An overview of Knowledge Compilation
Lower bounds, *December 08, 2017.*

1 Parametrized problems and FPT Algorithms

Flum and Grohe, *Parametrized Complexity Theory.*

1.1 Preliminaries

Definition 1.
- *A* parametrisation *is a ptime computable function* $k : \Sigma^* \to \mathbb{N}$.
- *A* parametrised problem *is a pair* (L, k) *of* $L \subseteq \Sigma^*$ *and* k *a parametrisation.*

Example 1.
- *SAT and the number of variables*
- *Clique* (G, k) *and* k
- *SAT and the number of variable per clauses...*

Definition 2.
- FPT-time algorithm wrt k *is an algorithm that decides on input* x *if* $x \in L$ *and runs in time* $f(k(x))\mathsf{poly}(|x|)$ *for a computable function* f.
- (L, k) *is* fixed parameter tractable *(FPT) if there exists an FPT-time algorithm wrt* k *for* (L, k)

Example 2.

SAT and number of variables

co-example : colorability

1.2 Example : vertex covers

Definition 3. *Let* $G = (V, E)$ *be a graph. A vertex cover is a set* $S \subseteq V$ *such that for every* $e \in E$, $S \cap e \neq \emptyset$.

Deciding given G and k if there exists a vertex cover of size $\leq k$ if a well-known NP-complete problem. However, it is FPT in k:

- If $|E| = 0$, return true.
- If $k = 0$, return false.
- Otherwise, choose $e = \{u, v\} \in E$. Return $VC(G \setminus \{u\}, k-1) \vee VC(G \setminus \{v\}, k-1)$.

At most 2^k calls of the function. And each call is polynomial in $|G|$.

Independent sets. Can we use this idea to detect independent sets in FPT time?

Definition 4. *Let $G = (V, E)$ be a graph. An independent set is a set $S \subseteq V$ such that for every $u, v \in S$, $\{u, v\} \notin E$.*

Lemma 5. *Let $G = (V, E)$ be a graph and $S \subseteq V$. S is a vertex cover iff $V \setminus S$ is an independent set.*

Previous algorithm gives an algorithm with runtime 2^{n-k} for Independent Sets. It is not FPT. But IS parametrized by $n - k$ is FPT.

2 Treewidth

Give intuition: how to measure the "distance between a graph and a tree".

2.1 Definition and examples

Definition 6. *A tree decomposition of a graph $G = (V, E)$ is a tree T and a labelling $B_t \subseteq V$ for every $t \in V(T)$ such that:*

- *for every $x \in V$, $\{t \mid x \in B_t\}$ is connected,*
- *for every $e \in E$, there exists t such that $e \subseteq B_t$.*

2.2 Treewidth of well-known graphs

Treewidth of trees.

Theorem 7. *A graph is a forest if and only if it has treewidth* 1.

Proof of the other way relies on:

Lemma 8. *If $H \subseteq G$ then $tw(H) \leq tw(G)$.*

Proof. Remove vertices of $V(G)$ from a decomposition of G. □

Treewidth of cycle. Cycle have treewidth 2.

Treewidth of clique. K_k is of treewidth $k - 1$.
Project examples. Proof of the lower bounds:

Lemma 9. *Let G be a graph and d be the minimal degree of its vertices. Then $tw(G) \geq d$.*

Proof. Let T be a tree decomposition of G of treewidth k. We claim that there exists a vertex $v \in V$ of degree k. Indeed, let t be a leaf of T with father t'. If $B_t \subseteq B_{t'}$ then we can remove t from T and still have a tree decomposition of G of treewidth k. Do this until you cannot any more to compute a new tree decomposition T' of G of treewidth k. Now let t be a leaf of T' and father t'. By definition, there exists $x \in B_t \setminus B_{t'}$. Since t is a leaf, x only appear in B_t by connectivity. Thus, every edge $\{x, y\} \subseteq B_t$, ie the degree of x is $\leq k$. □

Treewidth of grids.

Theorem 10. *Let G be a $n \times m$ grid with $n \leq m$. Then $tw(G) \leq m$. And $tw(G) \geq m/3$.*

3 Formulas of bounded treewidth

3.1 Graphs and formulas

Primal/Incidence graphs. On slides.

3.2 Primal treewidth

Solve $\#SAT$ for bounded primal treewidth.

Theorem 11. *$\#SAT$ parametrised by ptw can be solved in FPT time. More precisely, we can count the number of solution of F in time $2^{O(k)} \cdot poly(|F|)$ where $k = ptw(F)$.*

Proof. Start from a tree decomposition T of the primal graph of F, root it in a node r. The bags of T are denoted by B_t. Remember that $|B_t| \leq k+1$.

Given $t \in T$, define T_t to be the tree rooted in t, V_t to be the variables of F appearing in T_t and F_t to be CNF formula whose clauses are clauses C of F such that $var(C) \subseteq V_t$. Observe that $F_r = F$.

We will compute $\#F$ by dynamic programming. For every t and $\tau : B_t \to \{0,1\}$, we will compute $\#F_t[\tau]$. Observe that there is $|T| \cdot 2^{k+1}$ such values to compute.

We now explain how the dynamic programming works. If t is a leaf of the tree, then $\tau : B_t \to \{0,1\}$ assigns *all* variables of F_t. Thus $\#F_t[\tau]$ is either 0 or 1.

Now let t be a vertex of t and t_1, t_2 its children. Observe that $V_{t_1} \cap V_{t_2} \subseteq B_t$. We thus have $\#F_t[\tau] = \#F_{t_1}[\tau_1] \cdot \#F_{t_2}[\tau_2]$ where $\tau_1 = \tau|_{V_1}$ and $\tau_2 = \tau|_{V_2}$.

We conclude by observing that $\#F[\tau_1] = \sum_{\mu : B_{t_1} \setminus B_t \to \{0,1\}} \#F_1[\tau_1 \cup \mu]$ (symmetrically for t_2) which all have been precomputed. □

Change the proof to construct a d-DNNF:

- $F_t[\tau] = F_{t_1}[\tau_1] \wedge F_{t_2}[\tau_2]$ and this $\wedge$ is decomposable,
- $F[\tau_1] = \bigvee_{\mu : B_{t_1} \setminus B_t \to \{0,1\}} F_1[\tau_1 \cup \mu]$ and this $\vee$ is deterministic.

We can actually construct a dec-DNNF from this. Add decision tree for

$$\mu : B_{t_1} \setminus B_t \to \{0,1\}.$$

Relation with `c2d` and `d4`.

- 3-splitting: syntactic decompositions (what Pierre was presenting)
- `c2d` starts from a tree decomposition of the formula and compile it (not exactly the same algorithm).

3.3 Incidence treewidth

Compile formulas of bounded incidence treewidth toward d-DNNF.

Theorem 12. *Given F of itw k, we can construct in FPT time a d-DNNF of size $2^{O(k)} \cdot |F|$.*

Proof. Start from a tree decomposition T of the primal graph of F, root it in a node r. The bags of T are denoted by B_t. Remember that $|B_t| \leq k+1$. We denote by $var(B_t) = var(F) \cap B_t$ and $cla(B_t) = F \cap B_t$.

Given $t \in T$, define T_t to be the tree rooted in t, V_t to be the variables of F appearing in T_t and F_t to be CNF formula whose clauses are clauses of F appearing in T_t.

We will compute our d-DNNF for F by dynamic programming. For every t and $\tau : var(B_t) \to \{0,1\}$ and $\mathcal{C} \subseteq cla(F_t)$, we will compute a d-DNNF with a gate computing $(F_t \setminus C)[\tau] \wedge \bigwedge_{C \in \mathcal{C}} \neg C$. □

4 Toward more general parameters

Slide with the parameters zoo!