Minimizing Communication Costs of Distributed Local Computation*

Marc Pouly & Jürg Kohlas
Department of Informatics iiF
University of Fribourg
CH – 1700 Fribourg (Switzerland)
http://diuf.unifr.ch/tcs/

December 6, 2005

Abstract

A valuation algebra offers a suitable framework to represent knowledge and information. Based on this framework, several algorithms to process knowledge and pooled under the name local computation were mapped out in recent years. This paper proposes an extension of the valuation algebra framework that allows to express the costs of transmitting valuations between network hosts. Based on this model, we estimate the total computation costs caused by the local computation architectures and observe under which constraints these costs can be minimized. Once a sufficient condition is identified, we present an efficient algorithm to solve the minimization problem concretely.

Keywords: Valuation Algebra, Generic Weight Function, Communication Costs, Shenoy-Shafer Architecture, Partial Distribution Problem

*Research supported by grant No. 20-67996.02 of the Swiss National Foundation for Research.

Contents

1 Introduction

Over the last few decades, a lot of research was dedicated to fathom the definition and essence of information. A particular approach towards these objectives is known as a valuation algebra and has been introduced by (Shenoy & Shafer, 1990; Kohlas, 2003). This mathematical framework defines knowledge and information by their principal operation of combination and marginalization, which realize the natural aggregation and extraction of knowledge. Based on this mathematical system, a set of efficient algorithms to process this representation of knowledge were devised. They are pooled under the name local computation and among them, there exists different architectures which exploit the particular properties of valuation algebra instances to augment the computation's efficiency.

Although the worth of this approach is without controversy, an important property in the nature of knowledge and information is not satisfactory accommodated by the valuation algebra framework. In real life, information is a distributed resource. It may come from different sources, is shared and processed in a distributed manner. All these actions require that pieces of information can be transmitted between the hosts of a common network. Thus, the aim of this paper is to investigate this disregarded nature of knowledge in the context of a valuation algebra. We first extend the valuation framework in such a way, that communication costs of transmitting valuations can be modeled. Then, based on the work of (Schneuwly *et al.*, 2004), we show how the nodes in the underlying tree structure of local computation can be identified with real-world processors. This is a fair realization of classical local computation philosophy, were join tree nodes are regarded as virtual processors (Kohlas, 2003) and local computation messages are exchanged between them. Our main interest is directed towards the total communication costs caused by these messages and naturally, we will investigate under which constraints one can minimize communication costs in local computation.

We start by introducing quickly and in short the mathematical framework of a valuation algebra and give a handful of popular examples. Then, the cornerstone for our future studies is set by defining a generic weight function for valuation algebras. Of special interest is thereby the property of weight predictability which allows to compute a valuation's weight only from its domain. Based on these definitions, section 3 introduces a communication cost model which is used subsequently to measure the total communication costs of local computation. Thereby, we focus our interest on the Shenoy-Shafer architecture (Shenoy & Shafer, 1990), which serves as a representative for all other architectures of local computation (Lauritzen & Spiegelhalter, 1988; Jensen *et al.*, 1990). Section 5 is then addicted to the minimization of these costs. Primarily, we investigate under which constraints the minimization problem can effectively be solved and present an efficient algorithm for the manageable case.

2 Valuation Algebras

2.1 The Framework

The basic elements of a valuation algebra are so-called *valuations*. Intuitively, a valuation can be regarded as a representation of knowledge about the possible values of a set of variables. It can be said that each valuation ϕ refers to a finite set of variables $d(\phi)$, called its *domain*. For an arbitrary set s of variables, Φ_s denotes the set of valuations ϕ with $d(\phi) = s$. With this notation, the set of all possible valuations corresponding to a finite set of variables r can be defined as

$$\Phi = \bigcup_{s \subseteq r} \Phi_s.$$

Let D be the lattice of subsets (the powerset) of r. For a single variable X, Ω_X denotes the set of all its possible values. We call Ω_X the *frame* of the variable X. In an analogous way, we define the frame of a non-empty variable set $s \in D$ by the Cartesian product of frames Ω_X of each variable $X \in s$,

$$\Omega_s = \prod_{X \in s} \Omega_X. \tag{2.1}$$

The elements of Ω_s are called *configurations* of s. The frame of the empty variable set is defined by convention as $\Omega_\emptyset = \{\diamond\}$.

These rather basic definitions are sufficient to define a *valuation algebra*.

2.2 A formal Definition

Let Φ be a set of valuations with their domains in D. We assume the following operations defined on Φ and D:

1. *Labeling:* $\Phi \to D$; $\phi \mapsto d(\phi)$,
2. *Combination:* $\Phi \times \Phi \to \Phi$; $(\phi, \psi) \mapsto \phi \otimes \psi$,
3. *Marginalization:* $\Phi \times D \to \Phi$; $(\phi, x) \mapsto \phi^{\downarrow x}$, for $x \subseteq d(\phi)$.

These are the three basic operations of a valuation algebra. If we interpret valuations as information (pieces), the labeling operation tells us its domain. Combination can be understood as aggregation and marginalization as focusing or extraction of the part we are interested in.

We impose now the following set of axioms on Φ and D:

(A1) *Commutative Semigroup:* Φ is associative and commutative under combination.

(A2) *Labeling:* For $\phi, \psi \in \Phi$,

$$d(\phi \otimes \psi) = d(\phi) \cup d(\psi). \tag{2.2}$$

(A3) *Marginalization:* For $\phi \in \Phi$, $x \in D$, $x \subseteq d(\phi)$,

$$d(\phi^{\downarrow x}) = x. \tag{2.3}$$

(A4) *Transitivity:* For $\phi \in \Phi$ and $x \subseteq y \subseteq d(\phi)$,

$$(\phi^{\downarrow y})^{\downarrow x} = \phi^{\downarrow x}. \tag{2.4}$$

(A5) *Combination:* For $\phi, \psi \in \Phi$ with $d(\phi) = x$, $d(\psi) = y$ and $z \in D$ such that $x \subseteq z \subseteq x \cup y$,

$$(\phi \otimes \psi)^{\downarrow z} = \phi \otimes \psi^{\downarrow z \cap y}; \tag{2.5}$$

(A6) *Domain:* For $\phi \in \Phi$ with $d(\phi) = x$,

$$\phi^{\downarrow x} = \phi. \tag{2.6}$$

The axioms require natural properties of a valuation algebra regarding knowledge or information modelling. The first axiom indicates that Φ is a commutative semigroup under combination. If information comes in pieces, the sequence does not influence the overall knowledge. The labeling axiom tells us that the combination of valuations gives knowledge over the union of the domains involved. Neither variables vanish, nor new appear. The marginalization axiom expresses the natural functioning of focusing. Transitivity says that marginalization can be performed in several steps. Explaining the naturalness of the combination axiom is a little more difficult. Assume we have some knowledge over a domain in order to answer a question. It is interesting to see how the answer is affected if a new information piece arrives. The new piece extends probably the initially given question, since also new variables could come along. The combination axiom tells us that we either combine the new piece to the already given knowledge and focus afterwards to the (new) desired domain, or we cut first the uninteresting parts of the new information out and combine it afterwards. There is no difference between the two approaches. Finally, the domain axiom tells us that knowledge is not influenced by projecting it to the own domain. Without this axiom, this is not always the case (Shafer, 1991).

Definition 1 *A system (Φ, D) together with the operations of labeling, marginalization and combination satisfying these axioms is called a* valuation algebra.

The following lemma describes a few elementary properties derived directly from this set of axioms. Its simple proof can be found in (Kohlas, 2003).

Lemma 1 *1. If $\phi, \psi \in \Phi$ with $d(\phi) = x$ and $d(\psi) = y$, then*

$$(\phi \otimes \psi)^{\downarrow x \cap y} = \phi^{\downarrow x \cap y} \otimes \psi^{\downarrow x \cap y}. \tag{2.7}$$

2. *If* $\phi, \psi \in \Phi$ *with* $d(\phi) = x$, $d(\psi) = y$ *and* $z \subseteq x$, *then*

$$(\phi \otimes \psi)^{\downarrow z} = (\phi \otimes \psi^{\downarrow x \cap y})^{\downarrow z}. \tag{2.8}$$

Together with the definition of a valuation algebra given here, there are many variations being either more or less restrictive. We enumerate consecutively the most important ones and refer to (Kohlas, 2003) and (Schneuwly *et al.*, 2004) for more details.

- *Valuation algebras with neutral elements:* Every sub-semigroup Φ_s has a neutral element, i.e. for all $s \in D$ there is an element $e_s \in \Phi_s$ such that for all $\phi \in \Phi$ with $d(\phi) = s$ we get $\phi \otimes e_s = \phi$. For consistency, we add a neutrality axiom to the above system:

  (A7) *Neutrality:* For $x, y \in D$,

$$e_x \otimes e_y = e_{x \cup y}.$$

- *Separative & regular valuation algebras:* In separative and regular valuation algebras several conditions are required on (Φ, D) such that every valuation $\phi \in \Phi$ has a kind of inverse ϕ^{-1}. In the first, the newly required properties make an embedding of (Φ, D) into a valuation algebra (Φ^*, D) with a well-defined division operator possible. Unfortunately, the marginalization operation in (Φ^*, D) is only partial. This corresponds to a more general system of axioms whose marginalization operator is defined only partially. This system is fully described in (Schneuwly *et al.*, 2004). The regular kind of division requires stronger conditions, such that the complicated embedding with the drawback of the partial marginals can be avoided.

- *Information algebras:* An information algebra is a valuation algebra with neutral elements and the following additional axioms

  (A8) *Idempotency:* For all $\phi \in \Phi$ and $x \subseteq d(\phi)$,

$$\phi \otimes \phi^{\downarrow x} = \phi.$$

  (A9) *Stability:* For all $\phi \in \Phi$ and $x \subseteq y$,

$$e_y^{\downarrow x} = e_x.$$

  Axiom (A8) states that a valuation combined with a part of itself gives nothing new. This is a very natural property of information and is therefore eponym for this variation of a valuation algebra. Among the many strong consequences of this definition, which are altogether listed in (Kohlas, 2003), it permits to introduce an ordering relation of valuations. For an information algebra (Φ, D) and $\phi, \psi \in \Phi$, we say that $\phi \geq \psi$ if and only if $\phi \otimes \psi = \phi$.

As we have seen, the existence of neutral elements is not mandatory for a valuation algebra. This is justified by the fact, that we can adjoin a so-called *identity element* e to a valuation algebra whenever no neutral elements are available. This has been remarked for the first time by (Schneuwly *et al.*, 2004) and the consequences are important, especially also from the perspective of this paper.

Let (Φ, D) be an arbitrary valuation algebra according to the operations of labeling, combination and marginalization. We add a new valuation e to Φ and denote the resulting system by (Φ', D). The operations are extended from Φ to Φ' in the following way:

1. *Labeling:* $\Phi' \to D$; $\phi \mapsto d'(\phi)$,
   - $d'(\phi) = d(\phi)$, if $\phi \in \Phi$;
   - $d'(e) = \emptyset$;
2. *Combination:* $\Phi' \times \Phi' \to \Phi'$; $(\phi, \psi) \mapsto \phi \otimes' \psi$,
   - $\phi \otimes' \psi = \phi \otimes \psi$ if $\phi, \psi \in \Phi$;
   - $\phi \otimes' e = e \otimes' \phi = \phi$ if $\phi \in \Phi$;
   - $e \otimes' e = e$;
3. *Marginalization:* $\Phi' \times D \to \Phi'$; $(\phi, x) \mapsto \phi^{\downarrow' x}$, for $x \subseteq d(\phi)$
   - $\phi^{\downarrow' x} = \phi^{\downarrow x}$ if $\phi \in \Phi$;
   - $e^{\downarrow' \emptyset} = e$.

Lemma 2 *(Φ', D) with the extended operations d', $\otimes'$ and $\downarrow'$ is a valuation algebra.*

The proof of this and the following lemma can be found in (Schneuwly *et al.*, 2004). We usually identify the operators in (Φ', D) like in (Φ, D), i.e. d' by d, $\otimes'$ by $\otimes$ and $\downarrow'$ by $\downarrow$ if they are not used to distinguish between the two algebras. Furthermore, the next lemma states that there is no need to artificially adjoin an identity element if the valuation algebra possesses already neutral elements.

Lemma 3 *If (Φ, D) is a valuation algebra with neutral elements, then there is a valuation $e_\emptyset \in \Phi$ having the same properties in (Φ, D) as e in (Φ', D) .*

This concludes our short introduction to valuation algebras. The next subsection will give some concrete examples of valuation algebra instances and we refer to (Kohlas, 2003) and (Kohlas, 2004) for a rich collection of further examples.

2.3 Some Examples

We present a selection of well-known valuation algebra instances, which will exemplify the theory of the following sections:

Semiring-Valued Valuation Algebras:

A (commutative) semiring is a set A with two binary operations, which we denote by $+$ and $\times$, and which satisfy the following conditions:

1. $+$ and $\times$ are commutative and associative.
2. $\times$ distributes over $+$, i.e. for all $a, b, c, \in A$ we have that

$$a \times (b + c) = (a \times b) + (a \times c).$$

Examples of semirings are the *Boolean semiring*, where $A = \{0, 1\}$ and $a + b = \max\{a, b\}$, $a \times b = \min\{a, b\}$, the *Bottleneck Algebra*, where $+$ is the max operation and $\times$ the min operation on pairs of real numbers, augmented with $+\infty$ and $-\infty$, or $(\max / \min, +)$ semirings, where A consists of the nonnegative integers plus $+\infty$. Addition $+$ is taken as the min or, alternatively, the max-operation, whereas $\times$ is the ordinary multiplication. The nonnegative reals $\mathbb{R}^+$ together with the ordinary addition and multiplication forms also a semiring. Finally the interval $[0, 1]$ with $+$ as the max operation and any t-norm for multiplication yields another semiring (Menger, 1942; Schweizer & Sklar, 1960).

Assuming that A is a semiring, we may define valuations on s by $\phi : \Omega_s \to A$. If $\mathbf{x}$ is a configuration of s, and $t \subseteq s$, then let $\mathbf{x}^{\downarrow t}$ denote the configuration of t consisting of the components x_i of $\mathbf{x}$ with $i \in t$. Further, we define the following operations with respect to semiring-valued valuations:

1. *Labeling:* $d(\phi) = s$ if ϕ is a valuation on s,
2. *Combination:* If $d(\phi) = s$ and $d(\psi) = t$, and $\mathbf{x}$ is a configuration of $s \cup t$, then

$$\phi \otimes \psi(\mathbf{x}) = \phi(\mathbf{x}^{\downarrow s}) \times \psi(\mathbf{x}^{\downarrow t}), \tag{2.9}$$

3. *Marginalization:* If $t \subseteq d(\phi)$, and $\mathbf{x}$ is a configuration of t, then

$$\phi^{\downarrow t}(\mathbf{x}) = \sum_{\mathbf{y} \in \Omega_{s-t}} \phi(\mathbf{x}, \mathbf{y}). \tag{2.10}$$

It is easy to see that these semiring-valued valuations form indeed a valuation algebra. In the case of $\mathbb{R}_0^+$ with ordinary addition and multiplication this is the instance of *discrete probability potentials* as studied in (Lauritzen & Spiegelhalter, 1988) and (Shenoy & Shafer, 1990). Further instances of semiring-valued valuation algebras can be found in (Kohlas, 2004).

Relations:

A second important instance of a valuation algebra is the algebra of relations. Let $\mathcal{A}$ be a finite set of symbols, called *attributes*. For each $\alpha \in \mathcal{A}$ let U_α be a non-empty set, called the domain of attribute α. Let $s \subseteq \mathcal{A}$, then a *s-tuple* is a function f with domain s and $f(\alpha) \in U_\alpha$. The set of all s-tuples is called E_s. For a s-tuple and a subset t of s the restriction $f[t]$ is defined to be the t-tuple g such that $g(\alpha) = f(\alpha)$ for all $\alpha \in t$.

A *relation* R over s is a set of s-tuples, i.e. a subset of E_s. The set of attributes s is called the domain of R and denoted by $d(R) = s$. If R is a relation over s and t a subset of s, then the *projection* of R onto t is defined as:

$$\pi_s(R) = \{f[t] : f \in R\}. \tag{2.11}$$

The *natural join* of a relation R over s and a relation S over t is defined as

$$R \bowtie S = \{f \in E_{s\cup t} : f[s] \in R, f[t] \in S\}. \tag{2.12}$$

It is easy to see that the algebra of relations with join as combination and projection as marginalization is a valuation algebra. The full relations E_s are neutral elements in this algebra. A particularity of relations, is that they are *idempotent*: A relation combined (joined) by a projection of it returns the original relation. Such algebras are also called *information algebras* as we have seen before.

Belief Potentials:

A further well-known valuation algebra instance are belief potentials in the sense of Dempster-Shafer's theory of evidence (Dempster, 1967; Shafer, 1976). We consider variables $X \in s$ with finite frames Ω_X. The Cartesian product Ω_t represents the set of configurations with respect to a subset of variables $t \subseteq s$. For a set of configurations $A \subseteq \Omega_t$, $A^{\downarrow t'} = \{\mathbf{x}^{\downarrow t'} : \mathbf{x} \in A\}$ denotes the projection of A to $t' \subseteq t$. Similarly, $A^{\uparrow t''} = \{\mathbf{x} \in \Omega_{t''} : \mathbf{x}^{\downarrow t} \in A\}$ is the extension of A to a superset $t'' \supseteq t$.

A *belief potential* on a domain t is defined by a mapping $m : 2^{\Omega_t} \to [0, 1]$ where

$$\sum_{A\subseteq\Omega_t} m(A) = 1.$$

Such a mapping is called *mass function*. Subsets $A \subseteq \Omega_t$ for which $m(A) \neq 0$ are called *focal sets*. Combination of belief potentials is defined by Dempster's rule (Dempster, 1967). Let m_1 and m_2 be two belief potentials over $t_1 \subseteq V$ and $t_2 \subseteq V$. If $t = t_1 \cup t_2$, then $m_1 \otimes m_2$ is defined as:

$$m_1 \otimes m_2(A) = \sum_{A_1^{\uparrow t} \cap A_2^{\uparrow t} = A} m_1(A_1) \cdot m_2(A_2) \tag{2.13}$$

for all $A \subseteq \Omega_t$. For a belief potential m on $t \subseteq V$ and $t' \subseteq t$, marginalization is defined by

$$m^{\downarrow t'}(A) = \sum_{B^{\downarrow t'}=A} m(B) \tag{2.14}$$

for all $A \subseteq \Omega_{t'}$. (Haenni, 2004) shows that belief functions satisfy indeed the axioms of a valuation algebra.

2.4 Weight Functions for Valuation Algebras

This subsection is dedicated to the definition of a generic weight function for valuation algebras, which will serve afterwards to express communication costs caused by transmitting valuations over a network.

Definition 2 *Let (Φ, D) be a valuation algebra. A function $\omega : \Phi \to \mathbb{N}_0$ is called weight function if for all $\phi \in \Phi$ and $x \subseteq d(\phi)$ we have $\omega(\phi) \geq \omega(\phi^{\downarrow x})$.*

This definition can be extended to a valuation algebras Φ^* with an identity element e by assigning a constant value to $\omega(e)$. Because the identity element e has been adjoint artificially and has no informational content, it is in the general case reasonable to define $\omega(e) = 0$.

The requirement $\omega(\phi) \geq \omega(\phi^{\downarrow x})$ in the above definition deserves closer attention. Although it may seem reasonable that the weight decreases when the valuation is projected to a subdomain, there are instances which contradict this intuition. This is typically the case when valuation algebra instances are defined on language level rather than on models. A well-known example can be found in propositional logic. Defining valuations as being sets of clauses can cause an increase of the number of clauses when the marginalization operator is applied. But on the other hand, when the model-based dual representation of valuations is chosen, the above requirement is fulfilled. See (Lehmann, 2001) for an in-depth study of propositional logic as valuation algebra instance and (Kohlas, 2002) for a discussion of models and languages in general.

We give some simple examples of weight functions for the valuation algebra instances described in section 2.3.

Example 1:

Semiring-valued valuation algebras: We have seen that semiring-valued valuations are defined as $\phi : \Omega_s \to A$, where Ω_s denotes all possible configurations of a set of variables s and A the semiring whose values are assigned to the configurations. A general weight function for such valuations can be defined as

$$\omega(\phi) = \prod_{X \in s} |\Omega_X|. \tag{2.15}$$

Note, that generally, this weight function grows exponentially with the cardinality of the valuation's domain.

Relational Algebra: A possible weight function for relations $r \in \Phi$ is given by

$$\omega(r) = |d(r)| \cdot card(r), \tag{2.16}$$

where $card(r)$ denotes the number of tuples in the relation r. This definition is known from various database literature such as (Tamer Özsu & Valduriez, 1999).

Belief Potentials: A simple weight function for belief potentials with domain s is given by

$$\omega(\phi) = 2^{|\Omega_s|}. \tag{2.17}$$

This weight function even has a super-exponential character. (Haenni & Lehmann, 2003) motivate the representation of belief potentials by their focal sets. It is easy to see that focal sets shrink when the appropriate belief potential is marginalized to a subdomain. Therefore, counting focal sets is an alternative weight function for belief potentials. ⊖

Of special interest are valuation algebra instances whose weight function only depends on the valuation's domain. This is expressed by the following definition.

Definition 3 *Let ω be a weight function for a valuation algebra (Φ, D). ω is called a weight predictor for Φ if there exists a function $f : D \to \mathbb{N}_0$ such that*

$$\omega(\phi) = f(d(\phi))$$

for all $\phi \in \Phi$. A valuation algebra possessing a weight predictor is called weight predictable.

Example 2: All semiring-valued valuation algebras are weight predictable, since the weight function proposed in example 1 is clearly a weight predictor. On the other hand, the weight function defined for relations is not a weight predictor. Belief potentials are weight predictable if the super-exponential weight function is used. The more sophisticated version using focal sets is clearly not a weight predictor. ⊖

Lemma 4 *Let (Φ, D) be a valuation algebra with weight predictor ω. For $\phi, \psi, \eta \in \Phi$ we have*

1. $\omega(\phi),\ \omega(\psi)\ \leq\ \omega(\phi \otimes \psi)$,
2. $\omega(\phi) \leq \omega(\psi)\ \Rightarrow\ \omega(\phi \otimes \eta) \leq \omega(\psi \otimes \eta)$.

Proof. 1) If ω is a weight predictor, there exists a function $f : D \to \mathbb{N}_0$ such that $\omega(\phi \otimes \psi) = f(d(\phi) \cup d(\psi)) \geq f(d(\phi)) = \omega(\phi)$. The same holds for $\omega(\psi)$. 2) Again, if ω is a weight predictor, there exists a function $f : D \to \mathbb{N}_0$ such that

$$\omega(\phi) \leq \omega(\psi) \Rightarrow f(d(\phi)) \leq f(d(\psi)) \Rightarrow f(d(\phi) \cup f(\eta)) \leq f(d(\psi) \cup d(\eta)) = \omega(\psi \otimes \eta).$$

□

Note also, that if ω is a weight predictor for a valuation algebra Φ and if $\phi \in \Phi$ with $d(\phi) = \emptyset$, the weight of the identity element is naturally given by $\omega(e) = \omega(\phi)$.

In section 2.2 we have seen that the property of idempotency imposes a natural ordering of valuations. The following lemma shows that this ordering relation is reflected in the valuations' weights, presuming that the information algebra is weight predictable.

Lemma 5 *Let (Φ, D) be an information algebra with weight predictor ω. For $\phi, \psi, \in \Phi$, it holds that $\phi \geq \psi \Rightarrow \omega(\phi) \geq \omega(\psi)$.*

Proof. From the definition of information order and lemma 4, it follows that $\omega(\phi) = \omega(\phi \otimes \psi) \geq \omega(\psi)$. □

3 Modelling Communication Costs

We consider a set of remote processors $P = \{p_1, p_2, \ldots, p_p\}$ participating in a communication network. Processors are independent computing units with their own private memory space such that they can execute tasks without interactions and in parallel. To accentuate these abilities, the term processor has been chosen instead of network host.

In order to model the costs of transmitting valuations between processors, we need to define a cost function that satisfies a number of natural requirements. Let $c_\phi(i, j)$ denote the communication costs of sending valuation ϕ from processor i to processor j, presuming that ϕ is already stored on processor i. We impose the following requirements on c:

- $c_\phi(i, j) \geq 0$
- $c_\phi(i, j) = c_\phi(j, i)$
- $i = j \;\Rightarrow\; c_\phi(i, j) = 0$

It is reasonable to define communication costs as a positive and symmetric measure. The third requirement states that communication costs are negligible if no network activities are involved.

For the subsequent studies, we define a concrete cost function that satisfies these requirements. In order to have in some degree a realistic model, we assume that the distances between each pair of processors i and j in the underlying network topology are given in a distance matrix $D = [d_{i,j}]$. Furthermore, it is natural to assume $d_{i,i} = 0$ and $d_{i,j} = d_{j,i}$. Then, a *distance-dependent* cost model to transmit

valuations between processors is given as follows: Let (Φ, D) be a valuation algebra with a weight function ω. For all $\phi \in \Phi$ and $i, j \in P$, we define

$$c_\phi(i,j) \quad = \quad \omega(\phi) \cdot d_{i,j}. \tag{3.1}$$

Clearly, this definition satisfies the above requirements for communication cost functions. Furthermore, it allows to represent incomplete network topologies by defining $d_{i,j} = \infty$, if no communication channel between the processors i and j exists.

A corresponding *distance-independent* version of this model is defined by $d_{i,j} = 1$, for all $i, j \in P$ and $i \neq j$. Doing so, the above definition is rewritten as:

$$c_\phi(i,j) \quad = \quad \begin{cases} \omega(\phi) & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Note, that both of these models are simplified in the manner that they ignore additional costs caused by network access, serialization of data, message building and so on.

The following section will use the distance-dependent communication cost model just introduced to estimate the total communication costs of local computation.

4 Communication Costs in Local Computation

Section 2.2 introduced a valuation algebra as a suitable framework to represent knowledge and information. The following step is now to focus on the treatment of this knowledge, which means essentially query answering. In the terms of valuation algebras, this task is expressed by the (extended) projection problem formulated in (Schneuwly *et al.*, 2004). Thereby, all valuations within a knowledgebase $\{\phi_1, \phi_2, \ldots, \phi_n\}$ need to be combined and from the resulting total knowledge, projections to the domains of interest have to be computed. But this definition of the projection problem does not consider the distributed nature of information and knowledge at all. The distribution comes into play as soon as projection problems whose valuations reside on different processors are considered. This is expressed by the following extension of the projection problem definition:

Definition 4 *A distributed projection problem for a set of queries* $\{x_1, \ldots, x_s\}$, $x_i \subseteq d(\phi_1) \cup \ldots \cup d(\phi_n)$, *is given by the computational tasks*

$$(\phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_n)^{\downarrow x_i}$$

together with an assignment mapping $\chi : \{\phi_1, \ldots, \phi_n\} \to P$ *determining the host processor of each valuation* ϕ_i.

Executing such a sequence of combinations, followed by a single marginalization turns out to be computationally infeasible, because the complexity of the operations tends to increase exponentially with the size of the involved domains. Local

computation goes around this problem by offering a set of efficient algorithms that schedule the operations in such a way, that the domains do not grow significantly. These algorithms are described as message passing schemes on graphical structures called join trees.

Definition 5 *A join tree is a labeled tree (V, E) whose labeling function $\lambda : V \to D$ satisfies the running intersection property, i.e. for two nodes $v_1, v_2 \in V$, if $X \in \lambda(v_1) \cap \lambda(v_2)$, then X is contained in every node label on the unique path between v_1 and v_2.*

(Schneuwly *et al.*, 2004) describe in detail the way from a given projection problem towards a suitable join tree. In doing so, the important objective is that the join tree covers both, the domains of the factors in the projection problem, often called knowledgebase factors, and the queries. This is the case, if the join tree has the following properties:

- for each factor ϕ_i, there exists a node $v \in V$ with $d(\phi_i) \subseteq \lambda(v)$,
- for each query x_i, there exists a node $v \in V$ with $x_i \subseteq \lambda(v)$.

Such a join tree is called *covering join tree*.

Once a covering join tree has been found, we distribute the knowledgebase factors among its nodes. This is done by an assignment mapping $a : \{1, 2, \ldots, n\} \to V$, that assigns to every factor ϕ_i a join tree node $a(i) \in V$ with $d(\phi_i) \subseteq \lambda(a(i))$. This results in a new factorization

$$\phi_1 \otimes \ldots \otimes \phi_n \quad = \quad \psi_1 \otimes \ldots \otimes \psi_m, \tag{4.1}$$

where $m = |V|$ and

$$\psi_i \quad = \quad e \otimes \bigotimes_{j:a(j)=i} \phi_j. \tag{4.2}$$

The factors ψ_i are often called join tree factors and each of them is either a combination of the original knowledgebase factors or the identity element e.

However, in the case of a distributed projection problem, the accruement of the new factorization deserves closer attention. Let us assume that a given join tree factor ψ_i is a combination of two projection problem factors ϕ_j and ϕ_k, which do not reside on the same processor, i.e. $\chi(\phi_j) \neq \chi(\phi_k)$. To actually compute ψ_i, either ϕ_j or ϕ_k has to be transmitted over the network towards the processor of the other factor. According to our model, this causes communication costs of at least

$$d_{\chi(\phi_j),\chi(\phi_k)} \cdot \min\{\omega(\phi_j), \omega(\phi_k)\}.$$

Such antecedent costs are undesirable for our future studies. They are produced before the local computation algorithms are started and therefore falsify the total

communication costs of local computation that will be determined. To prevent such foregoing costs, we demand that each join tree factor ψ_i corresponds either to a knowledgebase factor ϕ_j or to the identity element e. This is not a restriction. (Lehmann, 2001) describes a bundle of efficient join tree construction algorithm that all respect this claim. Furthermore, each join tree containing nodes whose factors are made up of multiple knowledgebase factors can trivially be transformed in a new join tree respecting the claim. This is basically done by adding a sufficient number of new leaf nodes and distributing the supernumerary factors among them. Figure 4.1 illustrates this simple transformation.

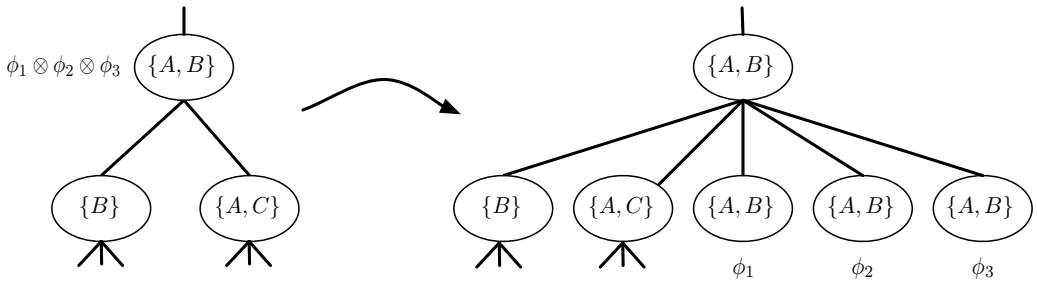

Figure 4.1: Transformation of a node whose factor is made up from multiple knowledgebase factors into a sub-tree with each node containing either a single knowledgebase factor or the identity element.

The property that each join tree factor ψ_i corresponds either to a knowledgebase factor ϕ_j or to the identity element induces naturally a new processor assignment mapping $\xi : V \to P$ which is defined as:

$$\xi(i) \;=\; \begin{cases} \chi(\phi_j) & \text{if } \psi_i = \phi_j, \\ p \in P \text{ arbitrary} & \text{otherwise.} \end{cases} \tag{4.3}$$

for all $i \in V$. To each node, we assign the processor where its factor is hosted. Because the identity elements have been introduced artificially, an arbitrary processor is assigned to these nodes. This colludes with classical theory of local computation, where join tree nodes are often regarded as virtual processors, exchanging messages with their neighbor nodes (Kohlas, 2003). The case at hand provides a natural realization of this rather philosophical view by identifying a real-world processor with each join tree node.

As already mentioned, local computation is a set of algorithms to solve projection problems efficiently. Among them, we distinguish different architectures, such as for example Shenoy-Shafer (Shenoy & Shafer, 1990), Lauritzen-Spiegelhalter (Lauritzen & Spiegelhalter, 1988) and HUGIN (Jensen *et al.*, 1990). All of them solve the same computational task. The Shenoy-Shafer is the most general architecture and valid for every valuation algebra. The other architectures exploit additional properties of the underlying valuations algebra to augment the algorithm's efficiency.

A common characteristic of these architectures is that they are best described as message passing schemes on join tree nodes. Each node can process incoming

messages, compute new messages and send them to other neighboring nodes. The underlying algorithms schedule these messages according to a numbering of the nodes which is introduced beforehand. We choose an arbitrary node as the join tree's root node and assign number $m = |V|$ to it. Then, by directing all edges towards this root node m, it is possible to determine a numbering in such a way, that if j is a node on the path from node i to m, then $j > i$. Note that this numbering is not unique. In the message passing scheme, each node $i \in V$ sends a message $\mu_{i\to j}$ to all its neighbors j and the algorithm terminates as soon as all nodes have received the messages from their neighbors.

We restrict our attention on the Shenoy-Shafer architecture. As mentioned above, it is the most general architecture and due to the uniform description as message passing schemes typical for all other architectures.

In the Shenoy-Shafer architecture, we assume the existence of *mailboxes* between each pair of linked nodes in the join tree. These mailboxes serve as message caches. Once a message is sent from one node to its neighbor, it is stored in the interjacent mailbox and stays available up to the end of the algorithms. The message itself, sent from node i to node j, is defined as:

$$\mu_{i\to j} = \left(\psi_i \otimes \bigotimes_{k\in ne(i), j\neq k} \mu_{k\to i}\right)^{\downarrow \sigma_{i\to j}\cap\lambda(j)}, \tag{4.4}$$

with

$$\sigma_{i\to j} = d(\psi_i) \cup \bigcup_{k\in ne(i), j\neq k} d(\mu_{k\to i}). \tag{4.5}$$

$ne(i)$ denotes the set of neighbor nodes of node i, i.e. all nodes that are directly linked with node i in the join tree. Each node then combines its initial node content with all messages received from the neighbors and the result of these computations is stated by the following theorem.

Theorem 1 *At the end of the message passing in the Shenoy-Shafer architecture, we obtain at node i*

$$\phi^{\downarrow\lambda(i)} = \psi_i \otimes \bigotimes_{j\in ne(i)} \mu_{j\to i}. \tag{4.6}$$

The proof of this theorem can be found in (Schneuwly *et al.*, 2004). Answering the projection problem from this result demands one last marginalization per query x_i,

$$\phi^{\downarrow x_i} = \left(\phi^{\downarrow\lambda(i)}\right)^{\downarrow x_i}.$$

According to the above definition, the communication costs caused by transmitting a single message $\mu_{i\to j}$ is defined as $c(\mu_{i\to j})$. Note, that we use an abbreviated

notation to avoid redundancy. For a given processor assignment mapping ξ, the total communication costs T of the Shenoy-Shafer architecture is given by:

$$T_\xi = \sum_{i=1}^{m} \sum_{j \in ne(i)} c(\mu_{i \to j}) = \sum_{i=1}^{m} \sum_{j \in ne(i)} \omega(\mu_{i \to j}) \cdot d_{\xi(i),\xi(j)} \tag{4.7}$$

Of special interest is again the case where ω is a weight predictor. Then, we can estimate an upper bound for the communication costs in the Shenoy-Shafer architecture. Let f be a function such that $\omega(\phi) = f(d(\phi))$ for all $\phi \in \Phi$. Then, we have

$$\begin{aligned} T_\xi &= \sum_{i=1}^{m} \sum_{j \in ne(i)} f(\sigma_{i \to j} \cap \lambda(j)) \cdot d_{\xi(i),\xi(j)} \\ &\leq \sum_{i=1}^{m} \sum_{j \in ne(i)} f(\lambda(i) \cap \lambda(j)) \cdot d_{\xi(i),\xi(j)} \\ &= 2 \sum_{i=1}^{m-1} f(\lambda(i) \cap \lambda(ch(i))) \cdot d_{\xi(i),\xi(ch(i))}, \end{aligned}$$

where $ch(i)$ denotes the unique neighbor of node i towards the root, i.e. its child node. Executing local computation on a weight predictable valuation algebra represents in a way the situation of full information. We can predict the communication costs of each message that will be sent during the algorithm's run without effectively computing this message.

5 Minimizing Communication Costs

This section goes deeper into the discussion of communication costs in local computation with the objective of minimizing these costs. We will again give special attention to weight predictable valuation algebras, because we have seen in the foregoing section that they allow to compute an upper bound for the total communication costs. This can be regarded as a worst-case analysis and will be the starting point of our investigations.

In section 4 we constructed a join tree factorization

$$\phi_1 \otimes \ldots \otimes \phi_n = \psi_1 \otimes \ldots \otimes \psi_m,$$

from a given projection problem in such a way that each factor ψ_i corresponds either to a knowledgebase factor or to the identity element e. We bring out this important partition as follows:

$$\Psi = \{\psi_1, \psi_2 \ldots \psi_m\} = \Psi_{\overline{e}} \cup \Psi_e,$$

where $\Psi_e = \{\psi \in \Psi : \psi = e\}$ and $\Psi_{\overline{e}} = \Psi \setminus \Psi_e$.

Using this notation, the processor assignment mapping introduced in (4.3) is rewritten as:

$$\xi(i) \quad = \quad \begin{cases} \chi(\psi_i) & \text{if } \psi_i \in \Psi_{\overline{e}}, \\ p \in P \text{ arbitrary} & \text{otherwise.} \end{cases} \tag{5.1}$$

for all $i \in V$. Afterwards, we bounded the total communication costs T of the Shenoy-Shafer architecture in the case of weight predictable valuation algebras by

$$\begin{aligned} T_\xi &\leq 2 \sum_{i=1}^{m-1} f(\lambda(i) \cap \lambda(ch(i))) \cdot d_{\xi(i),\xi(ch(i))} \\ &= 2 \sum_{i=1}^{m-1} f_i \cdot d_{\xi(i),\xi(ch(i))}, \end{aligned}$$

where $f_i = f(\lambda(i) \cap \lambda(ch(i)))$.

In order to minimize the total communication costs, we are looking for a processor assignment mapping ξ that minimizes T_ξ, or its upper bound. There are $|P|^{|\Psi_e|}$ possible assignment mappings ξ, such that a brute force determination becomes infeasible with an increasing cardinality of Ψ_e. Such optimization problems can typically be transformed to an equivalent decision problem (Garey & Johnson, 1990) with the processor assignment ξ as decision variable. In the case at hand, we need to find a processor assignment mapping ξ, such that

$$\sum_{i=1}^{m-1} f_i \cdot d_{\xi(i),\xi(ch(i))} \quad \leq \quad B \tag{5.2}$$

for a given upper bound $B \in \mathbb{N}$. Subsequently, we will refer to this decision problem as the *partial distribution problem* (*PDP*), due to its task of completing a partial processor distribution over join tree nodes.

The remaining part of this section is dedicated to a further analysis of the partial distribution problem. For this purpose, we will first introduce a very famous and well-studied decision problem called *multiway cut problem*, which has a variety of application in the field of parallel computing systems. Afterwards, we will show how the two problems are related and how insights on the multiway cut problem can be used to solve the partial distribution problem efficiently.

5.1 Analysis of the Partial Distribution Problem

(Dahlhaus *et al.*, 1994) originally initiated the study of the so-called *multiterminal cut problem*, also often named *multiway cut problem*. Meanwhile, this problem has been studied extensively because of its widespread application area. Instead of the original description of the multiway cut problem, we will use the more general and illustrative definition given in (Erdös & Szekely, 1994):

Instance: Given a simple graph $G = (V, E)$, a set of colors $C = \{1, 2, \ldots, r\}$, a positive number $B \in \mathbb{N}$ and a weight function $w : E \to \mathbb{N}$, assigning a weight $w(i, j)$ to each edge $(i, j) \in E$. Furthermore, we assume for $N \subseteq V$ a given partial coloration $\nu : N \to C$.

Question: Is there a completed mapping $\overline{\nu} : V \to C$, such that $\overline{\nu}(i) = \nu(i)$, if $i \in N$ and

$$\sum_{(i,j)\in E,\ \overline{\nu}(i)\neq\overline{\nu}(j)} w(i, j) \quad \leq \quad B.$$

A partial coloration ν defines a partition of N by $N_i = \{n \in N, \nu(n) = i\}$. A given edge $(i, j) \in E$ is called *color-changing* in the coloration $\overline{\nu}$, if $\overline{\nu}(i) \neq \overline{\nu}(j)$. The set of color-changing edges in the coloration $\overline{\nu}$ separates every N_i from all other N_j and is therefore called *multiterminal cut*. This constitutes the eponym of the decision problem, because originally, terminals were considered in the problem description instead of colors (Dahlhaus *et al.*, 1994). Figure 5.1 shows an instance of the multiway cut problem with three colors together with a possible coloration of total weight 28.

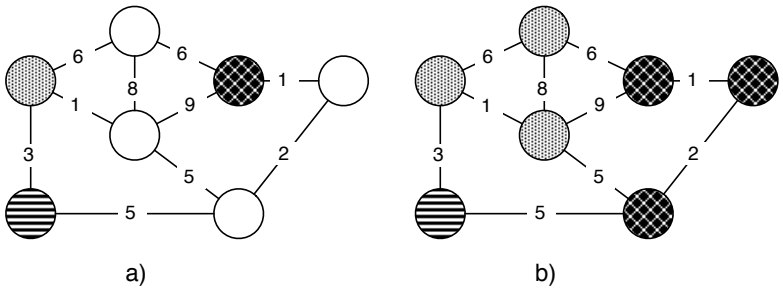

Figure 5.1: a) A multiway cut instance with three colors $C = \{dotted, dashed, grilled\}$. The numbers labeling the graph's edges represent the weight function w. b) A possible coloration with total weight 28.

The above definition of the multiway cut problem includes the so-called *color-independent* version of a weight function, which has also been used in (Dahlhaus *et al.*, 1994). A more general form is proposed by (Erdös & Szekely, 1994) and defined as $w : E \times C \times C \to \mathbb{N}$. In this case, the weight function is called *color-dependent* and the number $w(i, j, p, q)$ specifies the weight of the edge $(i, j) \in E$, if $\overline{\nu}(i) = p$ and $\overline{\nu}(j) = q$. Clearly, color-independence is reached, if for any $(i, j) \in E$, $p_1 \neq q_1$ and $p_2 \neq q_2$, we have $w(i, j, p_1, q_1) = w(i, j, p_2, q_2)$. Finally, if $w(i, j) = c$ for all $(i, j) \in E$, the weight function is said to be *constant*. Note, that without loss of generality, we can assume $c = 1$.

(Dahlhaus *et al.*, 1994) pointed out the following fundamental theorem about the general complexity of multiway cut.

Theorem 2 *The multiway cut problem is NP-complete even for* $|N| = 3$, $|N_i| = 1$ *and constant weight functions.*

For the special case of a planar graph $G = (V, E)$, (Dahlhaus *et al.*, 1994) showed that the multiway cut problem becomes solvable in polynomial time for any fixed $|N| \geq 3$ and $|N_i| = 1$. The corresponding algorithm has complexity

$$O((4|N|)^{|N|} n^{2|N|-1} \log n),$$

which is unfortunately exponential in $|N|$. A further important result has been proved by (Erdös & Szekely, 1994):

Lemma 6 *If* G *is a tree, the multiway cut problem can be solved in polynomial time even for color-dependent weight functions. The corresponding algorithm has time complexity* $O(|V| \cdot r^2)$.

This lemma finally determines the complexity of the partial distribution problem, associated with the minimization of communication costs in local computation with weight predictable valuation algebras.

Theorem 3 *PDP is solvable in polynomial time.*

Proof. We will show, how a given PDP instance can be interpreted as a color-dependent multiway cut instance on the same tree and conclude afterwards that each solution of the multiway cut problem provides directly a solution for the original PDP instance.

Assume a given PDP instance on a join tree $G = (V, E)$. We define $N = \Psi_{\overline{e}}$ and $C = \{c_1, c_2 \ldots, c_{|P|}\}$. The weights of the multiway cut instance are given by $w(i, j, p, q) = f_i \cdot d_{p,q}$ for $j = ch(i)$ and the initial coloration of the multiterminal cut instance $\nu : N \to C$ is defined as: $\nu(i) = c_j$ if $\chi(\psi_i) = p_j$ for $i \in N$ and χ being the processor assignment mapping of the distributed projection problem. The upper bound B is equal for both instances.

Let $\overline{\nu}$ be a solution of the constructed multiway cut problem. Clearly, this implies a solution ξ of the corresponding PDP instance by defining: $\xi(i) = p_j$ if $\overline{\nu}(i) = c_j$. Therefore, and because G is a tree, we know from lemma 6 that solving PDP has polynomial time complexity. □

This result shows that local computation is not only an efficient way to compute projection problems but the underlying join tree structure ensures also that in the case of weight predictable valuation algebras, communication costs caused by distributed projection problems can indeed be minimized with only little effort. A corresponding algorithm to solve the partial distribution problem efficiently is the objective of the following subsection.

5.2 Solving the Partial Distribution Problem

In the proof of theorem 3 we have seen that each PDP instance can easily be reduced to a corresponding multiway cut instance, or in other words, that PDP is an application of the color-dependent multiway cut problem. Because of this tight relation, we will now reformulate the polynomial algorithm for the multiway cut problem given in (Erdös & Szekely, 1994) in terms of the PDP, such that it can be applied directly to the problem at hand. To start with, we will consider a simplified version of the partial distribution problem, subsequently referred to as *autumnal PDP*.

5.2.1 An efficient Algorithm for the Autumnal PDP

Let $le(V) = \{i \in V : pa(i) = \emptyset\}$ denote the set of leaf nodes of the join tree $G = (V, E)$. We assume initially that every leaf node hosts a knowledgebase factor and that identity elements are assigned to all other nodes, i.e. $\{i \in V : \psi_i \neq e\} = le(V)$. We will call this simplified version autumnal PDP and a possible instance is shown in figure 5.2. As we will see below, this assumption does not restrict the generality of the algorithm because every PDP instance can easily be transformed into a autumnal PDP instance.

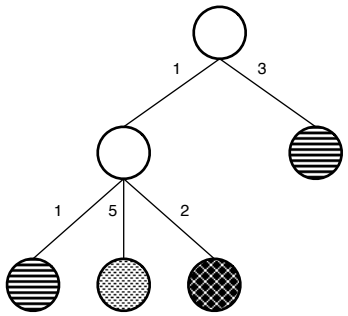

Figure 5.2: A PDP instance with three processors $P = \{dotted,\ dashed,\ grilled\}$ satisfying the imposed restriction that all but only leaves initially have an assigned processor. For simplicity, we assume that all distances in the processor network are constant, i.e. $d_{i,j} = 1$ for $i \neq j$ and $i, j \in P$. The numbers express the weight of the messages sent along the corresponding edge.

Definition 6 *Let $G = (V, E)$ be a tree. A penalty function is a map*

$$pen : V \rightarrow (\mathbb{N}_0 \cup \{\infty\})^{|P|}$$

such that $pen_i(v)$ constitutes the total weight of the sub-tree of node v, on condition that processor i has been assigned to node v.

The processor assignment algorithm that minimizes communication costs can now be formulated as a two-phase process. The first phase corresponds to an inward

tree propagation which assigns penalty values to each node. Afterwards, the second phase propagates outwards and assigns a processor to all interior nodes such that the penalty values are minimized.

Phase I: For each leaf $v \in le(V)$, we define:

$$pen_i(v) = \begin{cases} 0 & \text{if } \chi(\psi_v) = i, \\ \infty & \text{otherwise.} \end{cases}$$

Then, we compute recursively for each node $v \in V \setminus le(V)$:

$$pen_i(v) = \sum_{u \in pa(v)} \min_{j \in P} \{f_u \cdot d_{j,i} + pen_j(u)\}.$$

Figure 5.3 shows the penalty values obtained from applying the inward phase on the instance given in figure 5.2.

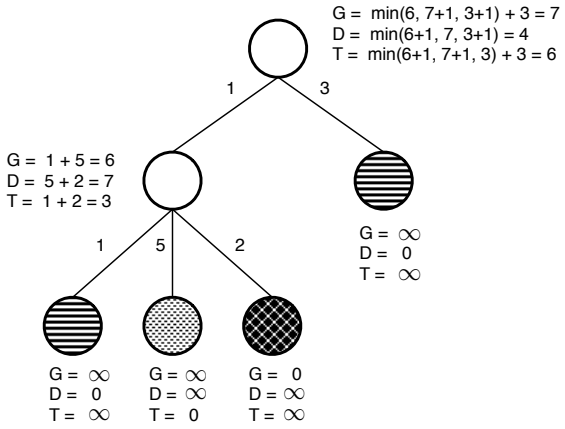

Figure 5.3: The penalty values computed during the inward phase. The capitals stand for: G = *grilled*, D = *dashed* and T = *dotted*.

Phase II: We now determine a complete processor assignment mapping $\xi : V \to P$ such that the total communication costs in the join tree are minimized. For the root node $r \in V$, we define $\xi(r) = i$ such that $pen_i(r) \leq pen_j(r)$ for all $i, j \in P$. Then we assign recursively to each other node $v \in V$, $\xi(v) = i$ if $f_v \cdot d_{i,\xi(ch(v))} + pen_i(v) \leq f_v \cdot d_{j,\xi(ch(v))} + pen_j(v)$ for all $i, j \in P$.

Note that whenever a new node is considered during the outward phase, it is ensured that a processor has already been assigned to its descendants. Therefore, the recursive minimization step of phase two is well-defined. Figure 5.4 finally shows how processors are assigned to the join tree nodes regarding the penalty values computed during the inward phase.

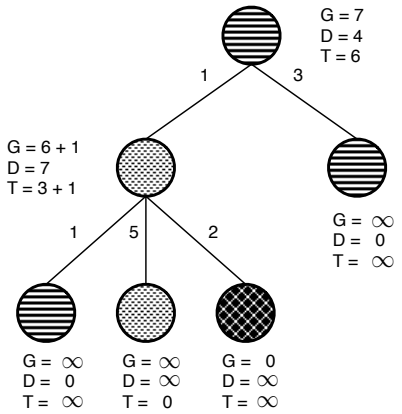

Figure 5.4: The completed processor assignment based on the penalty values from figure 5.3.

It is easy to see that at the end of the algorithm we have $\xi(v) = \chi(\psi_v)$ for all $v \in le(V)$. Since G is a tree, there are $|V| - 1$ edges and therefore the algorithm consists of roughly $2 \cdot |V|$ steps. At each step, we compute $|P|^2$ sums and take the minimum, which results in a time complexity of $O(|V| \cdot |P|^2)$.

5.2.2 Solving general PDP Instances

The above algorithm to minimize communication costs in join trees is based on the assumption that initially all but only leaf nodes possess an assigned processor. Because the initial processor assignment comes out of the factor distribution discussed in section 4, we should now stress a more general version of this algorithm in order to solve arbitrary PDP instances. To do so, we only need to change the penalty values computed during the inward phase, which leads to the following reformulation of phase I:

Phase I*: For each leaf $v \in le(V)$, we define:

$$pen_i(v) = \begin{cases} 0 & \text{if } \psi_v = e, \\ 0 & \text{if } \psi_v \neq e \wedge \chi(\psi_v) = i, \\ \infty & \text{otherwise.} \end{cases}$$

Then, we compute recursively for each node $v \in V \setminus le(V)$:

$$pen_i(v) = \begin{cases} \sum_{u \in pa(v)} \ \min_{j \in P} \ \{f_u \cdot d_{j,i} + pen_j(u)\} & \text{if } \psi_v = e, \\ 0 & \text{if } \psi_v \neq e \wedge \chi(\psi_v) = i, \\ \infty & \text{otherwise.} \end{cases}$$

The modification for leaf nodes states that if it contains an identity element, the assignment is recessed to the outward phase and because the penalty value is

set to zero, it will not influence the decisions in the outward phase but accept the same processor as assigned to its child. The second formula realizes an implicit tree splitting when inner nodes possess already a processor. For such a node i, we cut every edge that connects i with its neighbors such that $|ne(i)|$ new PDP instances are generated. Then, a copy of node i is again added to every former neighbor of i. We accent that every copy of node i is now either a leaf or the root node in the corresponding tree. In the latter case, a slightly modified tree numbering transforms root nodes again to leaf nodes. To sum it up, we find $|ne(i)|$ autumnal PDP instances which can now be treated independently of one another. Clearly, the sum of minimum costs of the created instances equals the minimum costs of the original instance. These transformations are wrapped into the second formula.

5.2.3 Optimization by Antecedent Transformations

We would like to point out that this more general algorithm computes the optimum processor assignment only in relation to the given initial distribution. Consider for example the general PDP instance shown in the left hand part of figure 5.5. The key point is that independently of the minimization algorithm's result, the total costs are built up from two weights w_{11} and $\min\{w_{12}, w_{13}\}$. Furthermore, if the minimization algorithm assigns the *dashed* processor to the root node, the message coming from the deepest node is sent to the *grilled* remote processor, treated and sent back to the *dashed* processor when the root node is reached. Clearly, this overhead is emerged because of the imprudent initial distribution.

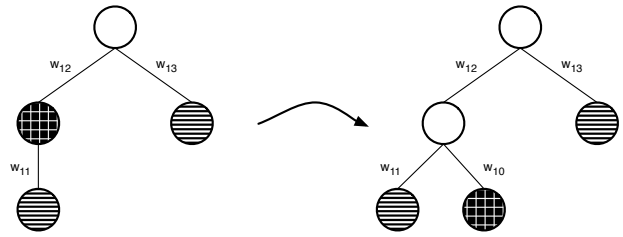

Figure 5.5: Delegation of assignment decisions to the minimization algorithm.

A possible way to avoid such inefficiencies is given by the transformation shown aside which extends essentially the decision scope of the minimization algorithm. Now, the total costs are $\min\{w_{10}, w_{11} + \min\{w_{12}, w_{13}\}\}$, in any case at most as high than before. This is summarized by the following lemma.

Lemma 7 *Every general PDP instance can be transformed into an equivalent autumnal PDP instance with at least as small communication costs.*

The equivalence notion in this lemma constitutes that the new join tree is again a covering join tree for the same projection problem. However, the drawback of this transformation is clearly, that the new join tree demands more computational

effort during the distribute phase of local computation. It is therefore a balance between communication costs and computational effort. This last remark closes our investigations on minimizing communication costs in local computation based on weight predictable valuation algebras.

6 Conclusion

The framework of a valuation algebra has proved its worth to be a suitable mathematical model to represent knowledge and information. Nevertheless, this is not yet fully satisfactory, because this model does not incorporate the distributed nature of information. This paper extended the valuation algebra framework by a weight function, which later has been used to model the communication costs caused by transmitting valuations over a network. Accordingly, we identified join tree nodes with real-world processors and estimated the total communication costs of the Shenoy-Shafer architecture, which has been chosen as a representative for all local computation architectures.

Furthermore, we have seen that the assignment of identity elements to join tree nodes which are not equipped with a factor from the initial knowledgebase, motivates the question on how these total costs can be minimized. Thereby, weight predictable cost functions turned out to be a sufficient condition for the existence of an efficient minimization algorithm. They allow indeed to predict an upper bound for each local computation message without effectively computing it. The main result of this paper states that in this case, the problem of minimizing local computation costs can be solved with only small effort. This is possible due to the underlying tree structure of local computation and increases beyond doubt the practical worth of this theory. Therefore, these insights will be a central point in the further development of local computation software, headed up by NENOK (Pouly, 2004), which is already equipped by the minimization algorithm presented in this paper.

Acknowledgments: Research supported by grant No. 20-67996.02 of the Swiss National Foundation for Research.

References

Ceri, Stefano, & Pelagatti, Giuseppe. 1984. *Distributed databases principles and systems*. New York, NY, USA: McGraw-Hill, Inc.

Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D., & Yannakakis, M. 1994. The Complexity of Multiterminal Cuts. *SIAM J. Comput.*, **23**(4), 864–894.

Dempster, A.P. 1967. Upper and Lower Probabilities Induced by a Multivalued Mapping. *Annals of Math. Stat.*, **38**, 325–339.

Erdös, Peter L., & Szekely, Laszlo A. 1994. On weighted multiway cuts in trees. *Math. Program.*, **65**(1), 93–105.

Garey, Michael R., & Johnson, David S. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co.

Haenni, R. 2004. Ordered Valuation Algebras: a Generic Framework for Approximating Inference. *International Journal of Approximate Reasoning*, **37**(1), 1–41.

Haenni, R., & Lehmann, N. 2003. Implementing belief function computations. *International Journal of Intelligent Systems*, **18**(1), 31–49.

Jensen, F.V., Lautitzen, S.L., & Olesen, K.G. 1990. Bayesian Updating in Causal Probabilistic Networks by Local Computation. *Computational Statistics Quarterly*, **4**, 269–282.

Kohlas, J. 2002. *Information in Context*. Tech. rept. 02-15. Department of Informatics, University of Fribourg.

Kohlas, J. 2003. *Information Algebras: Generic Structures for Inference*. Springer-Verlag.

Kohlas, J. 2004. *Valuation Algebras Induced by Semirings*. Tech. rept. 04-03. Department of Informatics, University of Fribourg.

Lauritzen, S. L., & Spiegelhalter, D. J. 1988. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statis. Soc. B*, **50**, 157–224.

Lehmann, N. 2001. *Argumentation System and Belief Functions*. Ph.D. thesis, Department of Informatics, University of Fribourg.

Menger, Karl. 1942. Statistical Metrics. *Proceedings of the National Academy of Sciences*, **28**, 535–537.

Pouly, M. 2004. *A generic Architecture for local Computation*. M.Phil. thesis, Department of Informatics, University of Fribourg.

Schneuwly, C., Pouly, M., & Kohlas, J. 2004. *Local Computation in Covering Join Trees*. Tech. rept. 04-16. Department of Informatics, University of Fribourg.

Schweizer, B., & Sklar, A. 1960. Statistical Metric Spaces. *Pacific J. Math.*, **10**, 313–334.

Shafer, G. 1976. *The Mathematical Theory of Evidence*. Princeton University Press.

Shafer, G. 1991. *An Axiomatic Study of Computation in Hypertrees*. Working Paper 232. School of Business, University of Kansas.

Shenoy, P. P., & Shafer, G. 1990. Axioms for probability and belief-function proagation. *Pages 169–198 of:* Shachter, Ross D., Levitt, Tod S., Kanal, Laveen N., & Lemmer, John F. (eds), *Uncertainty in Artificial Intelligence 4*. Machine intelligence and pattern recognition, vol. 9. Amsterdam: Elsevier.

Tamer Özsu, M., & Valduriez, P. 1999. *Principles of distributed database systems*. Vol. Second Edition. Prentice Hall.