Pedigree Traversal in FASTLINK

Alejandro A. Schäffer
Rice University

This document is meant to accompany FASTLINK, version 2.0 and beyond. It describes some aspects of pedigree traversal in FASTLINK. It is primarily intended for computer scientists and experts in linkage analysis who may wish to modify the LINKAGE or FASTLINK code in various ways.

Everything written in this document also applies to LINKAGE. This document is based on my own approach to understanding pedigree traversal and my own terminology. So far as I know, overlaps with other descriptions of pedigree traversal are largely incidental. Thanks to Meg Gelder Ehm (Rice University), Sandeep Gupta (Rice University), and Prof. Daniel Weeks (University of Pittsburgh) for asking enough questions about pedigree traversal to motivate me to write this document.

Thanks to Darrell Root (Oregon Health Sciences University) for contributing PostScript files with the first 3 sample pedigrees shown below. His pedigrees were drawn with:

PedDraw v.4.4r2 by Paul Mamelka, Bennet Dyke, and Jean MacCleur, Department of Genetics, Southwest Foundation for Biomedical Research, 1993.

Thanks to Prof. Elizabeth Thompson (University of Washington) for contributing a PostScript file with the last sample pedigree shown below. Her pedigree was drawn with psdraw, for which the reference is:

Geyer, C. J., Software for calculating gene survival and multigene descent probabilities and for pedigree manipulation and drawing, Technical Report No. 153, Department of Statistics, University of Washington, 1988.

This document describes pedigree traversal for loopless pedigrees. The issue of loops is dealt with in the accompanying document *Loops in FASTLINK*. This document should be read before the loops document.

Why Traverse the Pedigree

A fundamental goal in all the programs in LINKAGE/FASTLINK is to compute the pedigree likelihood for a candidate value of θ. How the candidates are generated varies from program to program. The computation is done by selecting one individual A to be the "proband" and computing for A and each genotype g, the probability that A has genotype g conditioned on the known genotype information of the rest of the pedigree and the candidate θ.

LINKAGE/FASTLINK uses a simple loop breaking strategy that allows us to assume in this document that all pedigrees are loopless. The issue of loops is discussed in detail in the accompanying document *Loops in FASTLINK*.

The reason for the term “traversal” is that it is natural to update the conditional genotype probabilities for one family at a time. This raises a major question: when updating the conditional genotype probability for members of some nuclear family, what should we condition on, besides θ?

A *nuclear family* consists of two parents and all the children which they produced together. Some geneticists use the term *full sibship* to mean the same thing as what I am calling nuclear family.

A (valid) traversal sequence is a sequence of visits to nuclear families done in such a way that:

1. Each nuclear family has one visit during which the genotype probabilities of one of its members are updated conditional on the genotypes of the other members of the nuclear family (which have been in turn conditioned on other individuals) and θ.

2. The nesting of the conditionals is such that at the last update, the proband’s genotype probabilities are updated (indirectly) conditional on the observed data for the rest of the pedigree and θ.

Elston and Stewart made the fundamental observation that (under some conditions, to be specified below), it is sufficient to condition the update of a given nuclear family on all of its descendant nuclear families. When arriving at a new nuclear family the probabilities of the parent that connects up the tree are updated conditional on the spouse, the children, and θ. The children already are conditioned on any descendants further down in the tree.

Such an upward traversal sequence will work if:

1. There is exactly one nuclear family T at the top generation.

2. Every other nuclear family has exactly one parent who is a direct descendant of the two parents in family T and one parent who has no ancestors in the pedigree (such a person is called a *founder*).

3. There are no multiple marriages.

4. One of the parents in T is treated as the proband.

It is useful to observe that for pedigrees that satisfy these four conditions, there may be many traversal sequences which are valid. For example, any sequence, which updates all nuclear families in the bottom generation, then all nuclear families in the next-to-bottom generation, and so on up the tree is valid. Within each generation any order of visits to nuclear families preserves validity.

Nuclear Family Graph

The description in Ott's book suggests that the pedigree traversal algorithm in LINKAGE is a minor modification of the Elston-Stewart bottom-up approach. I can see how this claim is plausible from a genetics or statistics point of view. However, as an algorithmicist, I find the suggestion that the Elston-Stewart and LINKAGE algorithms are similar to be misleading.

It is true that for pedigrees that satisfy the 4 conditions above, the two probability update algorithms can visit the nuclear families in the same order. However, this is far from obvious if one stares at the LINKAGE traversal code. I use the word "can" rather than "do" in the first sentence because both the Elston-Stewart algorithm and the LINKAGE algorithm allow for some arbitrary orderings in the nuclear family visits. It is possible to correctly implement the Elston-Stewart algorithm to visit the nuclear families in orders which are absolutely impossible with LINKAGE, no matter how LINKAGE makes its free choices. For example, in a pedigree that satisfies the 4 conditions above and has multiple nuclear families at each generation but the topmost, the generation-by-generation traversal sequences proposed above as possible for the Elston-Stewart algorithm are *not* possible in LINKAGE. This will be illustrated with an example below.

Since I am trained in graph algorithms, I prefer to think of the pedigree with the following auxiliary graph, which I call the *nuclear family graph*. First let's assume that there are no multiple marriages. In this graph each nuclear family becomes a vertex and two vertices are adjacent (i.e., connected by an undirected edge) if their nuclear families share an individual which is a parent in one family and a child in the other.

The case of multiple marriages is subtle. We call a nuclear family a "multiple marriage" if one of its parents parented offspring with multiple distinct mates both of whom are in the pedigree. We distinguish two types of multiple marriages. A f-multiple marriage (nuclear family) is one in which the multiply married spouse is a founder. A n-multiple marriage (nuclear family) is one in which the multiply married person is not a founder in the pedigree. Each multiple marriage corresponds to a distinct vertex in the nuclear family graph. There is an edge between an n-multiple marriage and the parent nuclear family in which the multiply married person is a child. Among all f-multiple marriages involving some individual A, one such family is designated as the first. There is an edge between that first family and all the other multiple marriages in which A is a parent.

Edge Labels

In the nuclear family graph, the edge $x - y$ is called a "down" edge w.r.t. x and an "up" edge w.r.t y if the shared individual is a child in nuclear family x and a parent in nuclear family y. In our drawings we show such an edge with x at a higher vertical position than y. The edge $x - y$ is called an "up" edge w.r.t. x and a "down" edge w.r.t y if the shared individual is a parent in nuclear family x and a child in nuclear family y. In the case of f-multiple marriages, the edges between them are down edges w.r.t. the first f-multiple marriage for a given individual and up edges w.r.t. to all the other f-multiple marriages w.r.t that same individual.

For each vertex divide its list of adjacent edges into two separate lists, one is the up list and the other is the down list with respect to that vertex. There can be at most one up edge per vertex. To the extent that we describe matters, the order of edges on the down list is not significant. It depends on the ordering of siblings in the input, which is somewhat arbitrary.

Traversal Order

Since the pedigree has no loops, it turns out that the nuclear family graph is always connected and has no cycles (i.e., it is what computer scientists call an unrooted tree). The tricky rules about multiple marriages are designed with the goal that the nuclear family graph should always be connected, but have no cycles. Mathematically inclined readers may find it instructive to prove this by induction on the number of nuclear families.

The traversal order uses the following rules:

1. If we first arrive at nuclear family x via an edge from nuclear family y, then x is updated only after all its neighbors, except y have been updated.
2. If nuclear family x is the first nuclear family, we update x only after all its neighbor nuclear families have been updated.

3. Up edges are always chosen before down edges, in deciding which neighbor to visit.

What this means is that we can think of the traversal routine with the following pseudocode. Caution: this is very different from the code in LINKAGE/FASTLINK but yields the same behavior:

Visit(w)
    While w has an unvisited neighbor x reachable via an up edge:
        Visit(x);
    While w has an unvisited neighbor y reachable via a down edge:
        Visit(y)
    Update w

We start by visiting a nuclear family containing the proband. If there is a nuclear family in which the proband is a child, we start with that one; otherwise, we start with the first nuclear family (dependent on input order) in which the proband is a parent.

The way we update w depends on how it was first reached. If it was reached via a down edge from z we update the parent in w that nuclear families z and w share. This is done with a call to to the procedure `segup` (`segsexup` for sexlinked data). In LINKAGE there were some alternatives to `segup` for special places in the pedigree; these are not used in FASTLINK.

If we reached w via an up edge from z then we update the child that w and z share using the procedure `segdown` (`segsexdown` for sexlinked data).

For the proband's nuclear family, we use whichever procedure would update the proband's conditional probabilities in that nuclear family.

By staring at the code one would get the impression that the interesting traversal work is done in the procedures `collapsedon` and `collapseup`. I do not agree with this view. The action is in `segdown` and `segup` and arguably in `seg`, which sets up the calls to `segdown` and `segup`. Roughly speaking, `collapsedown` traverses an up edge in the nuclear family graph and `collapseup` traverses a down edge in the nuclear family graph, but many recursive calls to these procedures do not actually result in a nuclear family update. Furthermore, `collapsedown` and `collapseup` work in the pedigree directly rather than in the nuclear family graph.

Examples

In this section we show four sample pedigrees. In each case, we selected the proband to be the leftmost individual at the topmost generation. This would be the typical choice that most users make.

For each pedigree, we show a picture of the pedigree, the corresponding nuclear family graph, and a transcript of how the traversal goes. In the nuclear family graph, each nuclear family is labeled with the number of its leftmost child.

The first pedigree shows the simple case, where all the updates go upward.

Here is a partial transcript of the traversal on pedigree 1.

Start at nuclear family 205

Visit nuclear family 304

Update person 205 conditioned on 204 and 304 going up

Backup to nuclear family 205

Visit nuclear family 302

Update person 202 conditioned on 203, 302, and 303 going up

Backup to nuclear family 205

Visit nuclear family 300

Visit nuclear family 400

Update person 300 conditioned on 301 and 400 going up

Back up to 300

Update person 200 conditioned on 201 and 300 going up

Backup to nuclear family 205

Update person 100 conditioned on 101 and 205, 202, and 200 going up

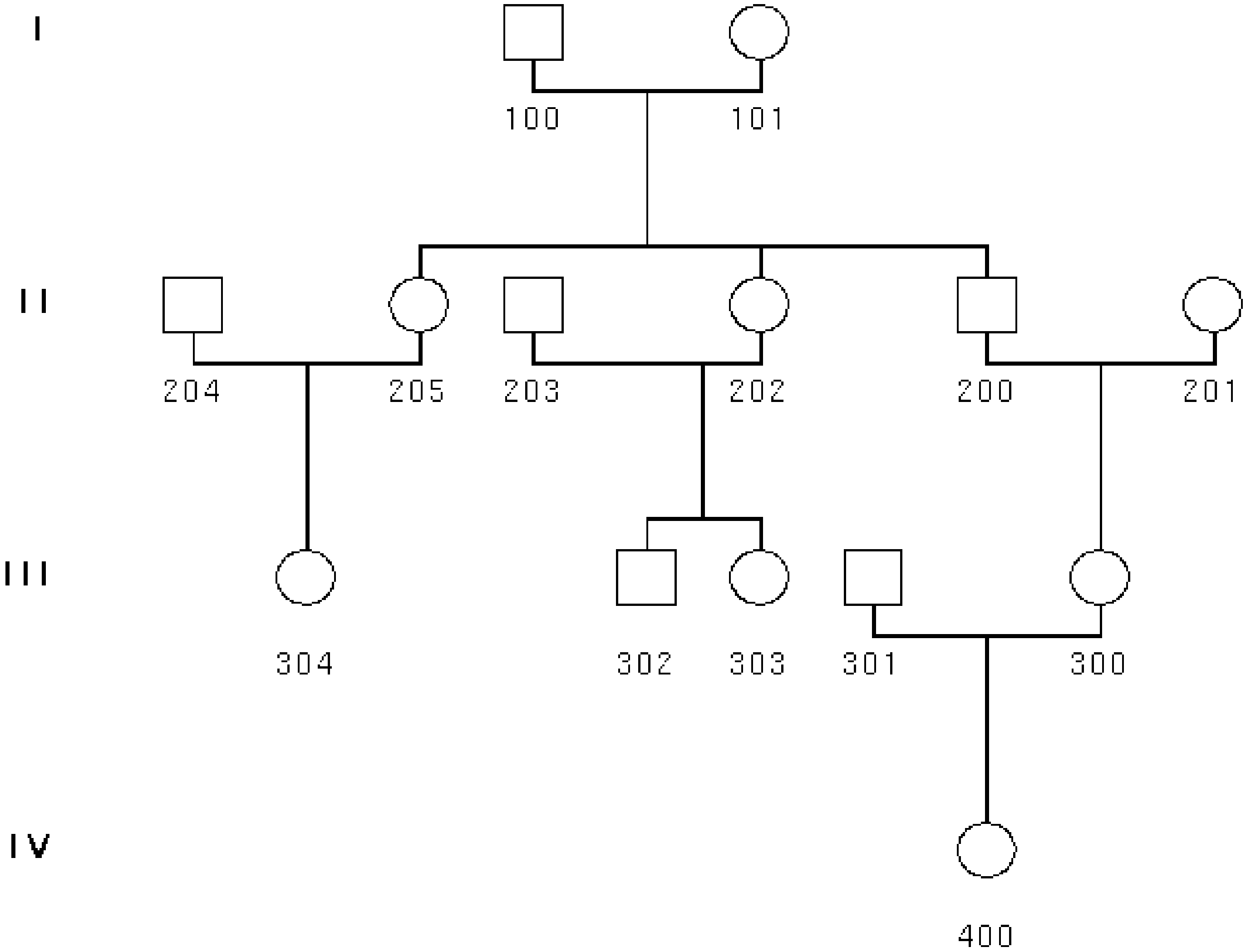

Figure 1: Pedigree 1

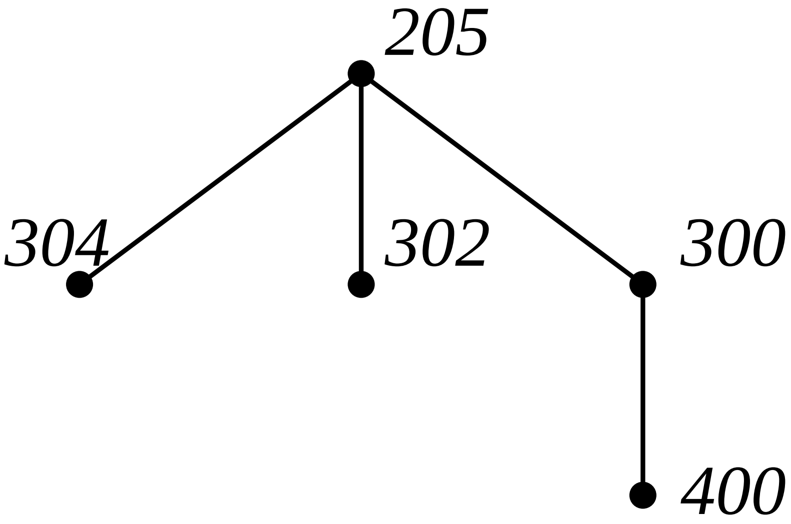

Figure 2: Nuclear family graph for Pedigree 1

Here is the second sample pedigree. It illustrates going down instead of going up.

Start at nuclear family 201
Visit nuclear family 304
Visit nuclear family 203
Update person 203 conditioned on 102 and 103, going down
Back to nuclear family 304
Visit nuclear family 404
Update person 304 conditioned on 305, 404, and 405, going up
Backup to nuclear family 304
Visit nuclear family 403
Update person 302, conditioned on 303 and 403, going up
Backup to nuclear family 304
Visit nuclear family 400
Update person 300, conditioned on 301, 400, 401, and 402, going up
Backup to nuclear family 304
Update person 202 conditioned on 203, 304, 302, and 300, going up
Backup to nuclear family 201
Update person 100, conditioned on 101, 201, and 202, going up

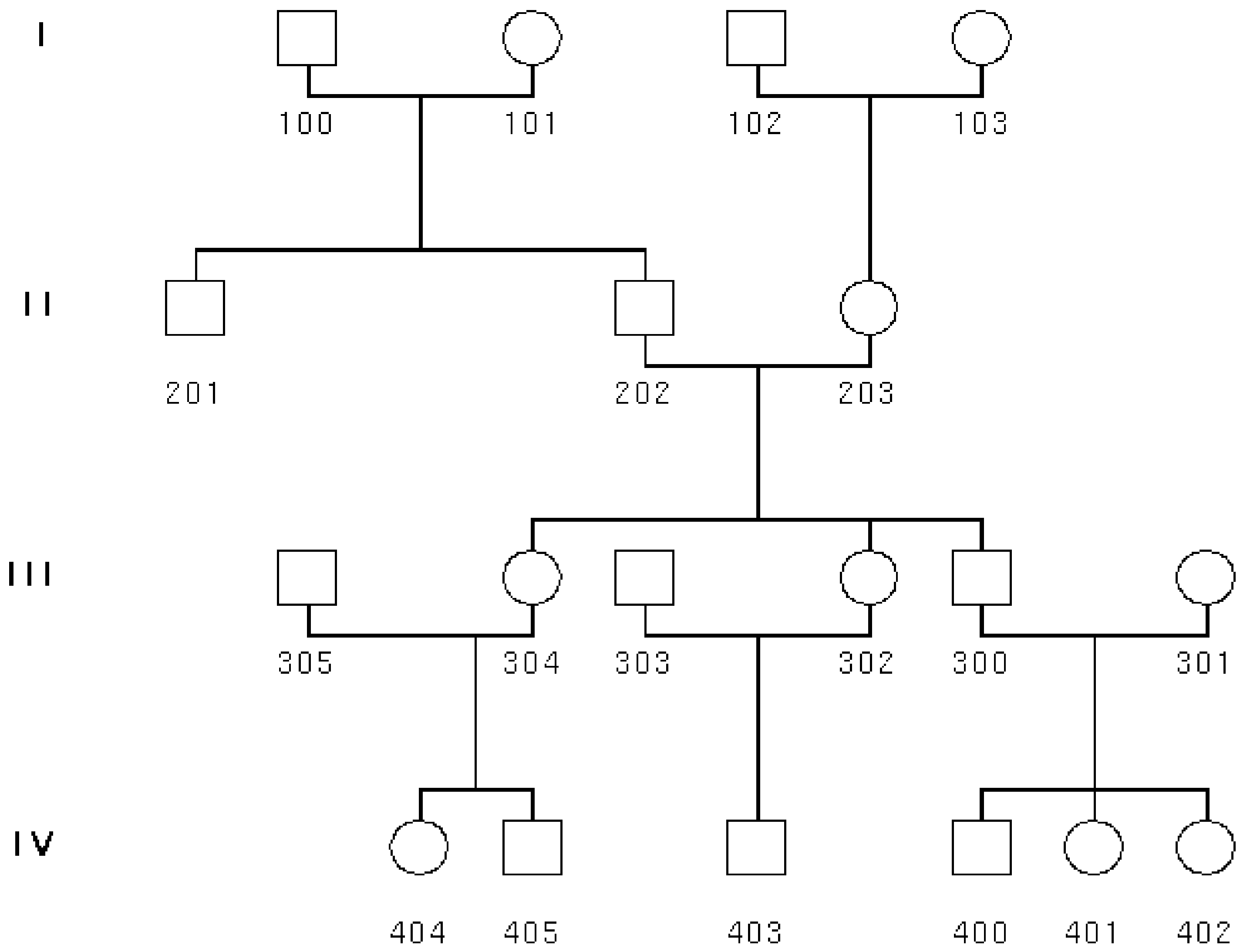

Figure 3: Pedigree 2

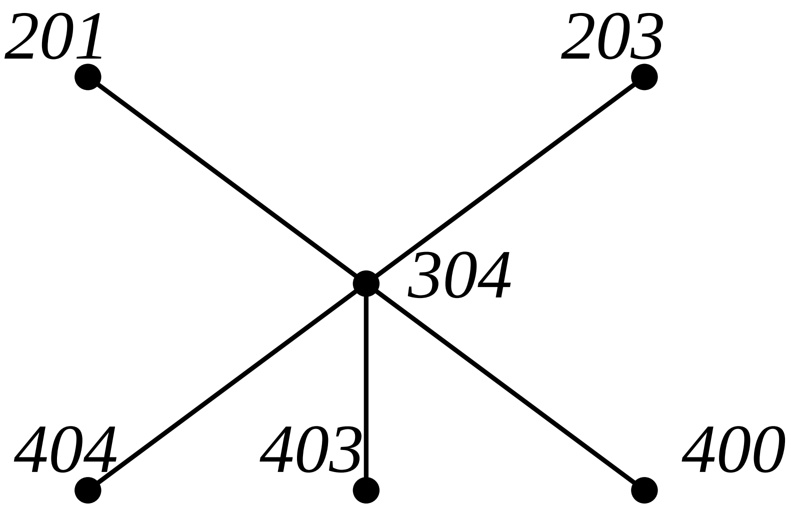

Figure 4: Nuclear family graph for Pedigree 2

Here is a third sample pedigree. It illustrates n-multiple marriages (individual 205). It can also be used to illustrate how LINKAGE deviates from the Elston-Stewart algorithm in an algorithmically fundamental way. The Elston-Stewart algorithm would allow a traversal order in which the nuclear families are updated in the order 402, 403, 404, 406, 307, 314, 310, 316. LINKAGE would never use this order no matter how the siblings in each nuclear family are arranged.

Here is the transcript for the third pedigree:

Start at nuclear family 202
Visit nuclear family 307
Visit nuclear family 402
Update person 307, conditioned on 306 and 402, going up
Backup to 307
Visit nuclear family 403
Update person 308, conditioned on 309 and 403, going up
Backup to 307
Update person 202, conditioned on 203, 307, and 308, going up
Backup to 202
Visit nuclear family 314
Update person 205, conditioned on 206 and 314, going up
Backup to 202

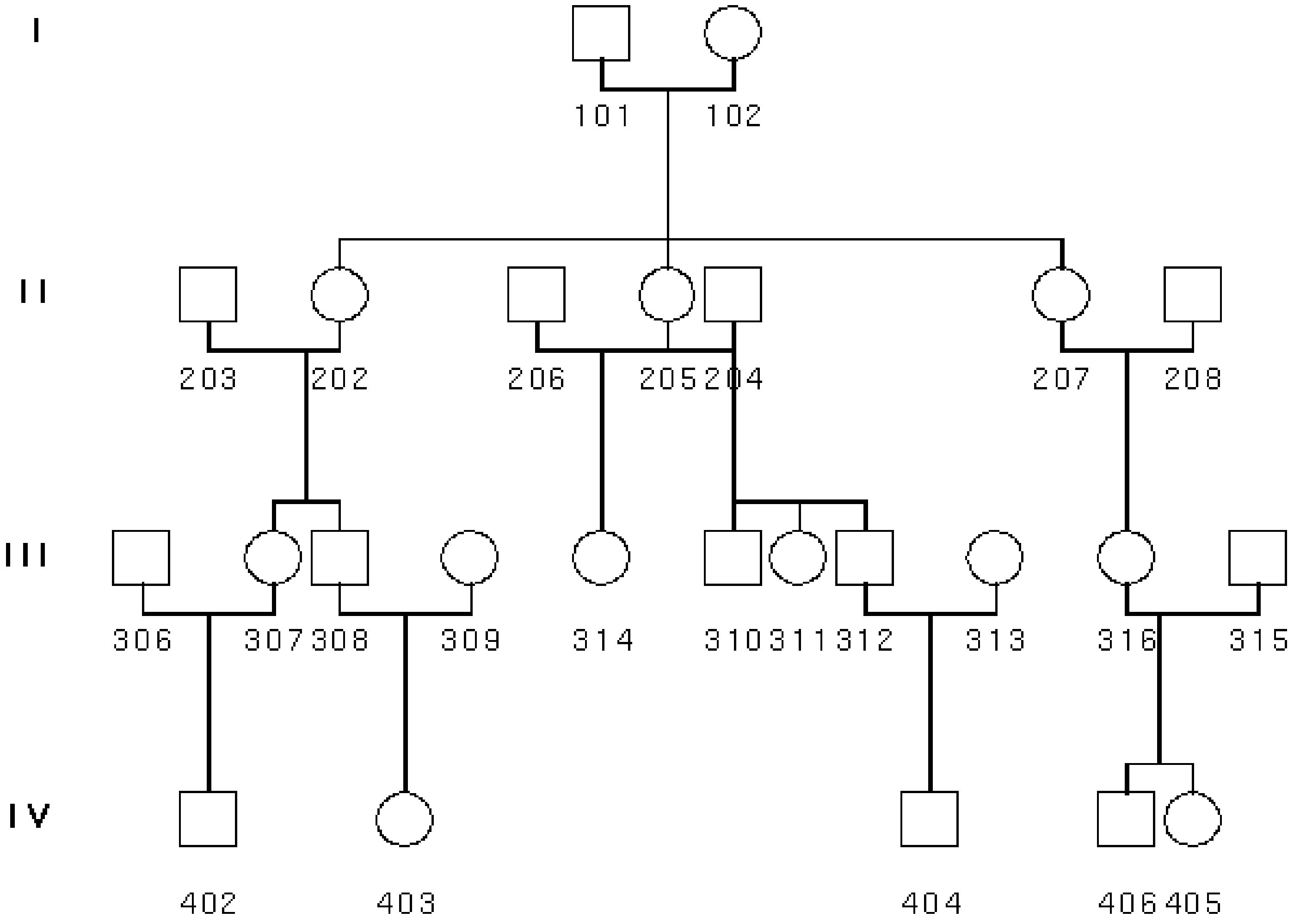

Figure 5: Pedigree 3

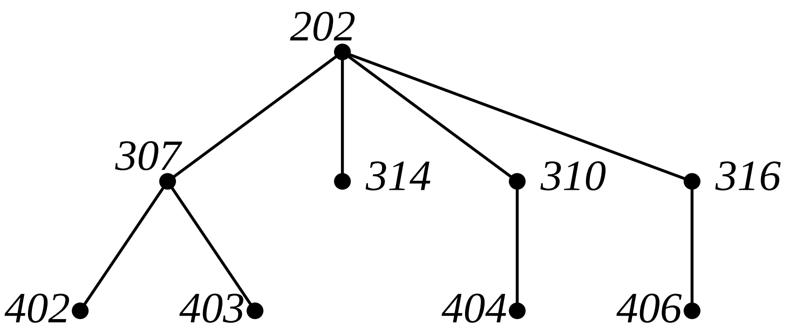

Figure 6: Nuclear family graph for pedigree 3

Visit nuclear family 310
Visit nuclear family 404
Update person 312, conditioned on 313 and 404, going up
Backup to 310
Update person 205, conditioned on 204, 310, 311, and 312, going up
Backup to 202
Visit nuclear family 316
Visit nuclear family 406
Update person 316, conditioned on 315, 406, and 405, going up
Backup to 316
Update person 207, conditioned on 208 and 316, going up
Backup to 202
Update person 101, conditioned on 102, 202, 205, and 207, going up

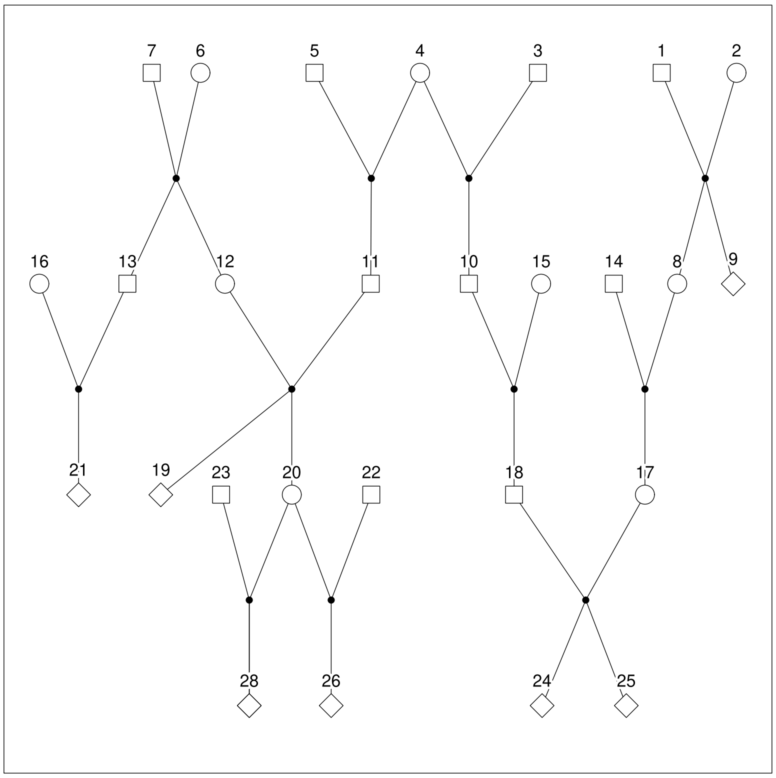

Figure 7: Pedigree 4, marriage graph format

The fourth pedigree illustrates the distinction between f-multiple marriages (individual 4) and n-multiple marriages (individual 20). It has the advantage of being in marriage graph format, which is much closer to what the nuclear family graph looks like. In particular there is a one-to-one correspondence between the filled in dots in the marriage graph and the vertices in the nuclear family graph.

Here is the traversal transcript for pedigree 4:

Start at nuclear family 13

Visit nuclear family 21

Update person 13, conditioned on 16 and 21, going up

Backup to 13

Visit nuclear family 19

Visit nuclear family 11

Visit nuclear family 10

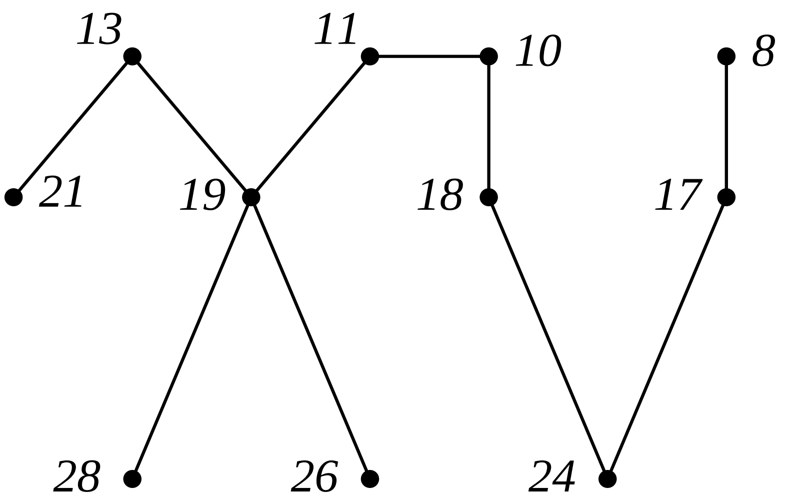

Figure 8: Nuclear family graph for pedigree 4

Visit nuclear family 18
Visit nuclear family 24
Visit nuclear family 17
Visit nuclear family 8
Update person 8, conditioned on 1,2, and 9, going down
Back to nuclear family 17
Update person 17, conditioned on 8 and 14, going down
Back to nuclear family 24
Update person 18, conditioned on 17. 24, and 25 going up
Backup to nuclear family 18
Update person 10 conditioned on 15, and 18, going up
Backup to nuclear family 10
Update person 4, conditioned on 3 and 10, going up
Backup to nuclear family 11
Update person 11 conditioned on 4 and 5, going down
Back to nuclear family 19
Visit nuclear family 28
Update person 20, conditioned on 23 and 28, going up
Backup to nuclear family 19
Visit nuclear family 26
Update person 20, conditioned on 22 and 26, going up

Backup to nuclear family 19
Update person 12, conditioned on 11, 19, and 20, going up
Backup to nuclear family 13
Update person 7, conditioned on 6, 13, and 12, going up