STK3405 – Lecture 5

A. B. Huseby & K. R. Dahl

Department of Mathematics
University of Oslo, Norway

Section 3.3

Modules of monotone systems

Modules of monotone systems

If $A \subseteq C$, then the complement set of A, i.e., $C \setminus A$, is denoted by $\bar{A}$. We also let $\boldsymbol{x}^A$ and $\boldsymbol{x}^{\bar{A}}$ denote the subvectors of $\boldsymbol{x}$ corresponding to the sets A and $\bar{A}$ respectively.

Definition

Let (C, ϕ) be a binary monotone system, and $A \subseteq C$. The monotone system (A, χ) is a *module* of (C, ϕ) if and only if the structure function ϕ can be written as:

$$\phi(\boldsymbol{x}) = \psi(\chi(\boldsymbol{x}^A), \boldsymbol{x}^{\bar{A}}), \quad \text{for all } \boldsymbol{x} \in \{0, 1\}^n,$$

where ψ is a monotone structure function. The set A is called a *modular set* of (C, ϕ).

Example: Series and parallel components

Example (Series and parallel components)

Let (C, ϕ) be a binary monotone system, and let $i, j \in C$.

We say that i and j are *in series* if ϕ depends on the component state variables, x_i and x_j, only through the product $x_i \cdot x_j$.

Thus, if i and j are *in series*, then (A, χ), where $A = \{i, j\}$ and $\chi(x_i, x_j) = x_i \cdot x_j$, is a *module* of (C, ϕ).

We say that i and j are *in parallel* if ϕ depends on the component state variables, x_i and x_j, only through the coproduct $x_i \amalg x_j$.

Thus, if i and j are *in parallel*, then (A, χ), where $A = \{i, j\}$ and $\chi(x_i, x_j) = x_i \amalg x_j$, is a *module* of (C, ϕ).

Modules of monotone systems

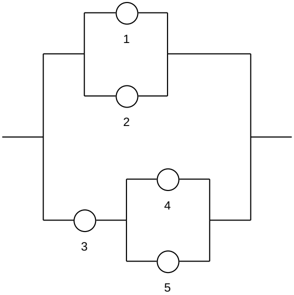
Definition

A *modular decomposition* of a monotone system (C, ϕ) is a set of modules $\{(A_j, \chi_j)\}_{j=1}^r$ connected by a binary monotone organisation structure function ψ. The following conditions must be satisfied:

- $C = \bigcup_{j=1}^r A_j$, and $A_j \cap A_k = \emptyset$ for $j \neq k$.
- $\phi(\boldsymbol{x}) = \psi[\chi_1(\boldsymbol{x}^{A_1}), \ldots, \chi_r(\boldsymbol{x}^{A_r})]$.

We observe that a modular decomposition is a disjoint partition of the component set into modules such that the structure function of the whole system is a function of the structure functions of these modules.

Modules of monotone systems (cont.)



Modules: (A_1, χ_1) and (A_2, χ_2) where $A_1 = \{1, 2\}$ and $A_2 = \{3, 4, 5\}$, and:

$$\chi_1(x_1, x_2) = x_1 \amalg x_2,$$

$$\chi_2(x_3, x_4, x_5) = x_3 \cdot (x_4 \amalg x_5)$$

$$\psi(\chi_1, \chi_2) = \chi_1 \amalg \chi_2$$

Section 3.4

Dynamic system analysis

Dynamic system analysis

Let (C, ϕ) be a binary monotone system, and introduce for $t \geq 0$:

$$X_i(t) = \text{ the state of component } i \text{ at time } t,\ i \in C,$$

$$\phi(\boldsymbol{X}(t)) = \text{ the state of the system at time } t.$$

- $X_i(t)$ is a random variable (for any given t).
- $\{X_i(t) : t \geq 0\}$, is a stochastic process.
- $\phi(\boldsymbol{X}(t))$ is a random variable (for any given t).
- $\{\phi(\boldsymbol{X}(t)) : t \geq 0\}$ is a stochastic process.

We assume that the stochastic processes $\{X_i(t) : t \geq 0\}_{i=1}^n$ are independent.

Dynamic system analysis (cont.)

We also introduce:

$$p_i(t) = P(X_i(t) = 1) = \text{ The reliability of component } i \text{ at time } t,$$

$$h(\boldsymbol{p}(t)) = P(\phi(\boldsymbol{X}(t)) = 1) = \text{ The reliability of the system at time } t.$$

We assume that the components cannot be repaired and let:

$$T_i = \text{ The lifetime of component } i,$$

$$S = \text{ The lifetime of the system.}$$

NOTE:

$$P(X_i(t) = 1) = P(T_i > t), \quad i \in C,$$

$$P(\phi(\boldsymbol{X}(t)) = 1) = P(S > t).$$

Dynamic system analysis (cont.)

We denote the cumulative distribution of T_i by F_i, $i \in C$, and the cumulative distribution of ϕ by G. We then have the following relations:

$$p_i(t) = P(X_i(t) = 1) = P(T_i > t) = 1 - F_i(t) =: \bar{F}_i(t), \quad i \in C,$$

$$h(t) = P(\phi(\boldsymbol{X}(t)) = 1) = P(S > t) = 1 - G(t) =: \bar{G}(t).$$

NOTE: Determining the lifetime distribution for the system is the same as finding the reliability of the system at time t, i.e., $h(t)$, for all time $t \geq 0$, and then letting $G(t) = 1 - h(t)$.

Dynamic system analysis (cont.)

Theorem

For a monotone system (C, ϕ) with minimal path sets $P_1, \ldots, P_p$ and minimal cut sets $K_1, \ldots, K_k$ we have:

$$S = \begin{cases} \max_{1 \leq j \leq p} \min_{i \in P_j} T_i \\ \min_{1 \leq j \leq k} \max_{i \in K_j} T_i \end{cases}$$

PROOF: The lifetime of the system equals the lifetime of the minimal path series structure which lives the longest.

The lifetime of a minimal path series structure equals the lifetime of the shortest living component in this path set.

The second equality can be proved similarly.

Chapter 4

Exact computation of reliability of binary monotone systems

Computational complexity

Let:

$$
\begin{aligned}
n &= \text{The size of the problem (e.g., number of components)} \\
t(n) &= \text{The worst case running time of the algorithm as a function of } n \\
f(n) &= \text{Some known non-negative increasing function of } n
\end{aligned}
$$

The order of the algorithm is said to be $O(f(n))$ if and only if there exists a positive constant M and a positive integer n_0 such that:

$$t(n) \leq Mf(n), \text{ for all } n \geq n_0.$$

If f is a polynomial in n, we say that the algorithm is a *polynomial time* algorithm, while if f is an exponential function of n, we say that the algorithm is an *exponential time* algorithm.

Computational complexity (cont.)

- NP (for nondeterministic polynomial time) is a complexity class used to describe certain types of problems.
- NP contains many important problems, the hardest of which are called *NP-complete* problems.
- Open question: Is it possible to find a polynomial time algorithm for solving NP-complete problems. Conjecture: *NO*.
- The class of *NP-hard* problems is a class of problems that are, informally, *at least as hard as the hardest problems in NP*.
- The problem of computing the reliability of a binary monotone system is known to be NP-hard in the general case.

Computational complexity (cont.)

EXAMPLE: In order to calculate the reliability of k-out-of-n system we need to do:

- $2 \cdot (2 + 3 + \cdots + n) = (n+2)(n-1)$ multiplications
- $1 + 2 + \cdots (n-1) = \frac{n(n-1)}{2}$ additions

Thus, we have:

$$
\begin{aligned}
t(n) &= (n+2)(n-1) + \frac{n(n-1)}{2} \\
&= \frac{3}{2}n^2 + \frac{1}{2}n - 2 \leq 2n^2
\end{aligned}
$$

This shows that the reliability of a k-out-of-n system can be calculated in $O(n^2)$ time.

Threshold systems

A *threshold system* is a binary monotone system (C, ϕ), where the structure function has the following form:

$$\phi(\boldsymbol{x}) = I(\sum_{i=1}^{n} a_i x_i \geq b),$$

where $a_1, \ldots, a_n$ and b are non-negative real numbers, and $I(\cdot)$ denotes the indicator function, i.e., a function defined for any event A which is 1 if A is true and zero otherwise.

NOTE: If $a_1 = \cdots = a_n = 1$ and $b = k$, the threshold system is reduced to a k-out-of-n system. Thus, threshold systems are a generalisation of k-out-of-n systems.

It can be shown that calculating the reliability of a threshold system in general is NP-hard.

Threshold systems (cont.)

Let (C, ϕ) a threshold system where $a_1, \ldots, a_n$ and b are positive integers, and introduce:

$$S_j = \sum_{i=1}^{j} a_i X_i, \quad j = 1, 2, \ldots, n.$$

By the assumptions it follows that $S_1, \ldots, S_n$ are integer valued stochastic variables.

Thus, the generating function for S_j, i.e., $G_{S_j}(y) = E[y^{S_j}]$ is a polynomial, and the distribution of S_j can be derived directly from the coefficients of $G_{S_j}(y)$, $j = 1, \ldots, n$.

Threshold systems (cont.)

We also introduce:

$$d_j = \sum_{i=1}^{j} a_i, \quad j = 1, 2, \ldots, n.$$

It follows that:

$$\deg(G_{S_j}(y)) = d_j, \quad j = 1, 2, \ldots, n.$$

Assuming $G_{S_j}(y)$ has been calculated, we can find $G_{S_{j+1}}(y)$ as:

$$G_{S_{j+1}}(y) = G_{S_j}(y) \cdot G_{a_{j+1} X_{j+1}}(y)$$

In the worst case this would require $2(d_j + 1)$ multiplications and d_j additions.

Threshold systems (cont.)

EXAMPLE: Assume that $a_j = 2^{j-1}$, $j = 1, \ldots, n$. We then have:

$$\deg(G_{S_j}(y)) = d_j = \sum_{i=1}^{j} 2^{i-1} = 2^j - 1, \quad j = 1, 2, \ldots, n.$$

In fact, in this case $G_{S_j}(y)$ consists of 2^j non-zero terms (including the constant term)!

Calculating $G_{S_{j+1}}(y)$ from $G_{S_j}(y)$ would require 2^{j+1} multiplications and $2^j - 1$ additions.

Thus, using generating functions for calculating the reliability of this threshold system takes $O(2^n)$ time.

Threshold systems (cont.)

EXAMPLE: Assume that $a_j \leq A$, $j = 1, \ldots, n$, where A is a fixed positive integer. We then have:

$$\deg(G_{S_j}(y)) = d_j \leq \sum_{i=1}^{j} A = Aj, \quad j = 1, 2, \ldots, n.$$

Calculating $G_{S_{j+1}}(y)$ from $G_{S_j}(y)$ would require at most $2(Aj + 1)$ multiplications and Aj additions.

Since A is a fixed constant, it follows that calculating the reliability of such a threshold system takes $O(n^2)$ time.