Fast Monotone Summation over Disjoint Sets*

Petteri Kaski[1], Mikko Koivisto[2], and Janne H. Korhonen[2]

[1] Helsinki Institute for Information Technology HIIT & Department of Information and Computer Science, Aalto University, Finland
[2] Helsinki Institute for Information Technology HIIT & Department of Computer Science, University of Helsinki, Finland

Abstract. We study the problem of computing an ensemble of multiple sums where the summands in each sum are indexed by subsets of size p of an n-element ground set. More precisely, the task is to compute, for each subset of size q of the ground set, the sum over the values of all subsets of size p that are *disjoint* from the subset of size q. We present an arithmetic circuit that, without subtraction, solves the problem using $O((n^p+n^q)\log n)$ arithmetic gates, all monotone; for constant p, q this is within the factor $\log n$ of the optimal. The circuit design is based on viewing the summation as a "set nucleation" task and using a tree-projection approach to implement the nucleation. Applications include improved algorithms for counting heaviest k-paths in a weighted graph, computing permanents of rectangular matrices, and dynamic feature selection in machine learning.

1 Introduction

Weak Algebrisation. Many hard combinatorial problems benefit from *algebrisation*, where the problem to be solved is cast in algebraic terms as the task of evaluating a particular expression or function over a suitably rich algebraic structure, such as a multivariate polynomial ring over a finite field. Recent advances in this direction include improved algorithms for the k-path [25], Hamiltonian path [4], k-coloring [9], Tutte polynomial [6], knapsack [21], and connectivity [14] problems. A key ingredient in all of these advances is the exploitation of an algebraic catalyst, such as the existence of additive inverses for inclusion–exclusion, or the existence of roots of unity for evaluation/interpolation, to obtain fast evaluation algorithms.

Such advances withstanding, it is a basic question whether the catalyst is *necessary* to obtain speedup. For example, fast algorithms for matrix multiplication [11,13] (and combinatorially related tasks such as finding a triangle in a graph [1,17]) rely on the assumption that the scalars have a ring structure, which prompts the question whether a weaker structure, such as a semiring without

* This research was supported in part by the Academy of Finland, Grants 252083 (P.K.), 256287 (P.K.), and 125637 (M.K.), and by the Helsinki Doctoral Programme in Computer Science - Advanced Computing and Intelligent Systems (J.K.).

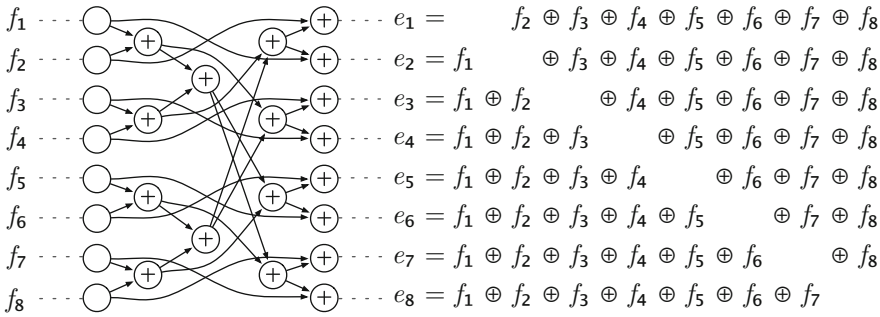
D.M. Thilikos and G.J. Woeginger (Eds.): IPEC 2012, LNCS 7535, pp. 159–170, 2012.

additive inverses, would still enable fast multiplication. The answer to this particular question is known to be negative [18], but for many of the recent advances such an analysis has not been carried out. In particular, many of the recent algebrisations have significant combinatorial structure, which gives hope for *positive* results even if algebraic catalysts are lacking. The objective of this paper is to present one such positive result by deploying *combinatorial* tools.

A Lemma of Valiant. Our present study stems from a technical lemma of Valiant [22] encountered in the study of circuit complexity over a monotone versus a universal basis. More specifically, starting from n variables $f_1, f_2, \ldots, f_n$, the objective is to use as few arithmetic operations as possible to compute the n sums of variables where the jth sum e_j includes all the other variables except the variable f_j, where $j = 1, 2, \ldots, n$.

If additive inverses are available, a solution using $O(n)$ arithmetic operations is immediate: first take the sum of all the n variables, and then for $j = 1, 2, \ldots, n$ compute e_j by subtracting the variable f_j.

Valiant [22] showed that $O(n)$ operations suffice also when additive inverses are *not* available; we display Valiant's elegant combinatorial solution for $n = 8$ below as an arithmetic circuit.



Generalising to Higher Dimensions. This paper generalises Valiant's lemma to higher dimensions using purely combinatorial tools. Accordingly, we assume that only very limited algebraic structure is available in the form of a commutative semigroup $(S, \oplus)$. That is, $\oplus$ satisfies the associative law $x \oplus (y \oplus z) = (x \oplus y) \oplus z$ and the commutative law $x \oplus y = y \oplus x$ for all $x, y, z \in S$, but nothing else is assumed.

By "higher dimensions" we refer to the input not consisting of n values ("variables" in the example above) in S, but rather $\binom{n}{p}$ values $f(X) \in S$ indexed by the p-subsets X of $[n] = \{1, 2, \ldots, n\}$. Accordingly, we also allow the output to have higher dimension. That is, given as input a function f from the p-subsets $[n]$ to the set S, the task is to output the function e defined for each q-subset Y of $[n]$ by

$$e(Y) = \bigoplus_{X : X \cap Y = \emptyset} f(X)\,, \tag{1}$$

where the sum is over all p-subsets X of $[n]$ satisfying the intersection constraint. Let us call this problem (p, q)-*disjoint summation*.

In analogy with Valiant's solution for the case $p = q = 1$ depicted above, an algorithm that solves the (p, q)-disjoint summation problem can now be viewed as a circuit consisting of two types of gates: *input gates* indexed by p-subsets X and *arithmetic gates* that perform the operation $\oplus$, with certain arithmetic gates designated as output gates indexed by q-subsets Y. We would like a circuit that has as few gates as possible. In particular, does there exist a circuit whose size for constant p, q is within a logarithmic factor of the lower bound $\Theta(n^p + n^q)$?

Main Result. In this paper we answer the question in the affirmative. Specifically, we show that a circuit of size $O\big((n^p + n^q) \log n\big)$ exists to compute e from f over an arbitrary commutative semigroup $(S, \oplus)$, and moreover, there is an algorithm that constructs the circuit in time $O\big((p^2 + q^2)(n^p + n^q) \log^3 n\big)$. These bounds hold uniformly for all p, q. That is, the coefficient hidden by O-notation does not depend on p and q.

From a technical perspective our main contribution is combinatorial and can be expressed as a solution to a specific *set nucleation* task. In such a task we start with a collection of "atomic compounds" (a collection of singleton sets), and the goal is to assemble a specified collection of "target compounds" (a collection of sets that are unions of the singletons). The assembly is to be executed by a straight-line program, where each operation in the program selects two *disjoint* sets in the collection and inserts their union into the collection. (Once a set is in the collection, it may be selected arbitrarily many times.) The assembly should be done in as few operations as possible.

Our main contribution can be viewed as a straight-line program of length $O\big((n^p + n^q) \log n\big)$ that assembles the collection $\{\{X : X \cap Y = \emptyset\} : Y\}$ starting from the collection $\{\{X\} : X\}$, where X ranges over the p-subsets of $[n]$ and Y ranges over the q-subsets of $[n]$. Valiant's lemma [22] in these terms provides an optimal solution of length $\Theta(n)$ for the specific case $p = q = 1$.

Applications. Many classical optimisation problems and counting problems can be algebrised over a commutative semigroup. A selection of applications will be reviewed in Sect. 3.

Related Work. "Nucleation" is implicit in the design of many fast algebraic algorithms, perhaps two of the most central are the fast Fourier transform of Cooley and Tukey [12] (as is witnessed by the butterfly circuit representation) and Yates's 1937 algorithm [26] for computing the product of a vector with the tensor product of n matrices of size 2×2. The latter can in fact be directly used to obtain a nucleation process for (p, q)-disjoint summation, even if an inefficient one. (For an exposition of Yates's method we recommend Knuth [19, §4.6.4]; take $m_i = 2$ and $g_i(s_i, t_i) = [s_i = 0 \text{ or } t_i = 0]$ for $i = 1, 2, \ldots, n$ to extract the following nucleation process implicit in the algorithm.) For all $Z \subseteq [n]$ and $i \in \{0, 1, \ldots, n\}$, let

$$a_i(Z) = \{X \subseteq [n] : X \cap [n - i] = Z \cap [n - i],\ X \cap Z \setminus [n - i] = \emptyset\}\,. \qquad (2)$$

Put otherwise, $a_i(Z)$ consists of X that agree with Z in the first $n - i$ elements of $[n]$ and are disjoint from Z in the last i elements of $[n]$. In particular, our objective

is to assemble the sets $a_n(Y) = \{X : X \cap Y = \emptyset\}$ for each $Y \subseteq [n]$ starting from the singletons $a_0(X) = \{X\}$ for each $X \subseteq [n]$. The nucleation process given by Yates' algorithm is, for all $i = 1, 2, \ldots, n$ and $Z \subseteq [n]$, to set

$$a_i(Z) = \begin{cases} a_{i-1}(Z \setminus \{n+1-i\}) & \text{if } n+1-i \in Z, \\ a_{i-1}(Z \cup \{n+1-i\}) \cup a_{i-1}(Z) & \text{if } n+1-i \notin Z. \end{cases} \tag{3}$$

This results in $2^{n-1}n$ disjoint unions. If we restrict to the case $|Y| \leq q$ and $|X| \leq p$, then it suffices to consider only Z with $|Z| \leq p+q$, which results in $O\big((p+q)\sum_{j=0}^{p+q}\binom{n}{j}\big)$ disjoint unions. Compared with our main result, this is not particularly efficient. In particular, our main result relies on "tree-projection" partitioning that enables a significant speedup over the "prefix-suffix" partitioning in (2) and (3).

We observe that "set nucleation" can also be viewed as a computational problem, where the output collection is given and the task is to decide whether there is a straight-line program of length at most ℓ that assembles the output using (disjoint) unions starting from singleton sets. This problem is known to be NP-complete even in the case where output sets have size 3 [15, Problem PO9]; moreover, the problem remains NP-complete if the unions are not required to be disjoint.

2 A Circuit for (p, q)-Disjoint Summation

Nucleation of p-Subsets with a Perfect Binary Tree. Looking at Valiant's circuit construction in the introduction, we observe that the left half of the circuit accumulates sums of variables (i.e., sums of 1-subsets of $[n]$) along what is a perfect binary tree. Our first objective is to develop a sufficient generalisation of this strategy to cover the setting where each summand is indexed by a p-subset of $[n]$ with $p \geq 1$.

Let us assume that $n = 2^b$ for a nonnegative integer b so that we can identify the elements of $[n]$ with binary strings of length b. We can view each binary string of length b as traversing a unique path starting from the root node of a perfect binary tree of height b and ending at a unique leaf node. Similarly, we may identify any node at level ℓ of the tree by a binary string of length ℓ, with $0 \leq \ell \leq b$. See Fig. 1(a) for an illustration. For $p = 1$ this correspondence suffices.

For $p > 1$, we are not studying individual binary strings of length b (that is, individual elements of $[n]$), but rather p-subsets of such strings. In particular, we can identify each p-subset of $[n]$ with a p-subset of leaf nodes in the binary tree. To nucleate such subsets it will be useful to be able to "project" sets upward in the tree. This motivates the following definitions.

Let us write $\{0,1\}^\ell$ for the set of all binary strings of length $0 \leq \ell \leq b$. For $\ell = 0$, we write ϵ for the empty string. For a subset $X \subseteq \{0,1\}^b$, we define the *projection of X to level ℓ* as

$$X|_\ell = \left\{x \in \{0,1\}^\ell \colon \exists y \in \{0,1\}^{b-\ell} \text{ such that } xy \in X\right\}. \tag{4}$$

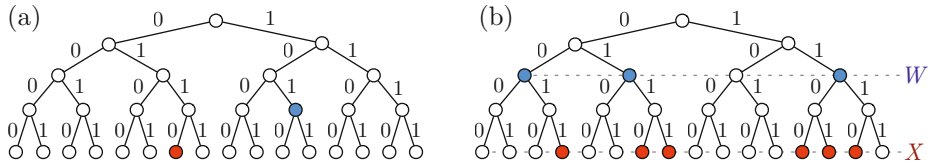


Fig. 1. Representing $\{0,1\}$-strings of length at most b as nodes in a perfect binary tree of height b. Here $b = 4$. (a) Each string traces a unique path down from the root node, with the empty string ϵ corresponding to the root node. The nodes at level $0 \leq \ell \leq b$ correspond to the strings of length ℓ. The red leaf node corresponds to 0110 and the blue node corresponds to 101. (b) A set of strings corresponds to a set of nodes in the tree. The set X is displayed in red, the set W in blue. The set W is the projection of the set X to level $\ell = 2$. Equivalently, $X|_\ell = W$.

That is, $X|_\ell$ is the set of length-ℓ prefixes of strings in X. Equivalently, in the binary tree we obtain $X|_\ell$ by lifting each element of X to its ancestor on level-ℓ in the tree. See Fig. 1(b) for an illustration. For the empty set we define $\emptyset|_\ell = \emptyset$.

Let us now study a set family $\mathcal{F} \subseteq 2^{\{0,1\}^b}$. The intuition here is that each member of $\mathcal{F}$ is a summand, and $\mathcal{F}$ represents the sum of its members. A circuit design must assemble (nucleate) $\mathcal{F}$ by taking disjoint unions of carefully selected subfamilies. This motivates the following definitions.

For a level $0 \leq \ell \leq b$ and a string $W \subseteq \{0,1\}^\ell$ let us define *the subfamily of* $\mathcal{F}$ *that projects to* W by

$$\mathcal{F}_W = \{X \in \mathcal{F} \colon X|_\ell = W\}\,. \tag{5}$$

That is, the family $\mathcal{F}_W$ consists of precisely those members $X \in \mathcal{F}$ that project to W. Again Fig. 1(b) provides an illustration: we select precisely those X whose projection is W.

The following technical observations are now immediate. For each $0 \leq \ell \leq b$, if $\emptyset \in \mathcal{F}$, then we have

$$\mathcal{F}_\emptyset = \{\emptyset\}\,. \tag{6}$$

Similarly, for $\ell = 0$ we have

$$\mathcal{F}_{\{\epsilon\}} = \mathcal{F} \setminus \{\emptyset\}\,. \tag{7}$$

For $\ell = b$ we have for every $W \in \mathcal{F}$ that

$$\mathcal{F}_W = \{W\}\,. \tag{8}$$

Now let us restrict our study to the situation where the family $\mathcal{F} \subseteq 2^{\{0,1\}^b}$ contains only sets of size at most p. In particular, this is the case in our applications. For a set U and an integer p, let us write $\binom{U}{p}$ for the family of all subsets of U of size p, and $\binom{U}{\downarrow p}$ for the family of all subsets of U with size at most p. Accordingly, for integers $0 \leq k \leq n$, let us use the shorthand $\binom{n}{\downarrow k} = \sum_{i=0}^{k} \binom{n}{i}$.

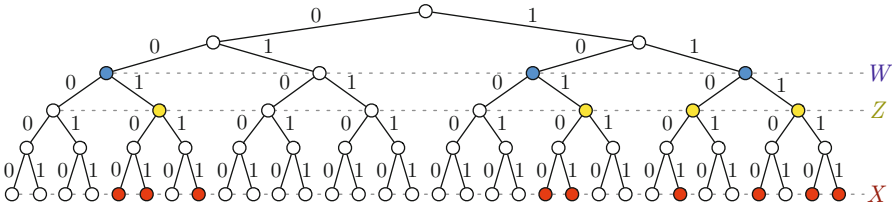


Fig. 2. Illustrating the proof of Lemma 1. Here $b = 5$. The set X (indicated with red nodes) projects to level $\ell = 2$ to the set W (indicated with blue nodes) and to level $\ell + 1 = 3$ to the set Z (indicated with yellow nodes). Furtermore, the projection of Z to level ℓ is W. Thus, each $X \in \mathcal{F}$ is included to $\mathcal{F}_W$ exactly from $\mathcal{F}_Z$ in Lemma 1.

The following lemma enables us to recursively nucleate any family $\mathcal{F} \subseteq \binom{\{0,1\}^b}{\downarrow p}$. In particular, we can nucleate the family $\mathcal{F}_W$ with W in level ℓ using the families $\mathcal{F}_Z$ with Z in level $\ell + 1$. Applied recursively, we obtain $\mathcal{F}$ by proceeding from the bottom up, that is, $\ell = b, b-1, \ldots, 1, 0$. The intuition underlying the lemma is illustrated in Fig. 2. We refer to the full version of this paper for the proof.

Lemma 1. *For all* $0 \leq \ell \leq b-1$, $\mathcal{F} \subseteq \binom{\{0,1\}^b}{\downarrow p}$, *and* $W \in \binom{\{0,1\}^\ell}{\downarrow p}$, *we have that the family* $\mathcal{F}_W$ *is a disjoint union* $\mathcal{F}_W = \bigcup \left\{ \mathcal{F}_Z \colon Z \in \binom{\{0,1\}^{\ell+1}}{\downarrow p}_W \right\}$.

A Generalisation: (p, q)-Intersection Summation. It will be convenient to study a minor generalisation of (p, q)-disjoint summation. Namely, instead of insisting on disjointness, we allow nonempty intersections to occur with "active" (or "avoided") q-subsets A, but require that elements in the intersection of each p-subset and each A are "individualized." That is, our input is not given by associating a value $f(X) \in S$ to each set $X \in \binom{[n]}{\downarrow p}$, but is instead given by associating a value $g(I, X) \in S$ to each pair (I, X) with $I \subseteq X \in \binom{[n]}{\downarrow p}$, where I indicates the elements of X that are "individualized." In particular, we may insist (by appending to S a formal identity element if such an element does not already exist in S) that $g(I, X)$ vanishes unless I is empty. This reduces (p, q)-disjoint summation to the following problem:

Problem 1. (**(p, q)-intersection summation**) Given as input a function g that maps each pair (I, X) with $I \subseteq X \in \binom{[n]}{\downarrow p}$ and $|I| \leq q$ to an element $g(I, X) \in S$, output the function $h \colon \binom{[n]}{\downarrow q} \to S$ defined for all $A \in \binom{[n]}{\downarrow q}$ by

$$h(A) = \bigoplus_{X \in \binom{[n]}{\downarrow p}} g(A \cap X, X)\,. \tag{9}$$

The Circuit Construction. We proceed to derive a recursion for the function h using Lemma 1 to carry out nucleation of p-subsets. The recursion proceeds

from the bottom up, that is, $\ell = b, b-1, \ldots, 1, 0$ in the binary tree representation. (Recall that we identify the elements of $[n]$ with the elements of $\{0,1\}^b$, where n is a power of 2 with $n = 2^b$.) The intermediate functions h_ℓ computed by the recursion are "projections" of (9) using (5). In more precise terms, for $\ell = b, b-1, \ldots, 1, 0$, the function $h_\ell \colon \binom{\{0,1\}^b}{\downarrow q} \times \binom{\{0,1\}^\ell}{\downarrow p} \to S$ is defined for all $W \in \binom{\{0,1\}^\ell}{\downarrow p}$ and $A \in \binom{\{0,1\}^b}{\downarrow q}$ by

$$h_\ell(A, W) = \bigoplus_{X \in \binom{\{0,1\}^b}{\downarrow p}_W} g(A \cap X, X)\,. \tag{10}$$

Let us now observe that we can indeed recover the function h from the case $\ell = 0$. Indeed, for the empty string ϵ, the empty set $\emptyset$ and every $A \in \binom{\{0,1\}^b}{\downarrow q}$ we have by (6) and (7) that

$$h(A) = h_0(A, \{\epsilon\}) \oplus h_0(A, \emptyset)\,. \tag{11}$$

It remains to derive the recursion that gives us h_0. Here we require one more technical observation, which enables us to narrow down the intermediate values $h_\ell(A, W)$ that need to be computed to obtain h_0. In particular, we may discard the part of the active set A that extends outside the "span" of W. This observation is the crux in deriving a succinct circuit design.

For $0 \le \ell \le b$ and $w \in \{0,1\}^\ell$, we define the *span* of w by

$$\langle w \rangle = \left\{ x \in \{0,1\}^b \colon \exists z \in \{0,1\}^{b-\ell} \text{ such that } wz = x \right\}.$$

In the binary tree, $\langle w \rangle$ consists of the leaf nodes in the subtree rooted at w. Let us extend this notation to subsets $W \subseteq \{0,1\}^\ell$ by $\langle W \rangle = \bigcup_{w \in W} \langle w \rangle$. The following lemma shows that it is sufficient to evaluate $h_\ell(A, W)$ only for $W \in \binom{\{0,1\}^\ell}{\downarrow p}$ and $A \in \binom{\{0,1\}^b}{\downarrow q}$ such that $A \subseteq \langle W \rangle$. We omit the proof; please refer to the full version of this paper for details.

Lemma 2. *For all $0 \le \ell \le b$, $W \in \binom{\{0,1\}^\ell}{\downarrow p}$, and $A \in \binom{\{0,1\}^b}{\downarrow q}$, we have*

$$h_\ell(A, W) = h_\ell(A \cap \langle W \rangle, W)\,. \tag{12}$$

We are now ready to present the recursion for $\ell = b, b-1, \ldots, 1, 0$. The base case $\ell = b$ is obtained directly based on the values of g, because we have by (8) for all $W \in \binom{\{0,1\}^b}{\downarrow p}$ and $A \in \binom{\{0,1\}^b}{\downarrow q}$ with $A \subseteq W$ that

$$h_b(A, W) = g(A, W)\,. \tag{13}$$

The following lemma gives the recursive step from $\ell + 1$ to ℓ by combining Lemma 1 and Lemma 2. Again, we defer the details of the proof to the full version of this paper.

Lemma 3. *For* $0 \leq \ell \leq b-1$, $W \in \binom{\{0,1\}^\ell}{\downarrow p}$, *and* $A \in \binom{\{0,1\}^b}{\downarrow q}$ *with* $A \subseteq \langle W \rangle$, *we have*

$$h_\ell(A, W) = \bigoplus_{Z \in \binom{\{0,1\}^{\ell+1}}{\downarrow p}_W} h_{\ell+1}(A \cap \langle Z \rangle, Z)\,. \tag{14}$$

The recursion given by (13), (14), and (12) now defines an arithmetic circuit that solves (p, q)-intersection summation.

Size of the circuit. By (13), the number of input gates in the circuit is equal to the number of pairs (I, X) with $I \subseteq X \in \binom{\{0,1\}^b}{\downarrow p}$ and $|X| \leq q$, which is

$$\sum_{i=0}^{p} \sum_{j=0}^{q} \binom{2^b}{i} \binom{i}{j}\,. \tag{15}$$

To derive an expression for the number of $\oplus$-gates, we count for each $0 \leq \ell \leq b-1$ the number of pairs (A, W) with $W \in \binom{\{0,1\}^\ell}{\downarrow p}$, $A \in \binom{\{0,1\}^b}{\downarrow q}$, and $A \subseteq \langle W \rangle$, and for each such pair (A, W) we count the number of $\oplus$-gates in the subcircuit that computes the value $h_\ell(A, W)$ from the values of $h_{\ell+1}$ using (14).

First, we observe that for each $W \in \binom{\{0,1\}^\ell}{\downarrow p}$ we have $|\langle W \rangle| = 2^{b-\ell}\,|W|$. Thus, the number of pairs (A, W) with $W \in \binom{\{0,1\}^\ell}{\downarrow p}$, $A \in \binom{\{0,1\}^b}{\downarrow q}$, and $A \subseteq \langle W \rangle$ is

$$\sum_{i=0}^{p} \sum_{j=0}^{q} \binom{2^\ell}{i} \binom{i2^{b-\ell}}{j}\,. \tag{16}$$

For each such pair (A, W), the number of $\oplus$-gates for (14) is $\left|\binom{\{0,1\}^{\ell+1}}{\downarrow p}_W\right| - 1$.

Lemma 4. *For all* $0 \leq \ell \leq b-1$, $W \in \binom{\{0,1\}^\ell}{\downarrow p}$, *and* $|W| = i$, *we have*

$$\left|\binom{\{0,1\}^{\ell+1}}{\downarrow p}_W\right| = \sum_{k=0}^{p-i} \binom{i}{k} 2^{i-k}\,. \tag{17}$$

Proof. A set $Z \in \binom{\{0,1\}^{\ell+1}}{\downarrow p}_W$ can contain either one or both of the strings $w0$ and $w1$ for each $w \in W$. The set Z may contain both elements for at most $p-i$ elements $w \in W$ because otherwise $|Z| > p$. Finally, for each $0 \leq k \leq p-i$, there are $\binom{i}{k}2^{i-k}$ ways to select a set $Z \in \binom{\{0,1\}^{\ell+1}}{\downarrow p}_W$ such that Z contains $w0$ and $w1$ for exactly k elements $w \in W$.

Finally, for each $A \in \binom{\{0,1\}^b}{\downarrow q}$ we require an $\oplus$-gate that is also designated as an output gate to implement (11). The number of these gates is

$$\sum_{j=0}^{q} \binom{2^b}{j}\,. \tag{18}$$

The total number of $\oplus$-gates in the circuit is obtained by combining (15), (16), (17), and (18). The number of $\oplus$-gates is thus

$$\begin{aligned}
&\sum_{i=0}^{p}\sum_{j=0}^{q}\binom{2^b}{i}\binom{i}{j}+\sum_{\ell=0}^{b-1}\sum_{i=0}^{p}\sum_{j=0}^{q}\binom{2^\ell}{i}\binom{i2^{b-\ell}}{j}\left(\sum_{k=0}^{p-i}\binom{i}{k}2^{i-k}-1\right)+\sum_{j=0}^{q}\binom{2^b}{j}\\
&\le\sum_{\ell=0}^{b}\sum_{i=0}^{p}\sum_{j=0}^{q}\binom{2^\ell}{i}\binom{i2^{b-\ell}}{j}3^i\le\sum_{\ell=0}^{b}\sum_{i=0}^{p}\sum_{j=0}^{q}\frac{(2^\ell)^i}{i!}\frac{i^j(2^{b-\ell})^j}{j!}3^i\\
&\le\sum_{\ell=0}^{b}\sum_{i=0}^{p}\sum_{j=0}^{q}\frac{(2^\ell)^{\max(p,q)}}{i!}\frac{i^j(2^{m-\ell})^{\max(p,q)}}{j!}3^i\\
&=n^{\max(p,q)}(1+\log_2 n)\sum_{i=0}^{p}\sum_{j=0}^{q}\frac{i^j3^i}{i!j!}\,.
\end{aligned}$$

The remaining double sum is bounded from above by a constant, and thus the circuit defined by (13), (14), and (12) has size $O((n^p+n^q)\log n)$, where the constant hidden by the O-notation does not depend on p and q.

The circuit can be constructed in time $O\big((p^2+q^2)(n^p+n^q)\log^3 n\big)$. We omit the details.

3 Concluding Remarks and Applications

We have generalised Valiant's [22] observation that negation is powerless for computing simultaneously the n different disjunctions of all but one of the given n variables: now we know that, in our terminology, subtraction is powerless for (p,q)-disjoint summation for any constant p and q. (Valiant proved this for $p=q=1$.) Interestingly, requiring p and q be constants turns out to be essential, namely, when subtraction is available, an inclusion–exclusion technique is known [5] to yield a circuit of size $O\big(p\binom{n}{\downarrow p}+q\binom{n}{\downarrow q}\big)$, which, in terms of p and q, is exponentially smaller than our bound $O\big((n^p+n^q)\log n\big)$. This gap highlights the difference of the algorithmic ideas behind the two results. Whether the gap can be improved to polynomial in p and q is an open question.

While we have dealed with the abstract notions of "monotone sums" or semigroup sums, in applications they most often materialise as maximisation or minimisation, as described in the next paragraphs. Also, in applications local terms are usually combined not only by one (monotone) operation but two different operations, such as "min" and "+". To facilitate the treatment of such applications, we extend the semigroup to a semiring $(S,\oplus,\odot)$ by introducing a product operation "$\odot$". Now the task is to evaluate

$$\bigoplus_{X,Y:X\cap Y=\emptyset} f(X)\odot g(Y)\,, \tag{19}$$

where X and Y run through all p-subsets and q-subsets of $[n]$, respectively, and f and g are given mappings to S. We immediately observe that the expression

(19) is equal to $\bigoplus_Y e(Y) \odot g(Y)$, where the sum is over all q-subsets of $[n]$ and e is as in (1). Thus, by our main result, it can be evaluated using a circuit with $O((n^p + n^q) \log n)$ gates.

Application to k-paths. We apply the semiring formulation to the problem of counting the maximum-weight k-edge paths from vertex s to vertex t in a given edge-weighted graph with real weights, where we assume that we are only allowed to add and compare real numbers and these operations take constant time (cf. [24]). By straightforward Bellman–Held–Karp type dynamic programming [2,3,16] (or, even by brute force) we can solve the problem in $\binom{n}{\downarrow k} n^{O(1)}$ time. However, our main result gives an algorithm that runs in $n^{k/2+O(1)}$ time by solving the problem in halves: Guess a middle vertex v and define $f_1(X)$ as the number of maximum-weight $k/2$-edge paths from s to v in the graph induced by the vertex set $X \cup \{v\}$; similarly define $g_1(X)$ for the $k/2$-edge paths from v to t. Furthermore, define $f_2(X)$ and $g_2(X)$ as the respective maximum weights and put $f(X) = (f_1(X), f_2(X))$ and $g(X) = (g_1(X), g_2(X))$. These values can be computed for all vertex subsets X of size $k/2$ in $\binom{n}{k/2} n^{O(1)}$ time. It remains to define the semiring operations in such a way that the expression (19) equals the desired number of k-edge paths; one can verify that the following definitions work correctly: $(c, w) \odot (c', w') = (c \cdot c', w + w')$ and

$$(c, w) \oplus (c', w') = \begin{cases} (c, w) & \text{if } w > w', \\ (c', w') & \text{if } w < w', \\ (c + c', w) & \text{if } w = w'. \end{cases}$$

Thus, the techniques of the present paper enable solving the problem essentially as fast as the fastest known algorithms for the special case of counting *all* the k-paths, for which quite different techniques relying on subtraction yield $\binom{n}{k/2} n^{O(1)}$ time bound [7]. On the other, for the more general problem of counting weighted subgraphs Vassilevska and Williams [23] give an algorithm whose running time, when applied to k-paths, is $O(n^{\omega k/3} + n^{2k/3+c})$, where $\omega < 2.3727$ is the exponent of matrix multiplication and c is a constant; this of course would remain worse than our bound even if $\omega = 2$.

Application to Matrix Permanent. Consider the problem of computing the permanent of a $k \times n$ matrix (a_{ij}) over a *noncommutative semiring*, with $k \leq n$ and even for simplicity, given by $\sum_\sigma a_{1\sigma(1)} a_{2\sigma(2)} \cdots a_{k\sigma(k)}$, where the sum is over all injective mappings σ from $[k]$ to $[n]$. We observe that the expression (19) equals the permanent if we let $p = q = k/2 = \ell$ and define $f(X)$ as the sum of $a_{1\sigma(1)} a_{2\sigma(2)} \cdots a_{\ell\sigma(\ell)}$ over all injective mappings σ from $\{1, 2, \ldots, \ell\}$ to X and, similarly, $g(Y)$ as the sum of $a_{\ell+1\sigma(\ell+1)} a_{\ell+2\sigma(\ell+2)} \cdots a_{k\sigma(k)}$ over all injective mappings σ from $\{\ell+1, \ell+2, \ldots, k\}$ to Y. Since the values $f(X)$ and $g(Y)$ for all relevant X and Y can be computed by dynamic programming in $\binom{n}{k/2} n^{O(1)}$ time, our main result yields the time bound $n^{k/2+O(1)}$ for computing the permanent.

Thus we improve significantly upon a Bellman–Held–Karp type dynamic programming algorithm that computes the permanent in $\binom{n}{\downarrow k} n^{O(1)}$ time, the best

previous upper bound we are aware of for noncommutative semirings [8]. It should be noted, however, that essentally as fast algorithms are already known for *noncommutative rings* [8], and that faster, $2^k n^{O(1)}$ time, algorithms are known for *commutative semirings* [8,20].

Application to Feature Selection. The extensively studied feature selection problem in machine learning asks for a subset X of a given set of available features A so as to maximise some objective function $f(X)$. Often the size of X can be bounded from above by some constant k, and sometimes the selection task needs to be solved repeatedly with the set of available features A changing dynamically across, say, the set $[n]$ of all features. Such constraints take place in a recent work [10] on Bayesian network structure learning by branch and bound: the algorithm proceeds by forcing some features, I, to be included in X and some other, E, to be excluded from X. Thus the key computational step becomes that of maximising $f(X)$ subject to $I \subseteq X \subseteq [n] \setminus E$ and $|X| \leq k$, which is repeated for varying I and E. We observe that instead of computing the maximum every time from scratch, it pays off precompute a solution to (p, q)-disjoint summation for all $0 \leq p, q \leq k$, since this takes about the same time as a single step for $I = \emptyset$ and any fixed E. Indeed, in the scenario where the branch and bound search proceeds to exclude each and every subset of k features in turn, but no larger subsets, such precomputation decreases the running time bound quite dramatically, from $O(n^{2k})$ to $O(n^k)$; typically, n ranges from tens to some hundreds and k from 2 to 7. Admitted, in practice, one can expect the search procedure match the said scenario only partially, and so the savings will be more modest yet significant.

Acknowledgment. We thank Jukka Suomela for useful discussions.

References

1. Alon, N., Yuster, R., Zwick, U.: Finding and counting given length cycles. Algorithmica 17(3), 209–223 (1997)
2. Bellman, R.: Combinatorial processes and dynamic programming. In: Combinatorial Analysis. Proceedings of Symposia in Applied Mathematics, vol. 10, pp. 217–249. ACM (1960)
3. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. J. ACM 9(1), 61–63 (1962)
4. Björklund, A.: Determinant sums for undirected Hamiltonicity. In: 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010), pp. 173–182. IEEE Computer Society, Los Alamitos (2010)
5. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets möbius: fast subset convolution (manuscript submitted for publication)
6. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Computing the Tutte polynomial in vertex-exponential time. In: 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008), pp. 677–686. IEEE Computer Society, Los Alamitos (2008)
7. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Counting Paths and Packings in Halves. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 578–586. Springer, Heidelberg (2009)

8. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Evaluation of permanents in rings and semirings. Information Processing Letters 110(20), 867–870 (2010)
9. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion-exclusion. SIAM Journal on Computing 39(2), 546–563 (2009)
10. de Campos, C.P., Ji, Q.: Efficient structure learning of bayesian networks using constraints. Journal of Machine Learning Research 12, 663–689 (2011)
11. Cohn, H., Kleinberg, R., Szegedy, B., Umans, C.: Group-theoretic algorithms for matrix multiplication. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), pp. 379–388. IEEE Computer Society, Los Alamitos (2005)
12. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation 19, 297–301 (1965)
13. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation 9(3), 251–280 (1990)
14. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time (2011) (manuscript)
15. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of NP-Completeness. W.H. Freeman and Company (1979)
16. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics 10(1), 196–210 (1962)
17. Itai, A., Rodeh, M.: Finding a minimum circuit in a graph. SIAM Journal on Computing 7(4), 413–423 (1978)
18. Kerr, L.R.: The effect of algebraic structure on the computational complexity of matrix multiplications. Ph.D. thesis, Cornell University (1970)
19. Knuth, D.E.: The Art of Computer Programming, 3rd edn. Seminumerical Algorithms, vol. 2. Addison–Wesley, Upper Saddle River (1998)
20. Koutis, I., Williams, R.: Limits and Applications of Group Algebras for Parameterized Problems. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009, Part I. LNCS, vol. 5555, pp. 653–664. Springer, Heidelberg (2009)
21. Lokshtanov, D., Nederlof, J.: Saving space by algebraization. In: 2010 ACM International Symposium on Theory of Computing (STOC 2010), pp. 321–330. ACM, New York (2010)
22. Valiant, L.G.: Negation is powerless for boolean slice functions. SIAM Journal on Computing 15, 531–535 (1986)
23. Vassilevska, V., Williams, R.: Finding, minimizing, and counting weighted subgraphs. In: 2009 ACM International Symposium on Theory of Computing (STOC 2009), pp. 455–464. ACM, New York (2009)
24. Vassilevska, V., Williams, R., Yuster, R.: Finding heaviest H-subgraphs in real weighted graphs, with applications. ACM Transactions on Algorithms 6(3), Art. 44, 23 (2010)
25. Williams, R.: Finding paths of length k in $O^*(2^k)$ time. Information Processing Letters 109(6), 315–318 (2009)
26. Yates, F.: The Design and Analysis of Factorial Experiments. Imperial Bureau of Soil Science, Harpenden, England (1937)