Day 6 Notes

Zach Neveu

May 14, 2019

1 Review

- NPC proof via sub-problem (see Venn diagram from day 5)
- If sub-problem is NPC, problem is NPC
- If problem is P, sub-problem also P
- Sub-problems can be organized recursively (Complexity Landscape^{TM})

Example: Procedure Constrained Scheduling (PCS)

- Given a set of tasks that each take one unit of time to complete, a partial order on the tasks, a number, m, of processors, and an integer deadline
- Question: Does a legal schedule allow the processes to be completed before the deadline?
- General PCS $\in$ NPC
- If constraints graph is tree, $PCS \in P$
- If constraints graph empty, $PCS \in P$
- Aside: Why m=2 solvable, but m=3 so hard?
- Consider: 3 is important.
- $2SAT \in P$, $3SAT \in NPC$, same for HC problem

Special Nodes

A node on the Complexity Landscape that has no known NPC children is called **Minimally NPC**
A node on the Complexity Landscape that is in P and has no parents in P is called **Maximally polynomially solvable**

2 Greedy Algorithms

- A **Greedy Algorithm** always makes what appears to be the best decision in the current moment
- A **Greedy Algorithm** does not utilize backtracking

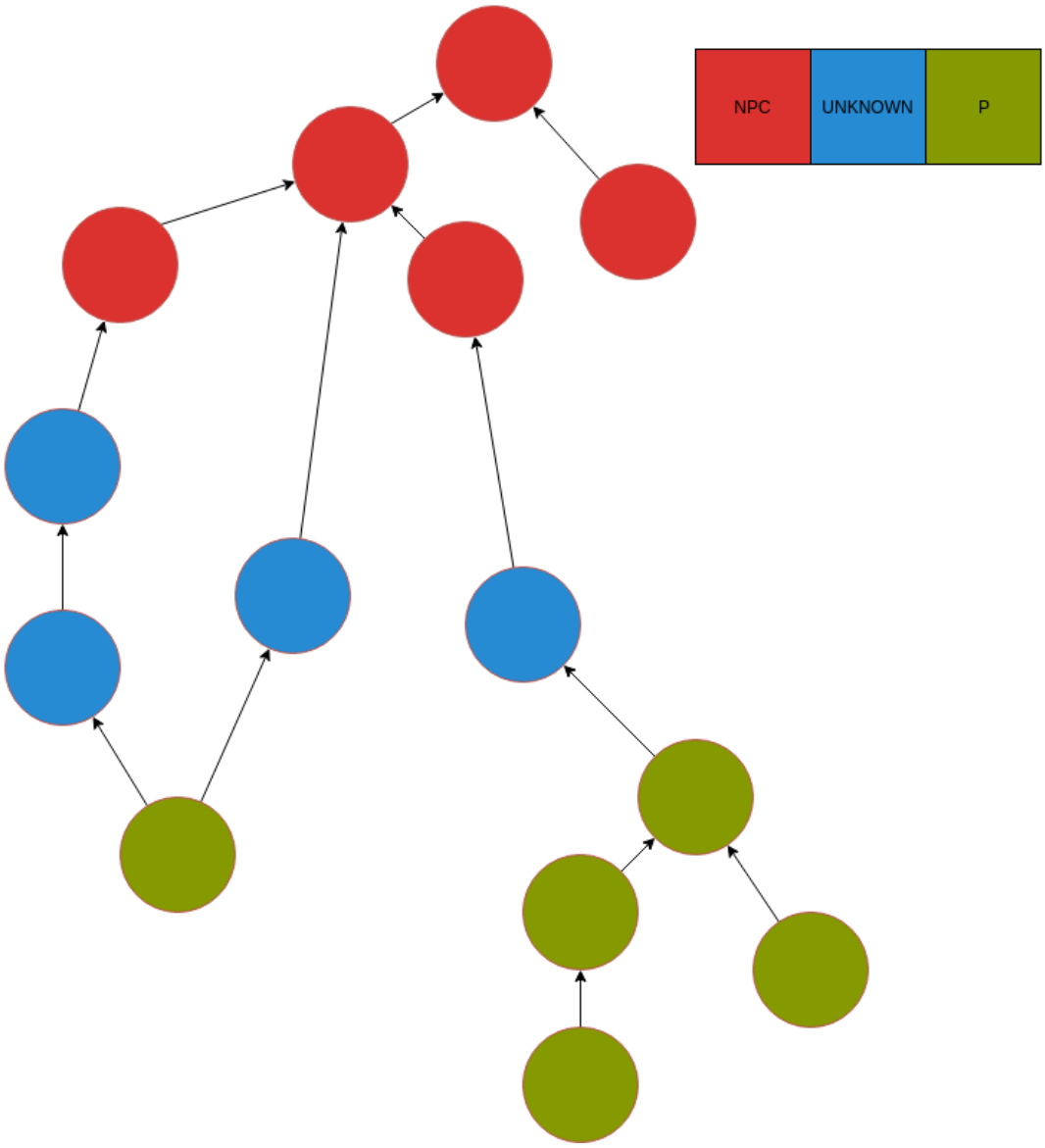

Figure 1: Example Complexity Landscape^{TM}

MST

MST: Given an undirected graph and a weight for each edge, find an acyclic subset of the edges that connects all nodes with minimum weight.

```
def MST():
  A=0
  while A is not spanning tree:
    find next edge (u,v) in increasing order by weight such that (u,v) is safe for A
    A += {(u,v)}
  return A
```

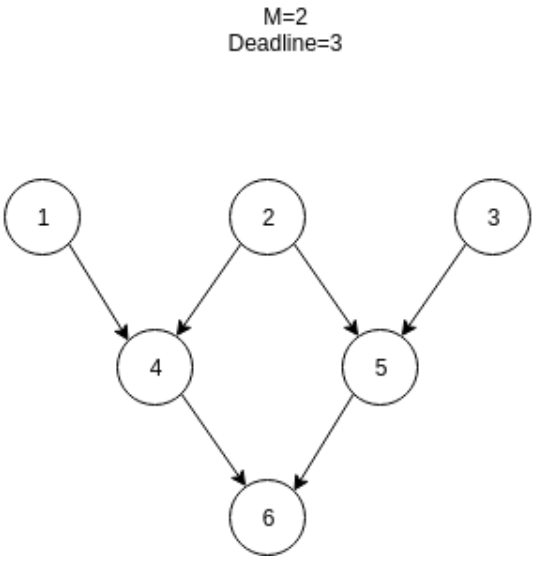

Figure 2: Example PCS Constraint

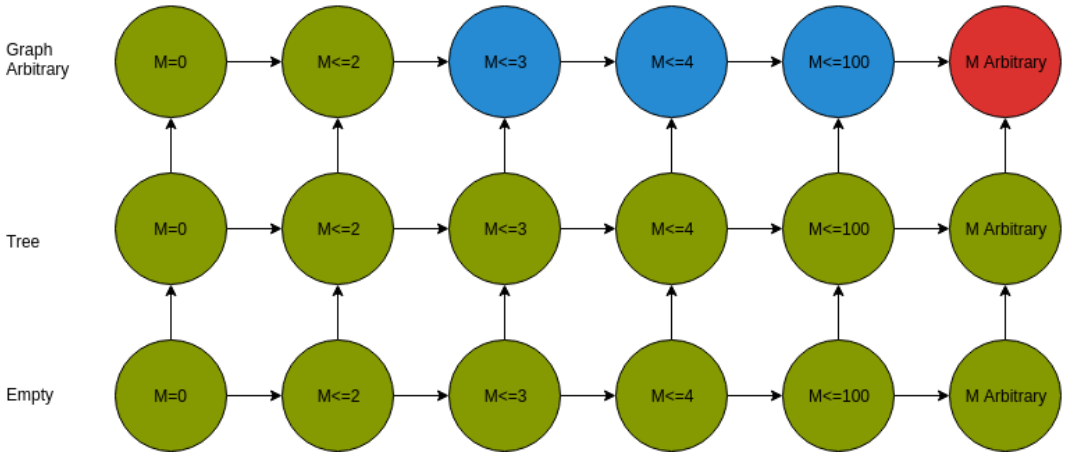

Figure 3: PCS Complexity Landscape^{TM}

Activity Selection

- Given: a set $S = \{a_1, a_2, \ldots, a_n\}$ of n activities, only one of which can take place at a time. Activity a_i starts at time s_i and finishes at time f_i. Two activities are **Compatible** if they do not conflict.
- Find: How many activities can we fit?
- Find a largest set of mutually compatible activities

```
# Solution
# a[i] is most recently selected activity
# a[m] activity we are considering adding
def activitySelection(a):
  sortByIncreaseFinish(a)
  n = number of Activities
  A = a[0] # first activity
```

	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	6	8	8	2	12	3	5
f_i	4	5	6	7	10	11	12	13	14	8	9

Table 1: Example Activity Problem

```
i = 1
for m in range(1,n):
  if not conflict(a[m], a[i]):
    A += a[m]
    i = m
return A
```

- Proof: Show that each step is in the right direction
- Prove by contradiction: assume that there is no maximal subset including a[0]. We can always swap the first event in any set with a[0] because of the sorting, so any maximal subset can include a[0].
- Repeat this problem recursively on all problems with $s_i > f_0$ - this step valid for all sub-problems

Head Partition Problem

- Given directed graph, find largest subset of edges which point to separate nodes

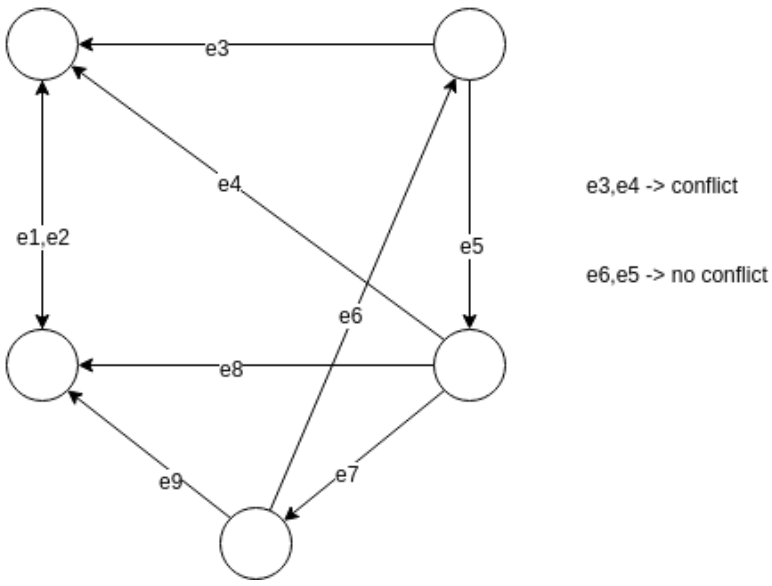

Figure 4: Head Partition Example

```
# Algorithm 1
def headPartition(g):
  for node in g:
    select any incoming arc
```

```
# Algorithm 2
# Version of "Generic Greedy Algorithm"
def headPartition2(g):
  edge_group = {}
  for arc in g.edges:
    if not conflict(arc, edge_group):
      edge_group += arc
```