prod \psi - \sum \prod \varphi \wedge \psi. \tag{11}$$

Lemma 4. *[Conditioning] Suppose φ a Boolean formula and v a Boolean variable occurring in φ. Fix a weight function $w : lits(\varphi) \to \mathcal{B}$. Then we have*

$$\sum \prod \varphi = \left(\sum \prod (\varphi|_v \wedge v)\right) + \left(\sum \prod (\varphi|_{\overline{v}} \wedge \overline{v})\right). \tag{12}$$

2.2 The upper-bound algorithm

Suppose φ a Boolean formula over variables V and $X \subseteq V$ as the setting of Definition 3. We refer to an instantiation of a subset of variables of X as a partial model.

```
1: procedure ub(φ, X, P, w)
2:     pm ← ∏_{p∈P} w(p)
3:     acc ← h(φ|_P, X, w)
4:     return pm × acc
5: end procedure
```

(a) The upper bound algorithm ub takes in a formula φ, a subset $X \subseteq vars(\varphi)$, P a partial model of X, and $w : lits(\varphi) \to \mathcal{B}$ a weight function.

```
1: procedure h(φ, X, w)
2:     if φ = ⊤ then return 1
3:     else if φ = ⊥ then return 0
4:     else choose v ∈ Vars(φ)
5:         if v ∈ X then return w(v)h(φ|_v) ⊔ w(v̄)h(φ|_v̄)
6:         else return w(v)h(φ|_v) + w(v̄)h(φ|_v̄)
7:         end if
8:     end if
9: end procedure
```

(b) The helper function h as seen on Line 3 in the above algorithm.

Fig. 1. A single top-down pass upper-bound function.

What does ub upper bound? The answer is subtle and we will first build supporting definitions and lemmata.

Definition 4. *Let φ be a Boolean formula and $X \subseteq vars(\varphi)$. A* partial model *of X is an instantiation of variables $Y \subsetneq X$. A* total model *of X is an instantiation of X.*

Lemma 5. *Let φ be a formula and X a set of Boolean variables disjoint from $vars(\varphi)$. Then for any weight function $w : lits(\varphi) \to \mathcal{B}$,*

$$h(\varphi, X, w) = \sum \prod \varphi. \tag{13}$$

Proof. The proof reduces to the fact that as long as Line 5 of h (as seen in Algorithm 1b) is never invoked, $h(\varphi, X, w)$ calculates the algebraic model count of φ with respect to w. □

Corollary 1. *Let φ be a formula and let $X \subseteq vars(\varphi)$. Let P be a partial model of X instantiating all variables of X. Let $w : lits(\varphi) \to \mathcal{B}$ be a weight function. Then*

$$ub(\varphi, X, P, w) = \sum_{m \models \varphi|_P} \prod_{\ell \in m, P} w(\ell). \tag{14}$$

In particular, if $f : inst(vars(\varphi)) \to \mathcal{B}$ is factorizable by w, we have

$$ub(\varphi, X, P, w) = \sum_{m \models \varphi|_P} f(m, P). \tag{15}$$

Lemma 6. *Let φ be a formula and let $X \subseteq vars(\varphi)$. Let T be a total model of X. Then for any $P \subseteq T$,*

$$ub(\varphi, X, T \setminus P, w) \sqsupseteq ub(\varphi, X, T, w). \tag{16}$$

In other words, every partial model upper-bounds its total extension.

Proof. We induct simultaneously on the size of P and $vars(\varphi)$. If $P = \varnothing$ or $vars(\varphi) = \varnothing$ the proof is trivial; assume as our inductive hypothesis that for all $P' \subsetneq P$

$$ub(\varphi, X, T \setminus P', w) \sqsupseteq ub(\varphi, X, T, w). \tag{IH}$$

Unfolding we can recover

$$h(\varphi|_{T\setminus P'}) \sqsupseteq h(\varphi|_T) \prod_{x \in P'} w(x). \tag{IH}$$

For the inductive case, it suffices to show

$$h(\varphi|_{T\setminus P}) \sqsupseteq h(\varphi|_T) \prod_{x \in P} w(x). \tag{17}$$

let x be the variable chosen by line 4 of Algorithm 1b. We case.

- If $x \in P$, then in particular $x \in T$; we take the join as per line 5. We observe

$$\begin{aligned}
h(\varphi|_{T\setminus P}) &= w(x)h(\varphi|_{T\setminus P}|_x) \sqcup w(\overline{x})h(\varphi|_{T\setminus P}|_{\overline{x}}) \\
&= w(x)h(\varphi|_{T\setminus (P\setminus\{x\})}) \sqcup w(\overline{x})h(\varphi|_{T\setminus P}|_{\overline{x}}) \\
&\sqsupseteq w(x)h(\varphi|_T) \prod_{y \in P\setminus\{x\}} w(y) \sqcup w(\overline{x})h(\varphi|_{T\setminus P}|_{\overline{x}}) && \text{(IH)} \\
&\sqsupseteq w(x)h(\varphi|_T) \prod_{y \in P\setminus\{x\}} w(x) = \boxed{h(\varphi|_T) \prod_{y\in P} w(x).}
\end{aligned}$$

- If $x \notin P$, then we only need to consider the case when $x \notin T$ also; we take the sum as per line 6. We continue recursing until we hit a variable $x' \in P$; then we reduce to case 1 and we are done.

□

Remark. It is important to note that the above lemma cannot be generalized to give a relation between two partial models $P \subsetneq P'$. This is because that the first inductive case crucially relies on the fact that we are not applying the IH twice. Indeed, in general,

$$\begin{aligned}
w(x)h(\varphi|_T) \prod_{y\in P\setminus\{x\}} w(y) \sqcup w(\overline{x})h(\varphi|_T) \prod_{y \in P\setminus\{\overline{x}\}} w(y) \\
\not\sqsupseteq \prod_{y\in P\setminus\{x,\overline{x}\}} w(y)\,(w(x)h(\varphi|_T) \sqcup w(\overline{x})h(\varphi|_T))
\end{aligned}$$

when $x \notin P'$; this is a manifestation of the more general phenomena that

$$a(b \sqcup c) \not\sqsubseteq ab \sqcup ac.$$

Now we can precisely define what *exactly we are upper bounding*. The following theorem is a trivial application of Corrollary 1 and Lemma 6.

Theorem 8. *Fix φ a Boolean formula and let $X \subseteq vars(\varphi)$. Let P a partial model of X. Then, for any T a total model extension of P,*

$$ub(\varphi, X, P, w) \sqsupseteq \sum_{m \models T} \prod_{\ell \in m, T} w(\ell). \tag{18}$$

2.3 The branch-and-bound algorithm

```
1:  procedure bb(φ, R, b, P_curr)
2:      if R = ∅ then
3:          u = ub(φ, X, P_curr, w)                    ▷ P_curr will be a total model
4:          return max(u, b)
5:      else
6:          r = pop(R)
7:          for ℓ ∈ {r, r̄} do
8:              u = ub(φ, X, P_curr ∪ {ℓ}, w)
9:              if u ⋢ b then
10:                 r = bb(φ, R, b, P_curr ∪ {ℓ})     ▷ r will always be from a total model
11:                 b = max(r, b)
12:             end if
13:             return b
14:         end for
15:     end if
16: end procedure
```

Fig. 2. A branch-and-bound style algorithm calculating the max-sum problem. φ is the Boolean formula with subset of variables X, R is a tracker for branch-and-bound variables initialized as X, b is a lower-bound initialized as 0, and P_{curr} is the current partial model.

Lemma 7. *In Algorithm 2, b will always either be*

$$\text{(1) } \sum \prod \varphi|_T \quad \text{for some total model } T\text{, or} \quad \text{(2) } 0.$$

Proof. The proof is a straightforward induction on R. □

Theorem 9. $bb(\varphi, X, 0, \emptyset) = MSP(\varphi)$.

Proof. It suffices to prove that $MSP(\varphi)$ is never pruned. That is, let T_{MSP} be the total model witnessing $MSP(\varphi)$. We claim that $T_{MSP} = P_{curr}$ at some recursive call of bb.

Suppose not. Then, following line 9 of Algorithm 2, there exists a partial model $P' \subset T_{MSP}$ such that $ub(\varphi, X, P', w) \sqsubseteq b$ for some b. By Lemma 7, we case on b.

If $b = 0$, then we have that $ub(\varphi, X, P', w) = 0$. By Lemma 6, we have that then $\sum \prod \varphi|_{T_{MSP}} = 0$. But since this is the maximum such, we have that for all total models T that $\sum \prod \varphi|_T = 0$; thus we are done.

If $b = \sum \prod \varphi|_T$ for some total model T, then we have that:

$$\sum \prod \varphi|_{T_{MSP}} \sqsubseteq ub(\varphi, X, P', w) \sqsubseteq \sum \prod \varphi|_T.$$

By compatibility we have that

$$\sum \prod \varphi|_{T_{MSP}} \leq \sum \prod \varphi|_T.$$

If the two sides are equal that means that $MSP(\varphi)$ has multiple witnesses, thus the branch was never pruned. Otherwise the contradiction is obvious and we are done. □