Odds and Ends: (=Promises made in CS6363)

1. Directed Spanning Tree Problem.

Let \( G \) be a directed graph \( G = (V; E) \) with edge weights \( W(e) \) \( e \in E \).

Also, \( r \in V \): root of the directed spanning tree.

( \( r \) is a directed path in the tree from \( r \) to every other node; and it is a spanning tree when directed edges are ignored)

Want: a rooted (at \( r \)) directed spanning tree with minimum total weight.

[This was also solved by J. Edmonds as a special case of something much bigger.] But we present a “much easier” version. It also involves “shrinking”. And in some sense, it is a “greedy” algorithm as well.

We describe the algorithm via an example (this will be the general result—far now believe me!).
Example: We assume $\exists$ a dir. path from $r$ to each other node in $G$. Moreover, we may assume w/o loss, $\exists$ no edge going into $r$ - these can be removed - they will not be part of any rooted (at $r$) dir. sp. tree in $G$.

Every dir. sp. tree has $|V|-1$ edges; moreover, in such trees, each node $\neq r$ has one edge coming into it. But not all subsets satisfying these two conditions is a dir. sp. tree.
Since each node has exactly one edge.

Coming into it, a greedy approach will be to select the cheapest edge into each node.

For our example, it would be:

This is "obviously" not a directed spanning tree. It has directed cycles.

Recall: If we subtract/add a constant for each edge going into a node, the problem solution does not change (although total weight will).
We do this to get some clarity.

\[ A: 2 \quad B: 1 \quad C: 4 \quad D: 2 \quad E: 4 \]

to get:

\[
\begin{array}{c}
R \\
8 \\
A \quad B \\
1 \\
C \\
6 \\
D \\
4 \\
E \\
0 \\
\end{array}
\]

Edge, with weight 0 here, were chosen by the first step of our greedy algorithm. But it produced a cycle.

We now "Shrink" this.

**Basis**

For any subset \( S \) of nodes such that \( r \in S \), and any directed spanning tree rooted at \( r \), there must be at least one edge entering this set.
Shrunken Graph (edge cuts modified)

Use greedy algorithm on this Shrunken Graph.

Ex: Prove this is optimal solution.

Of course, shrinking maybe done many time as also "unshrinking".

$$ABCD : 1$$
$$E : 0$$

Since $$E \ (r, B)$$ going into $$B$$, remove $$AB$$ to get a dir. sp. tree.
Ratio Optimization in Combinatorics:  
(See: N. Megiddo, L. Karp, Maffioli, RC)

Ratio Spanning Trees: (Maffioli, RC)

$G : \text{undir. } = [V; E] ; \text{ two numbers}$

for each edge $a(e), b(e) ; b(e) > 0 \forall e \in E$

Want: Sp. tree $T$ that $\min_{T} \sum_{e \in T} \frac{a(e)}{b(e)}$

An Example that shows that a potential greedy algorithm Candidate does not work.

Suppose we order edges so that $\frac{a(e_1)}{b(e_1)} \leq \frac{a(e_2)}{b(e_2)} \leq \ldots$

and select include edge, in that order unless the new edge creates a cycle (in which case we decide not include that edge).

Consider the example on the next page to show that this does not always produce the desired result.
\[ \sum a(e) = \frac{n + 9}{2n + 14} \rightarrow \frac{1}{2} \quad \text{as} \quad n \to \infty \]
Recall that only operation used in all algorithms for finding MST is Comparison. So, once an ordering of the edges is given, MST is uniquely determined.

We use this and the process of defining a family of regular MST problems to solve the ratio MST problem using a procedure which is due to W. Dinkelbach & R. Jagannathan for general ratio optimization.

Given $a(e), b(e); e \in E$, let us define a family of problems with:

$$W(e, \lambda) = a(e) - \lambda b(e) \quad e \in E$$

Now let $T^*(\lambda)$ be a min. Sp. tree on $G$ with weights $W(e, \lambda); e \in E$.

We discuss some properties of parametric MST in relation to Ratio MST.
Let $w_T(\lambda) = \sum_{e \in T} w(e, \lambda)$

And let $w^*(\lambda) = \min_T w_T(\lambda) = \frac{w_T^*(\lambda)}{\text{over all sp. trees of } T}$

Then if $\lambda > \lambda'$, $w_T(\lambda) < w_T(\lambda') \forall T$

Since $b(e) > 0 \forall e \in E$. \uparrow

\[w^*(\lambda') = \frac{w(\lambda')}{T^*(\lambda')} > \frac{w(\lambda)}{T^*(\lambda)} = \frac{w(\lambda)}{T^*(\lambda)} \quad \text{best tree for } \lambda'

\frac{w^*(\lambda)}{T^*(\lambda)} \quad \text{best tree for } \lambda

\therefore w^*(\lambda) \text{ is non-increasing function of } \lambda.

Lemma [Dinkelbach, Jagannathan]

Let \[\min_T \left\{ \frac{\sum_{e \in T} a(e)}{\sum_{e \in T} b(e)} \right\} = \lambda^* \quad ; \text{Then} \]

\[w^*(\lambda) = \begin{cases} > 0 & \lambda < \lambda^* \\ = 0 & \lambda = \lambda^* \rightarrow T^*(\lambda) \text{ is opt for RMST} \\ < 0 & \lambda > \lambda^* \end{cases} \]
Proof:

(i) \( \lambda > \lambda^* \):

\[
\lambda > \lambda^* = \frac{\sum b(e)}{\sum b(e)}
\]

\[
\therefore w(\lambda) < 0
\]

\[
\therefore w^*(\lambda) < 0
\]

(ii) \( \lambda < \lambda^* \):

\[
\lambda < \lambda^* = \frac{\sum a(e)}{\sum b(e)} \leq \frac{\sum a(e)}{\sum b(e)} \quad \forall T
\]

\[
\therefore w(\lambda) > 0 \quad \forall T
\]

\[
\therefore w^*(\lambda) > 0
\]

The case \( \lambda = \lambda^* \) follows from both of these, and \( w^*(\lambda) \) is a continuous function.
Recall \( w(e, \lambda) = a(e) - \lambda b(e) \)

For fixed \( \lambda \), finding \( T^*(\lambda) \) is MST problem and hence \( \exists \) an algorithm that uses only comparison between \( \{ w(e, \lambda) \} \); i.e.

Comparison of linear functions by \( \lambda \).

As we change \( \lambda \), the orderings of \( \{ w(e, \lambda) \} \) can only change at points of the type

\[ w(e', \lambda) = w(e, \lambda) \quad e \neq e' \]

i.e.

\[ a(e') - \lambda b(e') = a(e) - \lambda b(e) \]

\[ \Rightarrow \lambda = \frac{a(e) - a(e')}{b(e) - b(e')} \quad \text{assuming} \quad b(e) - b(e') \neq 0 \]

\[ \lambda(e, e') \]

Only \( O(|E|^2) \) values

Moreover, \( w^*(\lambda) \) is monotone non-decreasing.

Hence \( \exists \) a \( \text{polytime} \) algorithm to solve this problem. Best order in N. Megiddo's work.
Previous work applies to all problems which for the non-ratio optimization have a \( SP \) alg.
using only comparisons.

Now we turn to those that use additions and comparisons for the linear optimization.

This includes: Ratio trees, Ratio cycles, Ratio p. matchings, etc. \[ \text{[Caution: Ratio SP is NP-complete]} \]

Suppose \( \exists \) an algorithm that uses \( \Theta(p(n)) \) p(n) additions and \( \Theta(q(n)) \) comparisons to solve the problem:

\[
\min \left[ a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \right] \\
\text{subject to} \quad x \in S, \quad \forall a \in \mathbb{R}^{n+1}
\]

Where \( S \) is a set of Combinatorial Objects. Such as: set of spanning trees in a graph, perfect matching in a graph etc. \( x \) is the indicator vector of feasible solutions.

We assume \( p(n), q(n) \) are polynomial functions and work for arbitrary \( [a_0, a_1, \ldots, a_n] \).
Theorem (Megiddo).

There is an algorithm that solves the problem

\[
\begin{align*}
\text{Min} & \quad \frac{a_0 + a_1 x_1 + \ldots + a_n x_n}{b_0 + b_1 x_1 + \ldots + b_n x_n} \\
\text{subject to} & \quad x \in S
\end{align*}
\]

in time \( \Theta \left( \min \left( q(n), (p(n)+q(n)) \right) \right) \) assuming

\[ O \left( q(n) (p(n)+q(n)) \right) \]

that denominator is always positive.

We use Dinkelbach-Jagannathan result again:

Let optimal value be \( \lambda^* \). Consider the

parametric problem:

\[ W(\lambda) = \min \left\{ (a_0 - \lambda b_0) + (a_1 - \lambda b_1) x_1 + \ldots + (a_n - \lambda b_n) x_n \right\} \]

Since denominator is > 0 \( \forall x \in S \), it follows that

\( W(\lambda) \) is a monotone decreasing function of \( \lambda \).

Let algorithm \( A \) solve the linear problem in

\[ O(p(n)) \] comparisons and \( O(q(n)) \) additions.

Remark: Additions preserve linearity.
Lemma (D, J):

\[
W(\lambda) = \begin{cases} 
  > 0 & \lambda < \lambda^* \\
  = 0 & \lambda = \lambda^* \\
  < 0 & \lambda > \lambda^*
\end{cases}
\]

Proof: Similar to Ratio Sp. Tree Case.

This result lets us know which region to search for \( \lambda^* \) given \( W(\lambda) \) (rather its sign).

In algorithm A, whenever there is a comparison, it is a comparison of two linear functions of \( \lambda \). These two linear functions intersect at at most one value of \( \lambda \). If we know \( W(\lambda) \) (or its sign), then we can restrict ourselves to one of these regions, where the result of the comparison does not change.

Since there are \( O(p(n)) \) comparisons in algorithm A, we need to solve that many \( W(\lambda) \) problems, each of which take \( O(p(n)) \) comparisons and \( O(q(n)) \) addition. Hence, in \( O(p(n) (p(n)+q(n))) \) time, we can restrict our search for \( \lambda^* \) to an interval where \( W(\lambda) \) is linear.
Solving for $w(\lambda) = 0$ in this interval (solving a linear equation in $\lambda$) we get $x^*$ which in turn gets us $x^* \in S$.

Please be careful when reading N. Megiddo's paper. This cannot be used to solve ratio simple shortest path problem as claimed. This is because, in the presence of negative cycles, we cannot find shortest simple path in poly time. We can't even detect if there is a negative simple path which is what we need. However, we can do this latter part for shortest cycle (or detect negative cycle!) if Ratio Cycle problem is poly time solvable. There is no such difficulty for Ratio trees, Ratio Perfect Matching etc.

This result is immensely useful in many contexts.