Knapsack

KNAPSACK is a problem which generalises many natural scheduling and optimisation problems, and through reductions has been used to show many such problems \( \text{NP} \)-complete.

In the problem, we are given \( n \) items, each with a positive integer value \( v_i \) and weight \( w_i \).

We are also given a maximum total weight \( W \), and a minimum total value \( V \).

Can we select a subset of the items whose total weight does not exceed \( W \), and whose total value exceeds \( V \)?

Scheduling

Some examples of the kinds of scheduling tasks that have been proved \( \text{NP} \)-complete include:

Timetable Design

Given a set \( H \) of work periods, a set \( W \) of workers each with an associated subset of \( H \) (available periods), a set \( T \) of tasks and an assignment \( r : W \times T \rightarrow \mathbb{N} \) of required work, is there a mapping \( f : W \times T \times H \rightarrow \{0, 1\} \) which completes all tasks?

Sequencing with Deadlines

Given a set \( T \) of tasks and for each task a length \( l \in \mathbb{N} \), a release time \( r \in \mathbb{N} \) and a deadline \( d \in \mathbb{N} \), is there a work schedule which completes each task between its release time and its deadline?

Job Scheduling

Given a set \( T \) of tasks, a number \( m \in \mathbb{N} \) of processors a length \( l \in \mathbb{N} \) for each task, and an overall deadline \( D \in \mathbb{N} \), is there a multi-processor schedule which completes all tasks by the deadline?
Responses to NP-Completeness

Confronted by an NP-complete problem, say constructing a timetable, what can one do?

- It’s a single instance, does asymptotic complexity matter?
- What’s the critical size? Is scalability important?
- Are there guaranteed restrictions on the input? Will a special purpose algorithm suffice?
- Will an approximate solution suffice? Are performance guarantees required?
- Are there useful heuristics that can constrain a search? Ways of ordering choices to control backtracking?

Validity

We define VAL—the set of valid Boolean expressions—to be those Boolean expressions for which every assignment of truth values to variables yields an expression equivalent to true.

\[ \phi \in \text{VAL} \iff \neg \phi \not\in \text{SAT} \]

By an exhaustive search algorithm similar to the one for SAT, VAL is in \( \text{TIME}(n^2 2^n) \).

Is \( \text{VAL} \in \text{NP} \)?

Complementation

If we interchange accepting and rejecting states in a deterministic machine that accepts the language \( L \), we get one that accepts \( \overline{L} \).

If a language \( L \in \text{P} \), then also \( \overline{L} \in \text{P} \).

Complexity classes defined in terms of nondeterministic machine models are not necessarily closed under complementation of languages.

Define,

\( \text{co-NP} \) – the languages whose complements are in \( \text{NP} \).
Succinct Certificates

The complexity class \( \text{NP} \) can be characterised as the collection of languages of the form:

\[
L = \{ x \mid \exists y R(x, y) \}
\]

Where \( R \) is a relation on strings satisfying two key conditions

1. \( R \) is decidable in polynomial time.
2. \( R \) is \textit{polynomially balanced}. That is, there is a polynomial \( p \) such that if \( R(x, y) \) and the length of \( x \) is \( n \), then the length of \( y \) is no more than \( p(n) \).

\[
y \text{ is a certificate for the membership of } x \text{ in } L.
\]

Example: If \( L \) is \text{SAT}, then for a satisfiable expression \( x \), a certificate would be a satisfying truth assignment.

\text{co-NP}

As \text{co-NP} is the collection of complements of languages in \text{NP}, and \( \text{P} \) is closed under complementation, \text{co-NP} can also be characterised as the collection of languages of the form:

\[
L = \{ x \mid \forall y |y| < p(|x|) \rightarrow R'(x, y) \}
\]

\( \text{NP} \) – the collection of languages with succinct certificates of membership.

\( \text{co-NP} \) – the collection of languages with succinct certificates of disqualification.

Any of the situations is consistent with our present state of knowledge:

- \( \text{P} = \text{NP} = \text{co-NP} \)
- \( \text{P} = \text{NP} \cap \text{co-NP} \neq \text{NP} \neq \text{co-NP} \)
- \( \text{P} \neq \text{NP} \cap \text{co-NP} = \text{NP} = \text{co-NP} \)
- \( \text{P} \neq \text{NP} \cap \text{co-NP} \neq \text{NP} \neq \text{co-NP} \)