

2. Dynamic Programming

COMP6741: Parameterized and Exact Computation

Serge Gaspers^{1,2}

¹School of Computer Science and Engineering, UNSW Sydney, Australia

²Decision Sciences Group, Data61, CSIRO, Australia

Semester 2, 2017

1 Dynamic Programming Across Subsets

- Traveling Salesman Problem
- Coloring
- Dominating Set in bipartite graphs

2 Further Reading

1 Dynamic Programming Across Subsets

- Traveling Salesman Problem
- Coloring
- Dominating Set in bipartite graphs

2 Further Reading

Dynamic Programming across Subsets

- very general technique
- uses solutions of subproblems
- typically stored in a table of exponential size

1 Dynamic Programming Across Subsets

- Traveling Salesman Problem
- Coloring
- Dominating Set in bipartite graphs

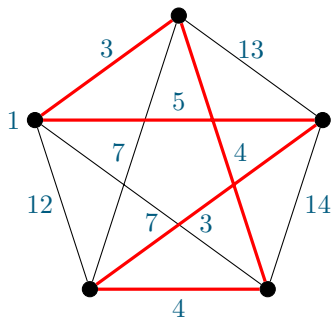
2 Further Reading

TRAVELING SALESMAN PROBLEM

TRAVELING SALESMAN PROBLEM (TSP)

Input: a set of n cities, the distance $d(i, j) \in \mathbb{N}$ between every two cities i and j , integer k

Question: Is there a permutation of the cities (a **tour**) such that the total distance when traveling from city to city in the specified order, and returning back to the origin, is at most k ?



Brute-force: Try all permutations of cities; $O^*(n!)$

Dynamic Programming for TSP I

For a non-empty subset of cities $S \subseteq \{2, 3, \dots, n\}$ and city $i \in S$:

- $\text{OPT}[S; i] \equiv$ length of the shortest path starting in city 1, visits all cities in $S \setminus \{i\}$ and ends in i .

Then,

$$\text{OPT}[\{i\}; i] = d(1, i)$$

$$\text{OPT}[S; i] = \min\{\text{OPT}[S \setminus \{i\}; j] + d(j, i) : j \in S \setminus \{i\}\}$$

- For each subset S in order of increasing cardinality, compute $\text{OPT}[S; i]$ for each i .
- Final solution:

$$\min_{2 \leq j \leq n} \{\text{OPT}[\{2, 3, \dots, n\}; j] + d(j, 1)\}$$

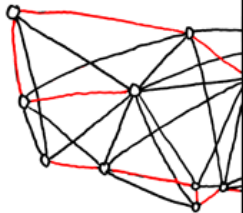
Theorem 1 (Held & Karp '62)

TSP can be solved in time $O(2^n n^2) = O^(2^n)$.*

- best known algo for TSP

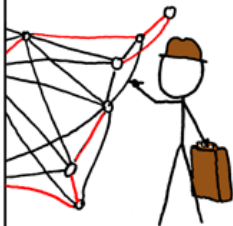
BRUTE-FORCE
SOLUTION:

$$O(n!)$$



DYNAMIC
PROGRAMMING
ALGORITHMS:

$$O(n^2 2^n)$$



SELLING ON EBAY:

$$O(1)$$

STILL WORKING
ON YOUR ROUTE?

SHUT THE
HELL UP.



1 Dynamic Programming Across Subsets

- Traveling Salesman Problem
- **Coloring**
- Dominating Set in bipartite graphs

2 Further Reading

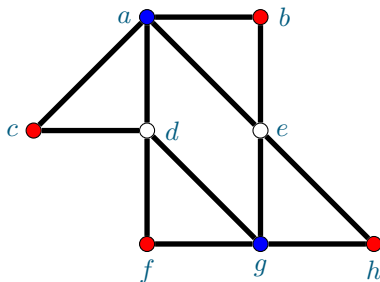
COLORING

A k -coloring of a graph $G = (V, E)$ is a function $f : V \rightarrow \{1, 2, \dots, k\}$ assigning colors to V such that no two adjacent vertices receive the same color.

COLORING

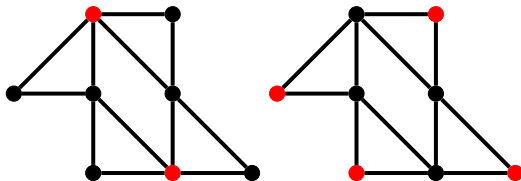
Input: Graph G , integer k

Question: Does G have a k -coloring?



Maximal Independent Sets

- An independent set is **maximal** if it is not a subset of any other independent set.
- Examples:



Theorem 2 ([Moon, Moser '65], [Johnson, Yannakakis, Papadimitriou '88])

A graph on n vertices contains at most $3^{n/3} \subseteq O(1.4423^n)$ maximal independent sets. Moreover, they can all be enumerated in time $O^(3^{n/3})$.*

Coloring and Maximal Independent Sets

Theorem 2 ([Moon, Moser '65], [Johnson, Yannakakis, Papadimitriou '88])

A graph on n vertices contains at most $3^{n/3} \subseteq O(1.4423^n)$ maximal independent sets. Moreover, they can all be enumerated in time $O^*(3^{n/3})$.

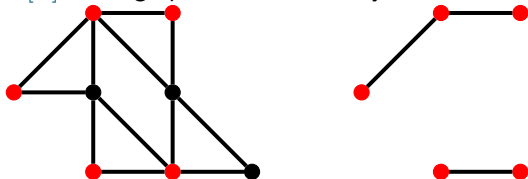
A coloring is **optimal** if it uses a smallest number of colors.

Lemma 3 ([Lawler '76])

For any graph G , there exists an optimal coloring for G where one color class is a maximal independent set in G .

Dynamic Programming for COLORING I

- $G[S] \equiv$ subgraph of G induced by the vertices in S



- $\text{OPT}[S] \equiv$ minimum k such that $G[S]$ is k -colorable.
- Then,

$$\text{OPT}[\emptyset] = 0$$

$$\text{OPT}[S] = 1 + \min\{\text{OPT}[S \setminus I] : I \text{ maximal ind. set in } G[S]\}$$

Dynamic Programming for COLORING II

$$\text{OPT}[\emptyset] = 0$$

$$\text{OPT}[S] = 1 + \min\{\text{OPT}[S \setminus I] : I \text{ maximal ind. set in } G[S]\}$$

- go through the sets S in order of increasing cardinality
- to compute $\text{OPT}[S]$, generate all maximal independent sets I of $G[S]$
- this can be done in time $|S|!3^{|S|/3}$
- time complexity:

$$\sum_{s=0}^n \binom{n}{s} s^2 3^{s/3} \leq n^2 \sum_{s=0}^n \binom{n}{s} 3^{s/3} = n^2 (1 + 3^{1/3})^n = O(2.4423^n)$$

[Recall the **Binomial Theorem**: $(x + y)^n = \sum_{k=0}^n \binom{n}{k} x^{n-k} y^k$.]

Theorem 4 ([Lawler '76])

COLORING can be solved in time $O(2.4423^n)$.

- was best known algorithm for 25 years (until [Eppstein '01])
- current best: $O^*(2^n)$ [Bjørklund & Husfeldt '06], [Koivisto '06]

k -Coloring for small k

k -COLORING

Input: Graph G , integer k

Question: Does G have a k -coloring?

- $k \leq 2$: polynomial
- $k > 2$: NP-complete

Theorem 5 ([Lawler '76])

3-COLORING can be decided in time $O(1.4423^n)$.

Algorithm for 3-Coloring

Theorem 5 ([Lawler '76])

3-COLORING can be decided in time $O(1.4423^n)$.

Proof.

For every maximal independent I set of G , check if $G - I$ is 2-colorable. \square

Algorithm for 3-Coloring

Theorem 5 ([Lawler '76])

3-COLORING can be decided in time $O(1.4423^n)$.

Proof.

For every maximal independent I set of G , check if $G - I$ is 2-colorable. \square

current best: $O(1.3289^n)$ [Eppstein '01]

Theorem 6

4-COLORING *can be decided in time* $O(1.7851^n)$.

Algorithm for 4-Coloring

Theorem 6

4-COLORING can be decided in time $O(1.7851^n)$.

Proof.

- By a generalization of Lemma 3, each 4-colorable graph G has a 4-coloring where one color class is a maximal i.s. of size $\geq n/4$.

Algorithm for 4-Coloring

Theorem 6

4-COLORING can be decided in time $O(1.7851^n)$.

Proof.

- By a generalization of Lemma 3, each 4-colorable graph G has a 4-coloring where one color class is a maximal i.s. of size $\geq n/4$.
- For each maximal independent set I of G of size at least $n/4$, check if $G - I$ is 3-colorable.
- Running time: $O(3^{n/3} 1.3289^{3n/4}) \subseteq O(1.7851^n)$



current best: $O(1.7272^n)$ [Fomin, Gaspers, Saurabh '07]

1 Dynamic Programming Across Subsets

- Traveling Salesman Problem
- Coloring
- Dominating Set in bipartite graphs

2 Further Reading

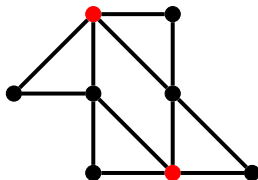
DOMINATING SET

A **dominating set** in a graph $G = (V, E)$ is a subset of vertices $S \subseteq V$ such that each vertex of G is either in S or adjacent to a vertex in S .

DOMINATING SET

Input: Graph G , integer k

Question: Does G have a dominating set of size k ?



Bipartite graphs

A graph $G = (V, E)$ is **bipartite** if its vertex set can be partitioned into two independent sets.

DOMINATING SET IN BIPARTITE GRAPHS

Input: Bipartite graph G , integer k

Question: Does G have a dominating set of size k ?

Note: DOMINATING SET IN BIPARTITE GRAPHS is **NP**-complete.

Algorithm for Dominating Set in Bipartite Graphs I

Partition V into independent sets A and B , with $|B| \geq |A|$.

The algorithm has 2 phases:

- **Preprocessing phase:** compute for each $X \subseteq A$ a subset $\text{Opt}[X]$ which is a smallest subset of B that dominates X .
- **Main phase:** for each subset $X \subseteq A$, compute a dominating set D of G of minimum size such that $D \cap A = X$.

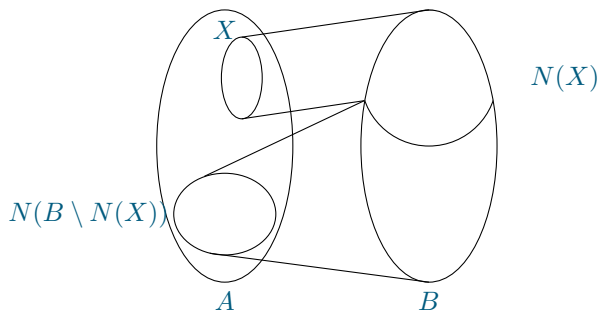
Algorithm for Dominating Set in Bipartite Graphs II

Main phase. For a vertex subset $X \subseteq A$, a dominating set D of G of minimum size such that $D \cap A = X$ is obtained by setting

$$D := X \cup (B \setminus N(X)) \cup \text{Opt}[A \setminus (X \cup N(B \setminus N(X)))]$$

if $A \setminus X$ contains no degree-0 vertex.

(If $A \setminus X$ contains a degree-0 vertex, we skip this set X , because there is no dominating set D of G with $D \cap A = X$.)



Algorithm for Dominating Set in Bipartite Graphs III

Preprocessing phase. Let $B = \{b_1, \dots, b_{|B|}\}$. We compute for each $X \subseteq A$ and integer k , $0 \leq k \leq |B|$, a subset $\text{Opt}[X, k] \subseteq \{b_1, \dots, b_k\}$ which is defined as

- a smallest subset of $\{b_1, \dots, b_k\}$ that dominates X if $X \subseteq N(\{b_1, \dots, b_k\})$, and
- B if $X \not\subseteq N(\{b_1, \dots, b_k\})$.

Note: $\text{Opt}[X, |B|] = \text{Opt}[X]$.

Algorithm for Dominating Set in Bipartite Graphs IV

Base cases

$$\begin{aligned}\text{Opt}[\emptyset, k] &= \emptyset & \forall k \in \{0, \dots, |B|\}, \\ \text{Opt}[X, 0] &= B & \forall X, \emptyset \subsetneq X \subseteq A.\end{aligned}$$

Dynamic Programming recurrence

$$\text{Opt}[X, k] = \begin{cases} \text{Opt}[X, k-1] & \text{if } |\text{Opt}[X, k-1]| < 1 + |\text{Opt}[X \setminus N(b_k), k-1]| \\ \{b_k\} \cup \text{Opt}[X \setminus N(b_k), k-1] & \text{otherwise} \end{cases}$$

for each X , $\emptyset \subsetneq X \subseteq A$ and $k \in \{1, \dots, |B|\}$.

Algorithm for Dominating Set in Bipartite Graphs V

Theorem 7 ([Liedloff '08])

DOMINATING SET IN BIPARTITE GRAPHS *can be solved in $O^*(2^{n/2})$ time, where n is the number of vertices of the input graph.*

1 Dynamic Programming Across Subsets

- Traveling Salesman Problem
- Coloring
- Dominating Set in bipartite graphs

2 Further Reading

- Chapter 3, *Dynamic Programming* in
Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer,
2010.