Function Problems

Function Problems

- Decisions problem are yes/no problems (SAT, TSP (D), etc.).
- **Function problems** require a solution (a satisfying truth assignment, a best TSP tour, etc.).
- Optimization problems are clearly function problems.
- What is the relation between function and decision problems?
- Which one is harder?

Function Problems Cannot Be Easier than Decision Problems

- If we know how to generate a solution, we can solve the corresponding decision problem.
  - If you can find a satisfying truth assignment efficiently, then SAT is in P.
  - If you can find the best TSP tour efficiently, then TSP (D) is in P.
- But decision problems can be as hard as the corresponding function problems.

FSAT

- FSAT is this function problem:
  - Let $\phi(x_1, x_2, \ldots, x_n)$ be a boolean expression.
  - If ϕ is satisfiable, then return a satisfying truth assignment.
  - Otherwise, return "no."
- We next show that if SAT $\in$ P, then FSAT has a polynomial-time algorithm.

An Algorithm for FSAT Using SAT

$t := \epsilon$;
if $\phi \in$ SAT **then**
  for $i = 1, 2, \ldots, n$ **do**
    if $\phi[\, x_i = \texttt{true} \,] \in$ SAT **then**
      $t := t \cup \{\, x_i = \texttt{true} \,\}$;
      $\phi := \phi[\, x_i = \texttt{true} \,]$;
    else
      $t := t \cup \{\, x_i = \texttt{false} \,\}$;
      $\phi := \phi[\, x_i = \texttt{false} \,]$;
    end if
  end for
  return t;
else
  return "no";
end if

Analysis

- There are $\leq n + 1$ calls to the algorithm for SAT.[a]
- Shorter boolean expressions than ϕ are used in each call to the algorithm for SAT.
- So if SAT can be solved in polynomial time, so can FSAT.
- Hence SAT and FSAT are equally hard (or easy).

[a] Contributed by Ms. Eva Ou (R93922132) on November 24, 2004.

TSP and TSP (D) Revisited

- We are given n cities $1, 2, \ldots, n$ and integer distances $d_{ij} = d_{ji}$ between any two cities i and j.
- The TSP asks for a tour with the shortest total distance (not just the shortest total distance, as earlier).
  - The shortest total distance must be at most $2^{|\, x \,|}$, where x is the input.
- TSP (D) asks if there is a tour with a total distance at most B.
- We next show that if TSP (D) $\in$ P, then TSP has a polynomial-time algorithm.

An Algorithm for TSP Using TSP (D)

Perform a binary search over interval $[\, 0, 2^{|\, x \,|} \,]$ by calling TSP (D) to obtain the shortest distance C;
for $i, j = 1, 2, \ldots, n$ **do**
  Call TSP (D) with $B = C$ and $d_{ij} = C + 1$;
  if "no" **then**
    Restore d_{ij} to old value; {Edge $[\, i, j \,]$ is critical.}
  end if
end for
return the tour with edges whose $d_{ij} \leq C$;

Analysis

- An edge that is not on *any* optimal tour will be eliminated, with its d_{ij} set to $C + 1$.
- An edge which is not on all remaining optimal tours will also be eliminated.
- So the algorithm ends with n edges which are not eliminated (why?).
- There are $O(|\,x\,| + n^2)$ calls to the algorithm for TSP (D).
- So if TSP (D) can be solved in polynomial time, so can TSP.
- Hence TSP (D) and TSP are equally hard (or easy).