On the tip selection algorithm. So, we want to defend against an attacker who secretly builds a chain/subtangle, occasionally referencing the main tangle to gain more height.

We are going to use the fact that the main tangle is supposed to have more (active) hashing power, and therefore manages to give more cumulative weight to more transactions than the attacker. The idea is to use a MCMC (Markov Chain Monte Carlo) algorithm to select the two tips to reference.

Let H_x be the current cumulative weight of a site (i.e., a transaction represented on the tangle graph). For simplicity, assume that all own weights are equal to 1; so, the cumulative weight of a tip is always 1, and the cumulative weight of other sites is at least 2.

The algorithm is then described in the following way:

1. consider all transactions with cumulative weight between L and (say) $2L$ (where L is large, to be chosen);

2. place N particles independently there (N is not so big, say, 10 or so);

3. these particles will perform discrete-time random walks "towards the tips" (i.e., transition from x to y is possible if and only if y approves x);

4. the two random walks that reach the tip set first will indicate our two tips to approve;

5. the transition probabilities of the walks are defined in the following way: if y approves x (we denote this $y \rightsquigarrow x$), then the transition probability P_{xy} is proportional to $\exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)$, that is

$$P_{xy} = \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_y)\big)\Big(\sum_{z:z\rightsquigarrow x} \exp\big(-\alpha(\mathcal{H}_x - \mathcal{H}_z)\big)\Big)^{-1}, \qquad (1)$$

   where $\alpha > 0$ is a parameter to be chosen (one can start e.g. with $\alpha = 1$).

In patricular, note that this algorithm is "local", one need not go to all the way to the genesis to calculate things.

To see that the algorithm works as intended, consider first the "lazy tips" (those that intentionally approve some old transactions to avoid doing verification), see Figure 1. Observe that, even if the particle is in a site approved by such a tip, it is not probable that the lazy tip would be selected, since the cumulative weights difference will be very large, look at (1).

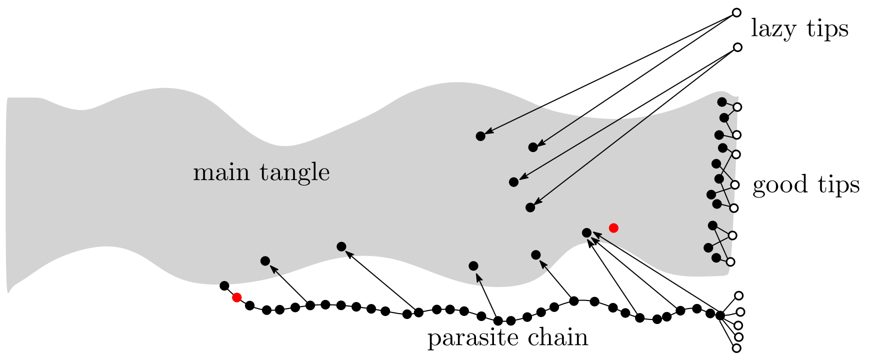

Figure 1: On the tip selection algorithm. The two red circles indicate an attempted double-spend.

Next, consider the following attack: the attacker secretly builds a chain (a “parasite chain”) containing a transaction that empties his account to another account under his control (indicated as the leftmost red circle on Figure 1). At some point the attacker issues a transaction in the main tangle (indicated as the other red circle), and waits until the merchant accepts it. The parasite chain occasionally references the main tangle (hence the name) and so its sites have good height/score (even better than those of the main tangle), although the cumulative weight is not so big in that chain. Note also that it cannot reference the main tangle after the merchant’s transaction. Also, the attacker might try to artificially inflate the number of “his” tips at the moment of the attack, as shown on the picture. The attacker’s idea is to make the nodes reference the parasite chain, so that the “good” tangle would become orphaned.

Now, it is easy to see why the MCMC selection algorithm with high probability won’t select one of the attacker’s tips. Basically, the reason is the same as why the algorithm doesn’t select the lazy tips: the sites of the parasite chain will have a much smaller cumulative weight than the main tangle’s sites they reference. Therefore, it is not probable that the random walk will ever jump to the parasite chain (unless it begins there, but this is not very probable too, since the main tangle contains more sites).