Proving Complexity Results for Real-Time Scheduling Problems

- Review NP-Completeness
- Pseudo-Polynomial algorithms and Strong NP-Completeness
- 3-Partition and its uses
- k-Simultaneous Congruences and its uses
- skip-over tasks

Decision Problem: A problem (denoted Π) that asks only for a yes/no answer is called a **decision problem.**

Does this graph contain a Clique of size k?

Is this task set feasible on 4 processors?

How could we use an algorithm for a decision problem to solve an optimization problem such as: What is the largest Clique in this graph of n vertices?

Encoding Scheme: An instance *I* of a problem is represented as a string of symbols with the following restrictions:

- the encoding of an instance *I* should be concise and not "padded" with unnecessary information or symbols
- numbers occurring in *I* should be represented in binary (or decimal, or octal, or in any fixed base other than 1)

The number of symbols in the input string that describes *I* is denoted **L(*I*)**.

The Class P: A decision problem is in **P** if there exists an algorithm **A** and a number **c** such that for every instance *I* of the problem the algorithm **A** will produce a solution in time $\mathbf{O(L(}I\mathbf{)}^{c}\mathbf{)}$. **A** is referred to as a **polynomial time algorithm**.

Is this Liu and Layland task set scheduable using EDF?

A decision problem is **intractable** if it can be **proven** that no polynomial time algorithm can possibly solve it.

The Class NP: A decision problem is in **NP** if there exists a polynomial time algorithm **A** that takes as input an instance and a *guess* that does the following:

- Associated with each 'yes' instance of the problem, there is a *guess* **G(*I*)** such that **A** answers 'yes'
- There is no *guess* such that **A** answers 'yes' when given a 'no' instance of the problem.

In **P** are the problems where it's easy to *find* a solution, and in **NP** are the problems where it's easy to *check* a solution that may have been hard to find.

The Travelling Salesman Problem (TS): Given n cities with a distance between each pair, and a bound *B*. Is there a tour that visits all n of the cities exactly once, returns to its starting point, and has a total length $\leq B$? Is TS in NP?

The complement of a decision problem Π, (denoted Π^c), is a decision problem for which the answer is 'yes' if and only if the answer to Π is 'no'.

The complement of TS, TS^c, asks the question "Is there *no* tour that visits all n of the cities exactly once, returns to its starting point, and has a total length $\leq B$?".

Is TS^c in NP?

The Class Co-NP: A decision problem is in Co-NP if its complement is in NP.

Is TS^c in Co-NP?

Polynomial Transformation Given 2 decision problems Π_1 and Π_2. Π_1 is *polynomially transformable* to Π_2 (denoted $\Pi_1 \propto \Pi_2$) if every instance I_1 of Π_1 can be transformed into an instance I_2 of Π_2 with the same yes/no answer in polynomal time.

Given $\Pi_1 \propto \Pi_2$.

If $\Pi_2 \in$ P then what about Π_1?

What if Π_1 is not in P?

A decision problem Π is **NP-Hard** if for all other decision problems $\Pi' \in$ NP, $\Pi' \propto \Pi$.

A decision problem Π is **NP-Complete** if $\Pi \in$ NP, and Π is NP-Hard.

A problem is **Co-NP-Hard** if its complement is NP-Hard, and it is **Co-NP-Complete** if its complement is NP-Complete.

Two important questions are: does **P = NP?** and does **Co-NP = NP?**

What if an NP-Complete problem was found to be in P?

What if an NP problem was found to be intractible?

What if a Co-NP-Hard problem was found to be in P?

What if NP $\neq$ Co-NP?

Satisfiability was the first discovered NP-Complete problem (Cook's Theorem).

Other example problems that are NP-Complete are **k-Clique**, **Hamiltonian Circuit**,**Vertex Cover**, **k-Colorability**, **Partition**, and **Bin Packing**.

How do you prove that a decision problem is NP-Complete?

Partition: Given a finite set A and a 'size' $s(a) \in Z^+$ for each $a \in A$. Is there a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)?$$

We can use dynamic programming to solve partition in time that is polynomial in nB where $n = \mid A \mid$ and $B = \sum_{a \in A} s(a)$

However, this algorithm is not polynomial in the *length* of the problem instance. The reason is because of our restrictions on encoding schemes. In particular, if numbers are represented in binary with b bits, then the maximum size of the numbers is 2^b which is exponential in the number of bits.

The maximum number occurring in a problem instance I is denoted $M(I)$.

An algorithm for a decision problem is **pseudo-polynomial** if its time complexity is bounded above by a polynomial in $L(I)$ and $M(I)$.

Partition can be solved with a pseudo-polynomial time algorithm.

An example from the real-time world is deciding scheduability under Rate-Monotonic for a given Liu and Layland Task Set.

What if $M(I)$ is bounded by a polynomial function of $L(I)$ (for example Clique).

A problem Π is a **number problem** if there exists no polynomial p such that $M(I) \leq p(L(I))$ for all instances. Is partition a number problem? How about Travelling Salesman? Feasibility of a non-preemptible set of tasks?

The only candidates for pseudo-polynomial time algorithms are number problems.

For any number problem, there exists a subproblem that is not a number problem.

For a decision problem Π and any polynomial p, Π_p denotes the subproblem of Π obtained by restricting Π to only those instances that satisfy $M(I) \leq p(L(I))$.

What does finding a pseudo-polynomial time algorithm for Π say about the complexity of Π_p? What if Π_p is NP-Complete?

A decision problem Π is **NP-Complete in the Strong Sense** if $\Pi \in NP$ and there is a polynomial p such that Π_p is NP-Complete.

The obvious way of proving NP-Completeness in the strong sense for Π is to find a polynomial p and prove that Π_p is NP-Complete.

For example, Traveling Salesman is NP-Complete in the strong sense. This can be proven using a reduction from Hamiltonian Circuit to a subproblem of Traveling Salesman.

Hamiltonian Circuit (HC): Given a graph $G = (V, E)$. Is there an ordering $< v_1, v_2, ..., v_n >$ of the vertices of G, where $n = \mid V \mid$, such that $\{v_n, v_1\} \in E$ and $\{v_i, v_{i+1}\} \in E$ for all $i, 1 \leq i < n$?

Proof: We already know that $TS \in NP$.

Hamiltonian Circuit is NP-Complete. Now consider the subproblem of Traveling Salesman TS_p where $p = 1$. That is every instance of TS_p satisfies $M(I) \leq L(I)$. Given an instance of HC where $\mid V \mid = m$. The corresponding instance of TS_p has a set C of cities that is identical to V. For any two cities $v_i, v_j \in C$, the intercity distance $d(v_i, v_j)$ is defined to be 1 if $\{v_i, v_j\} \in E$ and 2 otherwise. The bound B on the desired tour length is set equal to m.

Two other important problems (especially for real-time scheduling) shown to be NP-Complete in the Strong Sense using this method are **3-Partition** and **k-Simultaneous Congruences (SCP)** (see Baruah et. al. 1990).

3-Partition: Given a set A of $3m$ elements, a bound $B \in Z^+$, and a 'size' $s(a) \in Z^+$ for each $a \in A$, such that each $s(a)$ satisfies $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$. Can A be partitioned into m disjoint sets $S_1, S_2, ..., S_m$ such that, for $1 \leq i \leq m$, $\sum_{a \in S_i} s(a) = B$? (the above constraints on the item sizes imply that every such S_i must contain *exactly* three elements from A.)

Another way of showing NP-Completeness in the S.S. is to show that a known NP-Complete in the S.S. problem can be **pseudo-polynomially** reduced to the problem.

A pseudo-polynomial transformation from Π_1 to Π_2 is similar to a polynomial transformation except that the transformation can take pseudo-polynomial time and restrictions are placed on $L(I_2)$ and $M(I_2)$ to prevent exponential blowup of the integer sizes of the transformed instance.

Theorem: The problem of deciding whether it is possible to schedule a set of periodic processes which use semaphores only to enforce mutual exclusion is NP-Hard in the Strong Sense. (Mok 1983)

Proof: Do a pseudo-polynomial reduction from 3-Partition. Given an instance of 3-Partition, create $3m + 1$ tasks. There is only one semaphore. Each process grabs the semaphore, does some computation, and then releases the semaphore (in other words the tasks are non-preemptible). For each element in a_i in A, we create a task T_i with $p_i = d_i = mB + m$ and $c_i = s(a_i)$. In addition, we have a task T_{3m+1} with period $p_{3m+1} = B + 1$ and $d_{3m+1} = c_{3m+1} = 1$. Is this a pseudo-polynomial transformation? All feasible schedules must run T_{3m+1} in the intervals $[t, t + 1]$ where $t \equiv 0(modB + 1)$. What about the other tasks?

Note: this proof can also be used to demonstrate that non-preemptible task scheduling for concrete periodic tasks is NP-Hard in the Strong Sense (see also Jeffay et. al. 1991).

What about the complexity of an optimal scheduler?

Multiprocessor scheduling where the number of processors is variable can be proven to be NP-Hard in the S.S. using a very similar technique (see Leung & Whitehead '82)

As indicated, 3-Partition has been heavily used for showing NP-Hardness in the S.S. for problems involving multiprocessor scheduling, non-preemptive scheduling, and shared resources.

There is another NP-Complete in the S.S. problem, *k-Simultaneous Congruences*, that is useful for showing the difficulty of a class of problems including **Asynchronous Task Set Feasibility** and scheduling tasks that can suspend themselves, **I/O Gap Task Set Feasibility**.

k-Simultaneous Congruences (SCP): Given $A = \{(a_1, b_1), ..., (a_n, b_n)\} \subseteq NxN^+$ and $2 \leq k \leq n$. Is there a subset $A' \subseteq A$ of k pairs and a natural number x such that for every $(a_i, b_i) \in A', x \equiv a_i (mod b_i)$.

This problem was shown to be NP-Complete in the ordinary sense by Leung and Whitehead (1982) and later a strong sense proof was presented by Baruagh and others (1990).

The first use of this problem was to show that asynchronous task set feasbility where deadlines can be different than periods is co-NP-hard in the strong sense.