A Polynomial Time Delta-Decomposition Algorithm for Positive DNFs

Denis Ponomaryov

Ershov Institute of Informatics Systems, Novosibirsk State University
ponom@iis.nsk.su

Abstract. We consider the problem of decomposing a positive DNF into a conjunction of DNFs, which may share a (possibly empty) given set of variables \( \Delta \). This problem has interesting connections with traditional applications of positive DNFs, e.g., in game theory, and with the broad topic of minimization of boolean functions. We show that the finest \( \Delta \)-decomposition components of a positive DNF can be computed in polynomial time and provide a decomposition algorithm based on factorization of multilinear boolean polynomials.

1 Introduction

The interest in decomposition of positive DNFs stems from computationally hard problems in game theory, reliability theory, the theory of hypergraphs and set systems. A survey of relevant literature can be found in [1]. In the context of voting games, boolean variables are used to represent voters and the terms of a positive DNF correspond to the winning coalitions, i.e., the groups of voters, who, when simultaneously voting in favor of an issue, have the power to determine the outcome of the vote (i.e., in this case the DNF is evaluated as true). Dual to them are blocking coalitions, i.e., those that force the outcome of the vote to be negative, irrespective of the decisions made by the remaining voters. The problem to find blocking coalitions with a minimal number of voters is easily shown to be equivalent to the hitting set problem, which is NP-complete. Decomposing a DNF into components allows for reducing this problem to inputs having fewer variables.

In this paper, we consider decomposition of a positive DNF into a conjunction of DNFs sharing a given (possibly empty) subset of variables \( \Delta \) (in this case we say that a DNF is \( \Delta \)-decomposable). Decomposition of this kind facilitates finding a more compact representation of a positive boolean function. For example, the following positive DNF can be represented as a conjunction of two formulas:

\[
x a \lor x b \lor y a \lor y b \equiv (x \lor y) (a \lor b)
\]

i.e., it is \( \emptyset \)-decomposable into the components \( x \lor y \) and \( a \lor b \). The following positive DNF is not \( \emptyset \)-decomposable, but it is \( \{d_1, d_2\} \)-decomposable:

\[
x a d_1 \lor x b d_1 d_2 \lor y a d_1 d_2 \lor y b d_2 \equiv (x d_1 \lor y d_2)(a d_1 \lor b d_2)
\]
which can be easily verified by converting the expression into DNF.

In other words, $\Delta$-decomposition allows for common variables (from a set $\Delta$) between the components and partitions the remaining variables of a given formula. In particular, each of the components has fewer variables than the given formula. Decomposition into variable disjoint components (i.e., $\Delta$-decomposition for $\Delta = \emptyset$) is known as disjoint conjunctive decomposition, or simply as AND-decomposition. The notion of OR-decomposition is defined similarly.

The minimization of positive DNFs via decomposition in the sense above is related to open questions, not sufficiently addressed in the previous literature. For example, there is a fundamental work by Brayton et al. on the multilevel synthesis \cite{2}, which provides minimization methods with heuristics working well for arbitrary boolean functions. However, this contribution leaves space for research on minimization for special classes of functions, where the problem is potentially simpler. This is evidenced by the research in \cite{3} and \cite{4}, for example.

It has been observed that the quality of multilevel decomposition (i.e., alternating AND/OR–decomposition) of DNFs strongly depends on the kind of decomposition used at the topmost level. As a rule, OR–decomposition has a priority over AND-decomposition in applications, since it is computationally trivial (while AND-decomposition is considered to be hard and no specialized algorithms for DNFs are known). However, choosing AND-decomposition at the topmost level may provide a more compact representation of a boolean function. For example, application of “AND–first” strategy gives a representation of the following positive DNF

\[
\begin{align*}
\text{absu} \lor \text{absv} \lor \text{absw} \lor \text{abtu} \lor \text{abtv} \lor \text{abx} \lor \\
\text{acsu} \lor \text{acsu} \lor \text{acsu} \lor \text{actv} \lor \text{actw} \lor \text{acx} \lor \\
\text{desu} \lor \text{desv} \lor \text{desw} \lor \text{detu} \lor \text{detv} \lor \text{dex} \lor 
\end{align*}
\]

in the form $(ab \lor ac \lor de)(su \lor sv \lor sw \lor tu \lor tv \lor tw \lor xy \lor xz)$. Further, OR–decomposition of the second component gives $su \lor sv \lor sw \lor tu \lor tv \lor tw$ and $xy \lor xz$ (the first component similarly OR-decomposes syntactically). Finally, AND-decomposition of the obtained formulas gives a representation

\[
(a(c \lor b) \lor de)(x \lor z)(s \lor t)(u \lor v \lor w) \lor x(y \lor z),
\]

which is a read-once formula of depth 4 having 13 occurrences of variables.

On the contrary, Espresso\footnote{A well–known heuristic optimizer based on the work of Brayton et al., which is often used as a reference tool for optimization of boolean functions} which implements OR–decomposition at the topmost level, gives a longer expression:

\[
x(a(c \lor b) \lor de)(z \lor y)(a(c \lor b) \lor de)(t \lor s)(w \lor v \lor u)
\]

which is a formula of depth 5 having 18 occurrences of variables and this formula is not read-once.

Bioch \cite{1} studied (variable disjoint) decompositions of positive boolean functions in the form $\varphi \equiv F(G(X_A), X_B)$, where $\{X_A, X_B\}$ is a partition of the
variables of \( \varphi \) and \( F, G \) are some (positive) boolean functions. The set \( X_A \) is called a modular set of \( \varphi \) in this case. By taking \( \varphi = x_1 \lor \ldots \lor x_n \), one can see that the number of modular sets of \( \varphi \) is exponential in \( n \) (since any subset of the variables is modular). Bioch showed that one can compute a tree in time polynomial in the size of an input positive DNF, which succinctly represents all its modular sets. Given a modular set \( X_A \), the corresponding component \( G(X_A) \) can be also computed in polynomial time. These important results leave the question open however, which modular sets one should choose, when trying to find a compact representation of a boolean formula. For example, the modular tree for the DNF \( \varphi \) from equation (1) consists of the singleton variable subsets (plus the set of all the variables of \( \varphi \) being modular by definition), from which one can obtain representations of the form \( \varphi \equiv x(a \lor b) \lor ya \lor yb \) (and similarly for \( y \) and \( a, b \) selected). On the other hand, \( \emptyset \)-decomposition of this formula gives the representation \( \varphi \equiv (x \lor y)(a \lor b) \), which is more compact.

It has been shown in [5] that \( \emptyset \)-decomposition of a positive DNF can be computed in time polynomial in the size of the input formula (given as a string). In fact, it has been proved that \( \emptyset \)-decomposition reduces to factorization of a multilinear boolean polynomial efficiently obtained from the input positive DNF. For the latter problem the authors have provided a polynomial time algorithm based on the computation of formal derivatives. In this paper, we generalize the results from [5]. First, we provide an algorithm, which computes the finest \( \emptyset \)-decomposition components of a positive DNF. Since \( \emptyset \)-decomposable formulas are expected to be rare, we consider the more general notion of \( \Delta \)-decomposition. The problem of computing \( \Delta \)-decomposition (for an arbitrary given \( \Delta \)) can be reduced to \( \emptyset \)-decomposition: it suffices to test whether each of the (exponentially many) DNFs, obtained from the input one for (all possible) evaluations of \( \Delta \)-variables, is \( \emptyset \)-decomposable with the same variable partition. The reduction holds for arbitrary boolean expressions, not necessarily positive ones. We show however that for positive DNFs, it suffices to make \( \emptyset \)-decomposition tests only for a polynomial number of (positive) DNFs obtained from the input one. As a result, we obtain a polynomial time algorithm, which computes the finest \( \Delta \)-decomposition components of a positive DNF for a subset of variables \( \Delta \).

2 Preliminaries

A boolean expression is a combination of constants 0, 1 and boolean variables using conjunction, disjunction, and negation. A boolean expression is a DNF if it is a disjunction of terms (where each term is a conjunction of literals and constants). We assume there are no double negations of variables in boolean expressions, no double occurrences of the same term in a DNF, or of the same literal in a term of a DNF. A DNF is positive if it does not contain negated variables and the constant 0. For a set of variables \( V \), a \( V \)-literal is a literal with a variable from \( V \). We use the notation \( \text{vars}(\varphi) \) for the set of variables of an expression \( \varphi \). For boolean expressions \( \varphi \) and \( \psi \), we write \( \varphi \equiv \psi \) if they are logically equivalent.
Definition 1 (\(\Delta\)-decomposability). Let \(\varphi\) be a boolean expression and \(\Delta \subseteq \text{vars}(\varphi)\) a subset of variables. The expression \(\varphi\) is \(\Delta\)-decomposable if it is equivalent to the conjunction of boolean expressions \(\psi_1, \ldots, \psi_n\), where \(n \geq 2\), such that the following holds:

a. \(\bigcup_{i=1}^{n} \text{vars}(\psi_i) \subseteq \text{vars}(\varphi)\);

b. \(\text{vars}(\psi_i) \cap \text{vars}(\psi_j) \subseteq \Delta\), for all \(1 \leq i, j \leq n, i \neq j\);

c. \(\text{vars}(\psi_i) \setminus \Delta \neq \emptyset\), for \(i = 1, \ldots, n\).

The expressions \(\psi_1, \ldots, \psi_n\) are called (\(\Delta\)-)decomposition components of \(\varphi\). If \(\Delta = \emptyset\) and the above holds then we call \(\varphi\) decomposable, for short.

Clearly, \(\Delta\)-decomposition components can be subject to a more fine-grained decomposition, wrt the same or different delta’s. It immediately follows from this definition that a boolean expression, which contains at most one non-\(\Delta\)-variable, is not \(\Delta\)-decomposable. Observe that conditions a,c in the definition are important: if any of them is omitted then every boolean expression turns out to be \(\Delta\)-decomposable, for any (proper) subset of variables \(\Delta\).

Definition 2 (Finest Variable Partition wrt \(\Delta\)). Let \(\varphi\) be a boolean expression, \(\Delta \subseteq \text{vars}(\varphi)\) a subset, and \(\pi = \{V_1, \ldots, V_{|\pi|}\}\) a partition of \(\text{vars}(\varphi) \setminus \Delta\). The expression \(\varphi\) is said to be \(\Delta\)-decomposable with partition \(\pi\) if it has \(\Delta\)-decomposition components \(\psi_i\), for \(i = 1, \ldots, |\pi|\), such that \(\text{vars}(\psi_i) = V_i \cup \Delta\).

The finest variable partition of \(\varphi\) (wrt \(\Delta\)) is \(\{\text{vars}(\varphi)\}\) if \(\varphi\) is not \(\Delta\)-decomposable. Otherwise it is the partition \(\pi\), which corresponds to the non-\(\Delta\)-decomposable components \(\psi_i\) of \(\varphi\).

It will be clear from the results of this paper that for any \(\Delta \subseteq \text{vars}(\varphi)\), the finest variable partition of a positive DNF \(\varphi\) is unique. In the general case, (e.g., for arbitrary boolean expressions), this property follows from the result proved in [6] for a broad class of logical calculi including propositional logic. As will be shown below, once the finest variable partition of a DNF \(\varphi\) is computed, the corresponding (non-decomposable) components of \(\varphi\) are easily obtained.

Throughout the text, we use the term assignment as a synonym for a consistent set of literals. Given a set of variables \(V = \{x_1, \ldots, x_n\}\), where \(n \geq 1\), a \(V\)-assignment is a set of literals \(\{l_1, \ldots, l_n\}\), where \(l_i\) is a literal over variable \(x_i\), for \(i = 1, \ldots, n\).

Let \(\varphi\) be a DNF, \(V \subseteq \text{vars}(\varphi)\) a subset, and \(X\) a \(V\)-assignment such that there is a term in \(\varphi\), whose set of \(V\)-literals is contained in \(X\). Then the substitution of \(\varphi\) with \(X\), denoted by \(\varphi[X]\), is a DNF defined as follows:

- if there is a term \(t\) in \(\varphi\) such that every literal from \(t\) is contained in \(X\) then \(\varphi[X] = 1\) (and \(X\) is called satisfying assignment for \(\varphi\), notation: \(X \models \varphi\));
- otherwise \(\varphi[X]\) is the DNF obtained from \(\varphi\) by removing terms, whose set of \(V\)-literals is not contained in \(X\), and by removing \(V\)-literals in the remaining terms.
For example, for the positive DNF $\varphi$ from \ref{eq:positiveDNF} we have $\varphi[\{d_1, -d_2\}] = xa$.

For DNFs $\varphi$ and $\psi$, we say that $\varphi$ implies $\psi$ if for any assignment $X$, it holds $X \not\models \varphi$ or $X \models \psi$. For a set of variables $V$ and a DNF $\varphi$, a projection of $\varphi$ onto $V$, denoted as $\varphi|_V$, is the DNF obtained from $\varphi$ as follows. If there is a term in $\varphi$ which does not contain a variable from $V$ then $\varphi|_V = 1$. Otherwise $\varphi|_V$ is the DNF obtained from $\varphi$ by removing literals with a variable not from $V$ in the terms of $\varphi$. It should be clear from this definition that $\varphi$ implies $\varphi|_V$ (in the literature, projection is also known as a uniform interpolant or the strongest consequence of $\varphi$ wrt $V$). For instance, for the DNF $\varphi$ from \ref{eq:positiveDNF} we have $\varphi_{\{x,y,d_1,d_2\}} = xd_1 \lor xd_2 \lor yd_1d_2 \lor yd_2$.

**Lemma 1 (Decomposition Components as Projections).** Let $\varphi$ be a DNF, which is $\Delta$-decomposable with a variable partition $\pi = \{V_1, \ldots, V_n\}$. Then $\varphi|_{U_i}$, $i = 1, \ldots, n$, are $\Delta$-decomposition components of $\varphi$.

**Proof.** Since $\varphi$ implies $\varphi|_{U_i}$, for $i = 1, \ldots, n$, it suffices to demonstrate that $\bigwedge_{i=1}^{n} \varphi|_{U_i}$ implies $\varphi$. Assume $\varphi \equiv \varphi_1 \land \cdots \land \varphi_n$, where $\varphi_1, \ldots, \varphi_n$ are $\Delta$-decomposition components of $\varphi$, with $\text{vars}(\varphi_i) = U_i$, for $i = 1, \ldots, n$. We show that $\varphi|_{U_i}$ implies $\varphi_i$, for all $i = 1, \ldots, n$. For suppose there is a $U_i$-assignment $X$ such that $X \models \varphi|_{U_i}$ and $X \not\models \varphi_i$, for some $i \in \{1, \ldots, n\}$. Then by the definition of the projection $\varphi|_{U_i}$, there exists an assignment $X' \supseteq X$ such that $X' \models \varphi$ and $X' \not\models \varphi_i$, which is a contradiction, since $\varphi$ implies $\varphi_i$, for $i = 1, \ldots, n$. As $\varphi_1 \land \cdots \land \varphi_n$ implies $\varphi$, we conclude that $\bigwedge_{i=1}^{n} \varphi|_{U_i}$ implies $\varphi$. □

Let $V$ be a set of variables, $\Delta \subseteq V$ a subset, and $A$ a set of $V$-assignments. By $A|_{\Delta}$ we denote the set of all $\Delta$-assignments $d$, for which there is an assignment $X \in A$ such that $d \subseteq X$. For a $\Delta$-assignment $d$, the notation $A(d)$ stands for the set of $(V \setminus \Delta)$-assignments $X$ such that $X \cup d \in A$. Let $V_1, V_2$ be disjoint sets of variables and for $i = 1, 2$, let $A_i$ be a set of $V_i$-assignments. Then the notation $A_1 \bowtie A_2$ stands for the set of all assignments $X_1 \cup X_2$ such that $X_1 \in A_1$ and $X_2 \in A_2$. The intuitive relationship between the cartesian combinations of assignments is illustrated by the following remark and is put formally in the subsequent Lemma \ref{lem:cartesianProduct}.

**Remark 1 (Conjunction of DNFs is Similar to Cartesian Product).** Taking the conjunction of DNFs $\xi_1 \lor \cdots \lor \xi_m$ and $\zeta_1 \lor \cdots \lor \zeta_n$ gives a DNF, which has the form $\bigvee (\xi_i \land \zeta_j)$, for all pairs $i,j$, with $1 \leq i \leq m$, $1 \leq j \leq n$.

**Lemma 2 (Decomposability Criterion).** Let $\varphi$ be a DNF and $A$ the set of satisfying assignments for $\varphi$. Then $\varphi$ is $\Delta$-decomposable with a partition $\pi = \{V_1, \ldots, V_\ell\}$ iff for all $d \in A|_{\Delta}$ it holds that $A(d) = A(d)_{V_1} \bowtie \cdots \bowtie A(d)_{V_\ell}$.

In this paper, we are concerned with the problem of finding the finest variable partition of a positive DNF $\varphi$ wrt a subset $\Delta$ of its variables. By Lemma \ref{lem:decompositionComponents} decomposition components of $\varphi$ can be easily obtained from the finest variable partition. We assume that boolean expressions are given as strings and thus, the size of an expression $\varphi$ is the length of the string, which represents $\varphi$. 
For the sake of completeness, first we describe a factorization algorithm for multilinear boolean polynomials, which is based on the results from [5,7]. Then we provide a $\emptyset$-decomposition algorithm for a positive DNF $\varphi$ based on factorization of a boolean polynomial, which is obtained from $\varphi$ by a simple syntactic transformation. Finally, we demonstrate that $\Delta$-decomposition of a positive DNF reduces to $\emptyset$-decomposition of (a polynomial number of) positive DNFs obtained from the input one and devise the corresponding polynomial time $\Delta$-decomposition algorithm.

3 Factorization of Boolean Polynomials

In [8], Shpilka and Volkovich established the prominent result on the equivalence of polynomial factorization and identity testing. It follows from their result that a multilinear boolean polynomial can be factored in time cubic in the size of the polynomial given as a string. This result has been rediscovered in [5,7], where the authors have provided a factorization algorithm based on the computation of derivatives of multilinear boolean polynomials, which allows for deeper optimizations. Without going into implementation details, we employ this result here to formulate an algorithm, which computes the finest $\emptyset$-decomposition components of a positive DNF $\varphi$. Hereafter we assume that polynomials do not contain double occurrences of the same monomial.

Definition 3. A polynomial $F$ is factorable if $F = G_1 \cdot \ldots \cdot G_n$, where $n \geq 2$ and $G_1, \ldots, G_n$ are some non-constant polynomials. Otherwise $F$ is irreducible. The polynomials $G_1, \ldots, G_n$ are called factors of $F$. For a polynomial $F$, the finest variable partition of $F$ is \{vars($F$)\} if $F$ is irreducible and otherwise consists of the sets of the variables of the irreducible factors of $F$.

It is important to realize that since we consider multilinear polynomials (every variable can occur only in the power of $\leq 1$), the factors are polynomials over disjoint sets of variables. Note that the finest variable partition of a multilinear boolean polynomial is unique, since the ring of these polynomials is a unique factorization domain. We now formulate the first important observation, which is a strengthening of Theorem 5 from [5].

Theorem 1 (Computing Finest Variable Partition for Polynomial). For a multilinear boolean polynomial $F$, the (unique) finest partition of the variables of $F$ can be found in time polynomial in the size of $F$.

It is proved in [5] that testing whether $F$ is factorable and computing its factors can be done in time polynomial in the size of $F$ given as a string. By applying the factorization procedure to the obtained factors recursively, one obtains a partition of the variables of $F$, which corresponds to the irreducible factors of $F$. This is implemented in FindPartition procedure given below,
which is a modification of the factorization algorithm from [5]. It is also shown
in [5, 7] that once a partition of variables, which corresponds to the factors of
$F$, is computed, the factors can be easily obtained as projections of $F$ onto the
components of the partition (see the notion of projection below).

The \texttt{FindPartition} procedure takes a boolean polynomial $F$ as an input and
outputs the finest partition of $\text{vars}(F)$ in time polynomial in the size of $F$. A
few notations are required. For a polynomial $F$, we denote by $\text{vars}(F)$ the set of
the variables of $F$. For a variable $x \in \text{vars}(F)$ and a value $a \in \{0, 1\}$, we denote
by $F_{x=a}$ the polynomial obtained from $F$ by substituting $x$ with $a$. A derivative $\frac{\partial F}{\partial x}$ is a shorthand for the polynomial $F_{x=0} + F_{x=1}$. Given a set of variables $V$ and a monomial $m$, the projection of $m$ onto $V$ (denoted as $m|_V$) is 1 if $m$ does not contain any variable from $V$, or is equal to the monomial obtained from $m$ by removing all the variables not contained in $V$, otherwise. The projection of a polynomial $F$ onto $V$, denoted as $F|_V$, is the polynomial obtained by projecting the monomials of $F$ onto $V$ and by removing duplicate monomials.

Lines 2-4 of \texttt{FindPartition} is a test for a simple sufficient condition for irre-
ducibility: if a polynomial is a constant then it cannot be factorable. Lines 5-15
implement a test for trivial factors: if some variable $z$ is present in every mono-
monic of $F$, then $z$ is an irreducible factor. In the recursive part of the procedure,
the remaining sets from the finest variable partition of $F$ are computed as the
values of the variable $\Sigma$ and are added to $\text{FinestPartition}$.

1: \begin{verbatim}
procedure FindPartition(F)
2: if $F == 0$ or $F == 1$ then
3: return $\text{vars}(F)$
4: end if
5: for $z$ a variable occurring in every monomial of $F$ do
6: $\text{FinestPartition}.add(\{z\})$
7: $F \leftarrow F_{z=1}$
8: end for
9: if $F$ does not contain any variables then
10: return $\text{FinestPartition}$
11: end if
12: if $F$ contains a single variable, e.g., $x$ then
13: $\text{FinestPartition}.add(\{x\})$
14: return $\text{FinestPartition}$
15: end if
16: $V \leftarrow \text{variables of } F$
17: repeat
18: $\Sigma \leftarrow \emptyset$; $F \leftarrow F|_V$
19: pick a variable $x$ from $V$
20: $\Sigma.add(x)$; $V \leftarrow V \setminus \{x\}$
21: $G \leftarrow F_{x=0} \cdot \frac{\partial F}{\partial x}$
22: for a variable $y$ from $V$ do
23: if $\frac{\partial G}{\partial y} \neq 0$ then
24: $\Sigma.add(y)$
25: end if
26: end for
27: $\text{FinestPartition}.add(\Sigma)$
28: $V \leftarrow V \setminus \Sigma$
29: until $V = \emptyset$
30: return $\text{FinestPartition}$
31: end procedure
\end{verbatim}

4 \emptyset-decomposition of Positive DNFs

A term $t$ of a DNF $\varphi$ is called redundant in $\varphi$ if there exists another term
$t'$ of $\varphi$ such that every literal of $t'$ is present in $t$ (i.e., $t' \subseteq t$). For example, the
term $xy$ is redundant in $xy \lor x$. It is easy to see that removing redundant terms
gives a logically equivalent DNF.
Let us note the following simple fact:

**Lemma 3 (Existence of Positive Components).** Let \( \varphi \) be a positive boolean expression and \( \Delta \subseteq \text{vars}(\varphi) \) a subset of variables. If \( \varphi \) is \( \Delta \)-decomposable then it has decomposition components, which are positive expressions.

**Proof.** It is known (e.g., see Theorem 1.21 in [9]) that a boolean expression \( \psi \) is equivalent to a positive one in a variable \( x \) iff for the set of satisfying assignments \( A \) for \( \varphi \) the following property holds: if \( \{l_1, \ldots, l_n, \neg x\} \in A \), where \( l_1, \ldots, l_n \) are literals, then \( \{l_1, \ldots, l_n, x\} \in A \). Clearly, this property is preserved under decomposition: if a set of assignments \( A \) satisfies the property and it holds that \( A = A_1 \bowtie \ldots \bowtie A_n \), then so do the sets \( A_i \), for \( i = 1, \ldots, n \). Thus, the claim follows directly from Lemma 2. \( □ \)

The next important observation is a strengthening of the result from [5], which established the complexity of \( \varnothing \)-decomposition for positive DNFs.

**Theorem 2 (Computing the Finest Variable Partition wrt \( \Delta = \varnothing \)).** The finest variable partition of a positive DNF \( \varphi \) can be computed in time polynomial in the size of \( \varphi \).

Let \( P \) be a 1-1 mapping, which for a positive DNF \( \varphi \) gives a multilinear boolean polynomial \( P(\varphi) \) over \( \text{vars}(\varphi) \) obtained by replacing the conjunction and disjunction with \( \cdot \) and +, respectively. The theorem is proved by showing that decomposition components of a positive DNF \( \varphi \) can be recovered from factors of a polynomial \( P(\psi) \) constructed for a DNF \( \psi \), which is obtained from \( \varphi \) by removing redundant terms. The idea is illustrated in \( \varnothing \text{Decompose} \) procedure below, which for a given positive DNF \( \varphi \) computes the finest variable partition of \( \varphi \). It relies on the factorization procedure from Section 3 and is employed as a subroutine in \( \Delta \)-decomposition algorithm in Section 5. The procedure uses a simple preprocessing, which removes redundant terms. The preprocessing also allows for detecting those variables (line 5 of the procedure) that \( \varphi \) does not depend on. By the definition of decomposability, these variables are decomposition components of \( \varphi \), so they are added as singleton sets into the resulting finest variable partition (at line 6).

1: \textbf{procedure } \varnothing \text{Decompose}(\varphi) \\
2: \hspace{1em} \text{FinestPartition } \leftarrow \varnothing \\
3: \hspace{1em} \psi \leftarrow \text{REMOVEREDUNDTERMS}(\varphi) \\
4: \hspace{1em} \text{FinestPartition } \leftarrow \text{FINDPARTITION}(P(\psi)) \quad \text{▷ see Sect. 3} \\
5: \hspace{1em} \text{for all } x \in \text{vars}(\varphi) \setminus \text{vars}(\psi) \text{ do} \\
6: \hspace{2em} \text{FinestPartition.add(}\{x\}\text{)} \\
7: \hspace{1em} \text{end for} \\
8: \hspace{1em} \text{return } \text{FinestPartition} \\
9: \textbf{end procedure} \\

1: \textbf{procedure } \text{REMOVEREDUNDTERMS}(\varphi) \\
2: \hspace{1em} \text{for all terms } t \text{ in } \varphi \text{ do} \\
3: \hspace{2em} \text{if there exists a term } t' \text{ in } \varphi \text{ s.t. } t' \subseteq t \text{ then} \\
4: \hspace{3em} \text{remove } t \text{ from } \varphi \\
5: \hspace{2em} \text{end if} \\
6: \hspace{1em} \text{end for} \\
7: \hspace{1em} \text{return } \varphi \\
8: \textbf{end procedure}
5 \(\Delta\)-decomposition of Positive DNFs

**Definition 4 (\(\Delta\)-atom).** For a positive DNF \(\varphi\) and a subset \(\Delta \subseteq \text{vars}(\varphi)\), the set of \(\Delta\)-variables of a term of \(\varphi\) is called \(\Delta\)-atom of \(\varphi\).

Note that by definition a \(\Delta\)-atom can also be the empty set. Let \(U\) be the set of unions of \(\Delta\)-atoms of \(\varphi\). Given a set \(X \in U\), we introduce the notation \(\varphi(X)\) as a shortcut for the DNF \(\varphi[X \cup \bar{X}]\), where \(\bar{X} = \{-x \mid x \in \Delta \setminus X\}\).

Let \(\pi\) be a partition of \(\text{vars}(\varphi) \setminus \Delta\). We say that a formula \(\psi\) supports \(\pi\) if every set from the finest variable partition of \(\psi\) wrt \(\Delta\) is contained in some set from \(\pi\). It is easy to see that if \(\varphi\) is \(\Delta\)-decomposable with \(\pi\), then \(\varphi[X]\) supports \(\pi\), for any set of literals \(X\) such that \(\varphi[X]\) is defined.

We formulate two lemmas that are the key to the main result, Theorem 3 in this section.

**Lemma 4 (\(\Delta\)-Decomposability Criterion for Positive DNF).** Let \(\varphi\) be a positive DNF, \(\Delta \subseteq \text{vars}(\varphi)\) a subset, and \(U\) the set of unions of \(\Delta\)-atoms of \(\varphi\). Then \(\varphi\) is \(\Delta\)-decomposable with a variable partition \(\pi\) iff \(\varphi(X)\) supports \(\pi\), for all \(X \in U\).

**Proof.** (\(\Rightarrow\)): Take \(X \in U\). Since \(\varphi\) is positive, \(X\) is a consistent set of literals, \(\varphi(X)\) is defined, and clearly, supports \(\pi\).

(\(\Leftarrow\)): Let \(\pi = \{V_1, \ldots, V_{|\pi|}\}\) be the set of satisfying assignments for \(\varphi\) and \(d \in A\) a \(\Delta\)-assignment. Then there is \(X \in U\) such that \(X \subseteq d\), since \(\varphi\) is a DNF. Let \(X\) be the maximal set from \(U\) with this property. Then we have \(\varphi[d] = \varphi(X)\), so \(\varphi[d]\) supports \(\pi\). This yields \(A(d) = A(d)|_{V_1} \bowtie \ldots \bowtie A(d)|_{V_{|\pi|}}\), and since \(d\) was arbitrarily chosen, it follows from Lemma 2 that \(\varphi\) is \(\Delta\)-decomposable with \(\pi\).

**Lemma 5 (Decomposition Lemma).** Let \(\varphi_1, \ldots, \varphi_n\), where \(n \geq 1\), be positive DNFs such that \(\varphi_i \lor \varphi_j\), for \(1 \leq i, j \leq n\), is decomposable with a partition \(\pi\). Then so is \(\varphi_1 \lor \ldots \lor \varphi_n\).

**Proof.** Observe that the condition of the lemma implies that for all \(i = 1, \ldots, n\), \(\varphi_i\) is decomposable with partition \(\pi\). We prove the lemma by induction on \(n\) for the case \(|\pi| = 2\); the general case is proved similarly by simultaneous induction on \(n\) and \(|\pi|\).

The case \(n = 1, 2\) is trivial. For the induction step \(n \geq 3\), it suffices to show that \(\varphi_i \lor \varphi_{n-1} \lor \varphi_n\) is decomposable with a partition \(\pi\), for \(1 \leq i < n-1\). This yields that the condition of the lemma holds for \(\varphi_1 \lor \ldots \lor \varphi_{n-2} \lor \psi\), where \(\psi = \varphi_{n-1} \lor \varphi_n\), and the induction assumption applies.

That said, it suffices only to consider the case \(n = 3\). Let \(\pi = \{X, Y\}\) and \(\varphi = \varphi_1 \lor \varphi_2 \lor \varphi_3\). By Lemma 4 we have \(\varphi_i \equiv \varphi_i|_X \lor \varphi_i|_Y\), for \(1 \leq i \leq 3\). We need to show that \(\varphi \equiv \varphi|_X \lor \varphi|_Y\). For \(S \in \{X, Y\}\), we have \(\varphi|_S = \varphi_1|_S \lor \varphi_2|_S \lor \varphi_3|_S\).
and hence, $\varphi|_X \land \varphi|_Y$ is equivalent to

$$\varphi_1 \lor \varphi_2 \lor \varphi_3 \lor \left( (\varphi_1|_X \land \varphi_2|_Y) \lor (\varphi_2|_X \land \varphi_1|_Y) \lor (\varphi_3|_X \land \varphi_1|_Y) \lor (\varphi_2|_X \land \varphi_3|_Y) \lor (\varphi_3|_X \land \varphi_2|_Y) \right)$$

By the condition of the lemma it holds $\varphi_i \lor \varphi_j \equiv (\varphi_i \lor \varphi_j)|_X \land (\varphi_i \lor \varphi_j)|_Y$, for all $1 \leq i,j \leq 3$ and hence,

$$\varphi_i \lor \varphi_j \equiv \varphi_i \lor \varphi_j \lor (\varphi_i|_X \land \varphi_j|_Y) \lor (\varphi_j|_X \land \varphi_i|_Y)$$

It follows that (3) is equivalent to $\varphi_1 \lor \varphi_2 \lor \varphi_3$, so the lemma is proved. $\square$

**Theorem 3** (Computing the Finest Variable Partition wrt $\Delta$). Given a positive DNF $\varphi$ and a subset $\Delta \subseteq \text{vars}(\varphi)$, the finest variable partition of $\varphi$ wrt $\Delta$ can be computed in time polynomial in the size of $\varphi$.

**Proof.** Let $A$ be the set of $\Delta$-atoms of $\varphi$ and $U$ consist of all unions of sets from $A$. Note that $|A|$ is bounded by the size of $\varphi$, while $|U|$ is exponential. By Lemma 4, $\varphi$ is $\Delta$-decomposable with a partition $\pi$ iff $\varphi(X)$ supports $\pi$, for all $X \in U$.

For any $X \in U$, we have $\text{vars}(\varphi(X)) \subseteq \text{vars}(\varphi)$. Observe that $\varphi(X)$ is equivalent to the DNF $\psi = \varphi(X) \lor t$, where $t$ is a term redundant in $\psi$, $\text{vars}(t) = \text{vars}(\varphi(X)) \setminus \text{vars}(\varphi)$ (in case $\text{vars}(t) = \emptyset$ we assume that $\psi = \varphi(X)$) and it holds $\text{vars}(\psi) = \text{vars}(\varphi)$. Therefore, $\varphi(X)$ supports $\pi$ iff $\psi$ is decomposable with $\pi$.

For any $X \in U$, $\varphi(X)$ has the form $\varphi_1 \lor \ldots \lor \varphi_n$ where $n \geq 1$ and for $i = 1, \ldots, n$, $\varphi_i = \varphi(a_i)$, where $a_i \subseteq X$, a $\Delta$-atom of $\varphi$. The formula $\varphi(X)$ is equivalent to $\varphi'_1 \lor \ldots \lor \varphi'_n$, where $\varphi'_i = \varphi_i \lor t_i$ and $t_i$ is a redundant term as introduced above. By Lemma 4 if $\varphi$ is $\Delta$-decomposable with a partition $\pi$ then $\varphi(a_1 \cup a_2)$ supports $\pi$, for any $a_1, a_2 \in A$. For the other direction, if $\varphi(a_1 \cup a_2)$ supports $\pi$, for any $a_1, a_2 \in A$ then the condition of Lemma 5 holds for $\varphi'_1 \lor \ldots \lor \varphi'_n$. It follows that $\varphi(X)$ supports $\pi$, for any $X \in U$ and hence by Lemma 4 $\varphi$ is $\Delta$-decomposable with $\pi$.

By Theorem 2, a variable partition $\sigma$, which corresponds to the finest decomposition of $\varphi(a_1 \cup a_2)$, can be found in time polynomial in the size of $\varphi(a_1 \cup a_2)$ (and hence, in the size of $\varphi$, as well). For any variables $x, y \in \text{vars}(\varphi)$ and a set $S \in \sigma$, if $x, y \in S$ then $x$ and $y$ cannot belong to different decomposition components of $\varphi(a_1 \cup a_2)$. Let $\sim$ be an equivalence relation on $\text{vars}(\varphi)$ such that $x \sim y$ iff there are $a_1, a_2 \in A$ such that $x$ and $y$ belong to the same component of the finest variable partition of $\varphi(a_1 \cup a_2)$. Since $|A|$ is bounded by the size of $\varphi$, one can readily verify that the equivalence classes $\sim$ can be computed in time polynomial in the size of $\varphi$ and are equal to its finest variable partition. $\square$

We conclude the paper with a description of $\Delta\text{Decompose}$ procedure, which for a positive DNF $\varphi$ and a (possibly empty) subset $\Delta \subseteq \text{vars}(\varphi)$ computes the finest variable partition of $\varphi$ wrt $\Delta$ and outputs $\Delta$-decomposition components, which correspond to the partition.
In Lines 11-10 of the procedure, a set of \( \Delta \)-atoms of \( \varphi \) is computed, while skipping those ones, which subsume some term of \( \varphi \). Clearly, if there is a term \( t \) of \( \varphi \), which consists only of \( d \)-variables for some subset \( d \subseteq \Delta \), then it holds \( \varphi[d] = 1 \), which implies that \( \varphi(d) \) supports any partition \( \pi \) of \( \text{vars}(\varphi) \setminus \Delta \) (at this point \( \varphi \) necessarily contains at least 2 non-\( \Delta \)-variables due to the test in line 9). Therefore, these atoms are irrelevant in computing decomposition and they can be omitted (similarly, the unions of \( \Delta \)-atoms in line 13).

Lines 11-17 implement a call for computing the finest variable partition wrt the empty \( \Delta \) for each DNF \( \varphi(L) \) obtained from \( \varphi \) for a union \( L \) of relevant \( \Delta \)-atoms. The result is a family of partitions, which are further aligned by computing equivalence classes on the variables of \( \varphi \). This is implemented in AlignPartitions procedure by computing connected components of a graph, in which vertices correspond to the variables of \( \varphi \).

Finally, in lines 22-25 the decomposition components of \( \varphi \) are computed as projections onto the sets of variables corresponding to the finest partition. The components are cleaned up by removing redundant terms and are sent to the output.

```plaintext
1: procedure \( \Delta \)Decompose(\( \varphi \), \( \Delta \))
2:     FinestPartition ← \( \varnothing \)
3:     Components ← \( \varnothing \)
4:     if \( \varphi \) contains at most one non-\( \Delta \)-variable then
5:         return \{\( \varphi \)\} \( \triangleright \) \( \varphi \) is not \( \Delta \)-decomposable
6:     end if
7:     \( \Delta \)Atoms ← \( \varnothing \)
8:     for every term \( t \) of \( \varphi \), which contains at least one non-\( \Delta \)-variable do
9:         \( \Delta \)Atoms.add(\( \Delta \)-variables of \( t \))
10:     end for
11:     for all \( a_1, a_2 \) from \( \Delta \)Atoms do
12:         \( L \leftarrow a_1 \cup a_2 \)
13:         if there is no term \( t \) in \( \varphi \), whose every variable is from \( L \) then
14:             PartitionForL ← \( \varnothing \)Decompose(\( \varphi(L) \)) \( \triangleright \) see Sect. [I]
15:             PartitionFamily.add(PartitionForL)
16:         end if
17:     end for
18:     FinestPartition ← AlignPartitions(PartitionFamily)
19:     if FinestPartition.isSingleton then
20:         return \{\( \varphi \)\} \( \triangleright \) \( \varphi \) is not \( \Delta \)-decomposable
21:     else
22:         for \( V \in \text{FinestPartition} \) do
23:             \( \psi \leftarrow \text{RemoveRedundTerms}(\varphi|_{\text{vars}(\varphi) \cup \Delta}) \)
24:             Components.add(\( \psi \))
25:         end for
26:     return Components
27: end if
28: end procedure
```

```
1: procedure AlignPartitions(PFamily)
2:     \( G \leftarrow \varnothing \) \( \triangleright \) a graph with vertices being vars. of \( \varphi \)
3:     for Partition ∈ PFamily do
4:         for VarSet ∈ Partition do
5:             G.add(\( \text{a path involving all } x \in \text{VarSet} \))
6:         end for
7:     end for
8:     ResultPartition ← \( \varnothing \)
9:     for C a connected component of \( G \) do
10:         ResultPartition.add(\( \text{the set of vars from } C \))
11:     end for
12:     return ResultPartition
13: end procedure
```
References

1. Bioch, J.C.: Decomposition of Boolean functions. In Crama, Y., Hammer, P.L., eds.: Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Volume 134 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, New York, NY, USA (2010) 39–78
2. Villa, T., K. Brayton, R., Sangiovanni-Vincentelli, A.: Synthesis of multilevel boolean networks. In Crama, Y., Hammer, P.L., eds.: Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Volume 134 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, New York, NY, USA (2010) 675–722
3. Boros, E.: Horn functions. In Crama, Y., Hammer, P.L., eds.: Boolean Functions: Theory, Algorithms, and Applications. Volume 134 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, New York, NY, USA (2011) 269–325
4. Gursky, S.: Special classes of boolean functions with respect to the complexity of their minimization. PhD Thesis, Charles University in Prague (2014)
5. Emelyanov, P., Ponomaryov, D.: Algorithmic issues of conjunctive decomposition of boolean formulas. Programming and Computer Software 41(3) (2015) 162–169
6. Ponomaryov, D.: On decomposability in logical calculi. Bulletin of the Novosibirsk Computing Center 28 (2008) 111–120
7. Emelyanov, P., Ponomaryov, D.: On tractability of disjoint AND–decomposition of boolean formulas. In: Proceedings of the PSI 2014: 9th Ershov Informatics Conference. Volume 8974 of Lecture Notes in Computer Science., Springer (2015) 92–101
8. Shpilka, A., Volkovich, I.: On the relation between polynomial identity testing and finding variable disjoint factors. In: Proceedings of the 37th International Colloquium on Automata, Languages and Programming. Part 1 (ICALP 2010). Volume 6198 of Lecture Notes in Computer Science., Springer (2010) 408–419
9. Crama, Y., Hammer, P.L.: Boolean Functions - Theory, Algorithms, and Applications. Volume 142 of Encyclopedia of mathematics and its applications. Cambridge University Press (2011)
Appendix

Illustration of FindPartition procedure from Section 3.

For clarity of presentation, we show how the procedure is used to compute the finest variable partition for a factorable polynomial

$$F = xu + xv + yu + yv = (x + y) \cdot (u + v)$$

For the choice of variables $x$ and $y$ (see lines 19 – 22 of FindPartition), we have $F_x = 0 = yu + yv$ and $\frac{\partial F}{\partial x} = u + v$, and thus, $G = (yu + yv) \cdot (u + v) = yu + yuv + yuv + yv$ is computed. Further,

$$\frac{\partial G}{\partial y} = \frac{\partial (yu + yv)}{\partial y} \cdot (u + v) + (yu + yv) \cdot \frac{\partial (u + v)}{\partial y} = (u + v) \cdot (u + v) + 0 = u + v$$

which is not equal to 0, so the variable $y$ is put into $\Sigma$, and thus, $\Sigma = \{x, y\}$.

For a choice of any variable different from $y$ (and $x$) in line 22, say $v$, we have:

$$\frac{\partial G}{\partial v} = yu + yv + yu + yv = 0,$$

thus, $v$ is not put into $\Sigma$, and therefore, $\{u, v\}$ is a component of the finest variable partition of $F$; it is put into FinestPartition at line 27. At the second iteration step (line 18), it holds $V = \{u, v\}$, hence, $F|_V = u + v$, and the above mentioned sequence of steps is repeated for this polynomial, thus, giving $\{u, v\}$ as the second component of the finest variable partition of $F$.

Missing proof to Theorem 2 from Section 4.

Theorem 2 (Computing the Finest Variable Partition wrt $\Delta = \emptyset$) The finest variable partition of a positive DNF $\varphi$ can be computed in time polynomial in the size of $\varphi$.

Proof. Let $\psi$ be the positive DNF obtained from $\varphi$ by eliminating all redundant terms. Obviously, this operation can be done in time polynomial in the size of $\varphi$. Denote $\sigma = \{\{x\} \mid x \in \text{vars}(\varphi) \setminus \text{vars}(\psi)\}$. We have $\psi \equiv \varphi$ and it is easy to see that a set $\pi$ is the finest variable partition of $\psi$ iff so is $\pi \cup \sigma$ for $\varphi$. Consider the boolean polynomial $P(\psi)$ obtained from $\psi$. We show that $\psi$ is decomposable with a variable partition $\pi$ iff $P(\psi)$ is factorable with $\pi$. Then by Theorem 1 our claim is proved.

($\Leftarrow$): Let $F_1, \ldots, F_{|\pi|}$ be factors of $P(\psi)$ over the variable sets from $\pi$. Then it follows from Remark 1 that neither of these factors contains the monomial 1, because otherwise $\psi$ would contain a redundant term, and for $i = 1, \ldots, |\pi|$, $P^{-1}(F_i)$ is a decomposition component of $\psi$.

($\Rightarrow$): For $i = 1, \ldots, |\pi|$, let $\psi_i$ be DNFs, which are decomposition components of $\psi$ and do not contain redundant terms. Let $\psi'$ denote the expression $\psi_1 \land \ldots \land \psi_{|\pi|}$ given as DNF. By Lemma 3 we may assume that each $\psi_i$, $i = 1, \ldots, |\pi|$ is positive and hence, so is $\psi'$. It is also easy to verify by induction on $|\pi|$ that $\psi'$
does not contain redundant terms. We show that $\psi = \psi'$, i.e. $\psi$ and $\psi'$ consist of the same terms. Then by Remark 1, $P(\psi) = P(\psi_1) \cdot \ldots \cdot P(\psi_{|\pi|})$.

It is known (see, e.g., Theorem 1.22 in [9]) that a DNF $\alpha$ implies a positive DNF $\beta$ iff for every term $t$ of $\alpha$ there is a term $t'$ of $\beta$ such that $t' \subseteq t$. Assume there is a term $t$ of $\psi$, which is not a term of $\psi'$. Since $\psi$ implies $\psi'$, there is a term $t'$ of $\psi'$, with $t' \neq t$, such that $t' \subseteq t$. Similarly, since $\psi'$ implies $\psi$, there is a term $s$ in $\psi$ such that $s \subseteq t'$. Hence, $s$ and $t$ are terms of $\psi$ such that $s \subseteq t$. If $s \neq t$ then $s$ is redundant in $\psi$, which contradicts the assumption that $\psi$ does not contain redundant terms. Thus, $s = t$ and hence, $t = t'$, which is again a contradiction, since we have assumed that $t$ is not a term of $\psi'$. As $\psi'$ does not contain redundant terms, it can be shown similarly that every term of $\psi'$ is present in $\psi$, and thus, we have $\psi = \psi'$. □