Edmonds' Minimum Weight Perfect Matching Algorithm

Luis Goddyn, Math 408

1 The Linear Program

Let P be an even set of points in the plane. We will describe an algorithm which finds a $(0,1)$-solution to the following linear program. Since all perfect matchings of P correspond to feasible solutions, the solution found will correspond to a minimum weight perfect matching of P.

$$\begin{aligned} \min dx & & \\ x(\delta(u)) &= 1, & \text{for all } u \in P \\ x(\delta(S)) &\geq 1, & \text{for all } S \subseteq P,\ |S| \text{ odd} \\ x_{uv} &\geq 0, & \text{for } u, v \in P,\ u \neq v \end{aligned}$$

Here, the variables are $x = (x_{uv} : u, v \in P, u \neq v)$, and $d = (d(u,v) : u, v \in P, u \neq v)$ are distances between points. As usual, $\delta(S)$ denotes all pairs $u, v \in P$ such that exactly one of u, v belongs to S. We write $\delta(u)$ for $\delta(\{u\})$, and $x(\delta(S))$ means $\sum\{x_{uv} : uv \in \delta(S)\}$. We interpret $x_{uv} = 1$ to mean that u and v are matched, while $x_{uv} = 0$ means they are unmatched.

There is a dual variable r_u for each $u \in P$, which we interpret as the radius of a disc centered at u. There is a variable w_S for each odd subset $S \subseteq P$, which we interpret as the width of a moat surrounding the points in S. The dual linear program is as follows.

$$\begin{aligned} \max \sum\{r_u : u \in P\} + \sum\{w_S : S \subseteq P, |S| \text{ odd}\} & & \\ r_u + r_v + \sum\{w_S : u, v \in \delta(S)\} &\leq d(u,v), & \text{for all } u, v \in P, u \neq v \\ w_S &\geq 0, & \text{for all } S \subseteq P, |S| \text{ odd} \end{aligned}$$

Provided that $|P|$ is even, the algorithm always stops with an optimum $(0,1)$-solution for x, and an optimum dual solution (r, w) (in the form of a disc/moat packing).

Optimality is checked by observing that both x and (r, w) are feasible, and that the *complementary slackness conditions* hold:

- If $x_{uv} > 0$, then $r_u + r_v + \sum\{w_S : u, v \in \delta(S)\} = d(u,v)$. (That is, if two points are married then their discs and moats are tangent.)
- If $w_S > 0$, then $x(\delta(S)) = 1$. (That is, every positive-width moat is crossed exactly once.)

2 Definitions and Invariants

At any time during execution of the algorithm we have the following.

- a disc of radius r_u centered at each point $u \in P$,
- a moat having width $w_S \geq 0$ surrounding certain odd subsets $S \subseteq P$, where $3 \leq |S| \leq |P| - 3$.
- for any two moats, their subsets are either disjoint, or one contains the other.
- a set L of tight lines. (A line uv is *tight* if $r_u + r_v + \sum\{w_S : Su, v \in \delta(S)\} = d(u,v)$.)
- a set M of married (matched) pairs of points, where $M \subseteq L$.
- each moat surrounds a blossom. (A *blossom* contains an odd number of objects. Each object is either a point or a smaller blossom. These objects, which we call disc/moats, are joined in a circular fashion by tight lines in L. All of the points within each blossom are married in pairs, except for one point (the *base* of the blossom). The base is either unmarried or is married to a point outside of the blossom.)
- If two points are married to each other, then the two outermost discs or moats surrounding them are considered to be married to each other.

- each outermost disc or moat is either unmarried, or is married to some other outermost disc or moat. Any disc/moat which is not outermost is married.
- some of the unmarried outermost discs or moats are *roots* of trees.
- each tree consists of exactly one root, and a set of married pairs of outermost discs and moats, all joined in an alternating tree-like fashion by lines in L and M.
- to expand/shrink a tree T is to increment (decrement), by a constant amount, the width of each outermost moat or disc in T which is an even (odd) distance from its root.

3 The Algorithm

0. Input points P. Initialize all disk to have radius zero. (Alternatively, initialize the radius r_v of the disc of each point v to equal half the distance to its nearest neighboring point.) Initially, there are no blossoms or moats. The current matching M and the set L of tight lines are both empty.

1. If all of the outermost discs and moats are married, then exit with the perfect matching M and the current disc/moat packing. Else select at least one of the unmarried outermost discs or moats to be tree roots. Any trees which already exist from a previous iteration can be kept at this point.

2. Expand/shrink all of the trees until one of the following happens.

   Case 1 The width of a shrinking moat W goes to zero. Here the blossom of W dissolves. Some of the disc/moats surrounded by W will replace W in T; these disc/moats form a path which is chosen in such a way that the new tree is still alternating. One or two tight edges in the dissolving blossom will leave L. Repeat step 2.

   Case 2 An expanding disc/moat R from one of the trees T collides with another outermost disc/moat R'. Add to L the edge joining the two colliding points.

   Subcase 2.1 Both R and R' belong to the same tree T. Here we make a new blossom B by tracing back toward the root of T starting from both R and R' to find the odd cycle disc/moats that will form B. A new moat W having width zero surrounds all the disc/moats in B. Modify T by replacing all the disc/moats within B by the new moat W. Repeat step 2.

   Subcase 2.2 The disc/moat R' is either unmarried or (belongs to a different tree $T' \neq T$). Go to step 3.

3. We now find an augmenting path joining the unmarried point in the root of T to the unmarried point in R' (or in the root of T'). Traverse T from R, back toward its root. If a blossom is encountered in this traversal, then we traverse around it in whatever direction makes an alternating path. (If another tree T' is involved in the collision, then we do the same for T', and joint the two traversals together to obtain the alternating path.) Finally we augment the matching M along this path. Dismantle the trees structures T (and T'), leaving a set of married pairs of outermost disc/moats. Go to step 1.

4 Analysis

Suppose n points were input. Because moats can never "cross" each other, one can show that the greatest number of moats one can possibly have is $n-1$.

Step 3, and thus Step 1, is executed at most $O(n)$ times since each augmentation causes the number of married points to increase by 2. Step 2 may be repeated at most $O(n)$ times before going on to Step 3. To see this, we notice that any new blossom formed in Subcase 2.1 continues to grow in the tree T and will not dissolved while T still exists. Thus Subcase 2.1 can only be executed at most $O(n)$ between augmentations. As Case 2.2 decreases the total number of blossoms, it can be repeated at most $n-1$ times plus the number of times Subcase 2.1 is executed between augmentations.

In total, each step of the algorithm is executed at most $O(n^2)$ times. Using basic data structures, each step can be implemented to run in $O(n^2)$ time, so the algorithm stops in $O(n^4)$ time. It is known that this can be reduced to $O(n^3)$ time by the careful use of modern data structures.