# Semiring Labelled Decision Diagrams, Revisited: Canonicity and Spatial Efficiency Issues

**IRIT report RR–2013-21–FR,**
Hélène Fargier, Pierre Marquis, Nicolas Schmidt

## Abstract

Existing languages of valued decision diagrams (VDDs), including ADD, AADD and those of the SLDD family, prove valuable target languages for compiling mappings which associate valuations with assignments of values to variables. Among other operations, conditioning and optimization can typically be achieved in polynomial time from such diagrams. However, their efficiency is directly related to the size of the compiled forms. The notion of succinctness is a key to compare the spatial efficiency of representation languages from the theoretical side. In practice, the existence of canonical forms may also have a major impact on the size of the compiled VDDs. While efficient normalization procedures have been pointed out for ADD and AADD the canonicity issue for SLDD representations has not been addressed so far. In this paper, the SLDD family is revisited. We modify the algebraic requirements imposed on the valuation structure so as to ensure tractable optimization and normalization for revisited SLDD representations. We show that AADD is captured by the revisited family. We provide a number of succinctness results relating members of the revisited family with ADD and AADD. We finally report some results from experiments where we compiled some instances of an industrial configuration problem into each of those languages, which enables us to compare their spatial efficiency from the practical side.

## 1 Introduction

In configuration problems of combinatorial objects (like cars), there are two key tasks for which short, guaranteed response times are expected: conditioning (propagating the end-user's choices: version, engine, various options ...) and optimization (maintaining the minimum and/or the maximum cost of a feasible car satisfying the user's requirements). When the set of feasible objects and the corresponding cost functions are represented as a valued CSP (VCSP for short, see [Schiex *et al.*, 1995]), the optimization task is NP-hard in the general case, so short response times cannot be ensured.

Valued decision diagrams (VDDs) from the families ADD [Bahar *et al.*, 1993], EVBDD [Lai and Sastry, 1992; Lai *et al.*, 1996; Amilhastre *et al.*, 2002] and their generalization SLDD [Wilson, 2005], and AADD [Tafertshofer and Pedram, 1997; Sanner and McAllester, 2005] do not have such a drawback and appear as interesting representation languages for compiling mappings associating valuations with assignments of discrete variables (including utility functions and probability distributions). Valuations belong to a set $E$, which can be equipped with some operators, conveying to it a structure, which can be more or less sophisticated. For instance, EVBDD, and more generally additive SLDD, is a basic representation framework for strongly additive cost/preferences in configuration problems [Amilhastre *et al.*, 2002; Hadzic and Andersen, 2006].

The VDD languages offer tractable conditioning and tractable optimization (under some conditions in the SLDD case). The efficiency of these operations is directly related to the size of the compiled form. The choice issue of a target language for compiling a valued CSP (in our case, a configuration problem including cost information) must thus also be based on the spatial efficiency criterion, i.e., the (relative) ability of each language to represent information using little space. Following the principles stated in the knowledge compilation map [Darwiche and Marquis, 2002], the (worst case) theoretical succinctness of the representation language is crucial in this objective. From the practical side, the canonicity of the representation can be a critical factor since it facilitates the search for compiled forms of optimal size (see the discussion about it in [Darwiche, 2011]). Indeed, the ability to ensure a unique canonical form for subformulae prevents them from being represented twice or more in the data structure (caching mechanisms can then automatically merge equivalent subformulae).

In this paper, the SLDD family [Wilson, 2005] is revisited, focusing on the canonicity and the spatial efficiency issues. We extend the SLDD setting by relaxing some algebraic requirements on the valuation structure (requiring that it is a commutative semiring is too demanding for the normalization and optimization purposes). This extension allows us to capture the AADD language as an element of the revisited SLDD family. We point out a normalization procedure which extends the one for AADD to some representation languages of the extended SLDD family and we identify a simple condition on the valuation structure which is sufficient for ensuring that optimization is tractable. We also provide a number of succinctness results relating some elements of the e−SLDD

family with ADD and AADD. We finally report some results from experiments where we compiled some instances of an industrial configuration problem into each of those languages, thus comparing their spatial efficiency on the practical side.

The next section gives some formal preliminaries on valued decision diagrams. Section 3 presents the e−SLDD family, and describes the normalization procedure. In Section 4, succinctness results concerning ADD, some elements of the e−SLDD family, and AADD are pointed out. Section 5 gives some empirical results about the spatial efficiency of those languages, and discusses them. Finally, Section 6 concludes the paper. Proofs are omitted for space reasons.

## 2 Valued Decision Diagrams

Given a finite set $X = \{x_1, \ldots, x_n\}$ of variables where each variable $x \in X$ ranges over a finite domain $D_x$, we are interested in representing mappings associating an element from a valuation set $E$ with assignments $\vec{x} = \{(x_i, d_i) \mid d_i \in D_{x_i}, i = 1, \ldots, n\}$ ($\vec{X}$ will denote the set of all assignments over $X$). $E$ is the carrier of a valuation structure $\mathcal{E}$ which can be more or less rich from an algebraic point of view. in the ADD framework, no assumption is needed on $E$ (even if $E$ is often considered equal to $\mathbb{R}$); in the AADD framework, $E = \mathbb{R}^+$; in the SLDD one, $E$ is supposed to be equipped with two binary operations so that $\mathcal{E}$ is a semiring.

A representation language given $X$ w.r.t. a valuation structure $\mathcal{E}$ mainly is a set of data structures. The targeted mapping is called the *semantics* of the data structure and the data structure is a *representation* of the mapping:

**Definition 1 (representation language)** *(inspired from [Gogic* et al.*, 1995]) Given a valuation structure $\mathcal{E}$, a* representation language $\mathcal{L}$ *over $X$ w.r.t. $\mathcal{E}$, is a 4-tuple $\langle C_\mathcal{L}, Var_\mathcal{L}, I_\mathcal{L}, s_\mathcal{L}\rangle$ where $C_\mathcal{L}$ is a set of data structures $\alpha$ (also referred to as $C_\mathcal{L}$ representations or "formulae"), $Var_\mathcal{L} : C_\mathcal{L} \to 2^X$ is a scope function associating with each $C_\mathcal{L}$ representation the subset of $X$ it depends on, $I_\mathcal{L}$ is an interpretation function which associates with each $C_\mathcal{L}$ representation $\alpha$ a mapping $I_\mathcal{L}(\alpha)$ from the set of all assignments over $Var_\mathcal{L}(\alpha)$ to $E$, and $s_\mathcal{L}$ is a size function from $C_\mathcal{L}$ to $\mathbb{N}$ that provides the size of any $C_\mathcal{L}$ representation.*

In this paper, we are specifically interested in data structures of the form of *valued decision diagrams* (VDDs):

**Definition 2 (valued decision diagram)** *A* valued decision diagram *over $X$ w.r.t. $\mathcal{E}$ is a finite DAG $\alpha$ with a single root, s.t. every internal node $N$ is labelled with a variable $x \in X$ and if $D_x = \{d_1, \ldots, d_k\}$, then $N$ has $k$ outgoing arcs $a_1, \ldots, a_k$, so that the arc $a_i$ of $\alpha$ is valued by $v(a_i) = d_i$. We note $out(N)$ (resp. $in(N)$ ) the arcs outgoing from (resp. incoming to $N$). Arcs can also be labelled by elements of $E$: if $a_i$ is an arc of $\alpha$, then $\phi(a_i)$ denotes the label of $a_i$.*

When *ordered* valued decision diagrams are considered, a total ordering over $X$ is is chosen and the sequence of internal node labels corresponding to every path from the root of $\alpha$ to any leaf is required to be compatible w.r.t. this ordering.

The key problems we focus on are the *conditioning problem* (i.e., given a $C_\mathcal{L}$ formula $\alpha$ over $X$ w.r.t. $\mathcal{E}$ and an assignment $\vec{y} \in \vec{Y}$ where $Y \subseteq X$, compute a $C_\mathcal{L}$ formula representing the restriction of $I_\mathcal{L}(\alpha)$ to $\vec{y}$) and the *optimization problem(s)* (i.e., given an $C_\mathcal{L}$ formula $\alpha$ over $X$ w.r.t. $\mathcal{E}$, find an assignment $\vec{x^*} \in \vec{X}$ such that $I_\mathcal{L}(\alpha)(\vec{x^*})$ is optimal w.r.t. some ordering $\succeq$ over $E$).

Our purpose is to derive polynomial-time algorithms for both problems. Conditioning is an easy operation on a VDD $\alpha$. Mainly, for each $(y, d_i) \in \vec{y}$, just by-pass in $\alpha$ every node $N$ labeled by $y$ by linking directly each of its parents to the child $N_i$ of $N$ such that $v((N, N_i)) = d_i$ ($N$ and all its outgoing arcs are thus removed). Optimization is often more demanding, depending on the family of VDDs under consideration.

ADD, SLDD, and AADD are representation languages based on valued decision diagrams. The scope functions $Var_\texttt{ADD}$, $Var_\texttt{SLDD}$, and $Var_\texttt{AADD}$ are the same ones and they return the set of variables $Var(\alpha)$ from $X$ labeling at least one node in $\alpha$. The size functions $s_\texttt{ADD}$, $s_\texttt{SLDD}$, and $s_\texttt{AADD}$ are closely related: the size of a (labelled) decision graph $\alpha$ is the size $s(\alpha)$ of the graph (number of nodes plus number of arcs) plus the sizes of the labels in it. The main difference between ADD, SLDD, and AADD lies in the way the decision diagrams are labelled and interpreted.

For ADD, no specific assumption has to be made on valuation structure $\mathcal{E}$. Considering any valuation set $E$ is enough:

**Definition 3 (ADD)** ADD *is the 4-tuple $\langle C_\texttt{ADD}, Var_\texttt{ADD}, I_\texttt{ADD}, s_\texttt{ADD}\rangle$ where $C_\texttt{ADD}$ is a set of ordered decision diagrams $\alpha$ over $X$ such that sinks are labelled by elements of $E$, and the arcs are not labelled; $I_\texttt{ADD}$ is defined inductively by: for every assignment $\vec{x}$ over $X$,*

- *if $\alpha$ is a sink node, labelled by an element $a$ of $E$, then $I_\texttt{ADD}(\alpha)(\vec{x}) = a$,*
- *else the root $N$ of $\alpha$ is labelled by $x \in X$; let $d \in D_x$ such that $(x, d) \in \vec{x}$, $a = (N, M)$ the arc such that $v(a) = d$, and $\beta$ the* ADD *representation rooted at node $M$ in $\alpha$ ; we have $I_\texttt{ADD}(\alpha)(\vec{x}) = I_\texttt{ADD}(\beta)(\vec{x})$.*

Optimization is easy on an ADD formula: every path from the root of $\alpha$ to a leaf labelled by a best valuation of $\alpha$ is labelled by a (usually partial) variable assignment which can be extended to a (full) optimal assignment.

In the SLDD framework, the valuation structure $\mathcal{E}$ must take the form of a commutative semiring $\langle E, \oplus, \otimes, 0_s, 1_s\rangle$: $\oplus$ and $\otimes$ are associative and commutative mappings from $E \times E$ to $E$, with identity elements (respectively) $0_s$ and $1_s$, $\otimes$ left and right distributes over $\oplus$, and $0_s$ is an annihilator for $\otimes$ (i.e., $\forall a \in E, a \otimes 0_s = 0_s \otimes a = 0_s$).

**Definition 4 (SLDD)** *Let $\mathcal{E} = \langle E, \oplus, \otimes, 0_s, 1_s\rangle$ be a commutative semiring.* SLDD *is the 4-tuple $\langle C_\texttt{SLDD}, Var_\texttt{SLDD}, I_\texttt{SLDD}, s_\texttt{SLDD}\rangle$ where $C_\texttt{SLDD}$ is a set of valued decision diagrams $\alpha$ over $X$ with a unique sink and such that the arcs are labelled by elements of $E$, and $I_\texttt{SLDD}$ is defined inductively by: for every assignment $\vec{x}$ over $X$,*

- *if $\alpha$ is a sink node, then $I_\texttt{SLDD}(\alpha)(\vec{x}) = 1_s$,*
- *else the root $N$ of $\alpha$ is labelled by $x \in X$; let $d \in D_x$ such that $(x, d) \in \vec{x}$, $a = (N, M)$ the arc such that $v(a) = d$, and $\beta$ the* SLDD *representation rooted at node $M$ in $\alpha$; we have $I_\texttt{SLDD}(\alpha)(\vec{x}) = \phi(a) \otimes I_\texttt{SLDD}(\beta)(\vec{x})$.*

Observe that several choices for $\otimes$ remain usually possible when $E$ is fixed; we sometimes make the notation of the

language more precise (but not too heavy) and write $\texttt{SLDD}_\otimes$ instead of SLDD.

Some complexity assumptions are usually made on $\oplus$ and $\otimes$: they are supposed to be computable in polynomial time.

Note that SLDD languages are not specifically suited to optimization, but they enable for performing efficiently $\oplus$-variable elimination using dynamic programming, see [Wilson, 2005] for details. The point is that when $\oplus$ satisfies the following *addition-is-max* assumption about $\oplus$, i.e.,

$$\forall a, b \in E, a \oplus b \in \{a, b\},$$

and the relation $\succeq$ is a total order satisfying $b \succeq a$ iff $a \oplus b = a$, computing a solution maximal w.r.t. $\succeq$ amounts to performing $\oplus$-variable elimination; this can be achieved in polynomial-time under the poynomial-time assumption for $\otimes$ and $\oplus$.

Sanner and Mc Allester's AADD framework [2005] considers the valuation set $\mathbb{R}^+$ but enables decision graphs into which the arcs are labelled with *pairs* of values from $\mathbb{R}^+$ and consider *two* operators, namely $+$ and $\times$:

**Definition 5 (**AADD**)** *AADD is the 4-tuple* $\langle C_{\texttt{AADD}}, Var_{\texttt{AADD}}, I_{\texttt{AADD}}\ s_{\texttt{AADD}}\rangle$ *where* $C_{\texttt{AADD}}$ *is a set of ordered valued decision diagrams* $\alpha$ *over* $X$ *with a unique sink and such that the arcs are labelled by pairs* $\langle q, f\rangle$ *in* $\mathbb{R}^+ \times \mathbb{R}^+$*;* $I_{\texttt{AADD}}$ *is defined inductively by: for every assignment* $\vec{x}$ *over* $X$*,*

- *if* $\alpha$ *is reduced to the sink node, then* $I_{\texttt{AADD}}(\alpha)(\vec{x}) = 1$*,*
- *else the root* $N$ *of* $\alpha$ *is labelled by* $x \in X$ *; let* $d \in D_x$ *such that* $(x, d) \in \vec{x}$*,* $a = (N, M)$ *the arc such that* $v(a) = d$ *and* $\phi(a) = \langle q, f\rangle$*, and* $\beta$ *the* SLDD *representation rooted at node* $M$ *in* $\alpha$*; we have*

$$I_{\texttt{AADD}}(\alpha)(\vec{x}) = q + (f \times I_{\texttt{AADD}}(\beta)(\vec{x})).$$

*For the normalization purpose, each* $\alpha$ *is equipped with a pair* $\langle q_0, f_0\rangle$ *from* $\mathbb{R}^+ \times \mathbb{R}^+$ *(the "offset", labeling the root of* $\alpha$*); the interpretation function of the resulting "augmented"* AADD *is given by, for every assignment* $\vec{x}$ *over* $X$*,* $I^{\langle q_0, f_0\rangle}_{\texttt{AADD}}(\alpha)(\vec{x}) = q_0 + (f_0 \times I_{\texttt{AADD}}(\alpha)(\vec{x}))$.

AADD (see Figure 1 for an example) and ADD formulae can be normalized efficiently and conditioning and optimization are tractable on these structures.

## 3 Revisiting the SLDD Framework

We propose to extend the SLDD framework in two directions: we relax the algebraic requirements imposed on the valuation structure and we point out a normalization procedure which extends the one for AADD to some representation languages of the extended SLDD family (note that, while some simplification rules have been considered in [Wilson, 2005], no normalization procedure for SLDD has been considered so far). We also identify a simple condition on $\mathcal{E}$ which is sufficient for ensuring that optimization is tractable.

A first useful observation is that, in the SLDD framework, the aggregation operator $\oplus$ and the corresponding identity element $0_s$ are actually not used for defining the SLDD language. Thus, different $\oplus$ may be considered over the same valuation set (e.g., when SLDD is used to compile a Bayesian net, $\oplus = +$ can be used for marginalization purposes and

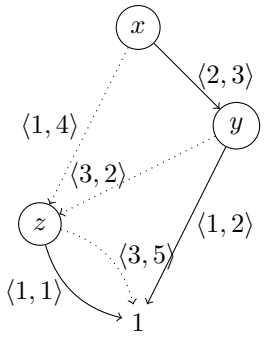


Figure 1: An AADD formula. $x$, $y$ and $z$ are Boolean variables. A (resp. plain) edge corresponds to the assignment of the variable labeling its source to 0 (resp. 1).

$\oplus = max$ can be considered when one is looking for a most probable explanation). That is why the following definition of the extended SLDD setting does not refer to $\oplus$. e−SLDD representations are mainly SLDD representations but the e−SLDD setting is less demanding than the SLDD one concerning the properties of the valuation structure:

**Definition 6 (**e−SLDD**)** *For any monoid* $\mathcal{E} = \langle E, \otimes, 1_s\rangle$*,* e−SLDD *is the 4-tuple* $\langle C_{\texttt{e-SLDD}}, Var_{\texttt{e-SLDD}}, I_{\texttt{e-SLDD}_\otimes}, s_{\texttt{e-SLDD}}\rangle$*, defined as the* SLDD *one, except that, for the normalization purpose, each* e−SLDD *representation* $\alpha$ *is associated with a value* $q_0 \in E$ *(the "offset" of the data structure, labeling its root); the interpretation function* $I^{q_0}_{\texttt{e-SLDD}}$ *of the extended* SLDD *setting is given by, for every assignment* $\vec{x}$ *over* $X$*,*

$$I^{q_0}_{\texttt{e-SLDD}}(\alpha)(\vec{x}) = q_0 \otimes I_{\texttt{e-SLDD}}(\alpha)(\vec{x}).$$

Now, several choices for $\otimes$ are also possible when $E$ is fixed. To avoid ambiguity (but also too heavy notations), we sometimes write $\texttt{e-SLDD}_\otimes$ instead of e−SLDD.

Obviously, the e−SLDD framework captures the SLDD one: when $\mathcal{E} = \langle E, \oplus, \otimes, 0_s, 1_s\rangle$ is a commutative semiring, $\langle E, \otimes, 1_s\rangle$ is a monoid, and every SLDD formula can be interpreted as an e−SLDD one (choose $q_0 = 1_s$). Interestingly, the e−SLDD framework also captures the AADD language:

**Proposition 1** *The valuation structure* $\mathcal{E} = \langle E, \otimes, 1_s\rangle$ *given by* $E = \mathbb{R}^+ \times \mathbb{R}^+$*,* $1_s = \langle 0, 1\rangle$ *and* $\otimes = \star$ *defined by* $\forall b, b', c, c' \in E$*,* $\langle b, c\rangle \star \langle b', c'\rangle = \langle b + c \times b', c \times c'\rangle$ *is a monoid.*

The correspondence between AADD and $\texttt{e-SLDD}_\star$ is made precise by the following proposition:

**Proposition 2** *Let* $\alpha$ *be an* AADD *formula over* $X$*, also viewed as an* $\texttt{e-SLDD}_\star$ *formula. We have:* $\forall \vec{x} \in \vec{X}$*, if* $I_{\texttt{AADD}}(\alpha)(\vec{x}) = a$ *and* $I_{\texttt{e-SLDD}_\star}(\alpha)(\vec{x}) = \langle b, c\rangle$*, then* $a = b + c$*.*

Observe that $\star$ is *not* commutative: the relaxation of this condition is necessary to capture the AADD framework within the e−SLDD family.

AADD is equipped with a normalization procedure that makes AADD representations canonical; canonicity is important for some queries, like deciding the equivalence of two

formulae. It is also important for computational reasons: formulae in canonical form can be efficiently recognized and cached; this has a significant impact on the size of the compiled form. We propose here to extend such a procedure to the extended SLDD setting.

**Definition 7 ($\oplus$-normalisation, $\oplus$-reduction)** *An* e−SLDD *formula $\alpha$ is* $\oplus$-normalized *iff for any node $N$ of $\alpha$, $\oplus_{a\in out(N)}\phi(a) = 1_s$. An* e−SLDD *formula $\alpha$ is* $\oplus$-reduced *iff it is $\oplus$-normalized, it does no not contain any pair of isomorphic nodes, and no node $N$ in $\alpha$ is such that all the arcs outgoing from $N$ join the same node and are valued with the same $\phi$-label.*

The idea at work in normalization is to propagate from the sinks to the root of the diagram the common valuations of its outgoing arcs in order to ensure a condition of minimality, in our case w.r.t. the relation $\trianglerighteq$ induced by $\oplus$ and given by:

$$\forall a, b \in E, a \trianglerighteq b \text{ iff } a \oplus b = b.$$

The existence of an identity element $0_s$ for $\oplus$ ensures that $\trianglerighteq$ is reflexive, and the fact that $\oplus$ is associative ensures that $\trianglerighteq$ is transitive. Furthermore, the fact that $\otimes$ is distributive over $\oplus$ ensures that $\otimes$ is monotonic w.r.t. $\trianglerighteq$, i.e., $\forall a, b, c \in E$, if $b \trianglerighteq a$, then $c \otimes b \trianglerighteq c \otimes a$. To allow such a propagation in VCSPs, [Cooper and Schiex, 2004] assume that $\trianglerighteq$ is a total order and that the aggregator $\otimes$ is monotonic and fair[1] w.r.t. $\trianglerighteq$. Here, we relax this condition so as to be able to encompass the case of the (possibly partial) relation $\trianglerighteq$ induced by $\oplus$.

**Definition 8 (extended SLDD condition)** *A valuation structure $\mathcal{E} = \langle E, \oplus, \otimes, 1_s\rangle$ satisfies* the extended SLDD condition *iff $\langle E, \otimes, 1_s\rangle$ is a monoid, $\oplus$ is a mapping from $E \times E$ to $E$, which is associative and commutative, $\otimes$ is distributive over $\oplus$ and is left-fair w.r.t. $\oplus$, i.e., $\forall a, b \in E$ such that $a \neq b$, if $a \oplus b = b$, then there exists a unique valuation $c \in E$, noted $a \otimes^{-1} b$, such that c is maximal w.r.t. $\trianglerighteq$ and $b \otimes c = a$.*

The extended SLDD condition is close to the commutative semiring assumption for SLDD. However, one the one hand, it requires neither the commutativity of $\otimes$, nor a neutral element (resp. an absorbing element) for $\oplus$ (resp. for $\otimes$); on the other hand, the left-fairness condition of $\otimes$ w.r.t. $\oplus$ is imposed.

**Proposition 3** *The valuation structure $\mathcal{E} = \langle \mathbb{R}^+ \times \mathbb{R}^+, \oplus, \star, \langle 0, 1\rangle\rangle$ where $\oplus = min_\star$ is defined by $\forall b, b', c, c' \in E$: $\langle b, b'\rangle$ $min_\star$ $\langle c, c'\rangle = \langle min(b, c),\ max(b + b', c + c') - min(b, c)\rangle$, satisfies the extended* SLDD *condition.*

Observe that $E$ is not totally ordered by $min_\star$ (for instance, $min_\star(\langle 0, 2\rangle, \langle 1, 2\rangle) = \langle 0, 3\rangle$). When $\langle a, a'\rangle \trianglerighteq \langle b, b'\rangle$ holds, we have:
$\langle a, a'\rangle \star^{-1} \langle b, b'\rangle = \langle 1, 0\rangle$ if $b' = 0$,
$\langle a, a'\rangle \star^{-1} \langle b, b'\rangle = \langle \frac{a-b}{b'}, \frac{a'}{b'}\rangle$ if $b' > 0$.
e−SLDD$_\star$ denotes the corresponding e−SLDD language.

We are now ready to extend the normalization procedure used for AADD representations to the case of e−SLDD$_\otimes$ representations $\alpha$, under the extended SLDD condition. The sink $1_s$ of any e−SLDD$_\otimes$ $\alpha$ is normalized by definition. Then for any internal node $N$ of $\alpha$ whose children have already been visited, the normalization procedure is as follows:

- $q_{min} := \oplus_{a\in out(N)}\ \phi(a)$
- For any $a \in out(N)$ do
  - $\phi(a) := 1_s$ if $\phi(a) == q_{\min}$,
  - $\phi(a) := \phi(a) \otimes^{-1} q_{min}$ otherwise
- For any $a \in in(N)$ do
  - $\phi(a) := \phi(a) \otimes q_{\min}$

If $R$ is the root of the $\alpha$, then its offset $q_0$ finally becomes $q_0 := q_0 \otimes q_{min}$. This normalization procedure runs in time linear in the size of $\alpha$ (which is not necessarily ordered). If $\alpha$ is ordered, the resulting normalized diagram can be reduced backwards efficiently by merging isomorphic nodes (nodes labelled by the same variable and having the same children) and deleting redundant nodes (nodes such that every outgoing arc reaches the same node), as done in the OBDD case.

**Proposition 4** *If $\mathcal{E} = \langle E, \oplus, \otimes, 1_s\rangle$ satisfies the extended* SLDD *condition then the $\oplus$-reduction of any ordered* e−SLDD *formula $\alpha$ is unique, equivalent to $\alpha$ and it can be computed in polynomial time.*

Weighted finite automata and edge-valued binary decision diagrams are recovered in the e−SLDD family by focusing on the valuation structure $\mathcal{E} = \langle \mathbb{R}^+, min, +, 0\rangle$; e−SLDD$_+$ denotes the corresponding language. Other significant valuation structures can be recovered as well, especially:

- $\mathcal{E} = \langle \mathbb{R}^+, max, \times, 1\rangle$: e−SLDD$_\times$ denotes the corresponding language.
- $\mathcal{E} = \langle \mathbb{R}^+ \cup \{+\infty\}, max, min, +\infty\rangle$: e−SLDD$_{min}$ denotes the corresponding language.
- $\mathcal{E} = \langle \mathbb{R}^+, min, max, 0\rangle$: e−SLDD$_{max}$ denotes the corresponding language.

**Proposition 5** *The valuation structures $\mathcal{E} = \langle \mathbb{R}^+, min, +, 0\rangle$, $\mathcal{E} = \langle \mathbb{R}^+, max, \times, 1\rangle$, $\mathcal{E} = \langle \mathbb{R}^+ \cup \{+\infty\}, max, min, +\infty\rangle$ and $\mathcal{E} = \langle \mathbb{R}^+, min, max, 0\rangle$ satisfy the extended* SLDD *condition.*

Clearly, the polynomial-time computability assumptions discussed previously are satisfied by these structures. Thus, the ordered representations from the corresponding languages can be normalized and reduced in polynomial time.

Let us finally switch to conditioning and optimization. First, conditioning does not preserve the $\oplus$-reduction of a formula in the general case, but this is computationally harmless since the $\oplus$-reduction of the conditioned formula can be done efficiently. As to optimization, when $\trianglerighteq$ is total, if an e−SLDD formula $\alpha$ is $\oplus$-reduced, then it contains a path the arcs of which are labelled by $1_s$. The (usually partial) variable assignment along this path can be extended to a full minimal solution $\vec{x^*}$ w.r.t. $\trianglerighteq$, and the offset of $\alpha$ is equal to $I_{\text{e−SLDD}}(\alpha)(\vec{x^*})$. In the general case, the ordering $\succeq$ is not equal to $\trianglerighteq$, so the normalization procedure does not help for determining a minimal solution $\vec{x^*}$ w.r.t. $\succeq$ (or equivalently, a maximal solution w.r.t. the inverse ordering $\preceq$). Nevertheless, simple monotonicity conditions over the valuation structure are enough for ensuring that a minimal solution $\vec{x^*}$ w.r.t. $\succeq$ can be computed in time polynomial in the size of the e−SLDD formula, by dynamic programming. Thus, given any monoid $\mathcal{E} = \langle E, \otimes, 1_s\rangle$ such that $E$ is ordered by

[1] $\otimes$ is fair w.r.t. some total order $\trianglerighteq$ iff for any pair of valuations $a, b \in E$ such that $a \trianglerighteq b$, there exists a unique valuation $c \in E$ such that $c$ is maximal w.r.t. $\trianglerighteq$ and $b \otimes c = a$.

$\succeq$, let us say that $\otimes$ is strictly monotonic w.r.t. $\succeq$ iff for any $a, b, c \in E$, we have $a \succeq b$ iff $c \otimes a \succeq c \otimes b$, and that $\otimes$ is weakly monotonic w.r.t. $\succeq$ iff for any $a, b, c \in E$, if $a \succeq b$ then $c \otimes a \succeq c \otimes b$. We have that:

**Proposition 6** *For any monoid $\mathcal{E} = \langle E, \otimes, 1_s \rangle$ such that $E$ is ordered by $\succeq$, if $\otimes$ is strictly monotonic w.r.t. $\succeq$, or if $\otimes$ is weakly monotonic w.r.t. $\succeq$ and $\succeq$ is total, then for any* e−SLDD *formula $\alpha$, a maximal solution $\vec{x^*}$ w.r.t. $\succeq$ can be computed in time polynomial in the size of $\alpha$.*

## 4 Succinctness of VDDs: Theoretical Results

Let $\mathcal{L}_1$ be a representation language over $X$ w.r.t. $\mathcal{E}_1$ and let $\mathcal{L}_2$ be a representation language over $X$ w.r.t. $\mathcal{E}_2$. Let us say that a $\mathcal{L}_1$ representation $\alpha$ is equivalent to a $\mathcal{L}_2$ representation $\beta$ (where $\mathcal{E}_1$ and $\mathcal{E}_2$ share the same valuation set $E$) iff $Var_{\mathcal{L}_1}(\alpha) = Var_{\mathcal{L}_2}(\beta)$ and $I_{\mathcal{L}_1}(\alpha) = I_{\mathcal{L}_2}(\beta)$. Once this is stated , the notion of succinctness and of translations usually considered over propositional languages (see [Darwiche and Marquis, 2002]) can be extended as follows:

**Definition 9 (succinctness)** *$\mathcal{L}_1$ is* at least as succinct as *$\mathcal{L}_2$, denoted $\mathcal{L}_1 \leq_s \mathcal{L}_2$, iff there exists a polynomial $p$ such that for every $\alpha \in C_{\mathcal{L}_2}$, there exists $\beta \in C_{\mathcal{L}_1}$ which is equivalent to $\alpha$ and such that $s_{L_1}(\beta) \leq p(s_{L_2}(\alpha))$.*

**Definition 10 (linear / polynomial translation)** *$\mathcal{L}_2$ is* linearly (resp. polynomially) translatable into $\mathcal{L}_1$, *denoted $\mathcal{L}_1 \leq_l \mathcal{L}_2$ (resp. $\mathcal{L}_1 \leq_p \mathcal{L}_2$), iff there exists a linear-time (resp. polynomial-time) algorithm $f$ from $C_{\mathcal{L}_2}$ to $C_{\mathcal{L}_1}$ such that for every $\alpha \in C_{\mathcal{L}_2}$, and $\alpha$ is equivalent to $f(\alpha)$.*

It is easy to check that $\leq_s$ (resp. $\leq_p$, $\leq_l$) are pre-orders over representation languages and that $\leq_l \subseteq \leq_p \subseteq \leq_s$. $<_s$ (resp. $<_p$, $<_l$) denotes the asymmetric part of $\leq_s$ (resp. $\leq_p$, $\leq_l$), and $\sim_s$ (resp. $\sim_p$, $\sim_l$) denotes the symmetric part of $\leq_s$ (resp. $\leq_p$, $\leq_l$). By construction, $\sim_s$, $\sim_p$, $\sim_l$ are equivalence relations.

We have obtained the following result showing that every ADD is linearly translatable into any e−SLDD (sharing the same valuation set $E$):

**Proposition 7** e−SLDD $\leq_l$ ADD.

We have also obtained the following results about the valuation set $E = \mathbb{R}^+$:

**Proposition 8**

- ADD $\sim_p$ e−SLDD$_{max}$.
- e−SLDD$_\times$ $\not\leq_s$ e−SLDD$_+$ *and* e−SLDD$_+$ $\not\leq_s$ e−SLDD$_\times$.
- AADD$<_s$ e−SLDD$_+$ $<_s$ ADD.
- AADD$<_s$ e−SLDD$_\times$ $<_s$ ADD.

Similarly, for $E = \mathbb{R}^+ \cup \{+\infty\}$, ADD $\sim_p$ e−SLDD$_{min}$ holds.

## 5 Succinctness of VDDs: Empirical Results

While taking advantage of succinctness is a way to compare representation languages w.r.t. the concept of spatial efficiency, it does not capture all aspects of this concept, for two reasons (at least). On the one hand, succinctness focuses on the worst case, only. On the other hand, it is of qualitative (ordinal) nature: succinctness indicates when an exponential separation can be achieved between two languages but does not enable to draw any quantitative conclusion on the sizes of the compiled forms. This is why it is also important to complete succinctness results with some size measurements.

To this aim, we made some experiments. We designed a bottom-up ordered e−SLDD compiler. This compiler takes as input VCSP instances in the XML format described in [Roussel and Lecoutre, 2009] or Bayesian networks conforming to the XML format given in [Cozman, 2002]. When VCSP instances are considered, the compiler generates a compiled representation of each valued constraint of the instance, under the form of a normalized e−SLDD$_+$ formula, and incrementally combines them w.r.t. $+$ using a simplified version of the $apply(+)$ procedure described in [Sanner and McAllester, 2005]. Similarly, when Bayesian network instances are considered, the conditional probability tables are first compiled into e−SLDD$_\times$ formulae, which are then combined using $\times$. At each combination step, the current e−SLDD formula is normalized (this limits the risk of size explosion). We developed a VDD toolbox which also contains procedures for transforming any e−SLDD$_+$ (resp. e−SLDD$_\times$) formula into an equivalent ADD formula, and any ADD formula into an equivalent AADD formula (the variable ordering is preserved via such transformations); the transformation procedure from e−SLDD$_+$ (resp. e−SLDD$_\times$) representations to ADD representations roughly consists in pushing the labels from the root to the last arcs of the diagram (we cannot describe them in depth here for space reasons).

We considered two families of benchmarks. The VCSP instances we used concern configurations problems, provided by the french car manufacturer Renault; these instances contain hard constraints and soft constraints, with valuation representing prices, to be aggregated additively. They have the following characteristic features:

- **Small**: #variables=139; max. domain size=16; #constraints=176 (including 29 soft constraints)
- **Medium**: #variables=148; max. domain size=20; #constraints=268 (including 94 soft constraints)
- **Large**: #variables=268; max. domain size=12; #constraints=2157 (including 1825 soft constraints)
- **Big**: #variables=268; max. domain size=32; #constraints=2157 (including 1825 soft constraints)

**Large** is obtained by considering only a subset of the domain of the most "central" variable of the **Big** instance (only 4 of 32 possible values of this variable are considered). We also compiled only the soft constraints of the benchmarks, leading to four other instances, referred to as {**Small, Medium, Large, Big**} **Price only**.

As to Bayesian networks, which are of multiplicative nature (joint probabilities are products of conditional probabilities), we used some standard benchmarks [Cozman, 2002].

Each configuration instance has been compiled into an e−SLDD$_+$ representation, and then transformed into an ADD representation, an e−SLDD$_\times$ representation, and an AADD representation - the time needed for the initial compilation and the sizes on the different representations are reported in Table 1. Similarly, each Bayesian net instance has been compiled into an e−SLDD$_\times$ representation, and then transformed into an an ADD representation, an e−SLDD$_\times$ representation, and an AADD representation - the time needed for the initial

Table 1: Compilation of VCSPs into e−SLDD$_+$, and transformations into ADD, e−SLDD$_\times$ and AADD.

| | e−SLDD$_+$ | | ADD | e−SLDD$_\times$ | AADD |
|---|---|---|---|---|---|
| Instance | nodes (edges) | time (s) | nodes (edges) | nodes (edges) | nodes (edges) |
| Small Price only | 36 (108) | $< 1$ | 4364 (7439) | 4364 (7439) | 36 (108) |
| Medium Price only | 169 (499) | $< 1$ | 37807 (99280) | 37807 (99280) | 169 (499) |
| Large Price only | 95 (545) | 5 | 142925 (319844) | 142925 (319844) | 95 (545) |
| Big Price only | 328 (1121) | 5 | 555141 (1336163) | 555141 (1336163) | 328 (1121) |
| Small | 2344 (5584) | 1 | 299960 (637319) | 299960 (637319) | 2344 (5584) |
| Medium | 6234 (17062) | 23 | 752466 (2071474) | 752466 (2071474) | 6234 (17062) |
| Large | 1492 (5745) | 194 | 1371641 (3177066) | 1371641 (3177066) | 1492 (5745) |
| Big | 15858 (56961) | 11513 | m-o | - | 15858 (56961) |

Table 2: Compilation of Bayesian networks into e−SLDD$_\times$, and transformations into ADD, e−SLDD$_+$ and AADD.

| | e−SLDD$_\times$ | | ADD | e−SLDD$_+$ | AADD |
|---|---|---|---|---|---|
| Instance | nodes (edges) | time (s) | nodes (edges) | nodes (edges) | nodes (edges) |
| Cancer | 13 (25) | $< 1$ | 45 (59) | 23 (45) | 11 (21) |
| Asia | 23 (45) | $< 1$ | 423 (447) | 216 (431) | 23 (45) |
| Car-start | 41 (83) | $< 1$ | 52869 (84285) | 19632 (39265) | 38 (77) |
| Alarm | 1301 (3993) | $< 1$ | m-o | - | 1301 (3993) |
| Hailfinder25 | 32718 (108083) | 11 | m-o | - | 32713 (108063) |

compilation and the sizes on the different representations are reported in Table 2. In order to determine a variable ordering, we used the *Maximum Cardinality Search* heuristic [Tarjan and Yannakakis, 1984] in reverse order, as proposed in [Amilhastre, 1999] for the compilation of (classical) CSPs this heuristic is both simple to compute and efficient; experiments reported in [Amilhastre, 1999] show that it typically outperforms several standard variable ordering heuristics (like *Most Constrained First* or *Band-Width*).

We ran all our experiments on a computer running at 800MHz with 256Mb of memory. "m-o" means that the available memory has been exhausted, and that the program aborted for this reason.

Our experiments confirm the results we get about the succinctness of ADD, which is low: compiling instances into e−SLDD$_+$, e−SLDD$_\times$, or AADD prove a better choice as well in practice. Unsurprisingly, when the values of the soft constraints are to be aggregated additively as this is the case for configuration instances (resp. multiplicatively, as this is the case for Bayesian nets), e−SLDD$_+$ (resp. e−SLDD$_\times$) performs better than e−SLDD$_\times$ (resp. e−SLDD$_+$). AADD does not prove better than e−SLDD$_+$ in the additive case, or better than e−SLDD$_\times$ in the multiplicative case. On the Bayesian instances, the ADD and AADD formulae are larger than the ones obtained by [Sanner and McAllester, 2005]. This is due to the way we merge numeric labels (remember that reals are approximated by finite-precision decimals – floating-point numbers on a computer.) [2]

[2]To be more precise, in our implementation, $e_1$ and $e_2$ are considered identical whenever $e_1 - e_2 < 10^{-9}.e_1$ (where $e_1 \geq e_2$). Since $e_1, e_2 \leq 1$ (they represent probabilities) the standard merging condition $e_1 - e_2 < 10^{-9}$ considered in [Sanner and McAllester, 2005] is subsumed by ours. This explains the size discrepancy.

## 6 Conclusion

In this paper, we have shown that the SLDD family of VDDs is rich enough to capture affine algebraic decision diagrams, provided a harmless relaxation of some requirements on the valuation structure. We have pointed out a normalization procedure for the e−SLDD languages based on a valuation structure satisfying a fairness condition inspired from the one used in VCSPs, and we have shown the classical requirement of monotony of the aggregation operator w.r.t. the ordering $\succeq$ allows for tractable optimization over e−SLDD representations. We have also compared the spatial efficiency of some elements of the e−SLDD family, i.e., e−SLDD$_+$ and e−SLDD$_\times$, with ADD and AADD, both from the theoretical side (thanks to the concept of succinctness) and from the pratical side (by reporting some empirical evidence). Though e−SLDD$_+$ (resp. e−SLDD$_\times$) is less succinct AADD from a theoretical point of view, it proves space-efficient enough for enabling the compilation of instances of cost-based configuration problems (resp. Bayesian networks).Going to the AADD framework does not lead to much better compiled representations from the spatial efficiency point of view, when the mapping to be represented is additive or multiplicative in essence, but not both.

Interestingly, the conditions pointed out in the e−SLDD setting for tractable normalization and tractable optimization do not impose the valuation set $E$ to be *totally* ordered. Clearly, this paves the way for the compilation of multi-criteria objective functions as e−SLDD representations. Investigating this issue is a major perspective for the future. Another important issue for further research is to draw the full KC map for VDD languages, which will require to identify the queries and transformations of interest which can be achieved in polynomial time, w.r.t the algebraic properties of the valuation structure.

## References


[Amilhastre *et al.*, 2002] Jérôme Amilhastre, Hélène Fargier, and Pierre Marquis. Consistency restoration and explanations in dynamic CSPs application to configuration. *Artif. Intell.*, 135(1-2):199–234, 2002.

[Amilhastre, 1999] Jérôme Amilhastre. *Représentation par automate d'ensemble de solutions de problèmes de satisfaction de contraintes*. PhD thesis, Université de Montpellier II, 1999.

[Bahar *et al.*, 1993] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. In *Proc. of ICCAD'93*, pages 188–191, 1993.

[Cooper and Schiex, 2004] Martin C. Cooper and Thomas Schiex. Arc consistency for soft constraints. *Artif. Intell.*, 154(1-2):199–227, 2004.

[Cozman, 2002] Fabio Gagliardi Cozman. JavaBayes Version 0.347, Bayesian Networks in Java, User Manual. Technical report, dec 2002. Benchmarks at http://sites.poli.usp.br/pmr/ltd/Software/javabayes/Home/node3.html.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res. (JAIR)*, 17:229–264, 2002.

[Darwiche, 2011] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *Proc. of IJCAI'11*, pages 819–826, 2011.

[Gogic *et al.*, 1995] G. Gogic, H.A. Kautz, Ch.H. Papadimitriou, and B. Selman. The comparative linguistics of knowledge representation. In *Proc. of IJCAI'95*, pages 862–869, 1995.

[Hadzic and Andersen, 2006] Tarik Hadzic and Henrik Reif Andersen. A BDD-based polytime algorithm for cost-bounded interactive configuration. In *Proc. of AAAI'06*, pages 62–67, 2006.

[Lai and Sastry, 1992] Yung-Te Lai and Sarma Sastry. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *Proc. of DAC'92*, pages 608–613, 1992.

[Lai *et al.*, 1996] Yung-Te Lai, Massoud Pedram, and Sarma B. K. Vrudhula. Formal verification using edge-valued binary decision diagrams. *IEEE Trans. on Computers*, 45(2):247–255, 1996.

[Roussel and Lecoutre, 2009] Olivier Roussel and Christophe Lecoutre. XML Representation of Constraint Networks: Format XCSP 2.1. Technical report, CoRR abs/0902.2362, feb 2009.

[Sanner and McAllester, 2005] Scott Sanner and David A. McAllester. Affine algebraic decision diagrams (AADDs) and their application to structured probabilistic inference. In *Proc. of IJCAI'05*, pages 1384–1390, 2005.

[Schiex *et al.*, 1995] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *IJCAI (1)*, pages 631–639, 1995.

[Tafertshofer and Pedram, 1997] Paul Tafertshofer and Massoud Pedram. Factored edge-valued binary decision diagrams. *Formal Methods in System Design*, 10(2/3):243–270, 1997.

[Tarjan and Yannakakis, 1984] Robert E. Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, July 1984.

[Wilson, 2005] Nic Wilson. Decision diagrams for the computation of semiring valuations. In *Proc. of IJCAI'05*, pages 331–336, 2005.