Make sure citations are typeset properly. Consult Petra.

Typeset long function names, for example *dnf*, with mathit. Source: `http://tex.stackexchange.com/a/6088`

¿ Change reference names to adhere to reference guidelines at `http://en.wikibooks.org/wiki/LaTeX/Labels_and_Cross-referencing`.

Kučera: Když je to odkaz na číslovanou definici, tak velké D, to se týká i dalších odkazovaných bloků, section, lemma, theorem atd.

Change all references to *cref* or *Cref* from *autoref* and *ref*.

Make a section about disjoint representations, collect (review) the results from the thesis. Can be done in Conclusion. When we discuss Scheiber's results, mention disjoint dnfs and the results - Sheiber has two algs. and results. Motivation: generating test data for software testing - esp. with disjoint dnfs - uniform polling

Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS

Filip Bártek

Minimum representations of Boolean functions defined by multiple intervals

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis:	RNDr. Petr Kučera, Ph.D.
Study programme:	Informatics
Study branch:	Theoretical Computer Science

Prague 2015

Title: Minimum representations of Boolean functions defined by multiple intervals

Author: Filip Bártek

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. Petr Kučera, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract:

When we interpret the input vector of a Boolean function as a binary number, we define interval Boolean function $f^n_{[a,b]}$ so that $f^n_{[a,b]}(x) = 1$ if and only if $a \leq x \leq b$. Disjunctive normal form is a common way of representing Boolean functions. Minimizing DNF representation of an interval Boolean function can be performed in linear time. The natural generalization to k-interval functions seems to be significantly harder to tackle. In this thesis, I discuss the difficulties with finding an optimal solution and introduce a $2k$-approximation algorithm.

¿ Remove mathematical expressions from abstract.

Keywords: Boolean minimization, disjunctive normal form, interval functions

Contents

Introduction

An n-ary Boolean function takes an n-tuple of Boolean values as input and outputs a single Boolean value. There are two Boolean values – we will represent them with the symbols 0 and 1 and we will call $\{0, 1\}$ the Boolean domain.

A Boolean function can be represented in various ways. A common way to represent Boolean functions is the truth table, which explicitly lists the output value for each possible input n-tuple. For example, the simple Boolean function $f^2_\wedge$ that realizes the binary logical operator of conjunction can be represented by the following table:

x_1	x_2	$f^2_\wedge(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

The upper index of $f^2_\wedge$ signifies the arity of the function. In this case it is 2, meaning it is a function of two Boolean variables, or equivalently of pairs of Boolean values, or binary vectors of length 2.

Other representations of Boolean functions include propositional formulas, circuits, decision diagrams and sets of intervals.

Cite proper literature, for example Wegener: The complexity of Boolean functions

In this thesis, we will deal with the interval representation of Boolean functions and relate it to propositional formula representation of a special form called disjunctive normal form.

Boolean functions defined by intervals were introduced by Schieber et al. [5]. A pair of n-bit numbers a, b defines a 1-interval function $f^n_{[a,b]}$:

$$f^n_{[a,b]}(x) = 1 \iff a \leq x \leq b$$

Note that such function is the characteristic function of the interval $[a, b]$.

Schieber et al. showed an efficient method to find a minimum disjunctive normal form (DNF) representation of any 1-interval Boolean function given by a pair of endpoints.[5]

Since the problem of Boolean minimization is in general Σ^p_2-complete [6], there is an interesting question of how difficult it is to minimize DNF representation of a function defined by a set of intervals. Dubovský investigated the problem in the case of 2-interval functions.[2]

In this thesis we review the existing results regarding minimization of DNF representations of single- and multi-interval functions, and then present four new results:

- a simplified algorithm for minimizing DNF representations of 2-switch 2-interval functions (Section 3.1),
- show that the technique used for proving the optimality of the algorithms that minimize DNF representation of 1- and 2-switch (that is all 1-interval and 2-switch 2-interval) functions can not be used in general for functions with 3 or more switches (Section 3.2),

- introduce an algorithm that, given the intervals of a k-interval function, finds the function's DNF representation that is at worst $2k$ times larger than an optimal DNF representation (Chapter 4) and
- propose an improved approximation algorithm for k-interval functions and show that it does not perform significantly better than $2k$-approximation for large k (Chapter 5).

1. Definitions

Add Hammer.

Kučera

Dále používáte obecnou booleovskou terminologii, ke které by se hodila citace třeba té booleovský knížky od Y. Cramy a P.L. Hammera. Crama, Hammer: Boolean Functions - Theory, Algorithms, and Applications

The basic terminology used in this thesis was introduced by Schieber et al. [5], Dubovský [2] and Hušek [3, 4].

1.1 Vector operations

Throughout the thesis we will use vectors over small finite domains extensively. We shall write a vector as a sequence of symbols, for example 00101 is a vector of length 5. The concatenation of two vectors u and v will be denoted simply as uv, for example $00v11$ denotes a vector which is formed as a concatenation of the three vectors 00, v and 11.

Definition 1.1.1 (Component extraction $v^{[i]}$ [5])**.** Let v be a vector of length n and $1 \leq i \leq n$. Then $v^{[i]}$ is the i-th component of v.

Definition 1.1.2 (Subsequence extraction $v^{[a,b]}$ [5])**.** Let v be a vector of length n and $1 \leq a \leq b \leq n$. Then $v^{[a,b]}$ is the subvector of v that starts at a-th position and ends at b-th position ($v^{[a,b]} = v^{[a]}v^{[a+1]} \dots v^{[b-1]}v^{[b]}$).

Notably, $v = v^{[1]} \dots v^{[n]} = v^{[1,n]}$.

Definition 1.1.3 (Symbol repetition $\alpha^{\{n\}}$ [5])**.** Let α be a symbol (for example 0 or 1) and $n \in \{0, 1, \dots\}$. Then $\alpha^{\{n\}}$ is the vector of length n each component of which is equal to α.

For example $0^{\{2\}} = 00$.

1.2 Binary vectors

Definition 1.2.1 (Binary vector [5])**.** We will use the term *binary vector* to denote a vector over the Boolean domain. That is, x is a binary vector if and only if $x \in \{0, 1\}^n$ for some $n \in \{0, 1, \dots\}$. We call such x an *n-bit binary vector*.

Note that an n-bit binary vector x represents the natural number $\sum_{i=1}^{n} x^{[i]} 2^{n-i} = \sum_{i|x^{[i]}=1} 2^{n-i}$. An n-bit vector corresponds to a number between 0 and $2^n - 1$. We will not differentiate between a number and its binary vector representation. Specifically, we will use the standard linear order of natural numbers to compare binary vectors. Note that the linear order on natural numbers corresponds to lexicographic order on binary vectors:

Observation 1.2.1. *If a and b are n-bit binary vectors, then $a < b$ if and only if there are binary vectors c, a' and b' such that $a = c0a'$ and $b = c1b'$. We call such c the* longest common prefix *of a and b.*

1.3 Boolean functions

Cite a source. **Definition 1.3.1** (Boolean function)**.** $f : \{0,1\}^n \to \{0,1\}$ is an *n-ary Boolean function.*

Since we will not be dealing with any non-Boolean functions, in the remainder of the text, we will use the terms "function" and "Boolean function" interchangeably.

Cite a source. **Definition 1.3.2** (True point)**.** Let f be an n-ary Boolean function. An n-bit binary vector x is a *true point* of f if and only if $f(x) = 1$.

Equivalently, we define a *false point* of f as any point x such that $f(x) = 0$.

We'll denote the set of all true points of f as $TP(f)$ and the set of all false points as $FP(f)$.

1.4 DNF representations of Boolean functions

Every n-ary Boolean function f can be expressed as a propositional formula $\mathcal{F}$ on n variables $x_1, \ldots, x_n$. There's a natural bijection between binary vectors of length n and valuations of n variables (1-bits in the binary vector correspond exactly to 1-valued variables in the valuation).

Cite a source. **Definition 1.4.1** (Disjunctive normal form)**.** A *literal* is an occurrence of a variable (x_i) or its negation ($\overline{x_i}$) in a propositional formula. A *term* is an elementary conjunction of literals, that is a conjunction in which every variable appears at most once. A *disjunctive normal form (DNF)* propositional formula is a disjunction of terms. We will often view terms as sets of literals and DNF formulas as sets of terms.

A DNF formula $\mathcal{F}$ is a *DNF representation* of a Boolean function f if and only if $(\forall v \in \{0,1\}^n)[f(v) = 1 \iff \mathcal{F}(v) \equiv 1]$, where $\mathcal{F}(v)$ is the formula given by substituting each occurrence of a variable x_i in $\mathcal{F}$ with the value $v^{[i]}$ for every $i \in \{1, \ldots, n\}$.

Cite a source. **Definition 1.4.2** (Minimum DNF representation)**.** A DNF representation of function f is *minimum* if and only if there is no DNF representation of f with fewer terms.

We shall denote the size of a minimum DNF representation of function f by $dnf(f)$.

Note that a function may have more than one minimum DNF representation.

The focus of this thesis is minimizing the number of terms of DNF representations of certain Boolean functions.

1.5 Ternary vectors

Definition 1.5.1 (Ternary vector [5])**.** A *ternary vector* is a vector over the set $\{0, 1, \phi\}$.

There's a natural bijection between ternary vectors and DNF terms (elementary conjunctions of literals). Each of the n components of a ternary vector T corresponds to an occurrence (or its absence) of one of the n variables of a term C:

$T^{[i]}$	$C \cap \{x_i, \overline{x_i}\}$
ϕ	$\emptyset$
1	$\{x_i\}$
0	$\{\overline{x_i}\}$

Note that since C is elementary, it can not contain both x_i and $\overline{x_i}$ for any i.

Namely, a *binary* vector (a ternary vector that does not contain any ϕ) corresponds to a full term (that is one that contains every variable), and the ternary vector $\phi^{\{n\}}$ corresponds to the empty term (tautology).

It is easy to see that a set of ternary vectors, each of length n, corresponds to a DNF *formula* on n variables.

Definition 1.5.2 (Spanning [5]). A ternary vector T of length n *spans* a binary vector x of length n if and only if $(\forall i \in \{1, \ldots, n\})[T^{[i]} = \phi$ or $T^{[i]} = x^{[i]}]$.

A set of ternary vectors $\mathcal{T}$ *spans* a binary vector x if and only if some vector in $\mathcal{T}$ spans x.

The i-th symbol of a ternary vector constrains the i-th symbol of the spanned binary vector (or, equivalently, the valuation of the i-th variable). If $T^{[i]}$ is a "fixed bit" (that is $T^{[i]} \in \{0, 1\}$), the spanned binary vector x must have the same value in its i-th position, that is $x^{[i]} = T^{[i]}$. If $T^{[i]}$ is the "don't care symbol" ϕ, the spanned binary vector may have any value in its i-th position.

In the correspondence between ternary vectors and terms, spanning corresponds to satisfying. The ternary vector T of length n spans x if and only if $C(x) \equiv 1$, where C is the term on n variables that corresponds to T. Similarly, it is easy to see that a set of ternary vectors $\mathcal{T}$ spans x if and only if $\mathcal{F}(x) \equiv 1$, where $\mathcal{F}$ is the DNF formula on n variables that corresponds to $\mathcal{T}$.

Definition 1.5.3 (Spanned set). Let T be a ternary vector of length n. We denote the set of points spanned by T as *span*(T):

$$\mathit{span}(T) = \{x \in \{0,1\}^n | T \text{ spans } x\}$$

Note that a ternary vector with m ϕ-positions spans 2^m binary vectors.

We'll also use a natural extension of spanning to sets of ternary vectors – if $\mathcal{T}$ is a set of ternary vectors, then

$$\mathit{span}(\mathcal{T}) = \bigcup_{T \in \mathcal{T}} \mathit{span}(T)$$

Definition 1.5.4 (Exact spanning). A ternary vector T of length n *spans exactly* a set of n-bit binary vectors S if and only if *span*$(T) = S$.

T *spans exactly* an n-ary Boolean function f if and only if *span*$(T) = TP(f)$.

Again, we will generalize *exact spanning* to sets of ternary vectors – the set of ternary vectors $\mathcal{T}$ *spans exactly* S if and only if $span(\mathcal{T}) = S$.

Definition 1.5.5 (Spanning set [5])**.** Let f be an n-ary Boolean function. The set $\mathcal{T}$ of n-bit ternary vectors is a *spanning set* of f if and only if it $\mathcal{T}$ spans exactly f, that is $span(\mathcal{T}) = TP(f)$.

In other words, a spanning set of a function spans all of its true points and none of its false points.

Note that spanning sets of a function correspond to DNF representations of the function. The correspondence preserves size (number of ternary vectors and terms, respectively), so a minimum spanning set corresponds to a minimum DNF representation.

Definition 1.5.6 (Disjoint spanning set [5, Section 4])**.** A spanning set $\mathcal{T}$ is *disjoint* if and only if the spanned sets of its vectors do not overlap, that is

$$(\forall T_i, T_j \in \mathcal{T})[T_i = T_j \text{ or } span(T_i) \cap span(T_j) = \emptyset]$$

In terms of DNF, disjoint spanning sets correspond to DNF formulas such that every valuation satisfies at most one term.

We have introduced two equivalent representations of Boolean functions – one based on ternary vectors and the other being the DNF representation. Table 1.1 shows the corresponding terms side by side.

ternary vector	term
ternary vector set	DNF formula
binary vector	variable valuation
ternary vector *spans* a binary vector	valuation *satisfies* a term
ternary vector set *spans* a binary vector	valuation *satisfies* a DNF formula

Table 1.1: Corresponding terms used in Boolean function representations

Definition 1.5.7 (Complement [5])**.** The complement of a binary symbol α ($\alpha \in \{0, 1\}$), denoted $\overline{\alpha}$, is $1 - \alpha$.

We get the complement of a binary vector x by flipping all of its bits, that is $\overline{x} = (\overline{x_1}, \ldots, \overline{x_n}) = 1^{\{n\}} - x$.

The complement of the ϕ symbol is ϕ ($\overline{\phi} = \phi$).

It follows that we obtain the complement of a ternary vector by flipping all of its fixed bits.

We generalize the notion of complement to sets of ternary vectors – if $\mathcal{T}$ is a set of ternary vectors, then $\overline{\mathcal{T}} = \{\overline{T} | T \in \mathcal{T}\}$.

1.6 Orthogonal sets and coverable functions

Orthogonal set is a useful tool for showing a lower bound on the size of all spanning sets of a function. Orthogonal sets have been used both by Schieber et al. [5] (implicitly) and Dubovský [2] (explicitly) in the proofs of optimality of spanning algorithms.

Definition 1.6.1 (Orthogonality [2, p. 6])**.** Let x and y be true points of a Boolean function f (that is $f(x) = f(y) = 1$). The vectors x and y are *orthogonal* with respect to f if and only if every ternary vector that spans both x and y spans some false point of f.

A set of true points S is *orthogonal* if and only if all the vectors in S are pairwise orthogonal.

We shall denote the size of a maximum orthogonal set of function f with $ortho(f)$.

¿ Get rid of collision, or prove that it is equivalent to orthogonality, preferably using established terminology.

¿ Note that x and y collide on z if and only if z separates x and y in terminology of Čepek et al. [7, Definition 3.3]. Consider using the same terminology.

Definition 1.6.2 (Collision)**.** Let x, y and z be n-bit binary vectors. We say that *x and y collide on the vector z* if and only if every ternary vector that spans both x and y necessarily also spans z.

Theorem 1.6.1 (Collision and orthogonality)**.** *Let x and y be true points of function f. If the vectors x and y collide on some false point of f, they are orthogonal.*

Proof. The statement is a trivial application of Definitions 1.6.1 and 1.6.2. □

Note that the inverse implication holds as well [7, Lemma 3.7] but we will not use it in this thesis.

Theorem 1.6.2 ($ortho(f) \leq dnf(f)$ [2, Observation 1.1] [7, Theorem 2.8, Corollary 3.2])**.** *For any Boolean function, the size of its maximum orthogonal set is at most as great as the size of its minimum DNF representation.*

Proof. We will prove the statement by contradiction. Let f be a Boolean function such that $ortho(f) > dnf(f)$. Let V be an orthogonal set of size $ortho(f)$ and $\mathcal{T}$ be any spanning set of size $dnf(f)$ (such spanning set exists because of the correspondence between DNF representations and spanning sets). Since $|\mathcal{T}| < |V|$ and $\mathcal{T}$ spans $TP(f) \supseteq S$, there must be a ternary vector $T \in \mathcal{T}$ that spans two distinct vectors $x, y \in V$. However, since V is orthogonal, x is orthogonal to y, so T necessarily spans a false point, which contradicts the premise that $\mathcal{T}$ is a feasible spanning set of f. □

¡ Is the parallel between orthogonal vectors and essential sets presented in the following paragraph correct?

Note that Čepek et al. show the inequality for disjoint essential sets of implicants of f in place of orthogonal vectors [7, Theorem 2.8]. An orthogonal set corresponds to a set of pairwise disjoint *truepoint* essential sets [7, Section 3], where truepoint essential set is a special case of essential set. Čepek et al. continue to show that the lower bound on $dnf(f)$ is maintained if we restrict ourselves to sets of truepoint essential sets [7, Corollary 3.2]. Also note that their results are presented in the dual terminology of conjunctive normal form (CNF),

¡ Does this Lemma 3.7 in Čepek et al. [7] really prove this? (I believe it does.)

Do we use the theorem in the thesis? If not, throw it away, possibly along with definition of collision. If yes, label and cref it.

¿ Add "the proof is more complicated". We would especially need some established terminology.

¿ Cite Boros et al. [1] instead of or

as opposed to the DNF terminology used throughout this thesis.

We conclude that the size of any orthogonal set is a lower bound on the size of minimum DNF representation. Specifically, if we show a DNF representation of a function and an orthogonal set of the same size, we conclude that the representation is minimum. The orthogonal set certifies the optimality of the DNF representation.

Definition 1.6.3 (Coverability [7, Definition 2.9])**.** Let f be a Boolean function. The function f is *coverable* if and only if $ortho(f) = dnf(f)$.

Informally speaking, a coverable function is one for which it may be easy to prove optimality of a spanning set. We will see in Chapter 2 and Section 3.1 that all the 1-interval and 2-switch 2-interval functions are coverable. We will show examples of functions which are not coverable in Section 3.2.

¡ Maybe remove or rephrase the preceding sentence since k-interval and l-switch functions are defined later.

1.7 k-interval Boolean functions

The notation for 1- and 2-interval functions was introduced by Schieber et al. and Dubovský respectively. We shall generalize it to k-interval functions for any $k \geq 1$.

Definition 1.7.1 (k-interval Boolean function)**.** Let $n \geq 0$ and $k \geq 1$. Let $a_1, b_1, \dots, a_k, b_k$ be n-bit binary vectors such that $0^{\{n\}} \leq a_1 \leq b_1 < a_2 \leq \cdots < a_i \leq b_i < a_{i+1} \leq \cdots < a_k \leq b_k \leq 1^{\{n\}}$. Then $f^n_{[a_1,b_1],\dots,[a_k,b_k]} : \{0,1\}^n \to \{0,1\}$ is a function defined as follows:

$$f^n_{[a_1,b_1],\dots,[a_k,b_k]}(x) = \begin{cases} 1 & \text{if } a_i \leq x \leq b_i \text{ for some } i \\ 0 & \text{otherwise} \end{cases}$$

We call $f^n_{[a_1,b_1],\dots,[a_k,b_k]}$ a *k-interval Boolean function* and the vectors $a_1, b_1, \dots, a_k, b_k$ its *endpoints*.

Note in particular that in case $n = 0$, the 1-interval 0-ary function $f^0_{[\epsilon,\epsilon]}$, where ϵ denotes the binary vector of length 0, is properly defined and equivalent to tautology.

Note that the inequalities ensure that the intervals $[a_i, b_i]$ are non-empty and don't intersect. The inequalities, however, allow the intervals to be adjoint. This means that a k-interval function may also be k'-interval for some $k' < k$. We will often find it useful to use a less ambiguous notion of the number of intervals of a function:

¿ Add "I have taken the liberty to translate.".

Definition 1.7.2 (Proper k-interval Boolean function)**.** Let $f^n_{[a_1,b_1],\dots,[a_k,b_k]}$ be a k-interval Boolean function. If the endpoints satisfy the inequalities $b_1 < a_2 - 1$, $\dots$, $b_i < a_{i+1} - 1$, $\dots$, $b_{k-1} < a_k - 1$ (in other words, adjacent intervals are separated by at least one false point), we call $f^n_{[a_1,b_1],\dots,[a_k,b_k]}$ a *proper k-interval Boolean function.*

Especially, $f^n_{[a,b]}$ is a function whose true points form the interval $[a, b]$.
Table 1.2 shows examples of interval functions.

Interval function	Propositional formula	Description
$f^n_{[1^{\{n\}},1^{\{n\}}]}$	$\bigwedge_i x_i$	Conjunction
$f^n_{[0^{\{n-1\}}1,1^{\{n\}}]}$	$\bigvee_i x_i$	Disjunction
$f^n_{[0^{\{n\}},1^{\{n\}}]}$	1 (an empty term)	Tautology
$f^n_{[0^{\{n-1\}}1,1^{\{n-1\}}0]}$	$\overline{x_1 \dots x_n} \wedge \overline{\overline{x_1} \dots \overline{x_n}} \equiv \bigvee_{i \neq j} x_i \overline{x_j}$	Non-equivalence
$f^n_{[0^{\{n\}},0^{\{n\}}],[1^{\{n\}},1^{\{n\}}]}$	$x_1 \dots x_n \vee \overline{x_1} \dots \overline{x_n}$	Equivalence

Table 1.2: Examples of interval functions

1.8 l-switch Boolean functions

Given a k-interval function, apparently the most important values are the endpoints of the intervals. The important property of an interval endpoint is that it is a place where the value of the function changes (considering proper k-interval functions). We can view these values as places of switch, where f switches its value from 0 to 1 or back.

The notion of switch was introduced by Hušek[1] and it turns out it is very useful when studying 2-interval functions.

Definition 1.8.1 (l-switch function [4, pp. 13-14])**.** A Boolean function f is l-*switch* if and only if there are exactly l input vectors x such that $f(x) \neq f(x+1)$.

We call such x a *switch* of f.

Example 1.1 (Switches and intervals). Depending on $f(0)$, an l-switch Boolean function f is proper $\lceil \frac{l}{2} \rceil$-interval in case $f(0) = 0$, or proper $\lceil \frac{l+1}{2} \rceil$-interval in case $f(0) = 1$.

Equivalently, a proper k-interval n-ary function is:

- $(2k)$-switch in case $f(0^{\{n\}}) = f(1^{\{n\}}) = 0$
- $(2k-1)$-switch in case $f(0^{\{n\}}) \neq f(1^{\{n\}})$
- $(2k-2)$-switch in case $f(0^{\{n\}}) = f(1^{\{n\}}) = 1$

¿ Show that complementing bits preserves number of switches.

[1] Hušek first used the Czech term “zlom” in his thesis [3, p. 13] and later translated the term as “switch” in personal communication [4].

2. 1-interval functions

In this chapter, we will show an efficient way to compute a minimum spanning set of any 1-interval Boolean function defined by the interval endpoints. The algorithm was originally shown by Schieber et al. [5].

In the chapters that follow, we will use this algorithm as a procedure in order to span multi-interval functions.

In the remainder of this chapter, a and b ($a \leq b$) will denote the endpoints of the interval in question and n the number of input bits. We will be looking for a minimum representation of the function $f^n_{[a,b]}$.

Input n-bit numbers a, b such that $a \leq b$ for $n \geq 0$.

Output A spanning set of $f^n_{[a,b]}$.

The algorithm differentiates various cases of input values. Solving the more difficult cases involves recursive calls, while the simpler cases are solved iteratively.

2.1 Trivial cases

We will first deal with the trivial intervals.

2.1.1 $n = 0$

Since the algorithm is recursive, we need to deal with the degenerate case of 0-bit endpoints. We span such interval using a single ternary vector of length 0. Note that such vector corresponds to the empty term, that is tautology.

From now on, let $n \geq 1$.

2.1.2 $a^{[1]} = b^{[1]}$

If a and b share the leading bit, let c be the longest common prefix of a and b and let j be its length. Note that $j \geq 1$. Let $a' = a^{[j+1,n]}$ and $b' = b^{[j+1,n]}$ ($a = ca'$ and $b = cb'$). Let $\mathcal{T}'$ be the spanning set of the function $f^{n-j}_{[a',b']}$. Since $n - j < n$, we can get such set by recursion. We get the spanning set of $f^n_{[a,b]}$ by prepending c to each of the vectors in $\mathcal{T}'$ ($\mathcal{T} = c\mathcal{T}' = \{cT' | T' \in \mathcal{T}'\}$). It is easy to see that if $\mathcal{T}'$ spans exactly $[a', b']$, then $c\mathcal{T}'$ spans exactly $[ca', cb'] = [a, b]$.

Note that in the special case $a = b$ (that is $c = a = b$ and $j = n$), $\mathcal{T}'$ consists of the single vector of length 0 and $\mathcal{T}$ correctly consists of the single vector $c = a = b$.

From now on let $a^{[1]} \neq b^{[1]}$. Since $a \leq b$, necessarily $a^{[1]} = 0$ and $b^{[1]} = 1$. From now on we may especially assume that $a < b$.

2.1.3 $a = 0^{\{n\}}$ and $b = 1^{\{n\}}$

We span the full interval $[0^{\{n\}}, 1^{\{n\}}]$ with the single ternary vector $\phi^{\{n\}}$, which corresponds to an empty term (tautology).

From now on, let $a > 0^{\{n\}}$ or $b < 1^{\{n\}}$.

2.2 Prefix and suffix case

Mention that the algorithm is linear time.

Since we have dealt with the trivial cases, we are now left with the situation $0^{\{n\}} \leq a < b \leq 1^{\{n\}}$ and $a > 0^{\{n\}}$ or $b < 1^{\{n\}}$.

In this section we shall consider the prefix case, that is $[0^{\{n\}}, b]$ ($a = 0^{\{n\}}$), and the suffix case, that is $[a, 1^{\{n\}}]$ ($b = 1^{\{n\}}$).

2.2.1 Suffix case

Add an example with a picture.

The prefix and suffix cases are complementary. There is a size-preserving one-to-one correspondence between the spanning sets of suffix intervals and the spanning sets of prefix intervals. The correspondence complements each of the ternary vectors in the spanning set – the spanning set $\mathcal{T}$ of a suffix interval $[a, 1^{\{n\}}]$ corresponds to the spanning set $\overline{\mathcal{T}} = \{\overline{T} | T \in \mathcal{T}\}$ of the prefix interval $[0^{\{n\}}, \overline{a}]$.

We will prove the correctness of this transformation by a series of simple lemmas.

Lemma 2.2.1 (Complementing flips inequality)**.** *Let a and b be binary vectors of equal length. Then $a \leq b$ if and only if $\overline{b} \leq \overline{a}$.*

Proof. The equivalence follows from Observation 1.2.1.

$$\begin{aligned} a \leq b &\iff a = b \text{ or } (\exists c, a', b')[a = c0a' \text{ and } b = c1b'] \\ &\iff \overline{a} = \overline{b} \text{ or } (\exists \overline{c}, \overline{a'}, \overline{b'})[\overline{a} = \overline{c}1\overline{a'} \text{ and } \overline{b} = \overline{c}0\overline{b'}] \\ &\iff \overline{a} \geq \overline{b} \end{aligned}$$

□

Lemma 2.2.2 (Complementing preserves interval membership)**.** *Let x, a and b be binary vectors of equal length. Then $x \in [a, b]$ if and only if $\overline{x} \in [\overline{b}, \overline{a}]$.*

Proof. The equivalence follows from Lemma 2.2.1:

$$\begin{aligned} x \in [a, b] &\iff x \geq a \text{ and } x \leq b \\ &\iff \overline{x} \leq \overline{a} \text{ and } \overline{x} \geq \overline{b} \\ &\iff \overline{x} \in [\overline{b}, \overline{a}] \end{aligned}$$

□

Lemma 2.2.3 (Complementing preserves interval spanning)**.** *Let a and b be n-bit binary vectors and let $\mathcal{T}$ be a set of ternary vectors of length n. Then $\mathcal{T}$ spans $[a, b]$ if and only if $\overline{\mathcal{T}}$ spans $[\overline{b}, \overline{a}]$.*

Proof. The equivalence follows from Lemma 2.2.2:

$$\begin{aligned} \mathcal{T} \text{ spans } [a, b] &\iff (\forall x)[\mathcal{T} \text{ spans } x \iff x \in [a, b]] \\ &\iff (\forall \overline{x})[\overline{\mathcal{T}} \text{ spans } \overline{x} \iff \overline{x} \in [\overline{b}, \overline{a}]] \\ &\iff \overline{\mathcal{T}} \text{ spans } [\overline{b}, \overline{a}] \end{aligned}$$

□

As a special application of Lemma 2.2.3 we get:

$$\mathcal{T} \text{ spans } [a, 1^{\{n\}}] \iff \overline{\mathcal{T}} \text{ spans } [0^{\{n\}}, \overline{a}]$$

Since the correspondence preserves the size of the spanning set, it also preserves optimality. Thus we can use the correspondence to optimally span suffix intervals using a procedure that spans prefix intervals. We will show such procedure in the following section.

2.2.2 Prefix case

Let's proceed to span a prefix interval. Let $a = 0^{\{n\}}$ and $b < 1^{\{n\}}$.

Let c be the n-bit number $b + 1$. Since $b < 1^{\{n\}}$, we do not need more than n bits to encode c.

The algorithm produces one ternary vector for each bit that is set to 1 in c. If o is a position of a 1 in c ($c^{[o]} = 1$), then the corresponding ternary vector is $c^{[1,o-1]}0\phi^{\{n-o\}}$. Thus we get the following spanning set:

$$\mathcal{T} = \{c^{[1,o-1]}0\phi^{\{n-o\}} | c^{[o]} = 1\}$$

Theorem 2.2.1 (Feasibility). *$\mathcal{T}$ spans exactly the interval* $[0^{\{n\}}, b]$.

Proof. To see that every number spanned by $\mathcal{T}$ is in $[0^{\{n\}}, b]$, note that given an index o of a bit which is set to 1 in c, the biggest number spanned by $c^{[1,o-1]}0\phi^{\{n-o\}}$ is $c^{[1,o-1]}01^{\{n-o\}}$ which is still strictly smaller than c.

On the other hand, consider a number x smaller than c and let o be the most significant bit in which x and c differ. It follows that $x^{[1,o-1]} = c^{[1,o-1]}$ and $0 = x^{[o]} < c^{[o]} = 1$. Then $c^{[1,o-1]}0\phi^{\{n-o\}} \in \mathcal{T}$ spans x. □

Relabel to "theorem:prefixoptimal".

Theorem 2.2.2 (Optimality). *$\mathcal{T}$ is the minimum spanning set of* $[0^{\{n\}}, b]$.

Proof. We will construct an orthogonal set V of size $|\mathcal{T}|$. With the aid of section 1.6.2, this proves that $\mathcal{T}$ is a minimum spanning set.

Similarly to the spanning vectors, the orthogonal true points correspond to 1-bits of c:

$$V = \{c^{[1,o-1]}0c^{[o+1,n]} | c^{[o]} = 1\}$$

Clearly $|V| = |\mathcal{T}|$. Also note that all points in V are smaller than c, so they are true points.

We need to prove that V is orthogonal. Let $x, y \in V$, $x \neq y$. Let o_x and o_y be the positions of the symbol 1 in c that were used to construct x and y ($x = c^{[1,o_x-1]}0c^{[o_x+1,n]}$, $y = c^{[1,o_y-1]}0c^{[o_y+1,n]}$). Since $x \neq y$, necessarily $o_x \neq o_y$.

Let T be any ternary vector that spans both x and y. To see that T must also span the false point c, note that every component i of c matches the respective component in at least one of x, y (for every i, $c^{[i]} = x^{[i]}$ or $c^{[i]} = y^{[i]}$).

Table 2.1 shows a more detailed comparison of the corresponding subvectors of x, y and c in case $o_x < o_y$.

We conclude that V is orthogonal and since $|\mathcal{T}| = |V|$, $\mathcal{T}$ is a minimum spanning set by section 1.6.2. □

Corollary 2.2.1. *Every prefix or suffix function is coverable.*

	$[1, o_x - 1]$	$[o_x]$	$[o_x + 1, o_y - 1]$	$[o_y]$	$[o_y + 1, n]$
x	$\underline{c^{[1,o_x-1]}}$	0	$\underline{c^{[o_x+1,o_y-1]}}$	$\underline{c^{[o_y]}}$	$\underline{c^{[o_y+1,n]}}$
y	$\underline{c^{[1,o_x-1]}}$	$\underline{c^{[o_x]}}$	$\underline{c^{[o_x+1,o_y-1]}}$	0	$\underline{c^{[o_y+1,n]}}$
c	$c^{[1,o_x-1]}$	$c^{[o_x]}$	$c^{[o_x+1,o_y-1]}$	$c^{[o_y]}$	$c^{[o_y+1,n]}$

Table 2.1: Subvectors of x, y and c in case $o_x < o_y$. The emphasized subvectors of x and y match their counterparts in c.

Corollary 2.2.2. *For every prefix function $f^n_{[0^{\{n\}},b]}$, there is an orthogonal set V of size $dnf(f^n_{[0^{\{n\}},b]})$ such that every pair of vectors $x, y \in V$, $x \neq y$, collides on the false point $b + 1$.*

It is easy to see that the spanning set $\mathcal{T}$ produced by the algorithm is disjoint. Since it is minimum in general, it is a minimum disjoint spanning set too.

2.3 General case

Having solved the trivial and prefix and suffix cases, we are left with the situation $0^{\{n\}} < a < b < 1^{\{n\}}$.

Since we have handled the case $a^{[1]} = b^{[1]}$ in Section 2.1.2, we may assume that $a^{[1]} = 0$ and $b^{[1]} = 1$. This restriction leaves us with four possible combinations of pairs of leading bits of a and b:

1. $a^{[1,2]} = 01$, $b^{[1,2]} = 10$
2. $a^{[1,2]} = 00$, $b^{[1,2]} = 10$
3. $a^{[1,2]} = 01$, $b^{[1,2]} = 11$
4. $a^{[1,2]} = 00$, $b^{[1,2]} = 11$

Following Schieber et al. [5], we will deal with each of these cases separately. Schieber et al.'s proofs of feasibility and optimality of the following algorithm are rather technical and out of scope of this text. Note, however, that the proof of optimality is implicitly based on building an orthogonal set of the same size as the computed spanning set.

In the sections that follow, we will denote $a^{[3,n]}$ with $\hat{a}$ ($a = a^{[1,2]}\hat{a}$) and $b^{[3,n]}$ with $\hat{b}$ ($b = b^{[1,2]}\hat{b}$).

2.3.1 $a^{[1,2]} = 01$ and $b^{[1,2]} = 10$

In this case we span the two sub-intervals $[01\hat{a}, 011^{\{n-2\}}]$ and $[100^{\{n-2\}}, 10\hat{b}]$ separately.

The endpoints of the sub-interval $[01\hat{a}, 011^{\{n-2\}}]$ share the leading bit 0. This means that this sub-interval is immediately reduced to the suffix instance $[1\hat{a}, 11^{\{n-2\}}]$, which can be spanned using the suffix algorithm show in Section 2.2.

The remaining sub-interval $[100^{\{n-2\}}, 10\hat{b}]$ is reduced to a prefix interval $[00^{\{n-2\}}, 0\hat{b}]$ in a similar fashion.

2.3.2 $a^{[1,2]} = 00$ and $b^{[1,2]} = 10$

In this case, we divide the interval into three sub-intervals:

- $[00\hat{a}, 001^{\{n-2\}}]$
- $[010^{\{n-2\}}, 011^{\{n-2\}}]$
- $[100^{\{n-2\}}, 10\hat{b}]$

The sub-interval $[010^{\{n-2\}}, 011^{\{n-2\}}]$ is spanned by the single ternary vector $01\phi^{\{n-2\}}$.

The sub-intervals $[00\hat{a}, 001^{\{n-2\}}]$ and $[100^{\{n-2\}}, 10\hat{b}]$ are spanned together as follows:

1. Recursively solve the $(n-1)$-bit instance $[0\hat{a}, 1\hat{b}] = [a^{[1]}a^{[3,n]}, b^{[1]}b^{[3,n]}]$
2. Insert a 0-bit in the second position of the vectors from the resulting spanning set

Note that $a^{[2]} = b^{[2]} = 0$, so the n-bit vectors produced this way exactly span the union of the intervals $[00\hat{a}, 001^{\{n-2\}}]$ and $[100^{\{n-2\}}, 10\hat{b}]$.

2.3.3 $a^{[1,2]} = 01$ and $b^{[1,2]} = 11$

This case is complementary to case 2.3.2. As in suffix case, note that flipping all the bits in the endpoints transforms this case to case 2.3.2 and flipping the fixed bits in the resulting spanning set preserves correctness.

Napiš explicitně, na který interval redukuji - $[\overline{b}, \overline{a}]$. Zamysli se, jestli se dá zformulovat obecně pro obě tyto aplikace.

2.3.4 $a^{[1,2]} = 00$ and $b^{[1,2]} = 11$

Let j be maximal such that $a^{[1,j]} = 0^{\{j\}}$ and $b^{[1,j]} = 1^{\{j\}}$. Since $a > 0^{\{n\}}$ (or $b < 1^{\{n\}}$), necessarily $j < n$.

The number j gives us three sub-intervals to span:

- $[a, 0^{\{j\}}1^{\{n-j\}}]$
- $[0^{\{j-1\}}10^{\{n-j\}}, 1^{\{j-1\}}01^{\{n-j\}}]$
- $[1^{\{j\}}0^{\{n-j\}}, b]$

Note that the second sub-interval contains exactly the numbers that don't start with either j zeros or j ones, and as such can be spanned by j ternary vectors. We construct $T_1, \ldots, T_j$ that span the second sub-interval by appending $\phi^{\{n-j\}}$ to each of the j cyclic shifts of $01\phi^{\{j-2\}}$. Thus, $T_1 = 01\phi^{\{j-2\}}\phi^{\{n-j\}}$, …, $T_i = \phi^{\{i-1\}}01\phi^{\{j-1-i\}}\phi^{\{n-j\}}$ (for $i \in \{1, \ldots, j-1\}$), …, $T_{j-1} = \phi^{\{j-2\}}01\phi^{\{n-j\}}$, $T_j = 1\phi^{\{j-2\}}0\phi^{\{n-j\}}$.

The other two sub-intervals are spanned recursively. Let $a'' = a^{[j,n]}$ and $b'' = b^{[j,n]}$. Note that $a''^{[1]} = 0$ and $b''^{[1]} = 1$. Note that $|a''| = |b''| = n - j + 1$. Since $j \geq 2$, $n - j + 1 < n$. Let $\mathcal{T}''$ be the spanning set of $[a'', b'']$ computed recursively.

The spanning set of $[a, b]$ will be computed from the vectors $T_1, \ldots, T_j$ and $\mathcal{T}''$ based on the relation between $(n-j)$-bit suffixes of a and b. Let $a' = a^{[j+1,n]}$ and $b' = b^{[j+1,n]}$.

$b' < a' - 1$

In this case, the spanning set of $[a, b]$ consists of the vectors $T_1, \ldots, T_j$ and vectors obtained by prepending each vector from $\mathcal{T}''$ with $\phi^{\{j-1\}}$.

$b' \geq a' - 1$

In this case, the spanning set of $[a, b]$ consists of the vectors $T_1, \ldots, T_{j-1}$ (omitting T_j) and a set $\mathcal{T}'$ which is derived from $\mathcal{T}''$ in the following way:

$$\begin{aligned}\mathcal{T}' = \{&\phi^{\{j-1\}}T | T \in \mathcal{T}'' \text{ and } T^{[1]} \in \{0, \phi\}\} \\ &\cup \{1\phi^{\{j-1\}}T^{[2,n-j+1]} | T \in \mathcal{T}'' \text{ and } T^{[1]} = 1\}\end{aligned}$$

¿ Go into more detail.

2.4 Coverability

Schieber et al. [5] implicitly use orthogonal sets to prove the lower bounds on the sizes of the spanning sets of 1-interval functions. They show a set with the properties of an orthogonal set for each of the cases. This implies the following observation:

Observation 2.4.1. *Every* 1*-interval function is coverable.*

We will not give a detailed proof of Observation 2.4.1 in the general case (Section 2.3). We have proved it in the prefix and suffix case (Corollary 2.2.1).

Add “due to its (vyrecnost)”.

3. 2-interval functions

Add an example with pictures.

In this chapter we will revise the existing results specific to 2-interval functions. Dubovský [2] provides a more detailed overview. We will also introduce a simplified minimization algorithm for 2-switch 2-interval functions and generalize the non-coverability property to all classes of l-switch functions for $l \geq 3$.

Dubovský identified several classes of 2-interval functions [2, p. 5] based on the number of switches, although he did not use the term "switch". Recall that an l-switch function has exactly l input vectors x such that $f(x) \neq f(x+1)$. It is easy to see that a 2-interval function can only have 2, 3 or 4 switches. For 2-switch functions Dubovský proposed a polynomial minimization algorithm while for 3- and 4-switch functions he proposed a polynomial 2-approximation algorithm.

In Section 3.1 we will present a simplified minimization algorithm for 2-switch 2-interval functions. In Section 3.2 we will show that functions with 3 and more switches are not coverable in general.

3.1 2-switch 2-interval functions

In this section we will introduce a polynomial algorithm that solves the following problem:

Problem 3.1 (Minimization of 2-switch 2-interval function)**.** Given a proper 2-interval n-ary function f such that $f(0^{\{n\}}) = 1$ and $f(1^{\{n\}}) = 1$, find a minimum spanning set of f.

Let f be a 2-switch 2-interval Boolean function (necessarily $f(0^{\{n\}}) = f(1^{\{n\}}) = 1$).[1] Dubovský described a polynomial minimization algorithm for this class of functions. The algorithm in Dubovský [2, p. 17] follows the approach from Schieber et al. [5] and relies on a quite complicated and technically involved case analysis. Here we will show that we can find a minimum DNF formula representing a 2-switch 2-interval function f by reducing the function to a 1-interval function f'. The 1-interval function f' is then spanned optimally using the polynomial algorithm shown in Chapter 2. This approach is much simpler than the case analysis in Dubovský [2].

A 2-switch 2-interval function f necessarily has $f(0^{\{n\}}) = f(1^{\{n\}}) = 1$ and thus there are two numbers b_1 and a_2 so that $f = f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$. Since f is 2-switch, a false point must separate the true points b_1 and a_2, thus $b_1 < a_2 - 1$ (f is *proper* 2-interval).

Note that f is a negation of a 2-switch 1-interval function. We can also say that f is a *false point* 1*-interval function* as the false points of f form a single interval $[b_1 + 1, a_2 - 1]$.

3.1.1 Description of the algorithm

Input A pair of n-bit numbers b_1, a_2 that satisfy the inequality $b_1 < a_2 - 1$.

[1]This definition corresponds to class A_0 in Dubovský's classification [2, p. 5].

The numbers b_1 and a_2 are two of the four endpoints of the 2-switch proper 2-interval function $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]} = f$.

Output A spanning set of f. We will denote the output set with $\mathcal{T}$.

Procedure The algoritm constructs $\mathcal{T}$ from two partial spanning sets, $\mathcal{T}_{out}$ and $\mathcal{T}_{in}$.

Let c be the longest common prefix of b_1 and a_2 and let j be its length. Since $b_1 < a_2$, necessarily $j < n$.

We will split the true points of f into two sets S_{out} and S_{in} based on the j-bit prefix. We will span the sets S_{out} and S_{in} separately.

Let S_{out} be the set of all true points of f that do not start with the prefix c. Note that all vectors that do not start with the prefix c are true points of f, since each of them is either smaller than b_1 or larger than a_2. The set S_{out} then contains precisely all the n-bit binary vectors that are strictly smaller than $c0^{\{n-j\}}$ or strictly larger than $c1^{\{n-j\}}$:

$$\begin{aligned} S_{out} &= \{x \in \{0,1\}^n | f(x) = 1 \text{ and } x^{[1,j]} \neq c\} \\ &= \{x \in \{0,1\}^n | x^{[1,j]} \neq c\} \\ &= \{x \in \{0,1\}^n | x < c0^{\{n-j\}} \text{ or } x > c1^{\{n-j\}}\} \end{aligned}$$

We get the set $\mathcal{T}_{out}$ that spans exactly S_{out} in the following way:

$$\mathcal{T}_{out} = \{c^{[1,o-1]}\overline{c^{[o]}}\phi^{\{n-o\}} | o \in \{1,\ldots,j\}\}$$

We still need to span all the true points of f that start with the prefix c. Let S_{in} denote the set of these points:

$$\begin{aligned} S_{in} &= \{x \in \{0,1\}^n | f(x) = 1 \text{ and } x^{[1,j]} = c\} \\ &= \{x \in \{0,1\}^n | (x \leq b_1 \text{ or } x \geq a_2) \text{ and } x^{[1,j]} = c\} \end{aligned}$$

Since $b_1 < a_2$, necessarily $b_1^{[j+1]} = 0$ and $a_2^{[j+1]} = 1$. Let $b_1' = b_1^{[j+2,n]}$ and $a_2' = a_2^{[j+2,n]}$. Note that $b_1 = c0b_1'$ and $a_2 = c1a_2'$.

Let's span the 1-interval function $f' = f^{n-j}_{[0a_2',1b_1']}$ optimally using the procedure shown in Chapter 2. Let's denote the resulting minimum spanning set of f' with $\mathcal{T}_{[0a_2',1b_1']}$. We can use this set to span S_{in}:

$$\mathcal{T}_{in} = \{c\overline{T^{[1]}}T^{[2,n-j]} | T \in \mathcal{T}_{[0a_2',1b_1']}\}$$

Having constructed both of the sets $\mathcal{T}_{out}$ and $\mathcal{T}_{in}$, the algorithm returns their union.

3.1.2 Feasibility

Theorem 3.1.1. *$\mathcal{T}$ spans exactly $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$.*

We will separately prove that $\mathcal{T}_{out}$ spans exactly S_{out} and that $\mathcal{T}_{in}$ spans exactly S_{in}. Since $\mathcal{T} = \mathcal{T}_{out} \cup \mathcal{T}_{in}$ and $TP(f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}) = S_{out} \cup S_{in}$, the correctness of Theorem 3.1.1 will follow.

Lemma 3.1.1. *$\mathcal{T}_{out}$ spans exactly S_{out}.*

Proof. In order to show that $span(\mathcal{T}_{out}) = S_{out}$, we will prove the two inclusions separately.

$span(\mathcal{T}_{out}) \subseteq S_{out}$ Let $c^{[1,o-1]}\overline{c^{[o]}}\phi^{\{n-o\}} \in \mathcal{T}_{out}$ span x. Since the j-bit prefix of x differs from c, we conclude that $x \in S_{out}$.

$span(\mathcal{T}_{out}) \supseteq S_{out}$ Let $x \in S_{out}$ such that $x < c0^{\{n-j\}}$. Recall that by Observation 1.2.1, there must be binary vectors $\hat{c}$, x' and c' such that $x = \hat{c}0x'$ and $c0^{\{n-j\}} = \hat{c}1c'$ (namely $\hat{c}$ is a proper prefix of c). Let $o = |\hat{c}| + 1$. It is easy to see that the vector $\hat{c}0\phi^{\{n-o\}} \in \mathcal{T}_{out}$ spans x. If $x > c1^{\{n-j\}}$, a symmetric argument shows a vector in $\mathcal{T}_{out}$ that spans x. We conclude that $\mathcal{T}_{out}$ spans x.

□

Lemma 3.1.2. *$\mathcal{T}_{in}$ spans exactly S_{in}.*

Proof. Again we will prove the two inclusions separately.

1. $span(\mathcal{T}_{in}) \subseteq S_{in}$:

   Let $c\overline{T^{[1]}}T^{[2,n-j]} \in \mathcal{T}_{in}$ span x. Clearly x starts with the prefix c. Let $x = c\alpha x'$ where $\alpha \in \{0,1\}$. Without loss of generality let $\alpha = 0$ (the complementary case is symmetric).

   Since $c\overline{T^{[1]}}T^{[2,n-j]}$ spans $x = c0x'$, T spans $1x'$. Since $T \in \mathcal{T}_{[0a'_2,1b'_1]}$, we have $1x' \in [0a'_2, 1b'_1]$ and especially $1x' \leq 1b'_1$. It follows that $x' \leq b'_1$ and $x = c0x' \leq c0b'_1 = b_1$.

2. $span(\mathcal{T}_{in}) \supseteq S_{in}$:

   Let $x \in S_{in}$. This means that $x^{[1,j]} = c$, and $x \leq b_1$ or $x \geq a_2$. Without loss of generality let $x^{[j+1]} = 0$ (the complementary case is symmetric) and let $x = c0x'$.

   Since $x^{[1,j+1]} = c0 < c1 = a_2^{[1,j+1]}$, x cannot be greater than or equal to a_2. Thus necessarily $x \leq b_1$. Since x and b_1 share the prefix $c0$, necessarily $x' \leq b'_1$ and by extension $1x' \leq 1b'_1$. Trivially $1x' \geq 0a'_2$.

   Since $1x' \in [0a'_2, 1b'_1]$, there must be some $T \in \mathcal{T}_{[0a'_2,1b'_1]}$ that spans $1x'$. The corresponding $c\overline{T^{[1]}}T^{[2,n-j]} \in \mathcal{T}_{in}$ clearly spans $x = c0x'$.

□

We conclude that $\mathcal{T}$ spans exactly $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$.

3.1.3 Optimality

To show that $\mathcal{T}$ is a minimum spanning set of $f = f^n_{[0\{n\},b_1],[a_2,1\{n\}]}$, we will construct an orthogonal set V of f of size $|\mathcal{T}|$. By Theorem 1.6.2 this proves that the spanning set $\mathcal{T}$ is minimum.

The set V consists of three subsets:

- $V_{in} = \{c\overline{v^{[1]}}v^{[2,n-j]} | v \in V_{[0a'_2,1b'_1]}\}$, where $V_{[0a'_2,1b'_1]}$ is an orthogonal set of $f_{[0a'_2,1b'_1]}$ of size $|\mathcal{T}_{[0a'_2,1b'_1]}| = |\mathcal{T}_{in}|$. Such set exists because all 1-interval functions are coverable (see Observation 2.4.1).
- $V_0 = \{c^{[1,o-1]}0c^{[o+1,j]}(b_1+1)^{[j+1,n]} | c^{[o]} = 1\}$
- $V_1 = \{c^{[1,o-1]}1c^{[o+1,j]}(a_2-1)^{[j+1,n]} | c^{[o]} = 0\}$

Note that the vector $c^{[1,o-1]}0c^{[o+1,j]}(b_1+1)^{[j+1,n]} \in V_0$ (for any o such that $c^{[o]} = 1$) is equal to $(b_1+1)^{[1,o-1]}0(b_1+1)^{[o+1,n]}$ because $b_1^{[j+1]} = 0$, and by extension $(b_1+1)^{[1,j]} = b_1^{[1,j]} = c$. A symmetric observation can be made for vectors in V_1.

It is easy to see that $|V_{in}| = |\mathcal{T}_{in}|$. Since there is one vector in V_0 for each 1 bit of c and one vector in V_1 for each 0 bit of c, $|V_0| + |V_1| = |c| = j = |\mathcal{T}_{out}|$.

To see that the sets V_{in}, V_0 and V_1 are pairwise disjoint, consider any $v_{in} \in V_{in}$, $v_0 \in V_0$ and $v_1 \in V_1$. The j-prefix of both v_0 and v_1 differs from c, so $v_0 \neq v_{in}$ and $v_1 \neq v_{in}$. If $v_0^{[1,j]} = c^{[1,o-1]}0c^{[o+1,j]}$, then v_0 differs from v_1 in the o-th position. We conclude that $|V| = |V_{in}| + |V_0| + |V_1|$. Since $|V_{in}| = |\mathcal{T}_{in}|$ and $|V_0| + |V_1| = |\mathcal{T}_{out}|$, $|V| = |\mathcal{T}|$.

We still need to prove that V is orthogonal with respect to f. Recall that by Definition 1.6.1, the set V is orthogonal if and only if it only contains true points and every pair of vectors in V is orthogonal.

Lemma 3.1.3. $V \subseteq TP(f^n_{[0\{n\},b_1],[a_2,1\{n\}]})$

Proof. It is easy to see that $v_0 \leq b_1$ and $v_1 \geq a_2$ for any $v_0 \in V_0$ and $v_1 \in V_1$.

Let $v_{in} \in V_{in}$. Let v be the corresponding vector in $V_{[0a'_2,1b'_1]}$ ($v_{in} = c\overline{v^{[1]}}v^{[2,n-j]}$). Since $V_{[0a'_2,1b'_1]}$ is orthogonal with respect to $f_{[0a'_2,1b'_1]}$, necessarily $v \geq 0a'_2$ and $v \leq 1b'_1$. Without loss of generality let $v^{[1]} = 0$ (the complementary case is symmetric). Since $v = 0v^{[2,n-j]} \geq 0a'_2$, $v_{in} = c1v^{[2,n-j]} \geq c1a'_2 = a_2$.

We conclude that all the vectors in V are true points of $f^n_{[0\{n\},b_1],[a_2,1\{n\}]}$. □

Lemma 3.1.4. *Let $x, y \in V$, $x \neq y$. Then x and y are orthogonal with respect to $f^n_{[0\{n\},b_1],[a_2,1\{n\}]}$.*

Proof. We need to show that x and y cannot be spanned by a single ternary vector. Let T be any ternary vector that spans both x and y. We shall consider all six possible combinations one by one:

- $x, y \in V_{in}$: Let $x = c\overline{v_x^{[1]}}v_x^{[2,n-j]}$ and $y = c\overline{v_y^{[1]}}v_y^{[2,n-j]}$. The vectors v_x and v_y are orthogonal with respect to $f_{[0a'_2,1b'_1]}$ by definition of $V_{[0a'_2,1b'_1]}$.

  Let $\hat{T} = \overline{T^{[j+1]}}T^{[j+2,n]}$. Since v_x and v_y are orhtogonal with respect to $f_{[0a'_2,1b'_1]}$ and since $\hat{T}$ spans both v_x and v_y, let $\hat{z}$ be a false point of $f_{[0a'_2,1b'_1]}$

that is spanned by $\hat{T}$. Without loss of generality let $\hat{z}^{[1]} = 0$ (the complementary case is symmetric). Since $\hat{z}$ is a false point of $f_{[0a'_2, 1b'_1]}$ and $\hat{z}^{[1]} = 0$, necessarily $\hat{z} = 0\hat{z}^{[2,n-j]} < 0a'_2$ and by extension $\hat{z}^{[2,n-j]} < a'_2$.

¿ Replace with "contraction".

Let $z = c\overline{\hat{z}^{[1]}}\hat{z}^{[2,n-j]} = c1\hat{z}^{[2,n-j]}$. To see that z is a false point of $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$, note that $z > b_1$ since $z^{[j+1]} = 1$ and $z < a_2$ since $z^{[j+2,n]} < a'_2$. To see that T spans z, note that $T^{[1,j]}$ spans $x^{[1,j]} = c = z^{[1,j]}$ and $T^{[j+1,n]} = \overline{\hat{T}^{[1]}}\hat{T}^{[2,n-j]}$ spans $\overline{\hat{z}^{[1]}}\hat{z}^{[2,n-j]} = z^{[j+1,n]}$.

- $x, y \in V_0$: T spans the false point $b_1 + 1$.
- $x, y \in V_1$: T spans the false point $a_2 - 1$.
- $x \in V_{in}, y \in V_0$: T spans the false point $b_1 + 1$.
- $x \in V_{in}, y \in V_1$: T spans the false point $a_2 - 1$.
- $x \in V_0, y \in V_1$: T spans the false point $b_1 + 1$.

We have shown that no matter which combination of different vectors x, y from V and a ternary vector that spans both x and y we choose, the ternary vector necessarily spans some false point of $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$. □

We conclude that V is orthogonal with respect to $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$. Since $|V| = |\mathcal{T}|$, by Theorem 1.6.2 $\mathcal{T}$ is a minimum spanning set of $f^n_{[0^{\{n\}},b_1],[a_2,1^{\{n\}}]}$.

Corollary 3.1.1. *Every 2-switch 2-interval function is coverable.*

Since every 2-switch function is either 1- or 2-interval (see Example 1.1) and every 1-switch function is 1-interval, we collect the results from Observation 2.4.1 and Corollary 3.1.1 in the following statement:

Corollary 3.1.2. *Every function with at most 2 switches is coverable.*

3.2 Functions with 3 and more switches

¿ Separate in a chapter.

The situation changes in 3-switch functions. In 1-switch (prefix and suffix) and 2-switch (1-interval and 2-interval with extreme outer endpoints) functions, we managed to efficiently find minimum spanning sets. The proofs of optimality of the spanning sets depend on construction of sufficiently large orthogonal sets.[2] The fact that this aproach succeeds depends on the fact that all the functions we have considered so far (1- and 2-switch) are *coverable* (according to Definition 1.6.3).

Update footnote once we construct an orthogonal set explicitly in section 3.1.

Dubovský has shown that the 3-switch function $f = f^4_{[0,4],[11,14]}$ is not coverable. The size of a maximum orthogonal set of f is 4 while the size of a minimum spanning set of f is 5. [2, p. 32] This shows that 3-switch functions are not coverable in general. Dubovský proved these bounds on sizes of orthogonal sets and spanning sets by exhaustion using software tools. We will generalize the result for functions with larger number of switches.

[2]While we have only shown an orthogonal set explicitly in the case of prefix interval (subsection 2.2.2), the other cases use orthogonal sets implicitly, both in [5] and section 3.1.

Lemma 3.2.1. *If there is an l-switch function that is not coverable, then there is also an $(l+1)$-switch function that is not coverable.*

Proof. Let f be an n-ary l-switch function that is not coverable. We will construct an $(n+1)$-ary $(l+1)$-switch function f' that is not coverable. We shall distinguish two cases depending on the value of f on $1^{\{n\}}$.

1. $f(1^{\{n\}}) = 1$:

   Let f' be an $(n+1)$-ary function defined in the following way:

$$f'(x') = \begin{cases} f(x'^{[2,n+1]}) & \text{if } x'^{[1]} = 0 \\ 0 & \text{if } x'^{[1]} = 1 \end{cases}$$

   Since f is l-switch, there are exactly l vectors x of length n ($x < 1^{\{n\}}$) such that $f(x) \neq f(x+1)$. Prepending a 0 to each of these vectors, we get l vectors x' of length $n+1$ such that $f'(x') \neq f'(x'+1)$. The $(l+1)$-st switch vector of f' is $01^{\{n\}}$, since $f'(01^{\{n\}}) = 1$ and $f'(10^{\{n\}}) = 0$. On the other hand, observe that all switch vectors of f' correspond to switch vectors of f with the single exception of $01^{\{n\}}$. Thus we have shown that f' is $(l+1)$-switch.

   There is a size preserving 1-1 correspondence between the spanning sets of f and f'. A spanning set $\mathcal{T}$ of f corresponds to the spanning set $0\mathcal{T} = \{0T | T \in \mathcal{T}\}$ of f'. The same holds for orthogonal sets. This proves that if f is not coverable, neither is f'.

2. $f(1^{\{n\}}) = 0$:

   Let f' be an $(n+1)$-ary function defined in the following way:

$$f'(x') = \begin{cases} f(x'^{[2,n+1]}) & \text{if } x'^{[1]} = 0 \\ 0 & \text{if } 10^{\{n\}} \leq x' \leq 1^{\{n\}}0 \\ 1 & \text{if } x' = 1^{\{n+1\}} \end{cases}$$

   Again, the l switches of f translate to l switches of f' by prepending a 0. The $(l+1)$-st switch is $1^{\{n\}}0$ in this case.

   To show that f' is not coverable, we will prove bounds on the size of its minimum spanning set and maximum orthogonal set.

   $dnf(f') \geq dnf(f) + 1$ We need a dedicated ternary vector to span the true point $1^{\{n+1\}}$ in f'. If a ternary vector spanned both $1^{\{n+1\}}$ and $0x$ for any n-bit x, it would necessarily also span the false point $01^{\{n\}}$.

   If we could span f' with less than $dnf(f) + 1$ vectors, we could span f with less than $dnf(f)$ vectors, since given a spanning set of f', leaving out the vector that spans the true point $1^{\{n+1\}}$ and removing the leading symbol of the rest (necessarily a 0) gives a spanning set of f.

$ortho(f') \leq ortho(f) + 1$ Let us proceed by contradiction and let V' be an orthogonal set of f' of size $ortho(f) + 2$. Let $V = \{v'^{[2,n+1]} | v' \in V' \text{ and } v'^{[1]} = 0\}$. Since V' only consists of true points of f', there can be at most one vector in V' that starts with a 1. It follows that $|V| \geq |V'| - 1 = ortho(f) + 1$. V is an orthogonal set of f – if $0u$ and $0v$ are orthogonal in f', every ternary vector that spans them must span a false point $0x$. Leaving out the leading symbol preserves the relation, and thus orthogonality as well. We have shown an orthogonal set of f of size at least $ortho(f) + 1$, which contradicts the premise that $ortho(f)$ is the size of a maximum orthogonal set of f.

Since f is not coverable, $ortho(f) < dnf(f)$.

Putting the inequalities together:

$$ortho(f') \leq ortho(f) + 1 < dnf(f) + 1 \leq dnf(f')$$

We conclude that f' is an $(l + 1)$-switch non-coverable function.

□

We conclude with the following theorem:

Theorem 3.2.1. *For any $l \geq 3$, there is an l-switch non-coverable function. Moreover, the arity of such function can be as small as $l + 1$.*

Proof. Starting with the 4-ary 3-switch function $f^4_{[0,4],[11,14]}$ from Dubovský [2], we can get a non-coverable l-switch function for any $l \geq 3$ by using Lemma 3.2.1 inductively. The arity of $f^4_{[0,4],[11,14]}$ is 4 and every time we increment l in Lemma 3.2.1, we increase the arity by 1. □

This result indicates that if we consider an algorithm that minimizes the representations of all l-switch functions for any $l \geq 3$, then we have to use a different tool than orthogonal sets to show that the algorithm really finds an optimal spanning set.

¿ Also differentiate the functions based on their values in the points $0^{\{n\}}$ and $1^{\{n\}}$ and show we can find an $(l + 1)$-switch function for all the relevant cases. – Not necessary, but if the proof is simple, present it.

4. Suffix-prefix decomposition approximation algorithm for k-interval functions

Rename the algorithm to "suffix-prefix decomposition algorithm".

In this chapter we shall show a simple $2k$-approximation algorithm for the following problem: *Given a proper k-interval function f, find a minimum spanning set of f.* We shall assume that the function f is represented by pairs of endpoints of the respective intervals. The algorithm we shall describe is a straightforward extension of Schieber et al.'s suffix-prefix approximation algorithm for disjoint spanning sets of 1-interval functions [5, Section 6].

Move problem definition in Definitions. Refer to the problem by a name or a number (LaTeXreference).

In the rest of this chapter, we will only consider *proper* k-interval Boolean functions, that is those whose adjacent intervals are separated by at least one false point (see Definition 1.7.2). A k-interval Boolean function which is not proper k-interval can be efficiently reduced to a proper l-interval where $l < k$ by joining the adjoint intervals.

4.1 Description of the algorithm

Unite with the following chapter.

Input Numbers $a_1, b_1, \ldots, a_k, b_k$ for $k \geq 0$ that satisfy the inequalities $0^{\{n\}} \leq a_1$, $a_1 \leq b_1$, $b_1 < a_2 - 1$, …, $a_i \leq b_i$, $b_i < a_{i+1} - 1$ (for $i \in \{1, \ldots, k-1\}$), …, $b_{k-1} < a_k - 1$, $a_k \leq b_k$, $b_k \leq 1^{\{n\}}$.

Such numbers are the endpoints of the intervals of the proper k-interval Boolean function $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Output A spanning set of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Update in the interval decomposition algorithm.

Procedure The algorithm spans each of the intervals $[a_i, b_i]$ separately. The set that spans $[a_i, b_i]$ will be denoted $\mathcal{T}_i$, and the output set will be denoted $\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i$.

Let's consider the interval $[a_i, b_i]$. Let c be the longest common prefix of a_i and b_i and let j be its length.

If $j = n$, that is $c = a_i = b_i$, $\mathcal{T}_i$ consists of the single vector $c = a_i = b_i$.

Simplify: if $j = n$, then $\mathcal{T}_{i,\alpha} = c$ for both α.

We are left with the situation $j < n$.

Note that $a_i^{[j+1]} = 0$ and $b_i^{[j+1]} = 1$, since $a_i \leq b_i$ and $a_i \neq b_i$. Now let $a' = a_i^{[j+2,n]}$ and $b' = b_i^{[j+2,n]}$ ($a_i = c0a'$ and $b_i = c1b'$). Optimally span the suffix interval $[a', 1^{\{n-j-1\}}]$ and the prefix interval $[0^{\{n-j-1\}}, b']$ using the algorithm introduced in Section 2.2 and producing spanning sets $\mathcal{T}'_{i,0}$ and $\mathcal{T}'_{i,1}$ respectively. Now simply prepend $a_i^{[1,j+1]} = c0$ to all vectors in $\mathcal{T}'_{i,0}$ and $b_i^{[1,j+1]} = c1$ to all vectors in $\mathcal{T}'_{i,1}$, producing $\mathcal{T}_{i,0}$ and $\mathcal{T}_{i,1}$ respectively. The spanning set of $[a_i, b_i]$ is then $\mathcal{T}_i = \mathcal{T}_{i,0} \cup \mathcal{T}_{i,1}$.

¿ Treat more efficiently the cases $a_i^{[j+1,n]} = 0^{\{n-j\}}$ (prefix) and $b_i^{[j+1,n]} = 1^{\{n-j\}}$ (suffix), especially $a_1 = 0^{\{n\}}$ and $b_k = 1^{\{n\}}$. This doesn't improve the approximation ratio though and may complicate the proof.

Kučera: Můžete to dát do poznámky k algoritmu, jako „possible improvements“ s tím, že to nehraje roli pro aproximační poměr, ale v některých případech to může být užitečné.

4.2 Feasibility

Theorem 4.2.1. *The algorithm spans exactly* $f^n_{[a_1,b_1],\dots,[a_k,b_k]}$.

Proof. We need to show that for each i, $\mathcal{T}_i$ spans exactly $[a_i, b_i]$. This is obvious in case $j = n$.

In the remaining case $j < n$, observe that $0\mathcal{T}'_{i,0} = \{0T | T \in \mathcal{T}'_{i,0}\}$ spans exactly the interval $[0a', 01^{\{n-j-1\}}]$ and $1\mathcal{T}'_{i,1}$ spans exactly the interval $[10^{\{n-j-1\}}, 1b']$. Since $10^{\{n-j-1\}} = 01^{\{n-j-1\}} + 1$, the union of these sets spans exactly the interval $[0a', 1b'] = [a_i^{[j+1,n]}, b_i^{[j+1,n]}]$. Prepending the common prefix c to $0\mathcal{T}'_{i,0} \cup 1\mathcal{T}'_{i,1}$ preserves the spanning to $[a_i, b_i]$ and matches the operation performed in the algorithm to produce $\mathcal{T}_i$.

The union of such $\mathcal{T}_i$s clearly spans exactly the union of the intervals. □

4.3 Approximation ratio

Unite terminology with chapter "betterapprox".

Theorem 4.3.1. *Let $\mathcal{T}_{opt}$ be an optimal spanning set of $f^n_{[a_1,b_1],\dots,[a_k,b_k]}$ and let $\mathcal{T}$ be the spanning set returned by the algorithm. We claim that:*

$$|\mathcal{T}| \leq 2k|\mathcal{T}_{opt}|$$

Proof. To prove the statement, we will show an orthogonal set V of the function $f^n_{[a_1,b_1],\dots,[a_k,b_k]}$ of size $|V| \geq \frac{|\mathcal{T}|}{2k}$. By section 1.6.2, the existence of such orthogonal set gives a lower bound $|V|$ on $|\mathcal{T}_{opt}|$.

Let's first consider the case that $|\mathcal{T}_i| \leq 2$ for every i. Since $|\mathcal{T}_i| \leq 2$ for every $i \in \{1, \dots, k\}$, $|\mathcal{T}| \leq 2k$. If $k = 0$ (in the degenerate case that there are no intervals), $|\mathcal{T}| \leq 2k = 0 = 2k|\mathcal{T}_{opt}|$. If $k \geq 1$, necessarily $|\mathcal{T}_{opt}| \geq 1$. It is then easy to see that $|\mathcal{T}| \leq 2k \leq 2k|\mathcal{T}_{opt}|$.

We are now left with the case that $|\mathcal{T}_i| > 2$ for some i. Necessarily $\mathcal{T}_i = \mathcal{T}_{i,0} \cup \mathcal{T}_{i,1}$ for such i. One of the sets $\mathcal{T}_{i,0}$ and $\mathcal{T}_{i,1}$ must contain at least 2 vectors because otherwise $|\mathcal{T}_i| = 2$.

Let $\mathcal{T}_{i,\alpha}$ ($i \in \{1, \dots, k\}, \alpha \in \{0, 1\}$) be the largest of the partial spanning sets of this type produced during the course of the algorithm. Note that $|\mathcal{T}_{i,\alpha}| \geq 2$.

Without loss of generality, let $\alpha = 1$. The set $\mathcal{T}'_{i,1}$ that spans exactly $[0^{\{n-j-1\}}, b' = b_i^{[j+2,n]}]$ was in this case obtained using the prefix algorithm introduced in Section 2.2.2. Note that $b' < 1^{\{n-j-1\}}$ because otherwise $|\mathcal{T}_{i,1}| = 1$. In the proof of optimality of this algorithm (see Theorem 2.2.2), we showed an orthogonal set

¿ Clarify more – what is "this type"?

Add "The complementary case $\alpha = 0$ is symmetric.".

that matches the size of the spanning set produced by the algorithm. Let's call this set $V'_{i,1}$. Recall that by Corollary 2.2.2 all the pairs of vectors in $V'_{i,1}$ collide on the false point $b'+1$ of $f^{n-j-1}_{[0\{n-j-1\},b']}$.

Let $\hat{b} = b_i^{[1,j+1]}$ $(b_i = \hat{b}b')$. Let $V = \{\hat{b}v | v \in V'_{i,1}\}$. We claim that V is an orthogonal set of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Let $\hat{b}x, \hat{b}y \in V$ and $x \neq y$. Since by Corollary 2.2.2 x and y collide on $b'+1$, $\hat{b}x$ and $\hat{b}y$ collide on $\hat{b}(b'+1) = b_i + 1$. To see that b_i+1 is a false point of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$, note that if $i < k$, then $b_i < a_{i+1} - 1$ by the requirements on input data (we only consider *proper* k-interval functions).

It follows that V is orthogonal with respect to $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$. From section 1.6.2 we get $|\mathcal{T}_{opt}| \geq |V|$.

Since $\mathcal{T}_{i,\alpha}$ is the largest of the at most $2k$ spanning sub-sets of this type and the spanning sets of the other types are of size 1, necessarily $|\mathcal{T}| \leq 2k|\mathcal{T}_{i,\alpha}| = 2k|V| \leq 2k|\mathcal{T}_{opt}|$. □

Clarify.

Note that the spanning set returned by the algorithm is disjoint. This makes the algorithm $2k$-approximation for the disjoint case as well:

Theorem 4.3.2. *Let $\mathcal{T}_{disjoint}$ be an optimal* disjoint *spanning set of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$ and let $\mathcal{T}$ be the (disjoint) spanning set returned by our algorithm. We claim that:*

$$|\mathcal{T}| \leq 2k|\mathcal{T}_{disjoint}|$$

Proof. Since every disjoint spanning set is a spanning set in general, $|\mathcal{T}_{opt}| \leq |\mathcal{T}_{disjoint}|$, where $\mathcal{T}_{opt}$ is an optimal (not necessarily disjoint) spanning set of $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$.

Using Theorem 4.3.1 we get:

$$|\mathcal{T}| \leq 2k|\mathcal{T}_{opt}| \leq 2k|\mathcal{T}_{disjoint}|$$

□

Theorem 4.3.3. *The approximation ratio of $2k$ is tight.*

Proof. For every k that is a power of 2 we will show a k-interval function such that $|\mathcal{T}| = 2k|\mathcal{T}_{opt}|$ (following the notation from Theorem 4.3.1).

Let $k = 2^{n_k}$. Let P be all the n_k-bit numbers, that is $P = \{0,1\}^{n_k}$. Note that $|P| = 2^{n_k} = k$. Let $n = n_k + 3$.

For each $p \in P$, we define the interval $[a_p, b_p]$ by appending certain 3-bit suffixes to p:

- $a_p = p000$
- $b_p = p011$

The first appended bit (0 for a_p and 0 for b_p) ensures that there is at least one false point between any pair of intervals defined this way (all the points that have a 1 in their (n_k+1)-st position are false points).

The remaining pair of appended bits (00 for a_p and 11 for b_p) ensures that $a_p^{[1,n-1]} \neq b_p^{[1,n-1]}$, so the algorithm will span the sub-intervals $[a_p = p000, p001]$

and $[p010, b_p = p011]$ separately, producing at least 2 vectors to span $[a_p, b_p]$. The configuration also enables the interval $[a_p, b_p]$ to be spanned by the single ternary vector $p0\phi^{\{2\}}$.

¿ Rephrase.

Thus we have k intervals none of which intersect or are adjoint.

Since $P = \{0,1\}^{n_k}$, we can span all the intervals by the single ternary vector $\phi^{\{n_k\}}0\phi^{\{2\}}$. Clearly $|\mathcal{T}_{opt}| = 1$.

However, the approximation algorithm uses $2k$ vectors to span the intervals, since it spans each of the intervals separately and uses the two vectors $p00\phi, p01\phi$ for each interval.

We get $|\mathcal{T}| = 2k = 2k|\mathcal{T}_{opt}|$.

Note that the optimal spanning set is trivially disjoint, so the ratio is tight in disjoint case as well. □

5. Interval decomposition approximation algorithm for k-interval functions

The $2k$-approximation suffix-prefix decomposition algorithm introduced in chapter 4 can be naturally improved by spanning each of the k intervals of the given k-interval function optimally using the minimization algorithm introduced in chapter 2 (originally in Schieber et al. [5]). We will call this improved algorithm "interval decomposition algorithm".

Since the interval decomposition algorithm clearly performs at least as well as the suffix-prefix decomposition algorithm, it is $2k$-approximation as well. Dubovský has managed to prove that on 2-interval functions, the interval decomposition algorithm has an approximation ratio 2 [2, p. 39], while the suffix-prefix decomposition algorithm has a tight approximation ratio 4 on 2-interval functions (see Theorem 4.3.3). This might lead us to expect that the interval decomposition algorithm performs significantly better in general. However, we will show in this chapter that its approximation ratio converges to $2k$ for large k (and namely that it is strictly larger than k for $k > 2$).

Do we really prove this for all ks? Try to!

5.1 Algorithm description

Label or name the algorithm properly and then refer to it consistently.

Input Numbers $a_1, b_1, \ldots, a_k, b_k$ encoded as binary vectors of length n that satisfy the inequalities $0^{\{n\}} \leq a_1$, $a_1 \leq b_1$, $b_1 < a_2 - 1$, $\ldots$, $a_i \leq b_i$, $b_i < a_{i+1} - 1$ (for $i \in \{1, \ldots, k-1\}$), $\ldots$, $b_{k-1} < a_k - 1$, $a_k \leq b_k$, $b_k \leq 1^{\{n\}}$.

Such numbers are endpoints of intervals of a proper k-interval Boolean function.

Output A set of ternary vectors.

¿ Add "that spans $f^n_{[a_1,b_1],\ldots,[a_k,b_k]}$".

Procedure The algorithm spans each of the k intervals $[a_i, b_i]$ separately. For every i, let $\mathcal{T}_i$ be the spanning set of the interval $[a_i, b_i]$ returned by the 1-interval optimization algorithm introduced in chapter 2. The algorithm returns $\mathcal{T} = \bigcup_{i=1}^{k} \mathcal{T}_i$.

Note that the spanning set constructed by the interval decomposition algorithm need not be disjoint (unlike the spanning set constructed by the suffix-prefix decomposition algorithm). If we used Schieber et al. disjoint 1-interval spanning set minimization algorithm [5], we would obtain a disjoint spanning set. We will not analyze the approximation ratio in this case.

5.2 Approximation ratio

Let F be a class of Boolean functions. We shall denote the worst case approximation ratio of the interval decomposition algorithm on functions from class F with $approx_{ID}(F)$:

$$approx_{ID}(F) = \sup_{f \in F} \frac{id(f)}{dnf(f)}$$

Is this definition of approximation ratio correct? I'm afraid it may be inconsistent with some parts of the text.

where $id(f)$ is the size of the spanning set of f returned by the interval decomposition algorithm.

We will analyse the approximation ratio of the interval decomposition algorithm on k-interval Boolean functions, that is $approx_{ID}(INT(k))$, where $INT(k)$ is the class of k-interval Boolean functions. We will parametrize the approximation ratio with the number of intervals k.

5.2.1 Upper bound of $2k$

Since the interval decomposition algorithm spans each of the k intervals separately and optimally, while the suffix-prefix decomposition algorithm spans each of them separately and possibly non-optimally, we get an upper bound $2k$ on $approx_{ID}(INT(k))$ by Theorem 4.3.1:

$$approx_{ID}(INT(k)) \leq approx_{SPD}(INT(k)) = 2k$$

However, the ratio $2k$ need not be tight for the interval decomposition algorithm. We will continue to analyze the lower bounds on the approximation ratio.

5.2.2 Known lower bounds for $k \in \{1, 2\}$

Dubovský has shown that the approximation ratio is strictly smaller than $2k$ for $k = 2$, as the algorithm is 2-approximation in this case [2, p. 39]. Similarly, it is obvious that the algorithm gives an optimal result in case $k = 1$ (which corresponds to the approximation ratio 1).

$$approx_{ID}(INT(1)) = 1$$
$$approx_{ID}(INT(2)) = 2$$

In the following sections, we will construct a class of Boolean functions on some of which the interval decomposition algorithm performs strictly worse than k-approximately for $k = 3$. We will generalize this to show that the approximation ratio converges to $2k$ for large k.

Reformulate the sentence.

¿ Try to prove this for all $k > 2$.

5.2.3 Approximation-hard functions

Don't call the functions "hard".

In this section we shall describe a class of functions on which the interval decomposition algorithm performs particularly badly. For every $l \geq 1$ and $n \geq l + 2$, we will construct a proper $(2^{l-1} + 1)$-interval n-ary function f_l^n.

Let $l \geq 1$ denote a parameter that determines the number of intervals of the target function f_l^n. Its exact meaning will be explained in the description of the construction of f_l^n. Let $n \geq l+2$ denote the arity of the target function f_l^n (and thus the length of its points, especially the interval endpoints). Let k denote the number of intervals of f_l^n. After we define the intervals, we will show that $k = 2^{l-1}+1$.

All the interval endpoints have a prefix of length l that specifies the "position" of the interval and a suffix of length $n-l$ that specifies the interval's "shape". We will denote the l-bit "position" prefix of an n-bit binary vector x with p_x ($p_x = x^{[1,l]}$) and the $(n-l)$-bit "shape" suffix with s_x ($s_x = x^{[l+1,n]}$). Note that $x = p_x s_x$.

The "shape" is identical for all the intervals of a given f_l^n – for every $i \in \{1,\ldots,k\}$:

- $s_{a_i} = 0^{\{n-l-1\}}1$
- $s_{b_i} = 1^{\{n-l-1\}}0$

Note that for $n \geq l+3$, this "shape" corresponds to an extreme non-trivial endpoint combination in the 1-interval optimization algorithm (namely case "$b' \geq a'-1$" in subsection 2.3.4), which happens to be the case where the improvement of the optimization algorithm over the suffix-prefix approximation algorithm is the largest.

We construct the "position" prefixes of the endpoints so that they satisfy the following requirements:

1. $p_{a_1} = p_{b_1} = 0^{\{l\}}$
2. $p_{a_k} = p_{b_k} = 1^{\{l\}}$
3. $p_{b_i} = p_{a_i}+1$ for $i \in \{2,\ldots,k-1\}$
4. $p_{a_{i+1}} = p_{b_i}+1$ for $i \in \{1,\ldots,k-1\}$

Using these requirements, we can inductively construct a sequence of l-bit numbers $p_{a_1}, p_{b_1}, p_{a_2}, p_{b_2}, \ldots, p_{a_k}, p_{b_k}$. We start with $p_{a_1} = p_{b_1} = 0^{\{l\}}$ by requirement 1. We follow by constructing $p_{a_{i+1}}$ from p_{b_i} for $i \in \{1,2,\ldots\}$ using requirement 4 and constructing p_{b_i} from p_{a_i} for $i \in \{2,3,\ldots\}$ using requirement 3. Note that the p_{a_i}s constructed this way for $i \in \{2,3,\ldots\}$ systematically exhaust odd numbers, since $p_{a_2} = 0^{\{l-1\}}1$ (recall that we require $l \geq 1$) and $p_{a_{i+1}} = p_{a_i}+2$. We stop the induction as soon as we get to an index k such that $p_{a_k} = 1^{\{l\}}$. Since p_{a_i}s systematically exhaust odd numbers, there must be such k. We terminate by constructing $p_{b_k} = 1^{\{l\}}$ by requirement 2.

¿ Use a better term.

¿ Add "Moreover, $k = 2^{l-1}+1$", possibly with some simple analysis.

Note that the sequence is defined unambiguously for a given l.

The construction of p_{a_i} and p_{b_i} together with the definition of s_{a_i} and s_{b_i} define a set of points $a_i = p_{a_i}s_{a_i}$ and $b_i = p_{b_i}s_{b_i}$ for $i \in \{1,\ldots,k\}$. To see that points defined this way satisfy the requirements on endpoints of a proper k-interval function (see 1.7.2), note that:

- $a_1 \leq b_1$ by requirement 1 and the fact that $s_{a_1} \leq s_{b_1}$
- $a_i \leq b_i$ for $i \in \{2,\ldots,k-1\}$ by requirement 3

- $b_i < a_{i+1} - 1$ for $i \in \{1, \dots, k-1\}$ by requirement 4 and the fact that $s_{a_i} > 0^{\{l\}}$ (or $s_{b_i} < 1^{\{l\}}$)
- $a_k \leq b_k$ by requirement 2 and the fact that $s_{a_k} \leq s_{b_k}$

To see that the number of intervals k is $2^{l-1} + 1$, we will count the "subtrees" intersected by the intervals.

Definition 5.2.1 (Subtree)**.** Let p be a binary vector of length l and $n \geq l$. Then *level l p-subtree* is the interval $[p\,0^{\{n-l\}}, p\,1^{\{n-l\}}]$.

Clearly there are 2^l level l subtrees that partition the full interval $[0^{\{n\}}, 1^{\{n\}}]$.

The requirements 1 and 2 force each of the intervals $[a_1, b_1]$ and $[a_k, b_k]$ to intersect exactly 1 level l subtree (the $0^{\{l\}}$-subtree and the $1^{\{l\}}$-subtree respectively).

The requirement 3 forces each of the intervals $[a_i, b_i]$ for $i \in \{2, \dots, k-1\}$ to intersect exactly 2 level l subtrees (the p_{a_i}-subtree and the $(p_{a_i} + 1)$-subtree).

The requirement 4 forces each adjacent pair of intervals to intersect adjacent level l subtrees and not to intersect a common subtree. Since by requirements 1 and 2 the $0^{\{l\}}$-subtree and the $1^{\{l\}}$-subtree are intersected by some intervals, each of the level l subtrees is intersected by some interval.

Since there are 2^l level l subtrees, each of them is intersected by exactly one interval, the first and the k-th interval intersect one subtree each and the remaining intervals intersect 2 subtrees each, there are $2 + \frac{2^l - 2}{2} = 2^{l-1} + 1 = k$ intervals in total.

Equivalently, if we are looking for a "hard" function with at least k proper intervals, we can construct it using the parameter $l = \lceil \log_2 (k-1) + 1 \rceil$.

5.2.4 Interval decomposition spanning set size

Refer by number.

Let $f_l^n = f^n_{[a_1,b_1],\dots,[a_k,b_k]}$ be a k-interval function constructed using the definition shown in the previous section. In this section we will show the size of the spanning set returned by the interval decomposition algorithm when run on this function. We will denote its size with $id(f_l^n)$.

The interval decomposition algorithm spans each of the intervals separately using the optimization algorithm from Chapter 2, so $id(f_l^n) = \sum_{i=1}^{k} |\mathcal{T}_i|$, where $\mathcal{T}_i$ is the spanning set of the interval $[a_i, b_i]$ returned by the optimization algorithm for 1-interval functions.

For every i, we will analyse how the 1-interval algorithm solves the instance $[a_i, b_i]$. We will divide the analysis between the intervals such that $p_{b_i} = p_{a_i}$ and the ones such that $p_{b_i} = p_{a_i} + 1$.

$i \in \{1, k\}$

In case $i \in \{1, k\}$, the endpoints have common prefixes of length l ($p_{b_i} = p_{a_i}$). Thus the instance $[a_i, b_i]$ is reduced to the instance $[s_{a_i}, s_{b_i}] = [0^{\{n-l-1\}}1, 1^{\{n-l-1\}}0]$.

Recall that $n \geq l + 2$.

If $n = l + 2$, the instance $[0^{\{n-l-1\}}1, 1^{\{n-l-1\}}0] = [01, 10]$ is spanned using the procedure shown in Section 2.3.1. This procedure spans the two trivial subintervals $[01, 01]$ and $[10, 10]$ separately, producing $2 = n - l$ ternary vectors.

If $n \geq l+3$, the instance $[0^{\{n-l-1\}}1, 1^{\{n-l-1\}}0]$ is spanned using the procedure shown in Section 2.3.4. This procedure reduces the instance to the instance $[01, 10]$, which again contributes 2 ternary vectors, and adds $n-l-2$ extra ternary vectors that span the sub-interval $[0^{\{n-l-2\}}10, 1^{\{n-l-2\}}01]$. The spanning set of the interval in this case has the size $2+n-l-2 = n-l$.

We conclude that in case $i \in \{1, k\}$, $|\mathcal{T}_i| = n-l$.

$i \in \{2, \ldots, k-1\}$

In case $i \in \{2, \ldots, k-1\}$, recall that $p_{b_i} = p_{a_i} + 1$. This means that $p_{a_i} = c01^{\{m\}}$ and $p_{b_i} = c10^{\{m\}}$ for some binary vector c and number $m \in \{0, \ldots, l-1\}$.

To see that $m \geq 1$, recall that p_{a_i} must be odd.

This means that a_i and b_i start with a common prefix c followed by the bits 01 in a_i and 10 in b_i. In the 1-interval optimization algorithm, the common prefix c is left out and then the procedure shown in Section 2.3.1 is applied to the remainder of the endpoints (that is $[011^{\{m-1\}}s_{a_i}, 100^{\{m-1\}}s_{b_i}]$). This procedure spans the two sub-intervals $[1^{\{m\}}s_{a_i}, 1^{\{m+n-l\}}]$ and $[0^{\{m+n-l\}}, 0^{\{m\}}s_{b_i}]$ separately using the suffix and prefix procedure respectively.

Let's consider the sub-interval $[0^{\{m+n-l\}}, 0^{\{m\}}s_{b_i}]$; the other one is symmetric.

Note that this sub-interval can be immediately reduced to $[0^{\{n-l\}}, s_{b_i}]$ by removing the common prefix of the endpoints.

Recall that the prefix algorithm produces one ternary vector for each bit that is set to 1 in $s_{b_i} + 1 = 1^{\{n-l\}}$. The returned spanning set of $[0^{\{n-l\}}, s_{b_i}]$ thus has the size $n-l$. Prepending the prefix p_{b_i} we get a spanning set of $[p_{b_i}0^{\{n-l\}}, p_{b_i}s_{b_i}]$.

We span the complementary sub-interval $[p_{a_i}s_{a_i}, p_{a_i}1^{\{n-l\}}]$ in a symmetric manner, getting another spanning set of size $n-l$.

Joining the sub-interval spanning sets together we get the spanning set $\mathcal{T}_i$ of size $2(n-l)$.

Putting the interval spanning sets together, we get:

$$\begin{aligned}
id(f_l^n) &= \sum_{i=1}^{k} |\mathcal{T}_i| \\
&= 2(n-l) + \sum_{i=2}^{k-1} 2(n-l) \\
&= 2(n-l) + (k-2)2(n-l) \\
&= 2(n-l)(k-1) \\
&= 2(n-l)((2^{l-1}+1)-1) \\
&= 2^l(n-l)
\end{aligned}$$

5.2.5 Minimum spanning set size

In this section we will show a spanning set of f_l^n of size $n+l-2$ and prove its optimality ($dnf(f_l^n) = n+l-2$).

Lemma 5.2.1. $dnf(f_l^n) \leq n+l-2$

Proof. We will show a spanning set $\mathcal{T}$ of f_l^n of size $n+l-2$.

We will span the union of the intervals $[p0^{\{n-l-1\}}1, p1^{\{n-l-1\}}0]$ for all $p \in \{0,1\}^l$ using $n-l$ ternary vectors and then we will add some extra vectors to span the remaining true points.

First observe that all the false points of f_l^n end with either $0^{\{n-l\}}$ or $1^{\{n-l\}}$. This is clear from the definition of the "shape" endpoint suffixes $s_{a_i} = 0^{\{n-l-1\}}1$ and $s_{b_i} = 1^{\{n-l-1\}}0$ and the fact that the intervals intersect every level l subtree.

This means we need to span all the points that end with different $(n-l)$-bit suffixes. We can do that with the set $\mathcal{T}_{out} = \{\phi^{\{l\}}s | s \text{ is a cyclic shift of } 01\phi^{\{n-l-2\}}\} = \{\phi^{\{l\}}01\phi^{\{n-l-2\}}, \phi^{\{l\}}\phi01\phi^{\{n-l-3\}}, \ldots, \phi^{\{l\}}1\phi^{\{n-l-2\}}0\}$. To see that $\mathcal{T}_{out}$ spans all the points that do not end with $0^{\{n-l\}}$ or $1^{\{n-l\}}$, note that for every binary vector x that contains both a 0 and a 1 there is a position i such that $x^{[i]} = 0$ and $x^{[(i+1) \bmod (n-l)]} = 1$. Thus we have spanned the union of the intervals $[p0^{\{n-l-1\}}1, p1^{\{n-l-1\}}0]$ for all $p \in \{0,1\}^l$ using $n-l$ ternary vectors.

Observe that the true points that end with $1^{\{n-l\}}$ are exactly all the points $p_{a_i}1^{\{n-l\}}$ for some $i \in \{2, \ldots, k-1\}$ and that all p_{a_i} are odd numbers, which means they end with a 1. In fact, p_{a_i} for $i \in \{2, \ldots, k-1\}$ are all the l-bit odd numbers except $1^{\{l\}}$. We will employ the technique of cyclic shifts again to span these points efficiently.

Let $\mathcal{T}_1 = \{c11^{\{n-l\}} | c \text{ is a cyclic shift of } 0\phi^{\{l-2\}}\}$. To see that $\mathcal{T}_1$ spans all the true points that end with $1^{\{n-l\}}$, note that the cyclic shifts of $0\phi^{\{l-2\}}$ span exactly all the $(l-1)$-bit numbers that contain a 0 and the vectors $c1$ where c is a cyclic shift of $0\phi^{\{l-2\}}$ span exactly all the l-bit *odd* numbers that contain a 0. Clearly $|\mathcal{T}_1| = l-1$.

Similarly we construct $\mathcal{T}_0$ of size $l-1$ that spans all the true points that end with $0^{\{n-l\}}$.

Putting the partial spanning sets together, we get the size of a spanning set of f_l^n:

$$\begin{aligned}|\mathcal{T}| &= |\mathcal{T}_{out}| + |\mathcal{T}_0| + |\mathcal{T}_1| \\ &= (n-l) + (l-1) + (l-1) \\ &= n+l-2\end{aligned}$$

□

Having shown a spanning set of f_l^n of size $n+l-2$, we conclude that $dnf(f_l^n) \leq n+l-2$.

Lemma 5.2.2. *$dnf(f_l^n) \geq n+l-2$*

Proof. We will show that f_l^n can not be spanned using fewer than $n+l-2$ ternary vectors by showing an orthogonal set of f_l^n of size $n+l-2$. By section 1.6.2, this proves that all the spanning sets of f_l^n have the size at least $n-l-2$.

The orthogonal set V consists of 3 components:

- $V_{out} = \{0^{\{l\}}s | s \text{ is a cyclic shift of } 10^{\{n-l-1\}}\}$
- $V_0 = \{c00^{\{n-l\}} | c \text{ is a cyclic shift of } 10^{\{l-2\}}\}$
- $V_1 = \{c11^{\{n-l\}} | c \text{ is a cyclic shift of } 01^{\{l-2\}}\}$

To see that the sets V_{out}, V_0 and V_1 are pairwise disjoint, consider any $v_{out} \in V_{out}$, $v_0 \in V_0$ and $v_1 \in V_1$. The vector v_1 differs from both v_{out} and v_0 in the l-th position. The vector v_0 differs from v_{out} in the position $i \in \{1, \dots, l-1\}$ such that $v_0^{[i]} = 1$.

Since the sets V_{out}, V_0 and V_1 are pairwise disjoint, their union V has the size $|V_{out}| + |V_0| + |V_1| = (n-l) + (l-1) + (l-1) = n + l - 2$.

To see that V is orthogonal with respect to f_l^n, we need to show that any pair of different vectors in V cannot be spanned by a single ternary vector T. Let $x, y \in V$ and $x \neq y$. Let T span both x and y. We shall consider 6 possible combinations one by one:

- $x, y \in V_{out}$: T spans the false point $0^{\{n\}}$.
- $x, y \in V_0$: T spans the false point $0^{\{n\}}$.
- $x, y \in V_1$: T spans the false point $1^{\{n\}}$.
- $x \in V_{out}, y \in V_0$: T spans the false point $0^{\{n\}}$.
- $x \in V_{out}, y \in V_1$: T spans the false point $0^{\{l\}}1^{\{n-l\}}$.
- $x \in V_0, y \in V_1$: T spans the false point $x^{[1,l]}y^{[l+1,n]} = x^{[1,l-1]}01^{\{n-l\}}$. To see that it is a false point, recall that all the true points that end with $1^{\{n-l\}}$ have an *odd* prefix of length l.

Thus we have found an orthogonal set V of f_l^n of size $n + l - 2$. We conclude that $dnf(f_l^n) \geq n + l - 2$. □

5.2.6 Approximation ratio on hard functions

In the previous section, we have shown that $dnf(f_l^n) = n + l - 2$. We can use this information to prove that the approximation ratio of the interval decomposition algorithm is close to $2k$ for large k.

The approximation ratio of the interval decomposition algorithm on the function f_l^n, denoted $approx_{ID}(f_l^n)$, can be computed like this:

$$approx_{ID}(f_l^n) = \frac{id(f_l^n)}{dnf(f_l^n)} = \frac{2^l(n-l)}{n+l-2}$$

Let $HARD(l) = \{f_l^n | n \geq l + 2\}$ be the set of all "hard" functions constructed using "position" prefixes of length l. Note that all of these functions are $(2^{l-1}+1)$-interval, which means $HARD(l) \subseteq INT(2^{l-1} + 1)$.

First see that for a fixed l and large n, the approximation ratio converges to 2^l:

$$\begin{aligned} approx_{ID}(HARD(l)) &= \sup_{f \in HARD(l)} \frac{id(f)}{dnf(f)} \\ &= \sup_{n \geq l+2} \frac{id(f_l^n)}{dnf(f_l^n)} \\ &= \sup_{n \geq l+2} \frac{2^l(n-l)}{n+l-2} \\ &= 2^l \end{aligned}$$

To see that for any $l \geq 1$ the number 2^l really is the supremum of the given sequence, note that $\frac{n-l}{n+l-2} = 1 - \frac{2l-2}{n+l-2} \leq 1$ for all $n \geq l+2$ and that $\lim_{n\to\infty} \frac{2^l(n-l)}{n+l-2} = \lim_{n\to\infty} \frac{2^l n}{n} = 2^l$.

Since 2^l is the supremum of the approximation ratio for large n, for any $l \geq 1$ and a number $a < 2^l$, we can find n such that $id(f_l^n) \geq a \cdot dnf(f_l^n)$, which corresponds to finding an instance on which the interval decomposition algorithm performs at least a times worse than the optimum.

Example 5.1. For example, let's try to find a 3-interval function f that can be spanned using as few as $dnf(f)$ ternary vectors while the interval decomposition algorithm uses strictly more than $3 \cdot dnf(f)$ vectors to span f.

We will find a combination of l and n such that f_l^n is a 3-interval function and $id(f_l^n) > 3 \cdot dnf(f_l^n)$.

¿ Use the inverse mapping ($k \to l$).

Recall that $k = 2^{l-1}+1$, so we need to use $l = 2$ in order for f_l^n to be 3-interval.

¿ Why?

Now let's choose the parameter a. It needs to satisfy the inequalities $a > 3$ and $a < 4 = 2^l$. Let's take for example $a = 3.5 = \frac{7}{2}$.

We need to find n such that:

$$\begin{aligned} id(f_l^n) &= 2^l(n-l) \\ &= 2^2(n-2) \\ &= 4(n-2) \\ &= 4n-8 \\ &\geq \frac{7}{2}n \\ &= \frac{7}{2}(n+2-2) \\ &= a(n+l-2) \\ &= a \cdot dnf(f_l^n) \end{aligned}$$

Any $n \geq 16$ satisfies the inequality:

$$\begin{aligned} 4n-8 &\geq \frac{7}{2}n \\ 8n-16 &\geq 7n \\ n &\geq 16 \end{aligned}$$

We conclude, for example for $n = 16$:

$$\begin{aligned} id(f_2^{16}) &= 2^2(16-2) \\ &= 56 \\ &> 48 \\ &= 3 \cdot (16-2+2) \\ &= 3 \cdot dnf(f_2^{16}) \end{aligned}$$

which means that the interval decomposition algorithm uses strictly more than 3-times as many ternary vectors to span the 3-interval function f_2^{16} as the optimum.

Note that this example namely shows that the interval decomposition algorithm is not k-approximative in general, since it is not 3-approximative on the 3-interval function f_2^{16}.

5.2.7 Approximation ratio lower bound on k-interval functions

In Example 5.1, we have shown a k-interval function on which the interval decomposition algorithm performs strictly more than k times worse than optimum. More specifically, the algorithm produces a spanning set of a ($k = 3$)-interval function that is at least $\frac{7}{2} = \frac{7}{6}3 = \frac{7}{6}k$ times as large as optimum. On 3 intervals, we could get this error ratio arbitrarily close to $\frac{4}{3}k$ by choosing a sufficiently large n. We will show that if we allow using more intervals (that is larger k), we can get the error ratio arbitrarily close to $2k$.

For every k such that there is an $l \geq 1$ such that $k = 2^{l-1} + 1$, we know that the respective l is equal to $\log_2(k-1) + 1$. Since $approx_{ID}(HARD(l)) = 2^l$ for all l, $approx_{ID}(INT(k)) \geq 2^{\log_2(k-1)+1} = 2(k-1)$ for all such k. It is easy to see that $2(k-1)$ converges to $2k$ for large k.

In other words, for any $b < 2$ we can find a k-interval function on which the interval decomposition algorithm performs at least bk times worse than the optimum.

For demonstration, Table 5.1 shows the limits of the approximation ratio on f_l^n for some small values of l.

l	k	$id(f_l^n)$	$dnf(f_l^n)$	$approx_{ID}(HARD(l))$	$\frac{approx_{ID}(HARD(l))}{k}$
1	2	$2n-4$	$n-1$	2	1
2	3	$4n-8$	n	4	$\frac{4}{3}$
3	5	$8n-24$	$n+1$	8	$\frac{8}{5}$
4	9	$16n-64$	$n+2$	16	$\frac{16}{9}$

Table 5.1: Limits of the approximation ratio on f_l^n for some small l

5.3 Conclusion

We have shown an approximation algorithm for the problem of minimization of k-interval Boolean functions. We continued to analyse its approximation ratio. An upper bound $2k$ on the approximation ratio directly follows from the analysis of the suffix-prefix decomposition algorithm presented in chapter 4. A lower bound $2^l = 2(k-1)$ for $k = 2^{l-1} + 1$ for every $l \geq 1$ follows from the analysis of the algorithm's performance on a special class of "hard" functions.

Conclusion

Make a section about disjoint representations, collect (review) the results from the thesis. Can be done in Conclusion. When we discuss Scheiber's results, mention disjoint dnfs and the results - Sheiber has two algs. and results. Motivation: generating test data for software testing - esp. with disjoint dnfs - uniform polling

We have shown that with respect to finding a minimum DNF representation, 2-switch 2-interval functions can be reduced to 1-interval functions.

Then we showed that for every $l \geq 3$, there is an l-switch function that is not coverable, extending Dubovský's result about 3-switch functions. This suggests that possible future algorithms that optimally span multi-interval functions are going to require a different approach for proving optimality.

To remedy the difficulty with spanning multi-interval functions, we introduced a $2k$-approximation algorithm for minimizing DNF representations of k-interval functions. The algorithm naturally extends Schieber et al.'s suffix-prefix approximation algorithm for disjoint spanning sets of 1-interval functions [5, section 6]. Our algorithm produces disjoint spanning sets and has the approximation ratio of $2k$ for the problem of finding a minimum disjoint representation as well.

Some interesting questions are still left to be answered regarding multi-interval Boolean functions. Since general Boolean minimization is Σ_2^p-hard [6], there must be some k such that minimization of k-interval functions is Σ_2^p-hard, and a k' for which minimization is NP-hard. Schieber et al. showed that such $k' > 1$, which is the only bound shown so far. There is a substantial difference between 1- and 2-interval functions since the former are coverable in general, while the latter are not. This prevents us from using a similar approach to minimizing 2-interval functions. I suspect this difference could correspond to a difference in computational complexity of the problem.

No – there may be an approximation scheme. For example, if the complexity is n^k, there's no k for which the problem is hard, while the problem remains hard in general.

Is it a good idea to "suspect" in a thesis?

¿ Suspect more explicitly, e.g. 3-switch functions are NP-hard.

The $2k$-approximation algorithm presented in this thesis is constructed so that we can use the orthogonal sets for proving the approximation ratio. There may be a better approximation algorithm for k-interval functions and searching for it could provide an interesting insight in proving lower bounds for minimum DNF representation size of the functions in question.

We may have shown a better one in `betterapprox`.

Bibliography

[1] Endre Boros, Ondřej Čepek, Alexander Kogan, and Petr Kučera. Exclusive and essential sets of implicates of boolean functions. *Discrete Applied Mathematics*, 158(2):81 – 96, 2010. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2009.08.012. Available from: `http://www.sciencedirect.com/science/article/pii/S0166218X09003394`.

[2] Jakub Dubovský. A construction of minimum DNF representations of 2-interval functions. Master's thesis, Charles University in Prague, 2012. Available from: `https://is.cuni.cz/webapps/zzp/detail/116613/`.

[3] Radek Hušek. Properties of interval boolean functions. Master's thesis, Charles University in Prague, 2014. Available from: `https://is.cuni.cz/webapps/zzp/detail/145114/`. [In Czech].

[4] Radek Hušek. personal communication, April 2015.

[5] Baruch Schieber, Daniel Geist, and Ayal Zaks. Computing the minimum DNF representation of boolean functions defined by intervals. *Discrete Applied Mathematics*, 149(1–3):154 – 173, 2005. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2004.08.009. Available from: `http://www.sciencedirect.com/science/article/pii/S0166218X05000752`.

[6] Christopher Umans. The minimum equivalent DNF problem and shortest implicants. In *Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on*, pages 556–563. IEEE, Nov 1998. doi: 10.1109/SFCS.1998.743506. Available from: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=743506`.

[7] Ondřej Čepek, Petr Kučera, and Petr Savický. Boolean functions with a simple certificate for CNF complexity. *Discrete Applied Mathematics*, 160(4–5):365 – 382, 2012. ISSN 0166-218X. doi: http://dx.doi.org/10.1016/j.dam.2011.05.013. Available from: `http://www.sciencedirect.com/science/article/pii/S0166218X11001958`.

Capitalize "Boolean" in bibliography properly.

Umans [6] should have pages listed after year.

List of Tables

List of Abbreviations

CNF	conjunctive normal form. 9
DNF	disjunctive normal form. ii, 1, 3, 4, 6–10, 18, 38

Todo list