Exact (Exponential) Algorithms for NP-hard Problems

Fabrizio Grandoni

Università di Roma “La Sapienza”
grandoni@di.uniroma1.it
Outline

- Exact Algorithms
- Independent Set Problem
- A Toy-Algorithm
- Memorization
- Folding
- Measure & Conquer
- Other Results
- Open Problems
**Prb:** Designing exact algorithms for NP-hard problems with the smallest possible worst-case running time.

- Need for exact solutions (e.g. decision problems).
- Reducing the running time from, say, $2^n$ to $1.5^n$ increases the size of the instances solvable by a constant multiplicative factor.
- Classical approaches (heuristics, approximation algorithms, parameterized algorithms...) have limits and drawbacks (no guaranty, hardness of approximation, $W[t]$-completeness...).
- New combinatorial and algorithmic challenges.
Maximum Independent Set

**Prb:** given an $n$-node graph $G = (V, E)$, determine the maximum cardinality of a subset of pairwise non-adjacent nodes (independent set).

![Graph](image)

$OPT = 2$
Maximum Independent Set

- NP-hard.
- Hard to approximate within $n^{1-\epsilon}$.
- No exact $c^{o(n)}$ algorithm (unless SNP $\subseteq$ SUBEXP).

$\Rightarrow$ The best we can hope for is a $2^{cn}$ algorithm for some small $c \in (0, 1]$. 
Maximum Independent Set

- [Tarjan&Trojanowski’77]: $O(2^{0.334n})$ poly-space.
- [Jian’86]: $O(2^{0.304n})$ poly-space.
- [Robson’86]: $O(2^{0.296n})$ poly-space, $O(2^{0.276n})$ exp-space.
- [Fomin,Grandoni&Kratsch’06]: $O(2^{0.288n})$ poly-space.
- [Beigel’99, Chen,Kanj&Xia’03]: better results for sparse graphs.
Domination

**Lem:** If there are two nodes $v$ and $w$ such that $N[v] \subseteq N[w]$, there is a maximum independent set which does not contain $w$ ($N[x] = N(x) \cup \{x\}$).

**Prf:**

![Diagram showing the relationship between nodes and their neighborhoods, illustrating the lemma and proof.](image)
A Toy-Algorithm

```c
int mis(G) {
    if(|G| \leq 1) return |G|; //Base case
    if(∃ component C ⊂ G) //Components
        return mis(C)+mis(G – C);
    if(∃ nodes v and w: N[v] ⊆ N[w]) //Domination
        return mis(G – {w});
    //“Greedy” branching
    select a node v of maximum degree; //d(v) \geq 2
    if(deg(v)=2) return poly-mis(G); //cycles
    return max{mis(G-{v}), 1+mis(G–N[v])};
}
```
A Toy-Algorithm

- The algorithm produces a search tree of exponential size, where branching takes polynomial time.
- Thus the analysis reduces to bounding the number of subproblems generated.
- The bound is obtained by defining a measure of the size of the subproblems. Each branching rule leads to some linear recurrences in the measure, which are used to lower-bound the progress made by the algorithm at each branching step.
A Toy-Algorithm

Lem: Algorithm mis runs in time $O(2^{0.465n})$.

Prf:

• Let $P(n)$ be the number of subproblems solved to solve a problem on $n$ nodes. Then

$$
P(n) \leq \begin{cases} 
1 & \text{base case/cycles;} \\
1 + P(|C|) + P(n - |C|) & \text{connected components;} \\
1 + P(n - 1) & \text{domination;} \\
1 + P(n - 1) + P(n - 4) & \text{branching ($d(v) \geq 3$)}.
\end{cases}
$$

• The base of the exponential factor is obtained from

$$
c^n \geq c^{n-1} + c^{n-4} \iff c^4 - c^3 - 1 \geq 0.
$$
Memorization

- The same subproblem may appear several times.
- **Memorization** consists in storing the solutions of the subproblems solved in an (exponential-size) **database**, which is queried each time a new subproblem is generated.
- This way, no subproblem is solved twice.
Memorization

**Thm:** Algorithm mis, combined with memorization, has running time $O(2^{0.426n})$.

**Prf:**

- The subproblems involve **induced subgraphs** of the original graph. Thus there are at most $\binom{n}{k}$ different subproblems on $k$ nodes.
- From standard analysis, such subproblems are upper bounded also by $2^{0.465(n-k)}$.
- Altogether, using Stirling’s formula,

$$P(n) \leq \sum_{k=1}^{n} \min \left\{ 2^{0.465(n-k)}, \binom{n}{k} \right\} = O(2^{0.426n}).$$
**Folding**

**Lem:** For every node $v$, there is a maximum independent set which either contains $v$ or at least two of its neighbors.

**Prf:**

![Diagram]

For every node $v$, there is a maximum independent set which either contains $v$ or at least two of its neighbors.
Folding

**Def:** Folding a node $v$, $N(v) = \{w, u\}$, with $w$ and $u$ not adjacent, means

- replacing $v$, $w$, and $u$ with a new node $v'$;
- adding edges between $v'$ and $N(w) \cup N(u) - \{v\}$.

**Lem:** When we fold a node $v$, the maximum independent set size decreases exactly by one.
Folding

```c
int mis'(G) {
    if(|G| \leq 1) return |G|; //Base case
    if(\exists \text{ component } C \subset G) //Components
        return mis'(C)+mis'(G - C);
    if(\exists \text{ nodes } v \text{ and } w: N[v] \subseteq N[w]) //Domination
        return mis'(G - \{w\});
    if(\exists v \text{ foldable}) //Folding
        return 1 + mis'(fold(v, G));
    //“Greedy” branching
    select a node v of maximum degree; //d(v) \geq 3
    return \max\{mis'(G - \{v\}), 1+mis'(G - N[v])\};
}
```
Folding

**Lem:** Algorithm mis’ runs in time $O(2^{0.406n})$.

**Prf:** If the algorithm branches at a node of degree 3, then a node of degree 2 is left

$$P(n) \leq \begin{cases} 
1 & \text{base case/poly-case;} \\
1 + P(|C|) + P(n - |C|) & \text{connected components;} \\
1 + P(n - 1) & \text{domination;} \\
1 + P(n - 1) + P(n - 5) & \text{branching ($d(v) \geq 4$);} \\
1 + P(n - 3) + P(n - 4) & \text{branching ($d(v) = 3$).}
\end{cases}$$

• Folding is not compatible with memorization.
Measure & Conquer

- Exact recursive algorithms are often very complicated (tedious case analysis).
- But the measure used in their analysis is usually trivial (e.g. number of nodes in IS, as before).

⇒ **Measure & Conquer** approach consists in focusing on the choice of the measure.
Measure & Conquer

- Removing nodes of high degree reduces the degree of many other nodes. This pays off on long term since nodes of degree $\leq 2$ can be filtered out without branching.

- This phenomenon is not taken into account with standard analysis/measure.

$\Rightarrow$ The idea is to give a different (smaller) weight to nodes of different (smaller) degree:

$$W(v) = \begin{cases} 0 & \text{if } d(v) \leq 2; \\ \alpha \in (0, 1] & \text{if } d(v) = 3; \\ 1 & \text{if } d(v) \geq 4. \end{cases}$$
Measure & Conquer

**Thm:** Algorithm $\text{mis'}$ has running time $O(2^{362n})$.

**Prf:**
- First we need to enforce that folding does not increase the size of the problem: $\alpha \geq 0.5$.
- When we branch by **discarding** a node $v$, the size of the problem decreases because of: (1) the removal of $v$, and (2) the decrease of the degree of the neighbors of $v$.
- When we branch by **selecting** a node $v$, the size of the problem decreases because of: (1) the removal of $v$, and (2) the removal of the neighbors of $v$. 
**Prf:** Let $P(k)$ be the number of subproblems solved to solve a problem of size $k \leq n$. Then

$$P(k) \leq \begin{cases} 
1 + P(k - 1) + P(k - 6); \\
1 + P(k - 1 - \alpha) + P(k - 5 - \alpha); \\
1 + P(k - 1 - 2\alpha) + P(k - 4 - 2\alpha); \\
1 + P(k - 1 - 3\alpha) + P(k - 3 - 3\alpha); \\
1 + P(k - 1 - 4\alpha) + P(k - 2 - 4\alpha); \\
1 + P(k - 5 + 4\alpha) + P(k - 5); \\
1 + P(k - 4 + 2\alpha) + P(k - 4 - \alpha); \\
1 + P(k - 3) + P(k - 3 - 2\alpha); \\
1 + P(k - 2 - 2\alpha) + P(k - 2 - 3\alpha); \\
1 + P(k - 1 - 3\alpha) + P(k - 1 - 3\alpha). 
\end{cases}$$
Prf:

• By solving the recurrences, \( P(k) = O(c^k) = O(c^n) \), where \( c = c(\alpha) \) is a quasi-convex function of \( \alpha \) [Eppstein’04].

• Imposing \( \alpha = 0.6 \), one obtains \( c < 2^{0.362} \).
Other Results

- **Minimum Dominating Set** in time $O(2^{0.598n})$.
- **Maximum Cut** in time $O(2^{0.792n})$.
- **Steiner Tree** in time $O(2^{0.773n})$.
- **Cubic TSP** in time $O(2^{0.334n})$.
- **Chromatic Number** in time $O(2.415^n)$.
- **3-Colorability** in time $O(1.3289^n)$.
- **3-Satisfiability** in time $O(1.4802^n)$.
- **Knapsack** in time $O(2^{0.5n})$.
- ....
Open Problems

- Current best for **Hamiltonian Path** is poly-space $\Omega(2^n)$.
- Same for **TSP**, but exp-space.
- Current best for **SAT** is trivial $\Omega(2^n)$.
- Current best for **Feedback Vertex Set** is trivial $\Omega(2^n)$.
- ....