Parameter $k = \text{function: instance} \rightarrow \mathbb{N}$
- usually one of the numbers in instance
- sometimes hard to compute e.g. OPT

Parameterized problem = decision problem + parameter
- e.g. (k-)Vertex Cover: is there a vertex cover of \(\leq k\)?
  - \(k\) is the natural parameter: comparing with OPT
- e.g. Vertex Cover with respect to OPT (Vertex Cover)
  - similar but \(k\) not given
  - for \(k=0,1,2,\ldots\): run k-Vertex Cover
- e.g. Vertex Cover w.r.t. crossing number

\[ \text{XP} = \{ \text{parameterized problems solvable in } n^{f(k)} \text{ time} \} \]

Fixed-parameter tractable (FPT)
= \{ parameterized problems solvable in \( f(k) \cdot n^{O(1)} \) time \}
= \{ parameterized problems solvable in \( f(k) + n^{O(1)} \) time \}
- motivation: confine exponential to parameter \(k\) which may be \(<<\) problem size \(n\)

Example: (k-)Vertex Cover
- \(\text{XP}\): guess \(k\) vertices, test coverage \(|V|^{k} \cdot |E|\)
- \(\text{FPT}\): take edge, guess endpoint, delete, repeat \(2^k\) "bounded search tree technique" depth \(\leq k\)
**EPTAS in PTAS** with running time \( f(1/\epsilon) \cdot n^{O(1)} \)

- i.e. FPT w.r.t. \( 1/\epsilon \)

\[ \Rightarrow \text{FPT w.r.t. natural parameter } k \quad (\Rightarrow \text{w.r.t. } \text{OPT}) \]

- \( \epsilon \text{ FPT } \Rightarrow \epsilon \text{ EPTAS} \)

**Parameterized reduction:** \((A, k) \rightarrow (B, k')\)

- instance \( x \) of \( A \) \( \Rightarrow \) instance \( x'=f(x) \) of \( B \)

\[ f(k(x)) \cdot |x|^\omega \text{ time } \Rightarrow |x'| \leq f(k(x)) \cdot |x|^\omega \]

- answer preserving: \( x \text{ YES for } A \Leftrightarrow x' \text{ YES for } B \)

  \( \text{(just like NP/Karp reductions)} \)

- parameter preserving: \( k'(x') \leq g(k(x)) \)

  \( \text{for some } g: \mathbb{N} \rightarrow \mathbb{N} \)

- \( B \in \text{FPT } \Rightarrow A \in \text{FPT} \)

\[ \forall x \text{ parameter blowup} \]

**Nonexample:** independent set \( \rightarrow \) vertex cover

\((G, k) \quad \Rightarrow \quad (G, n-k)\)

- preserves answer but \underline{not parameter}

- indeed, vertex cover \( \in \text{FPT} \)

  but independent set is \( \text{W[1]} \)-hard

\[ \Rightarrow \epsilon \text{ FPT unless } \text{FPT=\text{W[1]}} \]

**Example:** independent set \( \rightarrow \) clique \( \quad \) (or vice versa)

\((G, k) \quad \Rightarrow \quad (\bar{G}, k)\)
Canonical hard problem for \( W[1] \): (analogy to \( \text{NP} \))
- \( k \)-step nondeterministic Turing machine
  - given nondeterministic Turing machine
    - code, state, finger to \( k \)-cell memory?
  - \( O(n) \) lines, \( \Theta(n) \) options, \( \Theta(n) \) states
  - (guess can have \( n \) choices/branches)
- does some choice sequence finish in \( k \) steps?

Reduction to Independent Set:
- \( k^2 \) cliques, \( k' = k^2 \Rightarrow 1 \) node per clique
- clique \( (i,j) \) represents memory cell \( i \) at time \( j \) (\( n \) choices) + state of machine
  - (e.g. PC=which of \( n \) instructions next)
- add edges to forbid certain transitions \( j \to j' \); omit edges for allowed nondet. trans.

Reduction from Independent Set: \( k' = \Theta(k^2) \)
- guess \( k \) vertices
  - \( \Theta(k) \)
- for each pair of these vertices:
  - \( \Theta(k^2) \)
    - check no edge (lookup table in code)

\( \Rightarrow \) both \( W[1] \)-complete
Clique in regular graphs: reduction from Clique
- $\Delta = \text{max. degree}$
- $\Delta$ copies of graph
- vertex $v$ of degree $d \Rightarrow v_1, v_2, \ldots, v_\Delta$ copies
  - add $\Delta - d$ vertices
  - biclique between $\&$
  $\Rightarrow \Delta$-regular
- add no cliques ($\geq 3$):
  new vertices in no $\Delta$

Independent set in regular graphs - just take complement

Partial vertex cover:
are there $k$ vertices that cover $l$ edges?
- FPT w.r.t. $l$
- $W[1]$-complete w.r.t. $k$

Reduction from Independent set in regular graphs:
- $k' = k$
- $l' = \Delta k$

**Multicolored clique:** — like (Numerical) 3DM
- given graph & vertex k-coloring
- find k vertices, one of each color, that form a k-clique
  [Fellows, Hermelin, Rosamond, Vialette - TCS 2009]

**Reduction from Clique:**
- vertex $v \rightarrow k$ copies $v_1, v_2, \ldots, v_k$
  colors: $1, 2, \ldots, k$
- edge $(v, w) \rightarrow$ edges $(v_i, w_j)$ $\forall i \neq j$
- $k' = k$
- k-clique $\iff$ k-colored k-clique

**Reduction to Clique:**
- nothing: coloring $\Rightarrow$ all cliques are multicolored

**Multicolored independent set** — just take complement
Shortest common supersequence:
- given \( k \) strings over alphabet \( \Sigma_i \) & number \( l \)
- is there a common supersequence of length \( l \)
- \( W[1] \)-hard w.r.t. \( k \) for \( |\Sigma| = 2 \) \([\text{Pietrzak-JCSS2003}]\)
- reduction from Multicolored Clique

Reduces to restricted form where input strings never repeat character twice in a row parameterized by \( k \) & \( \Sigma \)
- add new symbol \( s_i \) after every character in string \( i \) \( \Rightarrow \) no repeats
- \( k' = k \)
- \( |\Sigma'| = |\Sigma| + 1 + k \)
- \( l' = l + \text{total length of input strings} \)

Reduces to Flood-It on trees w.r.t. \# colors \( (|\Sigma|) \) & \# leaves \( (k) \)
Dominating set: (based on Cygan et al. book 2015)

Reduction from Multicolored independent set:
- vertex → vertex
- connect each color class in clique
- also add 2 dummy vertices to each clique
- \( k' = k \) ⇒ dominating set chooses one vertex from each clique, representing one vertex of each color in ind. set
- for each edge \((v, w)\):
  - add vertex connected to all vertices in color classes of \( v \) & \( w \), except \( v \) & \( w \)
  ⇒ dominated \( ⇔ \) \( v \) & \( w \) not both chosen (i.e. independent set)

\[ \Rightarrow W[1]-\text{hard} \]
\[ \wedge W[2]-\text{complete in fact} \]
\[ \nRightarrow \notin \text{FPT unless } \text{FPT} = W[2] \] (weaker assumption)
\[ \Rightarrow \text{reverse reduction impossible unless } W[1]=W[2] \]

Reduction to Set Cover: same as \( L_{11} \)
- vertex \( v \) → set \( N(v) \cup \exists v \exists \)
- \( k' = k \)
**Weighted Circuit SAT** (Circuit k-Ones)
- given acyclic Boolean circuit & parameter k
- can we set k inputs to 1 to get output = 1?

\[ W[P] = \{ \text{parameterized problems reducible to Weighted Circuit SAT} \} \]

- \( \text{depth} = \text{longest input}\rightarrow\text{output path} \)
- \( \text{weft} = \max \# \text{big gates on input}\rightarrow\text{output path} \)
  
  \( \text{not } O(1) \text{ inputs: e.g. } \geq 3 \text{ inputs} \)

\[ W[t] = \{ \text{parameterized problems reducible to } O(1)\text{-depth weft}-t \text{ Weighted Circuit SAT} \} \]

\[ = \{ \text{parameterized problems reducible to depth}-t \text{ output=AND Weighted Circuit SAT} \} \]

\[ W[*] = W[O(1)] \]

- \( W[1]\text{-complete} \)
  - weighted \( O(1) \)-SAT  
    \( \text{(big AND of small ORs)} \)

- \( W[2]\text{-complete} \)
  - weighted CNF-SAT  
    \( \text{(big AND of big ORs)} \)
  - k-step 2-finger nondeterministic Turing machine
    \[ = 2\text{-tape} \]

- \( W[\text{SAT}] = \text{reducible to SAT} \)
  - SAT \( \rightarrow \) CNF-SAT reduction adds extra vars.
  - so weighted problems not the same