2016

Drawing DNA Sequence Networks

Julia Olivieri
Oberlin College

Drawing DNA Sequence Networks

Julia Olivieri

June 21, 2016

Abstract

We explore methods for drawing a graph of DNA sequences on a digital canvas such that the Euclidean distances between sequences on the canvas suggest the distances between the sequences as calculated from pairwise sequence alignment. We use data from three plant taxa, the genus *Castilleja* as well as the families Caryophyllaceae and Cactaceae, to test our methods. We discuss different possible measures of the cost of a drawing, and analyze heuristic approaches to the problem including random assignment, greedy assignment, the iterated hill-climber, and simulated annealing. We find that our hill-climbing method tends to return superior drawings. Our simulated annealing method also returns drawings with low costs, but in much less time than the hill-climbing method for large datasets.

1 Introduction

1.1 Why visualize DNA sequence similarity?

Today we are facing a huge influx of newly sequenced DNA available to the public for analysis. This has been instrumental in many recent biological advances, from creating flu vaccines to cloning sheep. One particularly fascinating application is the use of DNA sequences to infer evolutionary histories of organisms, including how recently different species diverged from a common ancestor.

These relationships are usually inferred by gathering DNA sequences for the groups in question and determining the most likely chain of events that could give rise to the sequences, creating a phylogeny. Then the phylogeny is displayed with a bifurcating phylogenetic tree.

Phylogenetic trees provide useful information about relationships among sister groups, but it would be valuable to have other strategies for portraying connections among sequences.

For one thing, phylogenetic trees far from portray the whole story. True trees of life often aren't strictly bifurcating: events such as hybridization and horizontal gene transfer can cause the true relationships to be more net-like. Also, finding alternate ways of portraying connections among DNA sequences visually could encourage new perspectives on the data.

There have been some studies that analyze population graphs of species and some that attempt to portray individual DNA sequences visually [1, 2]. However, we have not found any research attempting to graphically represent DNA sequence networks using the problem construction we are investigating here. We will explore different ways of representing DNA sequences on a canvas in a way that suggests relative differences among the DNA sequences. With so much new information, new ways to process and understand it are important.

1.2 Data sets

We choose three plant groups to use as case studies. Plants often have more entangled evolutionary histories than animals due to the rampancy of genome duplications and hybridization between taxa.

Castilleja

As botanist Harold William Rickett put it, "The genus *Castilleia*[1] is one of those that make botanists wish they had embraced some easy branch of science such as theoretical physics" [6]. It is in the Orobanchaceae family, which includes mostly herbaceous parasitic plants. This family is the largest and most diverse of all parasitic angiosperms (flowering plants), so understanding its phylogeny could throw light on which factors made it so successful [3]. Also, because the evolutionary history of *Castilleja* is so convoluted we may benefit from alternate methods of analysis to provide us with a clearer picture of its ancestral relationships. We have 19 sequences in our *Castilleja* dataset. See [3] for a phylogenetic tree.

Caryophyllales

This plant order accounts for 6% of angiosperm diversity, with about 11510 species across 34 families. This order originated between 67 and 121 million years ago, and contains species that span all seven continents and all terrestrial ecosystems. The amazing diversity of plants in this order makes it fascinating to study their phylogenetic history [7].

We chose two families from this plant order to include as datasets, Caryophyllaceae and and Cactaceae. Caryophyllaceae contains mostly small herbaceous plants which have complex morphological differences, making it difficult to determine the relationships between

[1] alternative spelling, genus names are italicized

them [4]. We have 16 sequences in our Caryophyllaceae dataset. We also chose the cactus family, which has between 1500 and 1800 species. Because many cacti need to survive under similar environmental pressures they can end up displaying common features that may not be related to shared genetic history [5]. We have 82 sequences in our cactus dataset. Both of these groups have complicated phylogenetic histories, so visualizing their relationships in a different way could help us understand them better. See [4] for a Caryophyllaceae phylogenetic tree and [5] for a Cactaceae phylogenetic tree.

1.3 Method for comparing sequences

Now that we have our datasets we need some way of accessing their DNA sequences and comparing them. There is a very large searchable database of DNA sequences on the website of the National Center for Biotechnology Information [8]. We found a subset of the exact sequences used to create the phylogenies in Figure 1 by searching their accession numbers in this database (see Appendix 2 for accession numbers).

We need a method for calculating the difference between two DNA sequences, s_1 and s_2. DNA sequences are made up of four nucleotide bases (*bases* for short), so they are denoted as strings of the letters A, T, G, and C. Because we can have insertions and deletions, we cannot simply line up two sequences and count the number of base differences. For example, the minimum-cost alignment of ATGCAA and TGGATA is

A T G C A – A
– T G G A T A

where we insert *gaps* (symbolized by – here) to create an optimal alignment.

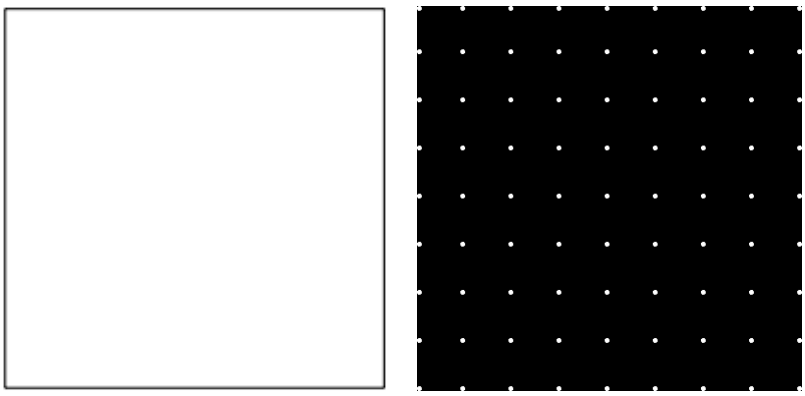
We can see that any alignment will pair bases of s_1 up with bases of s_2, although sometimes a base will be paired with a gap (a gap can never be paired with another gap). Then to find the cost of an alignment we can assign a value g as the gap penalty and a value σ as the substitution penalty (we used $g = 3$ and $\sigma = 2$ to compare our sequences). Because a substitution is more likely than an insertion or deletion, we will assume $\sigma < g$. Then we will have some number of gaps paired with bases (such pairs cost g), some number of bases paired with mismatched bases (such pairs cost σ) and some number of bases paired with the same base (such pairs cost 0). Adding all these costs together yields the cost of the alignment. For example, the alignment above has cost $2g + \sigma$. For larger pairs of sequences, the standard approach for finding an optimal alignment and its cost is to use dynamic programming [9].

Now that we have a method of comparing sequences we can approach the problem of drawing the sequences.

2 Graph Drawing

2.1 Formulation of the problem

We can represent DNA sequences as nodes in a weighted complete graph G. Let S be the set of DNA sequences, let $|S| = n$, and let the elements of S be $s_1, \ldots, s_n$. If $s_i, s_j \in S$, let the weight w_{ij} of the edge $s_i s_j$ be defined as the cost of an optimal alignment of the two sequences, calculated as shown above (note that $w_{ij} = w_{ji}$).



(a) (b)

Figure 1: It is possible to place a sequence in any area that is white, and impossible to place a sequence in any area that is black. (a) In the continuous version all placements on the canvas are possible. (b) An example of the discrete case described above with $l = 8$.

Points

We have a square digital canvas that has height and width l. We will define the bottom left corner of the canvas to be $(0, 0)$ and the top right corner to be (l, l). Let P be the set of *possible point locations* (*points* for short) on the canvas. These will be the places where we can draw nodes representing sequences in S. For $p \in P$, let (p_x, p_y) specify the point's x- and y-coordinates on the canvas. Then we can define P continuously as

$$P = \{(x, y) \,|\, 0 \leq x \leq l, 0 \leq y \leq l\}$$

as shown in Figure 1a. However, it will be useful for us to consider discrete subsets of the continuous case. For example, we can consider the case where

$$P = \{(x, y) \,|\, 0 \leq x \leq l, 0 \leq y \leq l, x, y \in \mathbb{Z}\}$$

as shown in Figure 1b.

Drawing

Let D be an injective mapping from S to P such that $D(s_i) \in P$, where $D(s_i)$ denotes the point where we will draw s_i. Let d_{ij} be the Euclidean distance between $D(s_i), D(s_j) \in P$, so

$$d_{ij} = \sqrt{(D(s_i)_x - D(s_j)_x)^2 + (D(s_i)_y - D(s_j)_y)^2}.$$

Our goal is to draw all the elements of S on the canvas such that the Euclidean distance between $D(s_i)$ and $D(s_j) \in P$ is in accordance with w_{ij} (up to scale). Preferably we would find a drawing that could be scaled such that $w_{ij} = d_{ij}$ for all $s_i, s_j \in S$. This would be called the *ideal* drawing I.

We know Euclidean distances satisfy the triangle inequality. One question immediately comes to mind: do the weights?

Claim 1 *Weights on G obey the triangle inequality.*

Proof. Let $s_i, s_j, s_k \in S$. Then we can change w_{ij} nucleotides of sequence s_i to make it identical to s_j. We also know that we can make w_{jk} changes to s_j to make it identical to s_k. Therefore we can transform s_i into s_j and then into s_k in at most $w_{ij} + w_{jk}$ nucleotide substitutions. Therefore $w_{ik} \leq w_{ij} + w_{jk}$, so weights on G obey the triangle inequality.

□

So we can see that the weights satisfy the triangle inequality. Does this mean that we will be able to draw all of our graphs ideally?

Claim 2 *An ideal placement does not always exist.*

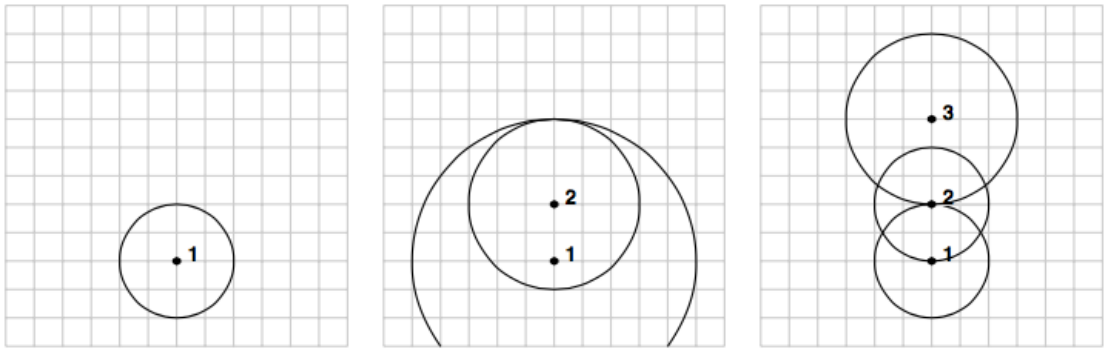
Proof. Let's look at a simple example. Let's say we have the following sequences:

1. AAAAA
2. ATATA
3. TTTTT
4. AAATT

Using a substitution penalty of $\sigma = 1$ and a gap penalty of $g = 2$ we get the weight matrix:

$$W = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0 & 2 & 5 & 2 \\ 2 & 0 & 3 & 2 \\ 5 & 3 & 0 & 3 \\ 2 & 2 & 3 & 0 \end{pmatrix} \end{array}.$$

Now, let's position s_1 at point $(6, 3)$. We can then place the second sequence anywhere on the circle of radius 2 around the origin (Figure 2a). Without loss of generality let's place it at $(6, 5)$. Now we must place s_3 and we have only one choice, $(6, 8)$ (see Figure 2b). Now we must place s_4. As we can see in Figure 2c, the circles around the sequence placements with radii corresponding to that sequence's distance from s_4 do not have a common intersection point. Therefore there is no place to position s_4 ideally, and therefore an ideal drawing does not exist in this case. □



(a) (b) (c)

Figure 2: Trying to place sequences on the canvas to make an ideal drawing.

Now we can see that the problem is more complicated that it initially seemed. Our goal is to find optimal (or close-to-optimal) drawings. Finding provably optimal drawings is extremely difficult. Our focus instead is on developing heuristic methods that produce close-to-optimal drawings.

2.2 The Quadratic Assignment Problem

This question is a variation on the quadratic assignment problem (QAP), a core problem of combinatorial optimization that is NP hard. In this problem, there are n facilities and n locations in which these facilities can be placed. We are then provided with information about the distance between each pair of locations and the flow between each pair of facilities. The idea is to place the facilities in locations to minimize the sum of distances multiplied by their flows. The QAP has many applications ranging in field from sports, to chemistry, to archaeology, and including everything from circuit wiring to typewriter keyboard design [10].

In our problem we have sequences rather than facilities and points rather than locations. Also, rather than flow between facilities we have weights due to nucleotide differences. Therefore, whereas in the QAP a higher flow means the facilities should be closer together,

in our problem a higher weight means the sequences should be further apart. Also, our weights obey the triangle inequality, which isn't guaranteed for flow. Additionally, we allow more points than sequences (analogous to having more locations than facilities in the QAP).

In 1998, it was impossible to solve the QAP to optimality within reasonable time limits for cases larger than $n = 20$. This was at a time when much progress had been made on other NP complete problems: the TSP could be solved with thousands of cities, and the maximum clique problem could be solved on graphs with thousands of vertices and millions of edges [10]. Even as recently as 2013 problems with $n > 35$ can't be solved within reasonable computational time, and even finding an approximate solution within some constant factor of the optimal solution cannot be completed in polynomial time unless P=NP. This holds true even when the values obey the triangle inequality. In fact, even finding local optima is nontrivial [11].

Therefore, we need a different approach to solve our problem. We will use heuristic methods to approach optimal solutions.

2.3 Point placement

Our problem involves deciding where to place points on the canvas. By limiting the size of the possible point placements P (while always ensuring that $|P| \geq n$) we limit our options, making the problem more tractable.

This means that we must find some method of choosing the elements of P. Figure 1b shows one method. We experimented with several others. Each exhibits different costs, as well as different aesthetics (see Figure 4). For consistency we chose to use the same point configuration for all of our images here. We chose the circle formation because it avoids the possibility of edges running over points that they don't connect, which can be misleading.

2.4 Determining Cost

We want to find a drawing that approximates the ideal drawing. But what do we mean by "approximates the ideal drawing"? How can we tell if one drawing is closer to the ideal than another? To accomplish this we need some measure of the distance of a drawing from the ideal, or the *cost* of a drawing.

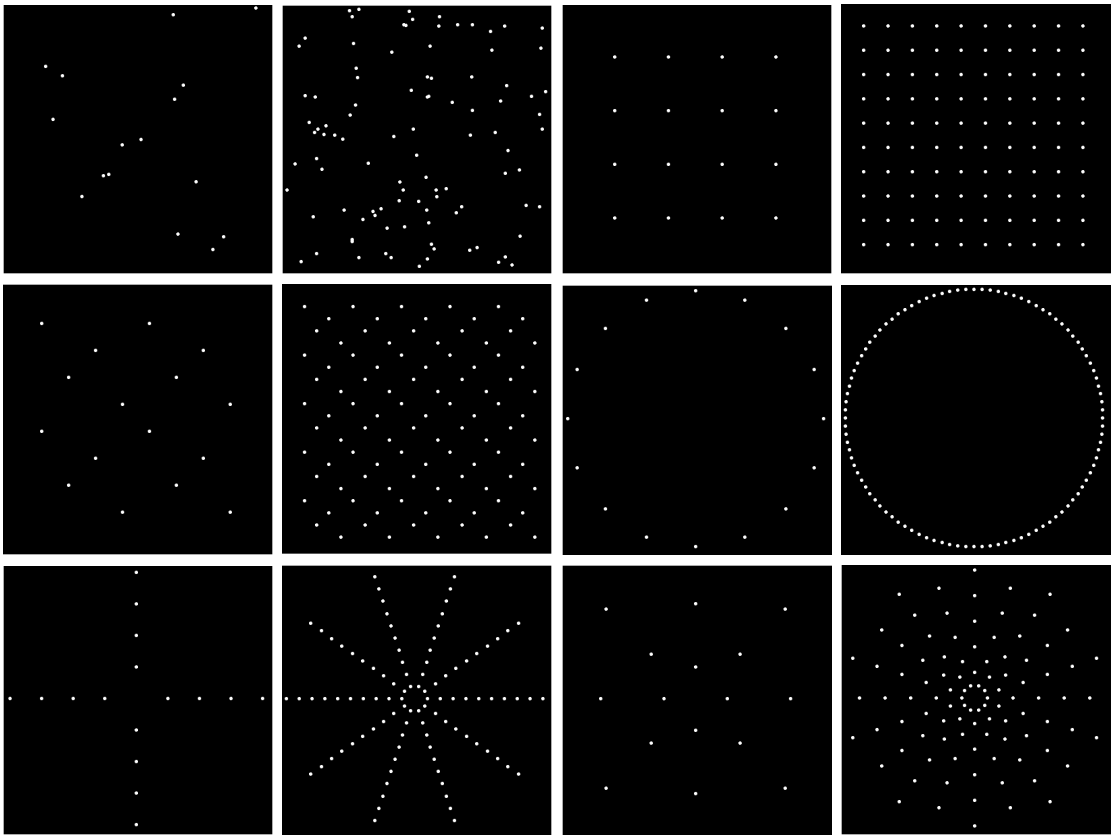


Figure 3: Several possible point placements, shown with both 16 and 100 nodes: random, square grid, skew grid, circle, multiple circles, skew multiple circles.

2.4.1 Euclidean cost

One possible measure of cost calculates the difference between the Euclidean distance between each pair of nodes on the canvas and the weight of each pair of nodes on the complete graph.

We can formulate this measurement as

$$C(D) = \sum_{s_i, s_j \in S} |kd_{ij} - w_{ij}|$$

where $k = \max(w_{ij})/\sqrt{2}l$. Dividing the maximum weight on an edge of G by the maximum distance between two points on the canvas gives us a scaling factor that makes distances comparable to weights. We can also adapt this cost more specifically to various point

configurations by using the maximum distance between two points in the configuration rather than $\sqrt{2}l$.

We can also use this method to calculate the cost of one node: for $s_i \in S$ that would be

$$C_{s_i} = \sum_{s_j \in S} |kd_{ij} - w_{ij}|.$$

This cost measure is attuned to even small changes in node placement. However, we may be interested in getting general placement right, without concerning ourselves too much with adjusting a specific placement. In that case we can try another method.

2.4.2 Ordering cost

A second possible measure of cost simply measures whether vertices are assigned positions on the canvas close to sequences that they have many nucleotides in common with.

Let $s \in S$ and let D be a drawing of S on the canvas. Then let $l_s = u_1, u_2, \ldots, u_n$ such that $u_i \in S$, each element of S appears exactly once in l_s, and $w_{s,u_i} \leq w_{s,u_{i+1}}$. Let $L_s = v_1, v_2, \ldots, v_n$ such that $v_i \in S$, each element of S appears exactly once in L_s, and $d_{D(s),D(v_i)} \leq d_{D(s),D(v_{i+1})}$.

If u_i is in l_s, then u_i must be in L_s. Let's say u_i is the entry v_j in L_s. Then $c_s(i) = |i - j|$.

Then define the cost of sequence s given D to be:

$$C_s = \sum_{i=1}^{n} c_s(i).$$

Then the cost of D is

$$C(D) = \sum_{s \in S} C_s.$$

In this cost system, the exact coordinate or distance on the canvas isn't important. A placement with cost 0 would merely mean that for any given sequence, the point closest to that sequence on the canvas would have the fewest number of nucleotide base differences from the first sequence, the sequence the next farthest away would have second fewest number of nucleotide base differences from the firs sequence, and so on for all points on the canvas.

2.4.3 Flow cost

This cost is informed by the QAP formulation. Instead of measuring based on nucleotide differences we can use nucleotide similarities. If this information is not readily available we can approximate it by taking the average length of our sequences L. Then for $s_i, s_j \in S$ the number of nucleotides in agreement between the two sequences, a_{ij}, is approximately $L - w_{ij}$.

Now, the higher a_{ij} is, the closer we want s_i and s_j. Therefore we can calculate

$$C(D) = \sum_{s_i, s_j \in S} d_{ij} a_{ij}.$$

Minimizing this cost measure encourages sequences that have more similarities to be closer together.

2.4.4 Cost choice

We report the same measure of cost for each image so that the costs can be directly compared. We chose the ordering cost because we are interested in the relative rather than absolute correct positioning. Also, the ordering cost yields the smallest values, so they are somewhat easier to understand and compare.

2.5 Heuristic methods

Given an $l \times l$ canvas, to find the optimal placement we might first think to try every combination of placing each $s \in S$ at each point in P and comparing the costs. However, if $|S| = n$ and $|P| = m$ with $n \leq m$ we can see that the number of possibilities is

$$\frac{m!}{(m-n)!}.$$

If $m = n = 13$ there are already $6,227,020,800$ possible placements. We need a different approach if we are to use sets S of nontrivial size.

Because we cannot feasibly analyze every possible drawing given nontrivial sets S and P, we need a method to help us decide which drawings to analyze to give us close-to-optimal solutions. We turn to heuristic algorithms. Results of these methods are displayed in Table 1 and Figures 6, 7, and 8.

2.5.1 Random assignment

We can search indiscriminately through the space of drawings by choosing many random drawings and recording the drawing with the lowest cost. One run of the random assignment procedure consists of assigning each $s \in S$ to a $p \in P$ that has yet to be assigned to a sequence, creating drawing D. We then save the cost of D. We perform some predetermined number of runs r, each time calculating the cost and saving any D with a minimum cost.

This method does not retain any information from one run to the next, so it is truly a blind search. Assuming there is only one optimal solution, the chance of landing on it is

$$1 - \Big(1 - \frac{(|P| - n)!}{|P|!}\Big)^r.$$

This means that if $n = |P| = 10$ and $r = 100000$, there is only about a 2.72% chance of obtaining the optimal solution (assuming there is one optimum). However, a benefit of this method is that the program doesn't get bogged down in any local optima.

2.5.2 Greedy assignment

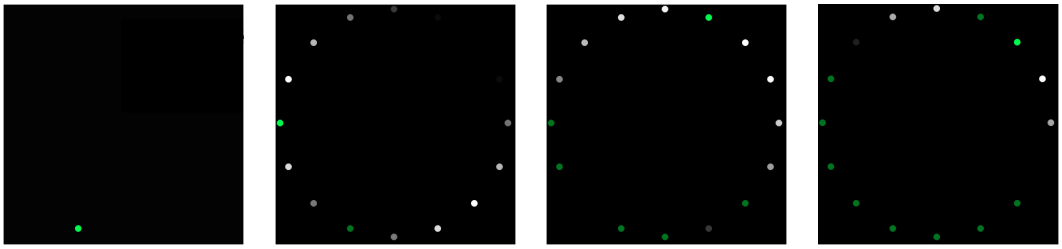
The idea of this method is to assign one sequence at a time to its 'best' placement on the canvas.

For one run we start by randomly choosing an order for the elements in S. Assume s_i is at the ith place in this order. Then we randomly choose $D(s_1)$ from P. We can then calculate the Euclidean cost of the single node placement of s_2 at p for each $p \in P$ that has not yet been assigned a sequence. We then let $D(s_2)$ equal the $p \in P$ for which the Euclidean cost of the sequence was smallest (if there are multiple of these points, choose one randomly). We repeat this procedure for the remaining sequences.

2.5.3 Iterated hill-climber

Hill-climbing algorithms search a neighborhood of solutions that are close to the current solution in some way and find the most optimal solution in that neighborhood [12]. They are iterative methods, so the process is repeated until a local optimum is found.

A *2-swap* of s_i and s_j is a switch from the current map D to a new map D' for which $D'(s_i) = D(s_j)$ and $D'(s_j) = D(s_i)$ for some $s_i, s_j \in S$, and $D'(s_k) = D(s_k)$ for all other $s_k \in S$.



(a) (b) (c) (d)

Figure 4: Performing the greedy method for Caryophyllaceae. Green circles are points that have been assigned sequences, light green circles are most recently assigned. Other circles are possible point placements, with lighter meaning lower cost for the light green circle being placed there. (a) Adding the first sequence randomly. (b) Adding the second sequence. (c) Adding the sixth sequence. (d) Adding the tenth sequence.

Each run of the hill-climber starts with a random assignment of sequences to points. Then for every $s_i, s_j \in S$ calculate the cost of the drawing resulting from the 2-swap of s_i and s_j. Next, we find the minimum of all of these costs and use that map for the next round. We continue like this until there is no 2-swap that decreases the cost.

The iterated hill-climber is guaranteed to find a local optimum. However, the danger lies in the possibility of getting stuck in local optima. Multiple runs starting with random configurations helps avoid this. If the same number of runs are used for random assignment and the iterated hill-climber, the the iterated hill-climber is guaranteed to find of solution of greater than (or equal) optimality, because it only improves each run's cost. However, random assignment is much faster than the iterated hill-climber.

A 3-or-more swap could lead to finding a more optimal solution in the same number of runs. Let's define a *3-swap* as a switch from the current map D to a new map D' for which $D'(s_i) = D(s_j)$, $D'(s_j) = D(s_k)$, and $D'(s_k) = D(s_i)$ for some $s_i, s_j, s_k \in S$, and $D'(s_l) = D(s_l)$ for all other $s_l \in S$. Then there are $n(n-1)(n-2)/3$ possible 3-swaps for a graph with n nodes as opposed to $n(n-1)/2$ possible 2-swaps. Therefore, for $n = 16$ there are 1120 possible 3-swaps and only 120 possible 2-swaps. We can see that a 3-swap method would significantly increase the runtime, so we will pursue a different method.

2.5.4 Simulated annealing

We want to try some method that moves steadily towards optimality, but that still has enough randomness to not get stuck in every local optimum. Simulated annealing is a method that involves adding a new parameter called the temperature that changes the probability of moving from one point of the search space to another [12].

In the simulated annealing procedure, we choose some *temperature* $T > 1$, ratio $q \in (0, 1)$, and number of runs r. We start with a random drawing D. The cost of our current drawing is cur. Then we perform a 2-swap and find the cost of D', new. If new is the lowest cost we have calculated so far we save new as opt and D' as $optD$. If the new cost is less than our current cost cur, we save it as cur and D' becomes our current drawing. If not, we choose a random $b \in [0, 1)$. Then if

$$b < e^{(cur-new)/T}$$

D' still becomes our current drawing. Otherwise, we maintain D as our current drawing.

Note that $cur - new \leq 0$, because the new cost must be greater than or equal to the current cost (otherwise we would have automatically accepted it).

After we've done this r times, we set T equal to qT and perform the procedure again. We continue until $T \leq 1$. Then we return $optD$.

We settled on $T = 100$ and $q = 0.75$ for our datasets because they resulted in drawings of comparable cost to those generated by the hill-climbing method, but still allowed the procedure to finish in a reasonable amount of time. Adjusting the initial value of T and the value of q could yield better drawings, but could also take more time.

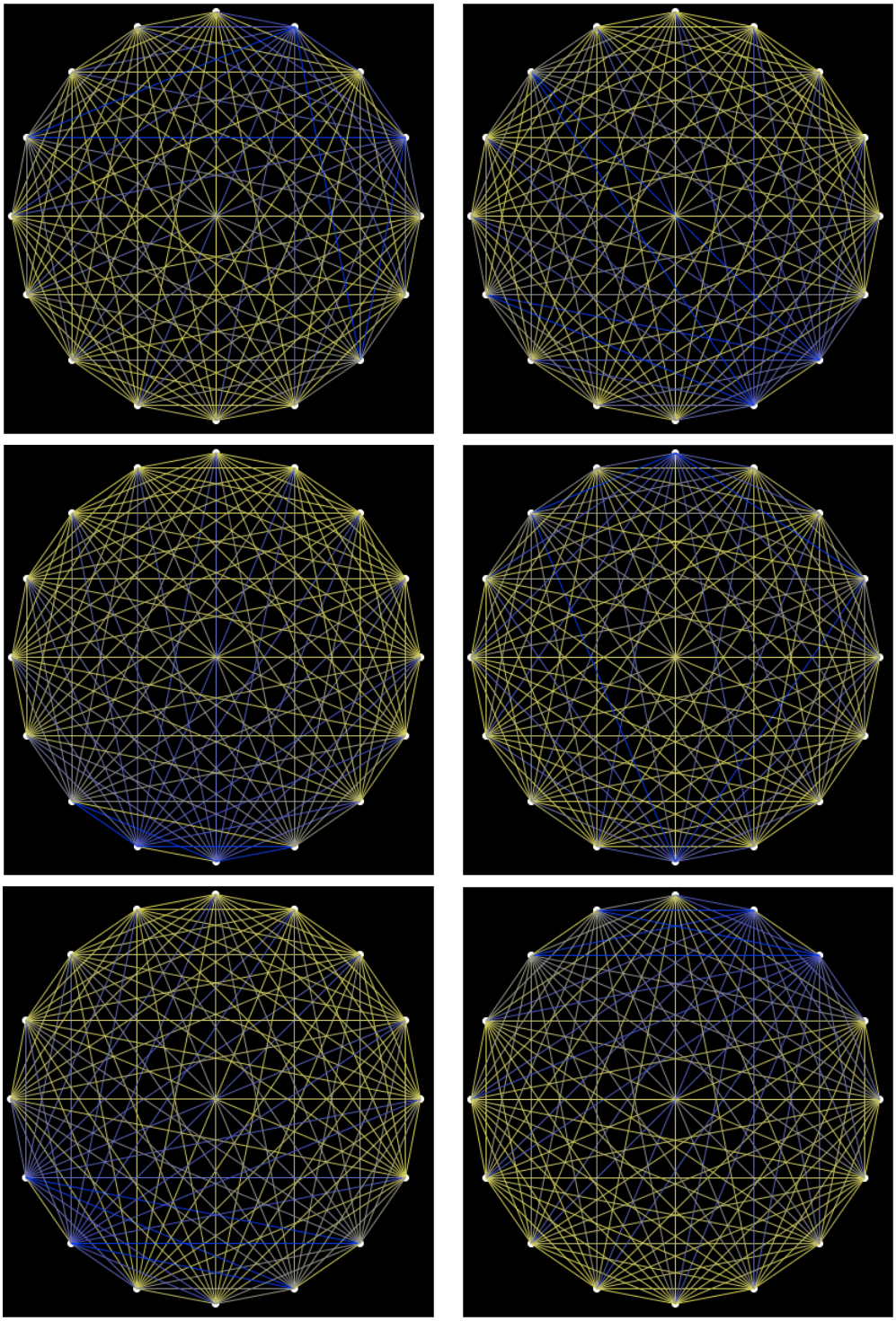


Figure 5: Caryophyllaceae images: For all images the color of each edge is scaled with sequence similarity, so the more yellow an edge is, the more similar the two sequences it connects are; the more blue it is, the more different the sequences it connects are. In order from left to right these are the images corresponding to the Caryophyllales results listed in Table 1.

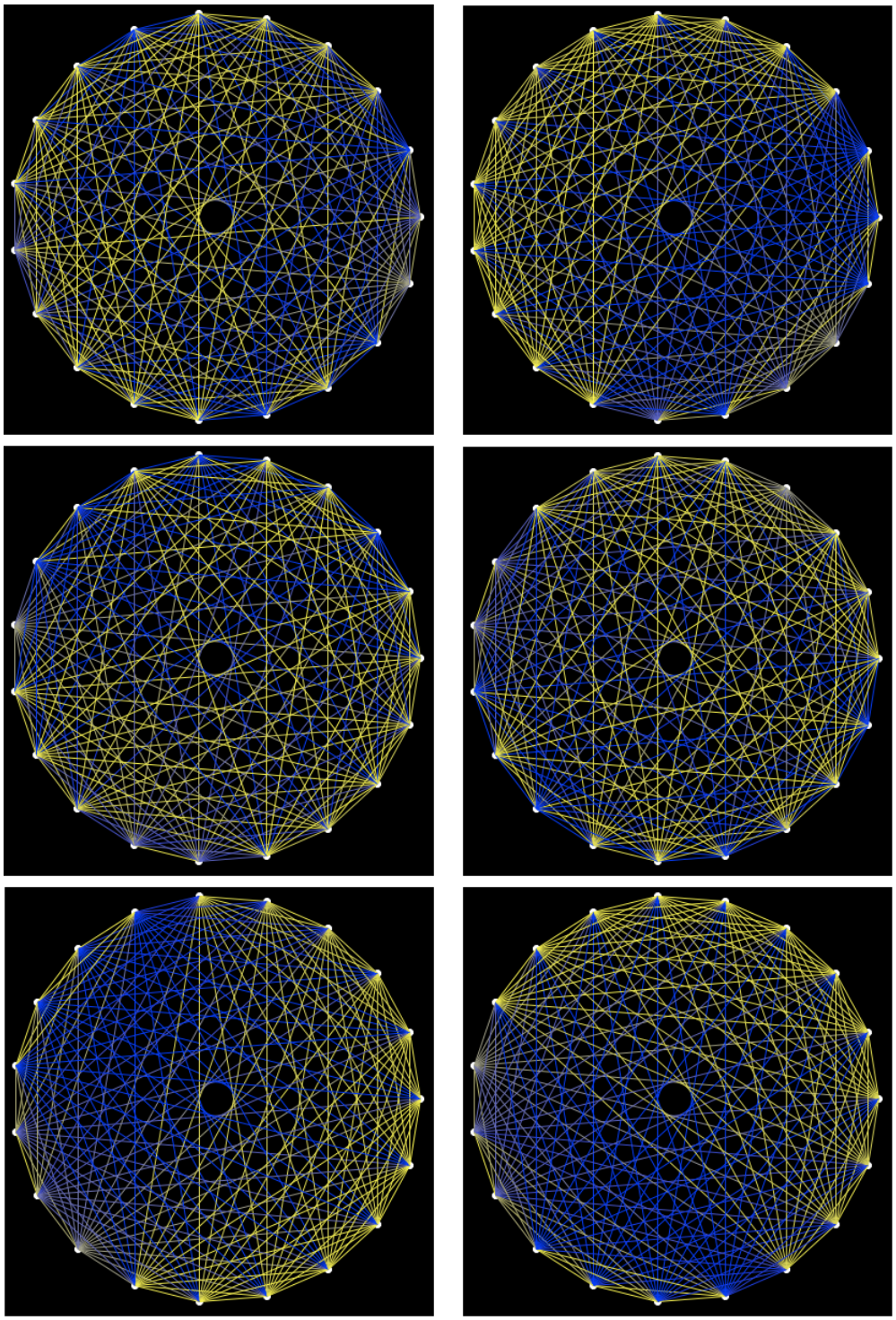


Figure 6: *Castilleja* images: In order from left to right these are the images corresponding to the *Castilleja* results listed in Table 1.

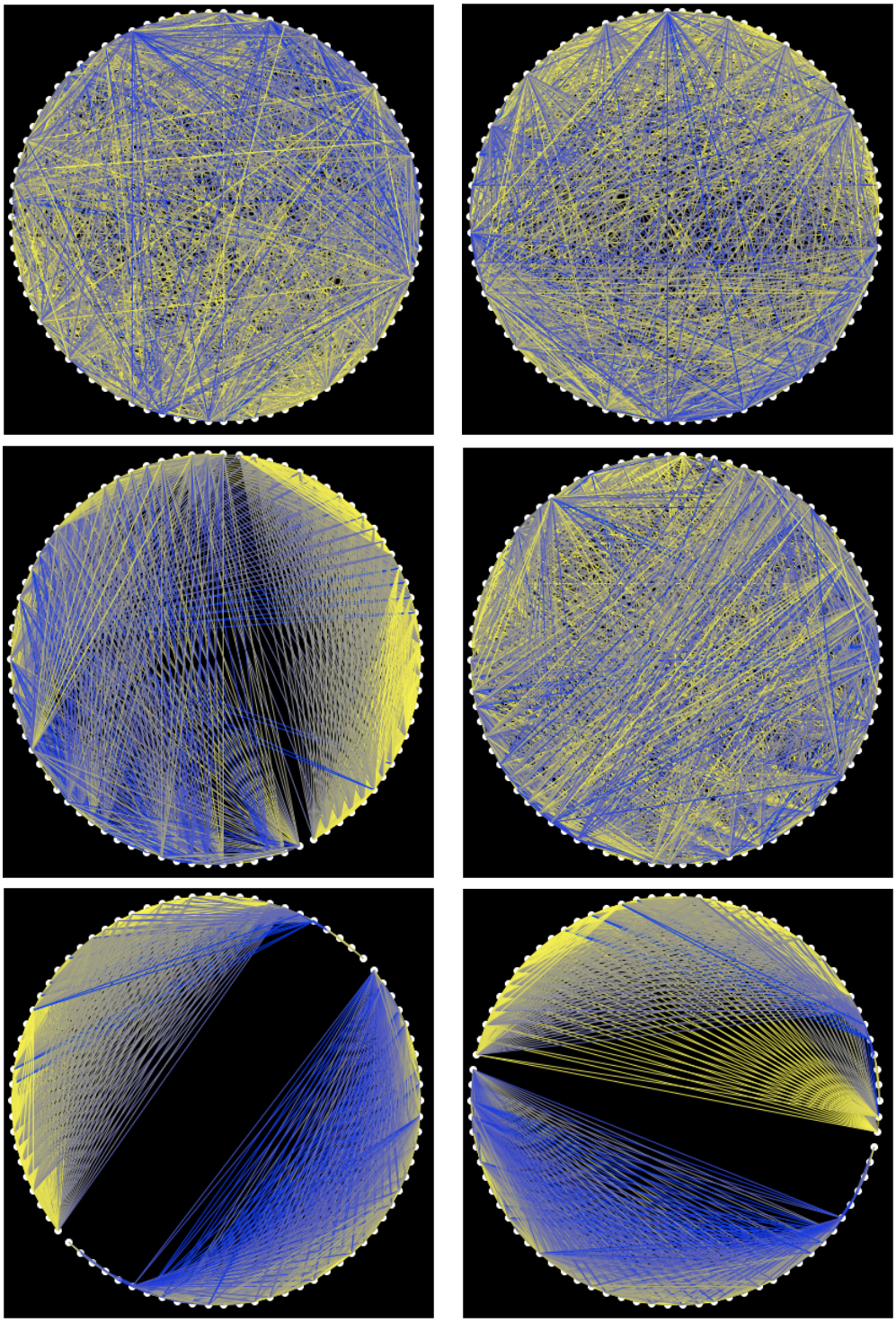


Figure 7: Cactaceae images: Not all edges are shown in these drawings; only those with a number of nucleotide differences less than or equal to the maximum number of differences divided by 5. This is to avoid many edges being completely covered up. In order from left to right these are the images corresponding to the Cactaceae results listed in Table 1.

Conclusion

We have now tried several methods for creating drawings of sets of DNA sequences. The question remains: which of the methods we have tried is best? The answer is not completely straightforward. Although the hill-climber consistently generated drawings with the lowest costs, we are also interested in differences in runtime that make some methods more feasible than others. See Table 1 for a breakdown of our results (the images corresponding to each of these results can be found in Figures 6, 7, and 8).

Dataset	**Result #**	**Method**	**Runs**	**Cost**	**Runtime (sec)**
Caryophyllaceae	1	random	1	1256	0.414
	2	random	10000	966	22.222
	3	greedy	1	1004	0.363
	4	greedy	10000	986	38.430
	5	hill climbing	10	856	31.750
	6	simulated annealing	1000	858	35.853
Castilleja	7	random	1	1958	0.547
	8	random	10000	1522	32.106
	9	greedy	1	2100	0.569
	10	greedy	10000	1430	59.848
	11	hill climbing	10	1164	78.945
	12	simulated annealing	1000	1180	53.047
Cactaceae	13	random	1	176386	11.903
	14	random	10000	164250	1304.604
	15	greedy	1	134830	8.035
	16	greedy	1000	170238	312.003
	17	hill climbing	1	96476	39303.943
	18	simulated annealing	1000	99678	2098.429

Table 1: Comparison of heuristic methods.

For results 3, 9, and 15 we didn't use a random ordering: instead, we used the order the sequences appeared on the phylogenetic tree.

For the small datasets, Caryophyllaceae and *Castilleja*, the greedy method in results 4 and 10 barely outperformed the random method in results 2 and 8, with slightly lower costs that took about double the time. The hill-climbing and simulated annealing methods performed best, both yielding comparable costs and runtimes. Tweaking the parameters T and q, as well as the number of runs, for the simulated annealing method could yield superior results within a reasonable amount of time. However, strictly from our results here the hill climbing and simulated annealing approaches both seem viable for creating images of small datasets. In both of these datasets the ranges of costs are relatively small,

especially for Caryophyllaceae. The images in Figure 6 show little visual difference between various methods.

For the large dataset, Cactaceae, the hill climbing and simulated annealing methods also performed best, returning comparable costs. The hill climbing algorithm performing slightly better, but the methods exhibited vastly different runtimes. The hill climbing algorithm took nearly 11 hours to perform a single run, whereas the simulated annealing procedure was completed in less than 35 minutes. This drastic difference in runtimes coupled with the relative similarity of costs suggests that simulated annealing is the preferred method for finding drawings of larger datasets. The random and greedy approaches in results 14 and 17 returned drawings with costs almost double those found in results 17 and 18. However, result 15 has a surprisingly low cost. It seems that for large datasets such as this, if some information is known about the dataset a pre-designed ordering could yield a drawing with a relatively low cost in a very short amount of time. Figure 8 shows a large visual difference between images with low costs and high costs.

Future Directions

We can now create images where distances between sequence nodes approximate distances between sequences, given a set of DNA sequences. It is important to note that these images should not be used to infer relationships among taxa; phylogenetic trees represent that information more accurately. We are representing different information: the degree of similarity among these specific DNA sequences. This can provide us with information that phylogenetic trees do not claim to provide, such as the relative difference between each pair of sequences (rather than just the relationships between sister taxa).

So far our methods have only been tested on datasets with fewer than 100 sequences. We would like to be able to draw larger sets of sequences. Drawing thousands of sequences at a time would require modified methods to make the problem tractable, but the simulated annealing approach seems promising. We could also combine multiple methods, such as using greedy drawings to begin each run of the hill climber rather than random ones. Additionally, now that full genome sequencing is possible, it could be valuable to use full genomes, or at least multiple genes, as data for drawings. Also, comparing drawings of different genes for the same set of taxa could provide interesting information about differing gene histories. We would also like to explore more deeply what we can learn from these images. What conclusions can we draw? Are there measures of similarity between different drawings that can tell us something about the communities they come from?

We believe that this project can be appreciated both as a new variation on a classic combinatorial optimization problem, and as a new method in the field of DNA sequence analysis that could help shed some light on sequence relationships.

References

[1] Rodney J. Dyer and John D. Nason. Population graphs: the graph theoretic shape of genetic structure. *Molecular Ecology*, 13:1713–1727, 2004.

[2] Ashesh Nandy, Marissa Harle, and Subhash C. Basak. Mathematical descriptors of DNA sequences: development and applications. *Archive of Organic Chemistry*, 9:211–238, 2006.

[3] Jonathan R. Bennett and Sarah Mathews. Phylogeny of the parasitic plant family Orobanchaceae inferred from phytochrome A. *American Journal of Botany*, 93(7):1039–1051, 2006.

[4] Simone Fior, Per Ola Karis, Gabriele Casazza, Luigi Minuto, and Francesco Sala. Molecular phylogeny of the Caryophyllaceae (Caryophyllales) inferred from chloroplast *matK* and nuclear rDNA ITS sequences. *American Journal of Botany*, 93(3):399–411, 2006.

[5] Reto Nyffeler. Phylogenetic relationships in the cactus family (Cactaceae) based on evidence from *trnK/matK* and *trnL-trnF* sequences. *American Journal of Botany*, 89(2):312–326, 2002.

[6] Harold William Rickett. *Wild Flowers of the United States*, volume 1. Hinkhouse Inc, New York, New York, 1966.

[7] Samuel F. Brockington, Ya Yang, Fernando Gandia-Herrero, Sarah Covshoff, Julian M. Hibberd, Rowan F. Sage, Gane K. S. Wong, Michael J. Moore, and Stephen A. Smith. Lineage-specific gene radiations underlie the evolution of novel betalain pigmentation in Caryophyllales. *New Phytologist*, 207(4):1170–1180, 2015.

[8] National Library of Medicine. *BLAST®*, 2016. http://blast.ncbi.nlm.nih.gov /Blast.cgi.

[9] István Miklós. *Introduction to Algorithms in Bioinformatics.* Budapest, Hungary, 2016. http://www.renyi.hu/ miklosi/AlgorithmsOfBioinformatics.pdf.

[10] Eranda Çela. *The Quadratic Assignment Problem: Theory and Algorithms.* Springer Science+Business Media, B.V., Dordrecht, Holland, 1998.

[11] Rainer E. Burkard. *Handbook of Combinatorial Optimization.* Springer Reference, Media, New York, 2013.

[12] Zbigniew Michalewicz and David B. Fogel. *How to Solve it: Modern Heuristics.* Springer, Berlin, Germany, 2002.

Apendix 1: Terminology

term	*definition*
–	the gap symbol
$C(D)$	the cost of drawing D
D	an injective map from S to P (a drawing of S on the canvas)
$D(s_i)$	the element of P that D maps s_i to
d_{ij}	the Euclidean distance between $D(s_i)$ and $D(s_j)$
G	the complete graph with sequences as nodes and w_{ij} as the weight on edge $s_i s_j$
g	the gap penalty ($\sigma < g$)
I	ideal drawing: injective map from S onto P for which $w_{ij} = d_{ij}$
k	$\max(w_{ij})/\sqrt{2}l$
l	side length of canvas
m	$\|P\|$
n	$\|S\|$
P	the set of points on the canvas available for assignment
p_x	for $p \in P$, this is the x coordinate of the position on the canvas
p_y	for $p \in P$, this is the y coordinate of the position on the canvas
q	the ratio for simulated annealing, $q \in [0, 1)$
r	the number of runs for a heuristic method
S	the set of DNA sequences
s_i	an element of S
σ	the substitution penalty ($\sigma < g$)
T	the temperature for simulated annealing, $T > 1$
w_{ij}	the cost of the alignment of s_i and s_j

Appendix 2: Sequence Accession Numbers

Search these numbers in BLAST to get the sequences [8].

Castilleja

AM233921, AM233922, AM233957, AM233926, AM234019, AM233969, AM233940, AM233945, AM233938, AM233997, AM233951, AM233946, AM233937, AM233941, AM233939, AM233942, AM233947, AM234032, AM233998

Caryophyllaceae

AY936267, AY936273, AY936259, AY936263, AY936241, AY936333, AY936279, AY936277, AY936235, AY286529, AY936247, AY936252, AY936254, AY936253, L78088, AF210907

Cactaceae

AY015345.1, AY015346.1, AY015347.1, AY015348.1, AY015349.1, AY015350.1, AY015351.1, AY015352.1, AY015353.1, AY015354.1, AY015355.1, AY015356.1, AY015357.1, AY015358.1, AY015359.1, AY015360.1, AY015361.1, AY015362.1, AY015363.1, AY015364.1, AY015365.1, AY015366.1, AY015367.1, AY015368.1, AY015369.1, AY015370.1, AY015371.1, AY015372.1, AY015373.1, AY015374.1, AY015375.1, AY015376.1, AY015377.1, AY015378.1, AY015379.1, AY015380.1, AY015381.1, AY015382.1, AY015383.1, AY015384.1, AY015385.1, AY015386.1, AY015387.1, AY015388.1, AY015389.1, AY015390.1, AY015391.1, AY015392.1, AY015393.1, AY015394.1, AY015395.1, AY015396.1, AY015397.1, AY015398.1, AY015399.1, AY015400.1, AY015401.1, AY015402.1, AY015403.1, AY015404.1, AY015405.1, AY015406.1, AY015407.1, AY015408.1, AY015409.1, AY015410.1, AY015411.1, AY015412.1, AY015413.1, AY015414.1, AY015415.1, AY015416.1, AY015417.1, AY015418.1, AY015419.1, AY015420.1, AY015421.1, AY015422.1, AY015423.1, AY015424.1, AY015425.1, AY015426.1