MNRAS 000, 1–22 (2021) Preprint 22 June 2021 Compiled using MNRAS LATEX style file v3.0

ACACIA: a new method to produce on-the-fly merger trees in the RAMSES code

Mladen Ivkovic,[1,2]⋆ Romain Teyssier[3]
[1] *Laboratoire d'Astrophysique, École Polytechnique Fédérale de Lausanne, 1290 Versoix, Switzerland*
[2] *Observatoire de Genève, Université de Genève, Chemin Pegasi 51, 1290 Versoix, Switzerland*
[3] *Institute for Computational Science, University of Zurich, 8057 Zurich, Switzerland*

Accepted XXX. Received YYY; in original form ZZZ

ABSTRACT

The implementation of `ACACIA`, a new algorithm to generate dark matter halo merger trees with the Adaptive Mesh Refinement (AMR) code `RAMSES`, is presented. The algorithm is fully parallel and based on the Message Passing Interface (MPI). As opposed to most available merger tree tools, it works on the fly during the course of the N body simulation. It can track dark matter substructures individually using the index of the most bound particle in the clump. Once a halo (or a sub-halo) merges into another one, the algorithm still tracks it through the last identified most bound particle in the clump, allowing to check at later snapshots whether the merging event was definitive, or whether it was only temporary, with the clump only traversing another one. The same technique can be used to track orphan galaxies that are not assigned to a parent clump anymore because the clump dissolved due to numerical over-merging. We study in detail the impact of various parameters on the resulting halo catalogues and corresponding merger histories. We then compare the performance of our method using standard validation diagnostics, demonstrating that we reach a quality similar to the best available and commonly used merger tree tools. As a proof of concept, we use our merger tree algorithm together with a parametrised stellar-mass-to-halo-mass relation and generate a mock galaxy catalogue that shows good agreement with observational data.

Key words: methods: numerical – galaxies: evolution – galaxies: haloes – dark matter

1 INTRODUCTION

Mock galaxy catalogues generated using N-body or hydrodynamical simulations are important tools for extragalactic astronomy and cosmology. They are used to test current theories of galaxy formation, to explore systematic and statistical errors in large scale galaxy surveys and to prepare analysis codes for future dark energy mission such as Euclid or LSST. There is a large variety of methods to generate such mock galaxy catalogues. The most ambitious line of products is based on full hydrodynamical simulations, where dark matter, gas, and star formation are directly simulated (Dubois et al. 2014; Khandai et al. 2015; Vogelsberger et al. 2014; Schaye et al. 2015). The intermediate approach is based on semi-analytic modelling (hereafter SAM) (White & Frenk 1991; Bower et al. 2006; Somerville & Primack 1999; Kauffmann et al. 1993; Kang et al. 2005; Croton et al. 2006; Somerville et al. 2008; Guo et al. 2011; Lu et al. 2011) for which galaxy formation physics, although simplified, is still at the origin of the mock galaxy properties. Finally, the simplest and most flexible approach is based on a purely empirical modelling of galaxy properties, sometimes called Halo Occupation Density (HOD hereafter) (e.g. Seljak 2000; Berlind & Weinberg 2002; Peacock & Smith 2000; Benson et al. 2000; Wechsler et al. 2001; Scoccimarro et al. 2001; Bullock et al. 2002; Vale & Ostriker 2006a; Yang et al. 2003, 2012; Vale & Ostriker 2004a; Van Den Bosch et al. 2003; Yang et al. 2009; Kravtsov et al. 2004; Vale & Ostriker 2004b; Conroy et al. 2006; Behroozi et al. 2013c; Moster et al. 2013; Guo et al. 2010; Yamamoto et al. 2015; Nuza et al. 2013). The last two techniques (SAM and HOD) both require the complete formation history of dark matter haloes, and possibly their sub-haloes. This formation history is described by halo '*merger trees*' (Roukema et al. 1993; Roukema & Yoshii 1993; Lacey & Cole 1993). Accurate merger trees are essential to obtain realistic mock galaxy catalogues, and constitute the backbone of SAM and HOD models.

⋆ E-mail: mladen.ivkovic@epfl.ch

The advantage of using SAM and HOD techniques to generate mock galaxy catalogues is that one does not need to model explicitly the gas component, but only the dark matter component. The corresponding N-body simulations are commonly referred to as '*dark matter only*' (DMO) simulations. With growing processing power, improved algorithms and the use of parallel computing tools and architectures, larger and better resolved DMO simulations are becoming possible. The current state-of-the-art is the Flagship simulation performed for the preparation of the Euclid mission (Potter et al. 2017) and featured 2 trillion dark matter particles. Such extreme simulations make post-processing analysis tool such as merger tree algorithms increasingly difficult to develop and to use, mostly because of the sheer size of the data to store on disk and to load up later from the same disk back into the processing unit memory. In some extreme cases, the amount of data that needs to be stored to perform a merger tree analysis in post-processing is simply too large. Storing just particle positions and velocities in single precision for trillions of particles requires dozens of terabytes per snapshot. Another issue is that most modern astrophysical simulations are executed on large supercomputers which offer large distributed memory. Post-processing the data they produce may also require just as much memory, so that the analysis will also have to be executed on the distributed memory infrastructures as well. The reading and writing of such vast amount of data to a permanent storage remains a considerable bottleneck, particularly so if the data needs to be read and written multiple times. One way to reduce the computational cost is to include analysis tools like halo-finding and the generation of merger trees in the simulations and run them "*on the fly*", i.e. run them during the simulation, while the necessary data is already in memory.

The main motivation for this work is precisely the necessity for such a merger tree tool for future "beyond trillion particle" simulations. To this end, a new algorithm that we named `ACACIA` was designed to work on the fly within the parallel AMR code `RAMSES`. One novel aspect of this work is the use of the halo finder `PHEW` (Bleuler et al. 2015) for the parent halo catalogue. Different halo finders have been shown to have a strong impact on the quality of the resulting merger trees (Avila et al. 2014). `PHEW` falls into the category of "watershed" algorithms that are not so common in the cosmological halo finding literature. This type of algorithm assigns particles (or grid cells) to density peaks above a prescribed density threshold and according to the so-called "watershed segmentation" of the negative density field.

This paper is structured as follows. In Section 2, a brief description of the `PHEW` halo finder and its new particle unbinding method is given. Section 3 describes the merger tree algorithm `ACACIA` and shows test results to determine what parameters give the best results. Using the halo catalogue and its corresponding merger tree generated on the fly by a cosmological N-body simulation, we use the stellar-mass-to-halo-mass (SMHM) relation from Behroozi et al. (2013c) to produce a mock galaxy catalogue in the light cone. We analyse in Section 6 the properties of our mock galaxy catalogue and show that the introduction of orphan galaxies improve the comparison to observations considerably. A detailed description of the `ACACIA` algorithm is given in Appendix A, and a detailed comparison with the other halo finding and tree-building algorithms presented in Avila et al. (2014) is given in Appendix B.

2 HALO FINDING AND PARTICLE UNBINDING

Halo finding plays a central role in the exploitation of N-body simulations. Paradoxically, a unique definition of what is a halo or a sub-halo has never been adopted so far. The current state of affairs in the halo finding business is quite the opposite, with a multitude of definitions emerging over the last decades, each definition corresponding to a different halo finding algorithm. The Halo Finder Comparison Project (Knebe et al. 2011) lists 29 different codes and roughly divides them into two distinct groups:

(i) Percolation algorithms, for which particles are linked together if closer to each other than some specified linking length. The typical example is the algorithm "*friends-of-friends*" (thereafter FOF) (Davis et al. 1985).
(ii) Segmentation algorithms, for which space is segmented into separate regions around local peaks of the density field. Particles within these regions are then collected and assigned to the same halo or sub-halo. The typical example is the "*Spherical Overdensity*" method (thereafter SOD) (Press & Schechter 1974).

The outer boundary of the haloes are defined in both method by a density iso-surface, whose exact value determines the properties of the resulting halo statistics. Halo catalogues derived from FOF and SOD and their corresponding merger trees have been studied quite extensively in the literature (see e.g. Avila et al. 2014).

2.1 The PHEW halo finder

In this paper, we extend these earlier studies to the `PHEW` halo finder (Bleuler et al. 2015) developed specifically for the `RAMSES` code (Teyssier, R. 2002). The `PHEW` algorithm belongs to the category of segmentation methods. Particle masses are first deposited to the AMR grid using the "*cloud-in-cell*" technique. All density maxima are then marked as potential sites for a *clump*. Clumps are what we call any structure, haloes and sub-haloes, in contexts where we don't need to differentiate between them. The volume is then segmented into peak patches by assigning each cell of the grid to the closest density maximum in the direction of the steepest density gradient.

This segmentation method provides well defined regions separated by density saddle surfaces. The minimum density in the saddle surface between two adjacent peaks marks the saddle point between the two peaks. This well known method is often called '*watershed segmentation*'. In order to define proper halo boundaries, `PHEW` uses an outer density isosurface, like most methods described above. Sub-haloes, on the other hand, are just the ensemble of all peak patches within the halo boundaries. This allows to identify haloes and sub-haloes without the assumption of spherical symmetry, unlike other popular methods such as SOD.

Sub-haloes can be organised into a hierarchy of substructures based on the same steepest gradient technique, for which individual clumps can be assigned to the closest

Table 1. Parameters used for the PHEW clump finder throughout this paper. The numerical values given can be directly used in the namelist file that RAMSES uses to read in runtime parameters. Here $\bar{\rho}$ here is the mean background density and m_p is the particle mass.

parameter	value	units
relevance threshold	3	1
density threshold	80	$\bar{\rho}$
saddle threshold	200	$\bar{\rho}$
mass threshold	10	m_p

densest peak. After this first pass, only a few sub-haloes survived the merging process, which is then repeated a second time, assigning these surviving sub-haloes to their densest neighbours. Ultimately, all sub-haloes will be collected into a single peak that corresponds to the main halo. Each pass defines a level in the hierarchy of sub-haloes. More details can be found in the original PHEW paper (Bleuler et al. 2015). As a consequence, a halo can have a number of sub-haloes, each one of them containing sub-sub-haloes and so on. This well defined hierarchy is a very important feature for us to uniquely assign particles to haloes and sub-haloes based on a binding energy criterion.

There are four parameters that PHEW requires a user to choose in order to identify clumps and haloes. Firstly, a "relevance threshold" needs to be defined. If for any given peak patch the ratio of the peak's density to the maximal density of the entire saddle surface of the respective peak patch is smaller than the chosen relevance threshold, then the peak patch is considered to be noise, not a genuine structure. The peak patch is then merged into a neighbour. Secondly, a density threshold determines the minimal density a cell needs to have to be part of any peak patch. Thirdly, a "saddle threshold" defines the maximal density for a saddle surface between two peak patches for the two patches to be considered parts of two different haloes. If the saddle surface density is above the threshold, then the peak patches will be parts of the same halo (but different sub-haloes within the host halo). Finally, a mass threshold determines the minimal mass a peak patch needs to have to be kept. We list the parameters that we used throughout this paper in Table 1.

2.2 Particle unbinding

We now describe how we assign each dark matter particle to a given sub-halo, a process that has not been implemented so far in the PHEW code. For this, we follow a physically motivated criterion, quite common in the halo finding literature, based on the binding energy of the particle (e.g. Knollmann & Knebe 2009; Springel et al. 2001; Stadel 2001). If a particle is not bound to the first sub-halo of the hierarchy, it is then passed recursively to the next sub-halo in the hierarchy, where the binding energy is checked again and so on. If the particle is not bound to any sub-halo, it is assigned to the main halo.

In the previous hierarchical unbinding process, the key component is the criterion adopted for deciding whether a particle is bound to a sub-halo or not. Traditionally, this is done using the *static* gravitational potential, since we are dealing with a single time step and we have to assume that

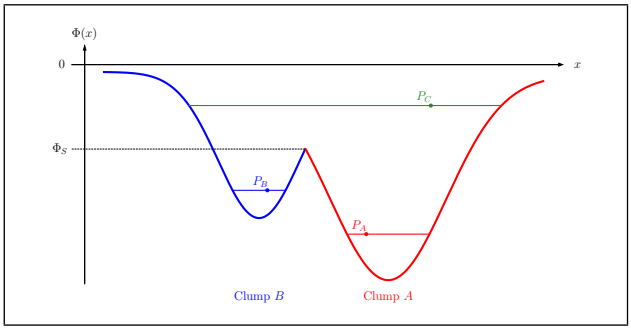

Figure 1. Simple sketch of the gravitational potential of a halo that consists of two clumps, A and B. The position of the horizontal lines marks the energies of three example particles, P_A, P_B, and P_C, while the length of the lines shows the spatial extent of the orbits. We call particles like P_A and P_B "strictly bound", as their predicted orbit boundaries don't allow them to escape from the clump they are assigned to. Particle P_C however, although energetically bound to clump A, can wander off deep into clump B, and for that reason is called "loosely bound". To discriminate between these two types of particles, we use the potential of the saddle point between clump A and B marked as Φ_S.

the N-body system is stationary. In this case, a particle at position $\mathbf{r}$ is considered as unbound if its velocity $\mathbf{v}$ exceeds the escape velocity given by

$$v_{\rm esc} = \sqrt{-2\Phi(\mathbf{r})} \tag{1}$$

More precisely, this means that the particle will be able to travel to infinity where the potential goes to zero. If the velocity is smaller than the escape velocity, the particle will follow a bound orbit and come back to its current location. Note that this orbit can leave the boundaries of the sub-halo. The particle will stay for some time in the sub-halo, but can visit at a later time a neighbouring sub-halo and then come back along the same bound orbit. This kind of particle does not *exclusively* belong to its original sub-halo. It should be in fact assigned to the parent sub-halo in the hierarchy. In order to identify particles as more strictly bound, we redefine the escape velocity using the potential of the closest saddle point Φ_S.

$$v_{\rm esc} = \sqrt{-2(\Phi(\mathbf{r}) - \Phi_S)} \tag{2}$$

This new escape velocity is smaller than the previous one, allowing more particle to exceed it and leak out of the current sub-halo into neighbouring ones. In what follows, we will use these two unbinding criteria, calling the first method the "loosely bound" criterion and calling the second one the "strictly bound" criterion. Figure 1 illustrates the difference between these two criteria. The gravitational potential of two neighbouring sub-haloes labelled A and B is represented. We show an example of a strictly bound particle in each sub-halo, and an example of a loosely bound particle that can wander from one sub-halo to the other one. If one uses the first binding criterion, this loosely bound particle will be assigned to sub-halo A, because this is where it is located at the present time. If one uses the second, stricter binding criterion, then the loosely bound particle will be assigned to the parent halo, but not to sub-halo A nor B.

When computing the velocity of the particle, it is important to use the velocity *relative* to the velocity of the sub-halo centre of mass (also called the bulk velocity of the

sub-halo). Because of this requirement, the unbinding process has to be performed iteratively, since removing a particle that is unbound requires to recompute the centre of mass velocity. Let us finally repeat that the particle unbinding is performed recursively, following the sub-halo hierarchy from the bottom up. Starting with the lowest (finest) level of sub-haloes, unbound particles are assigned to the higher (coarser) level of parent sub-halo for unbinding, and so on following the structure hierarchy. Particles that are unbound from all sub-haloes are collected into the main halo, marking the end of the hierarchical unbinding process.

3 MERGER TREES: BASIC PRINCIPLES

In this work, we adopt the terminology set by the "Sussing Merger Tree Comparison Project" (Srisawat et al. 2013; Wang et al. 2016; Avila et al. 2014; Lee et al. 2014). For sake of clarity, we repeat here some important definitions:

- For two snapshots at different times, a halo from the first one (i.e. higher redshift) is always referred to using the capital letter A and a halo from the second one (i.e. lower redshift) using B.
- Recursively, A itself and progenitors of A are all *progenitors* of B. When it is necessary to distinguish A from earlier progenitors, the term *direct progenitor* will be used.
- Recursively, B itself and descendants of B are all *descendants* of A. When it is necessary to distinguish B from later descendants, the term *direct descendant* will be used.
- In this work, we restrict ourselves to merger trees for which there is *precisely one direct descendant for every halo*.
- When there are multiple direct progenitors, it is required that one of these is identified as the *main progenitor*.
- The *main branch* of a halo is a complete list of main progenitors tracing back along its cosmic history.

We finally define an important convention we use here: when no distinction between sub-haloes and main haloes is necessary, they are collectively referred to as *clumps*.

3.1 Linking Clumps Across Snapshots

The aim of a merger tree code is to link haloes from an earlier snapshot to haloes in the consecutive snapshot, i.e. to find all the descendants of the haloes in the earlier snapshot. If we do this successfully each snapshot, then we can follow the formation history of haloes throughout the simulation. In particular, this will enable us to track the mass growth of haloes as well as the merging of different sub-haloes during the course of the simulation. To illustrate the idea, a merger tree of a main halo generated by `ACACIA` during a simulation is shown in Figure 2. In this particular case, we were able to track the formation history of the main halo down to redshifts $z > 3$. By linking progenitors and descendants throughout the simulation many branches of the tree are revealed. Each branch represents a clump that eventually merged into the main halo which we chose as the root of the tree at redshift zero.

Merger events occur when a clump, identified as such in a previous snapshot, disappears from the list of clumps in the next snapshot. In the case of a halo, it usually first becomes the sub-halo of another halo and can be followed as such

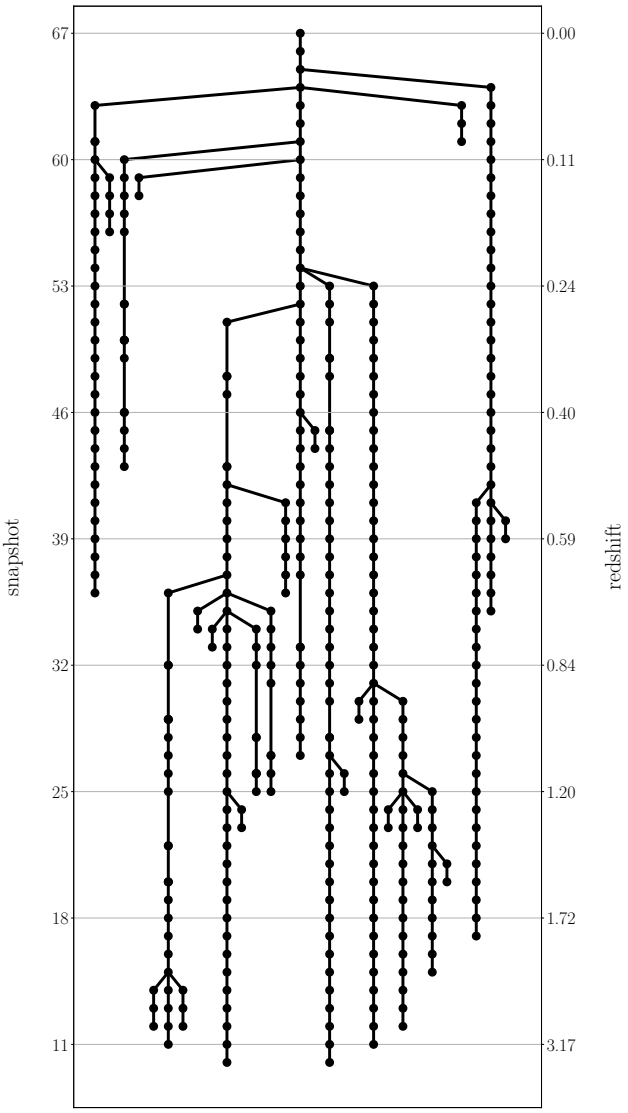

Figure 2. The merger tree of a main halo at redshift zero as found by `ACACIA`. This tree was extracted from a low resolution cosmological DMO simulation containing 64^3 particles for illustrative purposes. Using higher resolutions quickly leads to hundreds and thousands of branches, resulting in a rather messy plot. On the y axis, the snapshot numbers and their corresponding redshifts are given. The x axis has no physical meaning. Each dot represents a clump identified at the given snapshot. Two dots connected by a vertical line represent clumps that have been linked as main progenitor and main descendant. Diagonal lines depict merging events. In this tree, a few links between two dots are larger than others. These are cases where clumps merge temporarily, but then re-emerge later as separate clumps (see the example shown in Figure 4). We discuss these cases and how they are dealt with in Section 3.3.

in many subsequent snapshots. After some time, this sub-halo can merge into another sub-halo or dissolve completely due to numerical over-merging. In both cases, the sub-halo disappears completely from the clump catalogue.

A straightforward method to link progenitors with descendants in two consecutive snapshots is to trace individual particles using their unique particle ID. All merger tree codes use this simple technique (Behroozi et al. 2013b; Springel

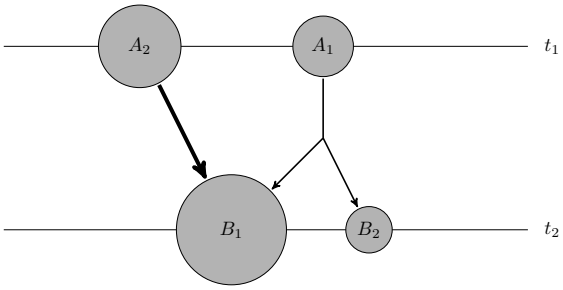

Figure 3. Illustration of a progenitor A_1 at time t_1 which is partially merged into a descendant B_1 at time $t_2 > t_1$, but some other part B_2 isn't. Because A_1 is not the main progenitor of B_1, by assigning its descendant only according to the merit function (6) would not pass on its formation history to B_2, but treat it as newly formed. The size of the circles represents the haloes' masses, the x-axis has no physical meaning.

et al. 2005; Jiang et al. 2014; Knebe et al. 2010; Tweed et al. 2009; Elahi et al. 2011; Jung et al. 2014; Rodriguez-Gomez et al. 2015) with the notable exception of the code `JMERGE` described and tested in Srisawat et al. (2013).

Linking a progenitor to a descendant means checking how many particles of the progenitor halo or sub-halo end up in the descendant halo or sub-halo. Naturally, these tracer particles may end up in multiple clumps, giving multiple descendant candidates for a progenitor. In such cases, the most promising descendant candidate will be called the *main descendant*. To find a main progenitor and a main descendant, a merit function $\mathcal{M}$ has to be defined, which is to be maximised or minimised, depending on its definition. An overview of the merit functions that are used in other merger tree algorithms is given in Table 1 of Srisawat et al. (2013). The merit function used in our implementation is given in Equation 6.

Sometimes, unfortunately, linking progenitors to descendants is not as straightforward as described so far. We now discuss two circumstances where special care must be taken to define robust links between different snapshots: fragmentation events and temporary merger events.

3.2 Fragmentation Events

In our current approach, each progenitor can have only one descendant[1]. We therefore need to pick only one descendant within a possibly large ranked list of descendant candidates, that all contain particles coming from the progenitor.

Normally, this choice is performed according to the ranking provided by the merit function, where the main descendant is ranked number 1. Problems arise for example when the progenitor A_1 is not the main progenitor of its main descendant B_1, but also has fragmented into another viable descendant candidate B_2. This situation is schematically shown in Figure 3.

Relying only on the merit function (6), progenitor A_1 will seem to have merged with A_2, the direct progenitor of B_1, in order to form B_1. The other fragment, B_2, will be treated as a newly formed clump and the entire formation history of B_2 would be lost. In order to preserve this history, we choose to prioritize the link from A_1 to B_2 over of merging progenitor A_1 into B_1.

It is simpler to deal with this case directly in the algorithm than via the merit function. The resulting logic can be summarized as follows: If A_1 is not the main progenitor of its main descendant B_2, then we don't merge it into B_2 but we link it instead with the first secondary descendants that considers A_1 as its main progenitor.

3.3 Temporary Merger Events

When a sub-halo travels towards the core of its parent halo, it will merge with the central clump and disappear from the sub-halo lists. It can however re-emerge at a later snapshot and will be added back to the list as a newly born halo. Such a scenario is shown in Figure 4. Indeed, when this occurs, the merger tree code will deem the sub-halo to have merged into the main halo, and will likely find no progenitor for the re-emerged sub-halo, thus treating it as newly formed.

This is a problematic case because we lose track of the growth history of the sub-halo, regardless of its size, and massive clumps may be found to just appear out of nowhere in the simulation. This is a well known problem for configuration-space halo finders (Onions et al. 2012), and phase-space halo finders like `ROCKSTAR` (Behroozi et al. 2013a) have been developed precisely to alleviate this issue. While they typically perform better than configuration-space halo finders, Srisawat et al. (2013) found that phase-space halo finders aren't infallible in recovering all such missing haloes, and strongly recommend checking for links between progenitors and descendants in non-consecutive snapshots as well.

As our merger tree code works on the fly, future snapshots will not be available at the time of the merger tree analysis, so it will be necessary to check for progenitors of a descendant across multiple snapshots. This can be achieved by keeping track of the most bound particles of each clump when it is merged into some other clump. These tracer particles are also used to track *orphan galaxies*. [2] For this reason, we call these tracer particles "orphan particles", and progenitor-descendant links over non-adjacent snapshots "jumpers".

These jumper links between different haloes widely separated in time are less reliable than proper links between progenitors and descendants from adjacent snapshots. As we will discuss in Section 5.3, the quality of the merger trees increases with the number of tracer particles used. Using jumpers corresponds to using only one tracer particle over a large time interval, much larger than the one between two adjacent snapshots.

For this reason, priority is given to direct progenitor candidates in adjacent snapshots. Only if no direct progenitor candidates have been found for some descendant, then progenitor candidates from non-adjacent snapshots are

[1] Note that Springel et al. (2020) proposed another approach that allows explicitly fragmentation events to be included in the merger tree analysis.

[2] In the context of SAM, orphan galaxies are galaxies born at the center of dark matter haloes that merged later into bigger haloes and eventually dissolved due to over-merging. As a consequence, these galaxies don't have a parent halo or sub-halo anymore.

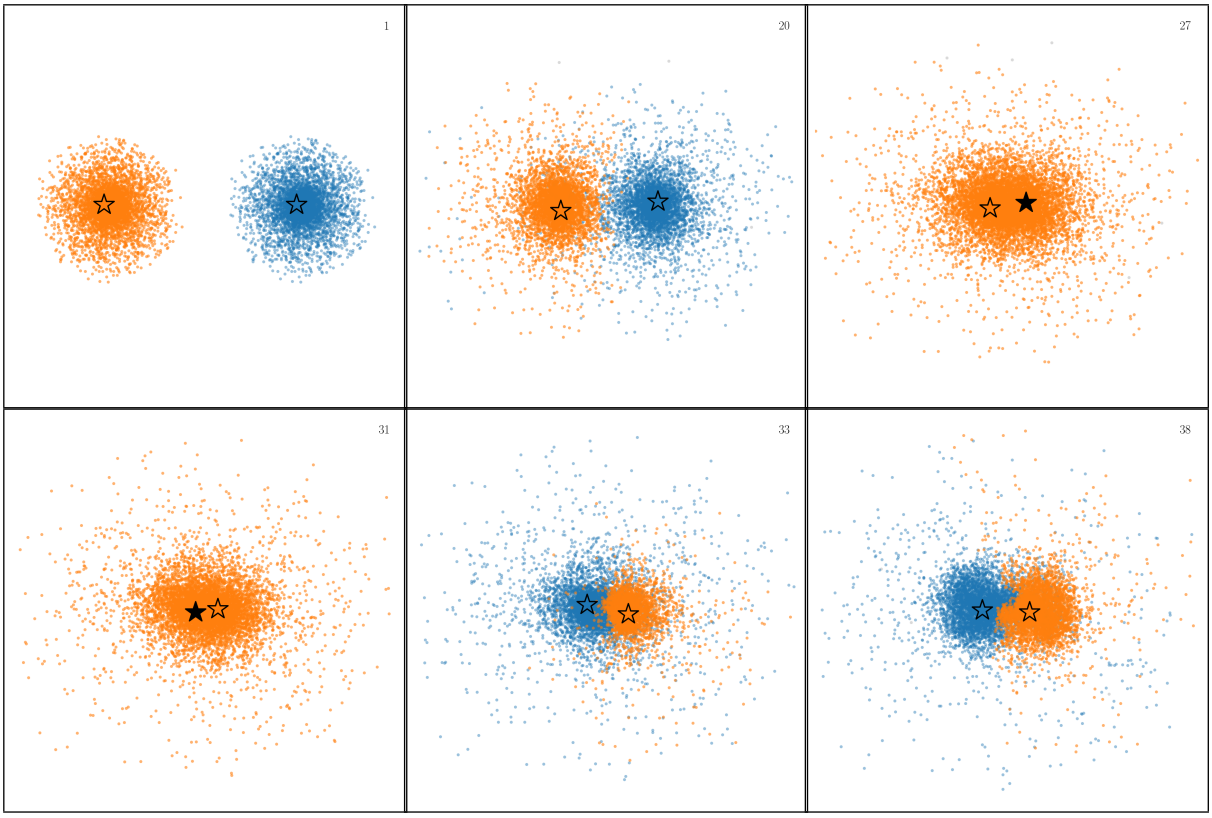

Figure 4. Illustration of how haloes can seemingly merge into another one and re-appear a few snapshots later. The blue and orange particles are two initially distinct haloes that pass through each other. The galaxies assigned to them are marked by a star with the same colour as the particles. Fully black stars mark orphan galaxies, which have lost their unique host (sub)halo. The number in the upper right corner of each plot is the snapshot number that is depicted. In snapshots 27-32, the halo-finding algorithm doesn't identify both haloes as distinct objects. However by tracing the blue halo's orphan galaxy it was possible to link the halo in snapshot 33 all the way back to snapshot 26. The initial conditions were created using `DICE` (Perret 2016).

searched for. Because these progenitors from non-adjacent snapshots are only tracked by one single particle, we don't use the merit function to rank them. Instead, we find the orphan particle within the descendant clump which is the most tightly bound.

In conclusion, although not ideal, using jumpers remains a necessity to track these temporary merger events. As a bonus, it allows us to track orphan galaxies as will be discussed in Section 6.

4 DETAILS OF OUR NEW ALGORITHM

The first step is to identify plausible progenitor candidates for descendant clumps, as well as descendant candidates for progenitor clumps. In our algorithm, this is done by tracking tracer particles across simulation snapshots. Tracer particles are selected within the list of all particles belonging to the clump, ranked from most bound to least bound. Indeed, the most bound particles are expected to remain well within the clump boundary between two snapshots. The main parameter of our method is the maximum number of these tracer particle used per progenitor clump, called $n_{\rm mb}$. The minimum number of tracer particles is obviously equal to $n_{\rm min}$, the minimum mass threshold adopted by the `PHEW` clump finder.

For every clump in the current snapshot, the $n_{\rm mb}$ most bound tracer particles are found and written to file. In the following output step, those files will be read in and the tracer particles will be used to determine which clumps in the previous snapshot are the progenitors of the clumps of the current snapshot.

As explained earlier, the main progenitor of each descendant and the main descendant of each progenitor need to be found. This search is performed iteratively. A main progenitor-descendant pair is established when the main progenitor of a descendant is the main descendant of said progenitor. At every iteration, all descendant candidates of all progenitors that haven't found their match yet are checked. The descendants without a matching progenitor however only move on to the next best progenitor candidate. This is how we deal with fragmentation events (see Section 3.2). For both descendants and progenitors, all candidates are ranked from "best" to "worst" based on the merit function we have introduced before and that we describe now in more details.

4.1 The Merit Function

Let $\mathcal{M}_{\rm pd}(A, B_i)$ be the merit function to be maximised for a list of descendant candidates B_i of a progenitor A. Let $n_{\rm mb}$ be the total number of particles of progenitor A that are being traced. Note that n_{mb} can be smaller than the total

number of particles in clump A. A straightforward ansatz for the merit function would be to based on the fraction of particle traced from the progenitor to the descendant candidate:

$$\mathcal{M}_{\rm pd}(A, B_i) \propto \frac{n_{A\cap B_i}}{n_{\rm mb}} \tag{3}$$

where $n_{A\cap B_i}$ is the number of tracer particles of A found in B. Similarly, we define $\mathcal{M}_{\rm dp}(A_i, B)$ as the merit function to be maximised for a list of progenitor candidates A_i of a descendant B. Another straightforward ansatz would be based on the fraction of particle traced from the progenitor candidates to the descendant:

$$\mathcal{M}_{\rm dp}(A_i, B) \propto \frac{n_{A_i\cap B}}{n_B} \tag{4}$$

where n_B is the total number of particles in the descendant B. In these two merit functions, $n_{\rm mb}$ and n_B are just normalizing factors. They are independent of the properties of the candidate and hence won't affect the selection process. We can therefore merge these two merit functions into one, defined as

$$\mathcal{M}(A, B) \propto n_{A\cap B} \tag{5}$$

The `PHEW` clump finder in `RAMSES` identifies the main halo as the clump with the highest density maximum. During a major merger event, the halo will have two clumps with similar masses and comparable maximum densities. It is then quite common that small variations in the value of the density maxima will cause the identification of the main halo to jump between these two clumps. Indeed, the particle unbinding algorithm will identify particles that are not bound to the sub-halo and pass them on to the main halo, modifying the resulting mass for the two clumps. This effect is particularly strong if one uses the *strictly bound* definition for the particle assignment. As a consequence, because the identification of the main halo varies, strong mass oscillations are expected. To counter this spurious effect, we modify the merit function to preferentially select candidates with similar masses:

$$\mathcal{M}(A, B) = \frac{n_{A\cap B}}{n_{\rm max} - n_{\rm min}} \tag{6}$$

where $n_{\rm max} = \max(n_A, n_B)$ and $n_{\rm min} = \min(n_A, n_B)$. An overview of other merit functions used in the literature is given in Table 1 of Srisawat et al. (2013).

4.2 Final Linking Process

The iteration is repeated until every descendant has been checked against every progenitor. Finally, the links in the tree making process are established following these two steps:

(i) Progenitors that have not found any available descendant will be considered to have merged into their highest ranked descendant candidate. These progenitors are recorded in the merger tree as *merged progenitors*. In this case, only one tracer particle is kept for future use, the most bound particle in the list of $n_{\rm mb}$ tracer particles. This single particle is referred to as the *orphan particle* of the merged progenitor. It is used to check whether the merger event was a final merger or only a temporary merger. It is also used to track orphan galaxies.

(ii) Descendants that have not found any available progenitor will be checked against non-consecutive past snapshots. The particles of the descendant are compared to the orphan particles in the list of past merged progenitors[3]. The most strongly bound orphan particle will be used to restore the broken link with its main progenitor. Finally, remaining descendants without a progenitor are considered as being newly formed.

For the interested reader, an even more detailed description of the merger tree algorithm is given in Appendix A.

5 TESTING AND OPTIMIZING THE ALGORITHM

The current implementation of our merger tree algorithm needs several free parameters to be chosen by the user. We present in this section multiple tests that reveal the recommended values for these parameters. We use for this a single reference cosmological simulation and analyze the merger trees we obtained for different set of parameters. A similar methodology was used in the Sussing Merger Trees Comparison Project (Srisawat et al. 2013; Wang et al. 2016; Avila et al. 2014; Lee et al. 2014). This is why we adopt in this section several tests from this seminal work, as they are quite efficient at testing the strengths and the weaknesses of merger tree codes. They also allow for a direct comparison of our new implementation with many other state-of-the-art merger tree codes in the community.

5.1 Test Suite

We now list the different diagnostics we use to characterize the quality of our merger tree algorithm.

(i) *Length of the Main Branch of a Halo*
The length of the main branch of a halo is simply defined as the number of snapshots in which a halo and all its progenitors are detected. A halo at $z = 0$ without any progenitors will be considered as newly formed and thus will have a main branch of length 1. If a halo appears to merge temporarily into another and re-emerges at a later snapshot, the missing snapshots will be counted towards the length of the main branch as if they weren't missing. Traditionally, finding long main branches is considered as a good thing for a merger tree code.

(ii) *Number of Branches of a Halo*
Another popular quantity is the number of branches of the tree leading to the formation of a halo at $z = 0$. The main branch is included in this count, thus the minimal number of branches is 1. If a different choice of parameters leads to a reduction of the number of branches, it usually corresponds to an increase of the average length of the main branches and a smaller number of merger events. In the hierarchical picture of structure formation, one would expect more massive clumps to have longer main branches and a higher number of branches.

[3] There is an option to remove past merged progenitors from the list if they have merged into their main descendant too many snapshots ago. By default, however, the algorithm will store them all until the very end of the simulation.

(iii) *Logarithmic Mass Growth of a Halo*
The logarithmic mass growth rate of a halo is computed using the following finite difference approximation:

$$\frac{\mathrm{d}\log M}{\mathrm{d}\log t} \simeq \frac{(t_{k+1}+t_k)(M_{k+1}-M_k)}{(t_{k+1}-t_k)(M_{k+1}+M_k)} \equiv \alpha_M(k,k+1) \tag{7}$$

where k and $k+1$ are two consecutive snapshots, with the corresponding halo mass M_k and M_{k+1} and times t_k and t_{k+1}. A convenient approach was proposed by Wang et al. (2016) to reduce the range of values to the interval $(-1,1)$ using the new variable

$$\beta_M = \frac{2}{\pi}\arctan(\alpha_M) \tag{8}$$

Note that we expect the mass of dark matter haloes to increase systematically with time. We also expect in some cases the mass to remain constant or even to decrease slightly. We nevertheless expect the distribution of β_M to be skewed towards $\beta_M > 0$. $\beta_M \to \pm 1$ imply $\alpha_M \to \pm\infty$, indicating suspiciously extreme cases of mass growth or mass loss.

(iv) *Mass Growth Fluctuation of a Halo*
Mass growth fluctuations are defined similarly as

$$\xi_M = \frac{\beta_M(k,k+1) - \beta_M(k-1,k)}{2} \tag{9}$$

where $k-1$, k, $k+1$ are three consecutive snapshots. A smooth mass accretion history generally leads to $\xi_M \simeq 0$. Strong deviations from zero could indicate an erratic behaviour, indicating extreme mass loss followed by extreme mass growth and vice versa. Within the standard model of structure formation, this behaviour is expected only during major merger events. Otherwise, it might indicate either a misidentification by the merger tree code or a misdetection by the halo finder.

Ideally, we should have tested `ACACIA` on the dataset used in Srisawat et al. (2013) and Avila et al. (2014). This would have enabled a direct comparison of the performance to other merger tree codes. However, `ACACIA` was designed to work on the fly with the `RAMSES` code. Using it as a post-processing tool would defeat its purpose and as a matter of fact handling foreign halo catalogues has proven technically impossible. Furthermore, we also want to demonstrate that the `PHEW` halo finder can be used within the `RAMSES` code to produce reliable merger trees. For these various reasons, we have decided to perform the same tests but using our own dataset generated on the fly by `RAMSES`.

Despite this limitation, we have performed a direct comparison to other halo finders and merger tree codes using the exact same merger tree parameters as in Avila et al. (2014). The results are given in Appendix B. Our results are comparable to e.g. the `Sublink` and `VELOCIraptor` tree builders with `AHF` or `Subfind` halo finders as presented in Avila et al. (2014), demonstrating that `ACACIA` performs similarly than other state-of-the-art tools.

In this section, we would like to explore different parameters and see how they affect the quality of the merger tree. Our tests are performed on a single DMO simulation with $256^3 \approx 1.7 \times 10^7$ particles of identical mass $m_p = 1.55 \times 10^9 M_\odot$. We have adopted the following cosmological parameters: $H_0 = 70.4$ km s^{-1}Mpc^{-1}, $\Omega_m = 0.272$ and $\Omega_\Lambda = 0.728$. For the clump finder, we have adopted a outer density threshold of $80\bar{\rho}$ and a saddle surface density threshold of $200\bar{\rho}$, where $\bar{\rho} = \Omega_m \frac{3H^2}{8\pi G}$ is the average density. The minimal mass for clumps is set to 10 particles. Note that for the histogram of the logarithmic mass growth and mass growth fluctuations, we adopt a threshold for the clump mass of 200 particles for sake of visibility. Choosing a smaller mass threshold would indeed give too much weight to small mass, poorly resolved haloes in our statistical analysis.

The snapshot generation was set as follows: As virtually no haloes are detected before $a \leq 0.1$, only a few snapshots are stored up to $a = 0.1$ in steps of $\Delta a \simeq 0.02$. From this point on, more frequent snapshots are created every $\Delta t \simeq 0.3$ Gyrs up until $a = \frac{1}{3}$, after which an even smaller time interval of $\Delta t \simeq 0.2$ Gyrs was chosen. This choice resulted in 67 snapshots generated up to $z = 0$. The simulation was then continued for 3 further snapshots with $\Delta t \simeq 0.2$ Gyrs to ensure that the merging events at $z = 0$ are actual mergers and not temporary mergers that will re-emerge later.

5.2 Varying the Clump Mass Definition

In our current implementation, there are two important parameters that can have a strong effect on the halo catalogue (beside the mass and the density thresholds mentioned earlier) and the corresponding merger tree.

The first one is the exact definition adopted for the mass of the sub-haloes in the merit function. For main haloes, there is no ambiguity as the mass is defined as the sum of the masses of all particles contained within the boundary of the halo (set by the outer density isosurface). This is not the case for sub-haloes, because of the unbinding process described in Section 2. Indeed, unbound particles are removed from their original sub-halo and passed to the parent sub-halo in the hierarchy. Clump masses are therefore defined as the sum of the mass of all bound particles. In the merit function evaluation, we however consider two different cases to compute the mass: 1- the mass is equal to the sum of the masses of only the bound particles, like for sub-haloes or 2- the mass is equal to the sum of the masses of all particles within their boundaries (set by the saddle surface with neighbouring clumps), like for main haloes. In the former case, the mass used in the merit function is identical to the clump mass. It is referred to as the `exclusive` case. In the latter case, the mass in the merit function is different that the sub-halo mass definition but identical to the main halo mass definition. We refer to this case as `inclusive`.

The second important definition is the exact boundedness criterion adopted for the unbinding process. As discussed in Section 2, we explored two different cases: When particles are allowed to leave the outer boundary of their host clump (and possibly come back later) or when particles are not allowed to cross the saddle surface during their orbital evolution. In the first case, we only require the binding energy to be negative, while in the second case, the binding energy has to be smaller than the gravitational potential of the nearest saddle point. We call the first case `loosely bound` and the second case `strictly bound`.

We now test our algorithm with these four different options for the clump masses, using the simulation presented in the previous section. Note that we used here $n_{\mathrm{mb}} = 200$

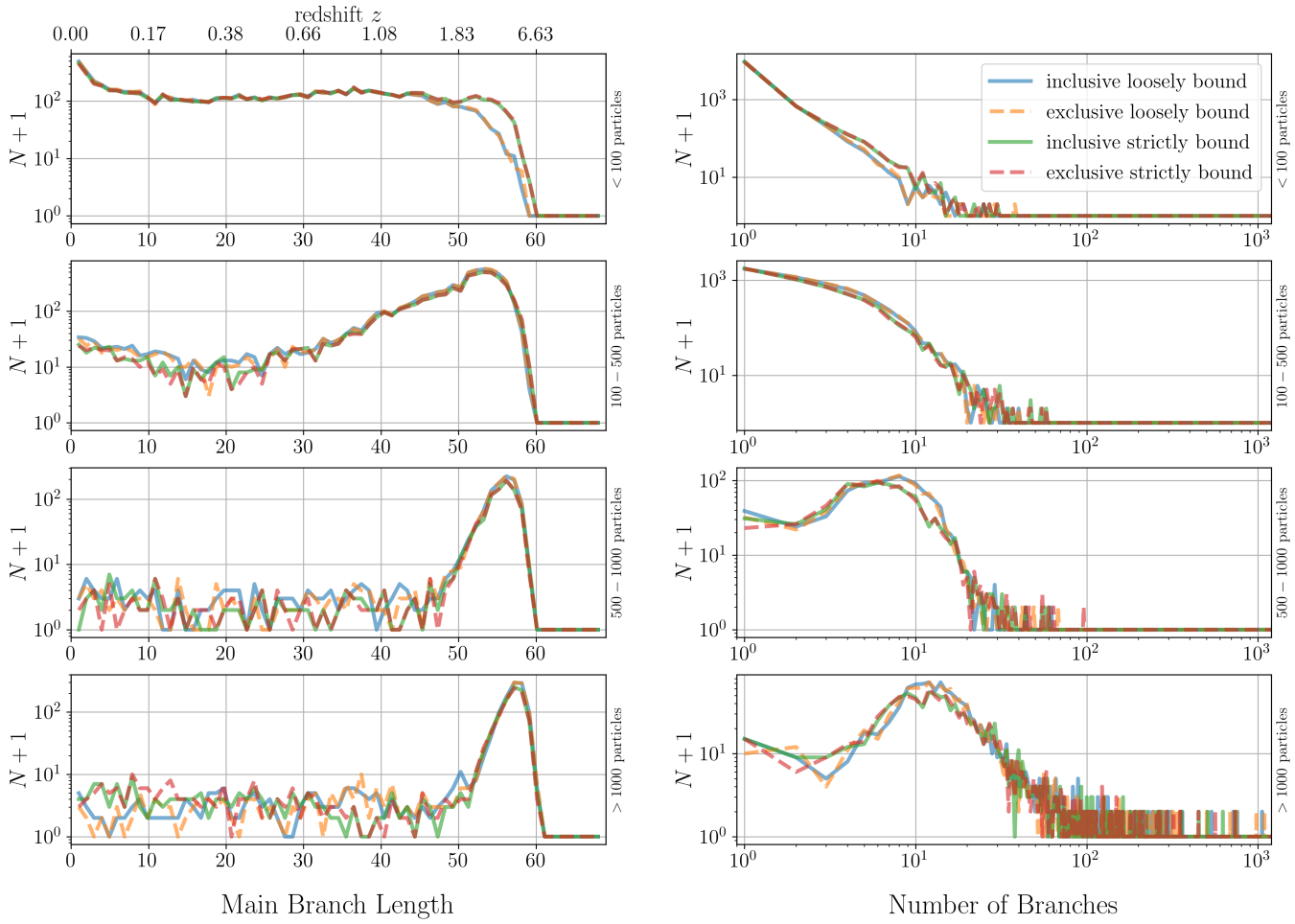

Figure 5. Histogram of the length of main branch (left) and of the number of branches (right) for all clumps (halo and sub-halo) detected at $z = 0$. Each row corresponds to a different range of clump masses (expressed in particle numbers): less then 100 (top), 100-500, 500-1000 and more than 1000 (bottom). We compare these histograms for four different cases: whether unbound particles are included (`inclusive`) or excluded (`exclusive`) in the evaluation of the merit function, and whether bound particles are `loosely bound` or `strictly bound`.

Table 2. Average data for all clumps at $z = 0$ depending on whether to consider particles which might wander off into another clump as bound (`loosely bound`) or not (`strictly bound`). The results shown are for the `exclusive` mass definition, which show no significant difference to when the `inclusive` mass definition is used.

	`strictly bound`	`loosely bound`
total clumps	16262	17242
median number of particles in a clump	83	93
average main branch length		
clumps with $<$ 100 particles	23.2	20.5
clumps with 100-500 particles	48.8	48.6
clumps with 500-1000 particles	54.7	54.9
clumps with $>$ 1000 particles	54.0	56.3
average number of branches		
clumps with $<$ 100 particles	1.3	1.2
clumps with 100-500 particles	3.4	3.6
clumps with 500-1000 particles	8.7	8.7
clumps with $>$ 1000 particles	29.5	28.6

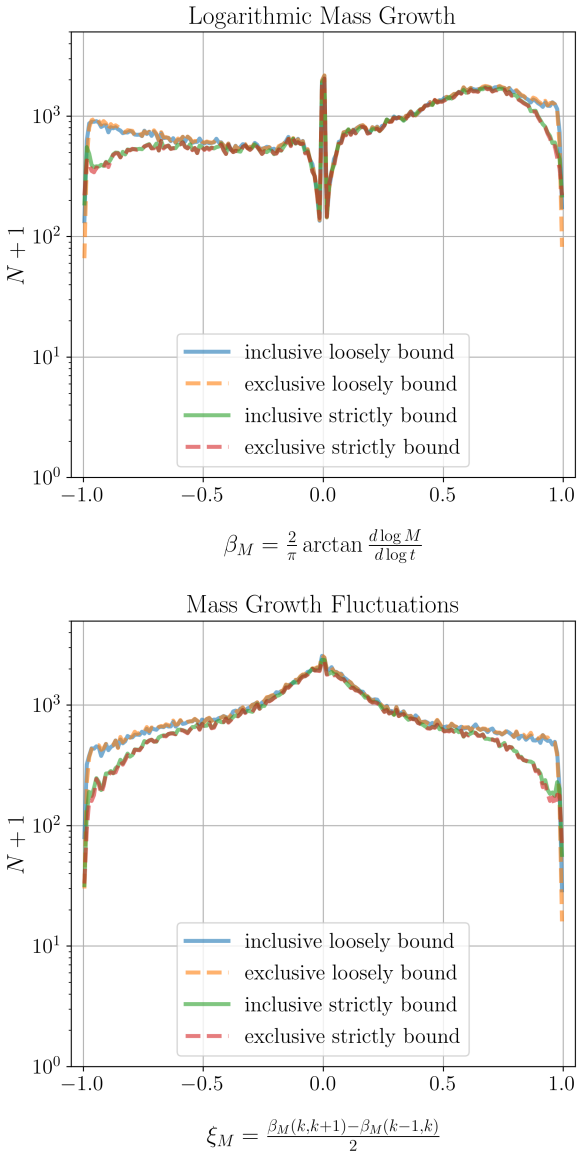

Figure 6. Histogram of the logarithmic mass growth (top) and the mass growth fluctuation (bottom) for all clumps (halo and sub-halo) detected in two (top) or three (bottom) consecutive snapshots of the simulation and with more than 200 particles. We compare these histograms for four different cases: whether unbound particles are included (`inclusive`) or excluded (`exclusive`) in the evaluation of the merit function, and whether bound particles are `loosely bound` or `strictly bound`.

tracer particles to identify links in the merger tree. We will study the impact of this other important parameter in the next section.

We show in Figure 5 the histogram of the length of the main branch and the histogram of the number of branches for each clump (halo and sub-halo) at $z = 0$ and for each of our four different mass definitions. In all four cases, we see that more massive clumps tend to have longer main branches and a higher number of branches. This is also visible from the average length of the main branch and the average number of branches in different bins of halo masses given in Table 2. This is a well-known property of cosmological simulations in the hierarchical scenario of structure formation.

Whether clump masses are defined in an `exclusive` manner (like for sub-haloes) or in an `inclusive` manner (like for main haloes) in the merit function has negligible effect on these two statistics. This means that this a priori large difference in the mass definition of the merit function has no effect on the linking process of the merger tree. The distinction between `strictly bound` and `loosely bound` particles does not change much for larger clumps but does change the length of the main branch (and to a lesser extent the number of branches) for small mass clumps (less than 100 particles). Our first idea was that `strictly bound` particles might be better at identifying robust links between snapshots. It turned out that the main effect of changing the mass definition from `loosely bound` to `strictly bound` is to reduce the mass of the clump and to promote them systematically from a larger mass bin to a smaller mass bin. We see indeed in Figure 5 and Table 2 that the number of clumps is reduced in the large mass bins and increased in the smallest mass bin, explaining that this change in the mass definition merely transfers clumps between different bins and affects the statistics accordingly.

We also note in Figure 5 that a few large clumps (with mass larger than 500 particles) at $z = 0$ have a main branch length of unity. These large clumps don't have any progenitor and thus essentially appeared out of nowhere. As explained in Srisawat et al. (2013), this effect is present in many state-of-the-art merger tree codes and is believed to be due to fragmentation events at the periphery of large haloes leading to a misidentification of a few rare progenitor-descendant links.

The histogram of the logarithmic mass growth shown in Figure 6 is indeed skewed towards $\beta_M > 0$, demonstrating that clump masses are on average growing. Adopting a merit function based on the `inclusive` or `exclusive` mass definition has here also no effect on the mass growth and mass growth fluctuation statistics. At first sight, using the `strictly bound` instead of the `loosely bound` definition would have led to more robust links and a smoother mass growth. On Figure 6, we do see in the latter case more extreme mass growth around $\beta_M \to \pm 1$ and mass growth fluctuations around $\xi_M \to \pm 1$. We verified that the increase in the number of these extreme events for the `loosely bound` case is in fact due to a larger number of small sub-haloes that satisfy the adopted mass threshold of 200 particles. This just means that mass growth statistics is more robust for large, well resolved haloes, while smaller clumps, closer to the resolution limit (between 10 and 100 particles; see discussion below) are less reliable.

In conclusion, whether to use `inclusive` or `exclusive` mass definitions in the merit function has no effect on the final merger trees, while using a `strictly bound` definition for the clump mass is preferable, since it naturally selects better resolved, higher mass clumps from the halo catalogue.

5.3 Varying the Number of Tracer Particles

In this section, we study the effect of varying the number of tracer particles on our various diagnostics of tree quality. We show in Table 3 the average number of branches and the average length of the main branch for all our detected clumps at $z = 0$ organised in different mass bins. We see that the effect of the number of tracer particles used is quite mild.

Table 3. Average length of main branch and average number of branches for clumps in different mass bins at $z = 0$ and for varying numbers of clump tracer particles $n_{\rm mb}$.

$n_{\rm mb}$ =	1	10	50	100	200	500	1000
Average main branch length							
clumps with $<$ 100 particles	24.2	24.3	23.6	23.4	23.2	22.9	22.7
clumps with 100-500 particles	50.4	50.1	49.5	49.1	48.8	48.8	48.8
clumps with 500-1000 particles	55.2	54.9	53.3	54.1	54.7	54.3	54.2
clumps with $>$ 1000 particles	56.7	54.9	52.3	52.9	54.0	55.8	56.4
Average number of branches							
clumps with $<$ 100 particles	1.2	1.3	1.3	1.3	1.3	1.4	1.4
clumps with 100-500 particles	2.7	3.0	3.3	3.3	3.4	3.6	3.6
clumps with 500-1000 particles	6.6	7.2	8.1	8.2	8.7	8.9	9.1
clumps with $>$ 1000 particles	20.4	25.2	27.3	28.6	29.5	30.4	31.4

Table 4. Number of dead trees pruned from the merger tree catalogue for varying numbers of tracer particles $n_{\rm mb}$ throughout all snapshots. "LIDIT" is an abbreviation for "last identifiable descendant in tree". For a LIDIT, no descendant could have been identified throughout the simulation and consequently the corresponding tree is considered dead and pruned from the merger tree catalogue. LIDITS are obviously a spurious feature of the merger tree algorithm.

$n_{\rm mb}$ =	1	10	50	100	200	500	1000
dead trees pruned from tree catalogue	33924	23091	22146	22131	22130	22129	22129
highest particle number of a LIDIT	1369	236	236	236	236	157	157
median particle number of a LIDIT	19	20	20	20	20	20	20
LIDITs with $>$100 particles pruned	513	42	32	26	25	24	24

Table 5. Number of "jumpers" (progenitor-descendant links found across non-adjacent snapshots) during the entire simulation for varying number of tracer particles $n_{\rm mb}$.

$n_{\rm mb}$ =	1	10	50	100	200	500	1000
jumpers	20176	20905	22074	22041	20970	19307	18249
jumper progenitors							
clumps with $<$ 100 particles	18776	19421	20175	19643	18028	15972	14823
clumps with 100- 500 particles	1348	1413	1796	2266	2760	3045	3082
clumps with 500-1000 particles	40	50	74	91	126	196	226
clumps with $>$ 1000 particles	12	21	29	41	56	94	118

Even with as few as one tracer particle do we manage to recover the correct average main branch length. This is also true for small mass haloes, although with a slightly reduced accuracy. This also validates our orphan particle technique to track temporary merger events.

On a closer look, the average number of branch seems however more affected by the number of tracer particles. With only one tracer particle we loose 30% of the links associated to merger events. This can be easily explained by the fact that too few tracer particles cannot be distributed across enough descendant candidates to identify potential links. It appears that using 100 tracer particles seems to be enough to recover most of the otherwise broken links. These conclusions remain the same after looking at the histogram of the number of branches and the histogram of the main branch length for the same clumps in Figure 7. Here again, we see the peak of the histogram of the number of branches being shifted to the right when increasing the number of tracer particles. We also see that using 100 tracer particles seems enough to almost recover the correct distribution.

We now examine the effect of the number of tracers on the mass growth (and on the mass growth fluctuations) of all our detected clumps within the entire redshift range. We see in Figure 8 that the effect is very weak, except for the extreme cases $\beta_M \simeq \pm 1$ and $\xi_M \simeq \pm 1$, corresponding to spurious links in the merger tree. We believe that these extreme mass growth cases correspond to broken links due to the small number of tracer particles. Here again, using more than 100 tracer particles seem to get rid of most of these spurious cases. Note that we include in these histograms only clumps with more than 200 particles.

We now study in details another spurious effect of our merger tree algorithm, shared by many other merger tree code in the literature, namely dead tree branches. A dead branch arises when no descendant could have been identified after a certain redshift, even after looking for all subsequent snapshots using the corresponding orphan particle. Such an event is called a "Last Identifiable Descendant In Tree" or LIDIT. When such a case occurs, it is customary to prune the corresponding tree from the tree catalogue. When not enough tracer particles are used, we expect such spurious dead links to appear. Table 4 shows the statistics of these LIDITs (or tree pruning events), which confirms that the number of LIDITs decreases strongly when using more

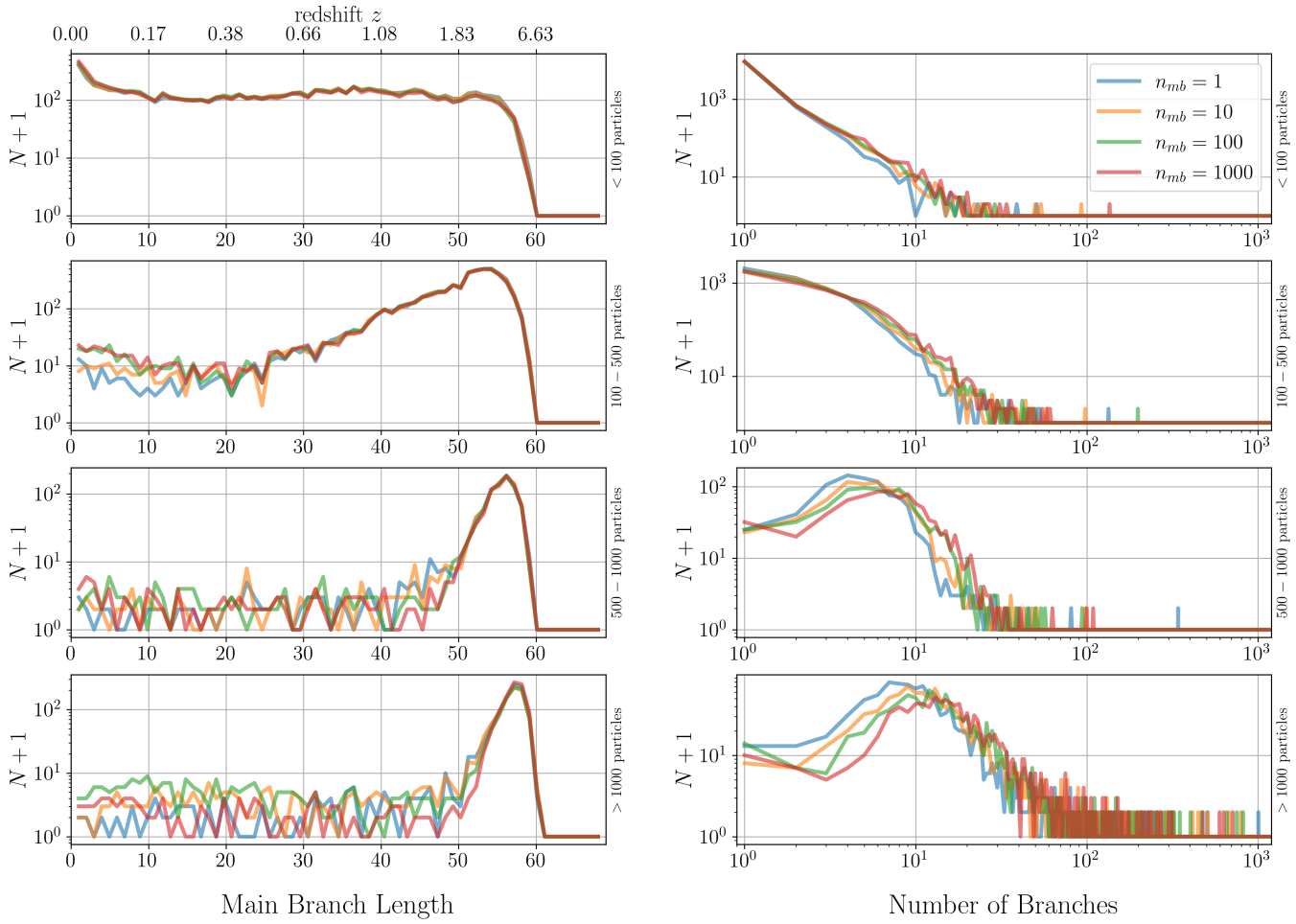

Figure 7. Histogram of the length of the main branch (left) and histogram of the number of branches (right) for all clumps (haloes and sub-haloes) detected at $z = 0$ for different numbers of tracer particles n_{mb} indicated in the legend. Each row corresponds to a different range of clump masses (expressed in particle numbers): less then 100 (top), 100-500, 500-1000 and more than 1000 (bottom). In all cases, we used the `exclusive` and `strictly bound` clump mass definitions.

and more tracer particles. We also show in the same table the typical and maximum mass of the LIDITs. Interestingly, the maximum mass also strongly decreases when using more tracer particles. When enough tracer particles are used, we see that LIDITs are typically less massive than 200 particles. We believe they correspond to poorly resolved clumps that are subject to all sorts of spurious numerical effects. Adopting a conservative resolution limit of 200 particles per halo removes all the LIDITs from our catalogue, as long as one uses more than 200 tracer particles.

We finally study a specific aspect of our merger tree algorithm, namely the possibility to follow the temporary mergers of clumps that travel through another clumps to emerges later as a distinct object. We show in Table 5 the number of these temporary mergers that we call "jumpers" as they represent links across non-adjacent snapshots that we are able to "repair" using orphan particles. When we increase the number of tracer particles, the total number of jumpers decreases steadily, but on a closer look, this is only true for poorly resolved clumps with less than 100 particles. For larger clumps, the number of jumpers actually increases with the number of tracer particles. This is a similar behavior than for the number of branches: More tracer particles allows more merger events to be detected, but also allows more non-adjacent descendant candidates to be found. We here also recommend to use 200 tracer particles as a compromise between speed and proper detection of jumpers in the simulation.

We show in Figure 9 the histogram of the distance in time between the two non-adjacent snapshot of all jumpers in our merger tree. We see that most jumpers have a distance of only 2 snapshots. They corresponds to clumps traversing another clump and re-emerging a snapshot later as a distinct halo. We also see in this histogram that the number of jumpers increases with the number of tracer particles. But overall, the statistics of the distance between jumpers is relatively robust, with only very rare cases with a distance larger than 10 snapshots. Figure 10 shows the histogram of the mass ratio between the jumper progenitor and the jumper descendant. As expected, it peaks at one, which means that the mass of the clump that re-emerges in a later snapshot is close to the mass of the clump that disappeared in an earlier snapshot. Note that this is not due to the merit function, because for jumpers we only use a single orphan particle to repair the link. This is a clear sign that the same clump is identified before and after the temporary merger. We also

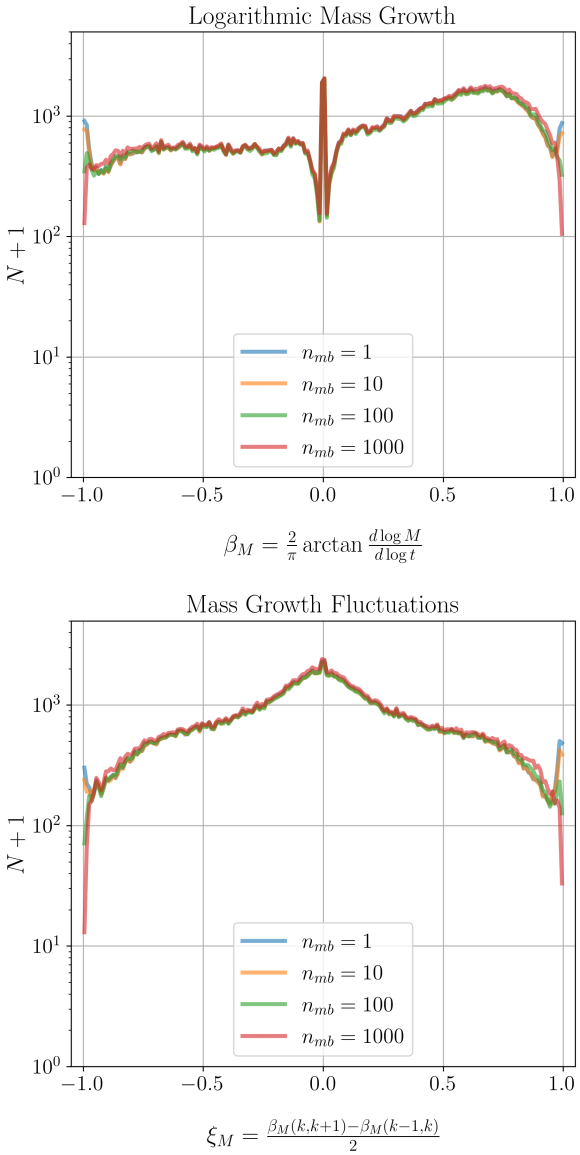

Figure 8. Histogram of the logarithmic mass growth (top) and histogram of the mass growth fluctuation (bottom) for all clumps (halo and sub-halo) detected in two (top) or three (bottom) consecutive snapshots of the simulation and with more than 200 particles. We compare these histograms for four different numbers of tracer particles $n_{\rm mb}$ as indicated in the legend. In all cases, we used the `exclusive` and `strictly bound` clump mass definitions.

see that the distribution is slightly skewed toward mass ratio smaller than 1, with values always bounded between 0.3 and 3. Only a few very rare cases show more extreme mass ratios. This means that clumps hosting our orphan particles either preserve their mass (over 2 snapshots) or loose mass (on average), usually when the time between non-adjacent snapshots increases.

In conclusion, we found that $n_{\rm mb} \simeq 200$ is a safe choice to obtain robust results for our merger tree algorithm, in light of the diagnostics we have used in this section. We also recommend adopting a conservative mass threshold of 200 particles per clumps to get rid of a few rare spurious dead branches that would need to be pruned from the halo catalogue anyway.

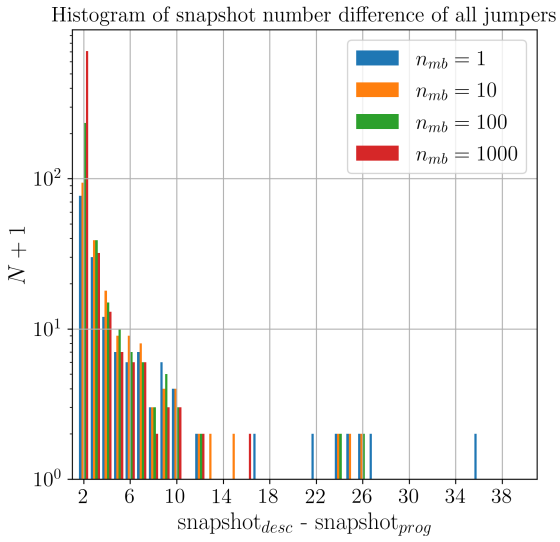

Figure 9. Histogram of the difference in snapshot numbers for jumpers, i.e. progenitor-descendant links which are made across non-adjacent snapshots. Only pairs where both the progenitor's and the descendant's masses exceed 200 particle masses were included in this plot. We compare these histograms for four different numbers of tracer particles $n_{\rm mb}$ as indicated in the legend. In all cases, we used the `exclusive` and `strictly bound` clump mass definitions.

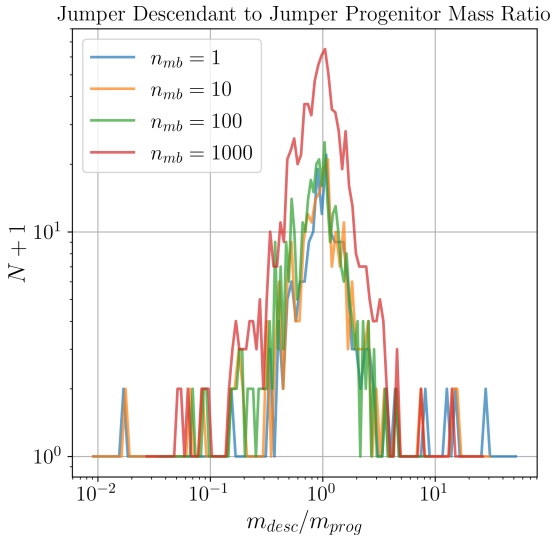

Figure 10. Histogram of the ratio of descendant mass to progenitor mass for jumpers, i.e. progenitor-descendant links which are made across non-adjacent snapshots. Only pairs where both the progenitor's and the descendant's masses exceed 200 particle masses were included in this plot. We compare these histograms for four different numbers of tracer particles $n_{\rm mb}$ as indicated in the legend. In all cases, we used the `exclusive` and `strictly bound` clump mass definitions.

6 APPLICATION OF THE MERGER TREE ALGORITHM: CREATING A MOCK GALAXY CATALOGUE

Now that we know the optimal parameters to create a merger tree with `ACACIA`, we use it to generate a mock galaxy catalogue. We summarize here the main data products generated by our code.

(i) For every snapshot of the N body simulation, we have the full clump catalogue generated by `PHEW`. Every clump is uniquely classified as a main halo or as a sub-halo.
(ii) Every dark matter particle is given the clump index it belongs to, or zero if it belongs to the smooth background.
(iii) For each clump, we follow and store the index of the n_{mb} most strongly bound particles, the position of the density peak, the clump bulk velocities, centre of mass, mass, and other clump properties.
(iv) For each clump, we store the index of the direct progenitors (in particular its main progenitor) and its peak mass over its entire past formation history.
(v) We augment our clump database with orphan particles, storing for each of them the index of the last known main progenitor and its peak mass.

To generate the mock galaxy catalogue, we use the well-established technique of Sub-Halo Abundance Matching (SHAM). This technique was introduced more than ten years ago as a surprisingly simple and accurate method to populate a pure dark matter simulation with galaxies with the correct clustering statistics (Vale & Ostriker 2006b; Shankar et al. 2006; Conroy et al. 2006).

Although several implementations of SHAM exist in the recent literature (Guo et al. 2010; Wetzel & White 2010; Moster et al. 2010; Trujillo-Gomez et al. 2011; Nuza et al. 2013; Zentner et al. 2014; Chaves-Montero et al. 2016) we use here the variant based on the peak clump mass as a proxy for the stellar mass (Reddick et al. 2013), using the Stellar-Mass-to-Halo-Mass (SMHM) relation of Behroozi et al. (2013c).

Using the peak clump mass is believed to mimick the actual stellar mass growth of a galaxy, first as a central galaxy when the host clump was a main halo, then as a satellite galaxy when the halo was accreted and became a sub-halo. After infall, although the clump mass might decrease quickly due to interactions within the parent main halo, this model assumes that the stellar mass in the galaxy remains constant (Nagai & Kravtsov 2005).

Note that in the SHAM methodology, the merger tree algorithm plays a central role.

(i) In order to compute the clump peak mass, we need the entire mass growth past history.
(ii) In order to follow galaxies even when the parent clump has dissolved due to numerical overmerging, we need to follow orphan particles and their peak clump mass.

Our parent DMO simulation uses $512^3 \simeq 1.3 \times 10^8$ particles and a box wize of 100 comoving Mpc with a particle mass resolution of $m_p \simeq 3.1 \times 10^8 M_\odot$. The cosmological parameters are taken from the 2015 Planck Collaboration results (Planck Collaboration et al. 2016), with Hubble constant $H_0 = 67.74$ km s^{-1}Mpc^{-1}, density parameters $\Omega_m = 0.309$, $\Omega_\Lambda = 0.691$, scalar spectral index $n_s = 0.967$, and fluctuation amplitude $\sigma_8 = 0.816$. The initial conditions were created using the `MUSIC` code (Hahn & Abel 2011). As explained before, the density threshold for clump finding was chosen to be 80 times the mean background density, $\bar{\rho} = \Omega_m \rho_c$, and the saddle threshold for haloes was set to $200\bar{\rho}$, where $\rho_c = \frac{3H_0^2}{8\pi G}$ is the cosmological critical density.

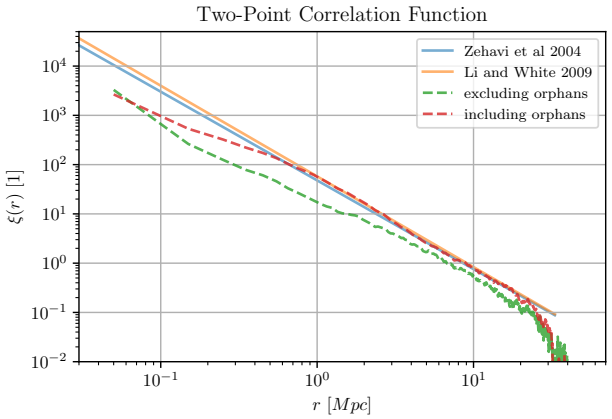

Figure 11. The predicted stellar mass 2-point correlation function (2PCF) $\xi(r)$ of our SHAM model, including and excluding orphan galaxies, compared to the power law fits of the observed 2PCF in Li & White (2009) and Zehavi et al. (2004).

As a first application of our merger tree code, we will now compute the two point correlation functions and the average radial profiles of galaxy clusters in our simulation. We will compare our results to observational data, and demonstrate that this is only when we include orphan galaxies that our results are in good agreement with observations.

6.1 The Stellar Mass Correlation Function

The stellar mass two-point correlation function (2PCF) $\xi(r)$ is computed via inverse Fourier transform of the power spectrum $P(k)$ (e.g. Mo et al. 2011), which itself can be obtained from the Fourier transform of the stellar mass density contrast field $\delta(\mathbf{r})$:

$$\delta_\mathbf{k} = \frac{1}{V} \int e^{i\mathbf{kr}} \delta(\mathbf{r}) \mathrm{d}^3\mathbf{r} \tag{10}$$

with

$$\delta(\mathbf{r}) = \frac{\rho(\mathbf{r})}{\langle \rho(\mathbf{r}) \rangle} - 1 \tag{11}$$

Where $\rho(\mathbf{r})$ is the galaxy stellar mass density field and $\langle \rho(\mathbf{r}) \rangle$ is the corresponding mean density, $V = L^3$ is the volume of our large box on which the density field is assumed periodic, and $\mathbf{k} = \frac{2\pi}{L}(i_x, i_y, i_z)$, where i_x, i_y, i_z are integers. The Fourier transform is performed using the FFTW library (Frigo & Johnson 2005). The power spectrum $P(k)$ and the 2PCF $\xi(r)$ are given by

$$P(k) = V \langle |\delta_\mathbf{k}|^2 \rangle \tag{12}$$

$$\xi(r) = \frac{1}{(2\pi)^3} \int e^{-i\mathbf{kr}} P(k) \mathrm{d}^3\mathbf{k} \tag{13}$$

The simulation box is divided in a uniform grid of 1024^3 cells and the stellar mass is deposited on the grid using a

cloud-in-cell interpolation scheme. The cloud-in-cell scheme consists of assigning each galaxy a cubic volume ("cloud") the size of a grid cell centered on the galaxy's position. The galaxy stellar mass is assumed to be uniformly distributed within the cloud, and is deposited on the uniform grid cells according to the volume fraction of the cloud that resides within each cell.

We only include galaxies with masses above $10^9 M_\odot$. Using our adopted SMHM relation, these galaxies are hosted in haloes with mass larger than $\sim 10^{11} M_\odot$, or more than 300 particles. This threshold ensures that we only use well resolved clumps for our analysis, as discussed in the previous sections.

The predicted 2PCF $\xi(r)$ is shown in Figure 11, and is compared to the observational results of Li & White (2009) and Zehavi et al. (2004). Note that the observed galaxy catalogue is presented as complete down to $10^8 M_\odot$, one order of magnitude smaller than our simulated catalogue. To highlight the influence of orphan galaxies, we have computed the predicted 2PCF both with and without orphan galaxies. Including orphan galaxies produces a correlation function in much better agreement with observations. Our theoretical 2PCF obtained reproduces the observed power law fit over two orders of magnitude in scale of $r \sim 0.3 - 25$ Mpc. On the smallest scales, below 0.3 Mpc, our predictions are likely to be affected by our limited mass resolution.

The role of the orphan galaxies is particularly important on intermediate scales ~ 0.2 Mpc $< r < 2$ Mpc. This behaviour is explained by the fact that orphan galaxies are located within host haloes, thus contributing to the correlations at small distances, the so-called 1-halo term. Our conclusion are in agreement with those of Campbell et al. (2018), who have found that the inclusion of orphan galaxies for mass-based SHAM models improves the clustering statistics of mock galaxy catalogues, particularly so at small scales.

6.2 Radial Profiles of Satellites in Large Clusters

In this section, we compute the radial profiles of number and mass densities of satellite galaxies in the 10 largest clusters in our simulation. In order to compare to observations, we will follow as closely as possible the method described in van der Burg et al. (2015), who analyzed 60 massive clusters between $0.04<z<0.26$ in the Multi-Epoch Nearby Cluster Survey and the Canadian Cluster Comparison Project.

We identified 10 haloes at $z = 0$ with the highest mass. The mass is defined here like in van der Burg et al. (2015) as M_{200c}, the mass included within radius R_{200c} at which the average enclosed mass density of the halo is 200 times the cosmological critical density ρ_c. The M_{200c} masses of our 10 selected haloes range between $1.4 \times 10^{14} M_\odot$ and $4.8 \times 10^{14} M_\odot$ with a median value of $1.6 \times 10^{14} M_\odot$. Due to our limited box size, these haloes are on the lower end of the sample observed in van der Burg et al. (2015), who reported masses ranging from $0.8 \times 10^{14} M_\odot$ and $1.6 \times 10^{15} M_\odot$ with a median value of $8.6 \times 10^{14} M_\odot$.

We compute the projected number density profiles and projected mass density profiles as follows: For each halo, we first project the galaxies (excluding the central galaxy) along each coordinate axis obtaining three images. We then compute cylindrical profiles using radial shells equally space in

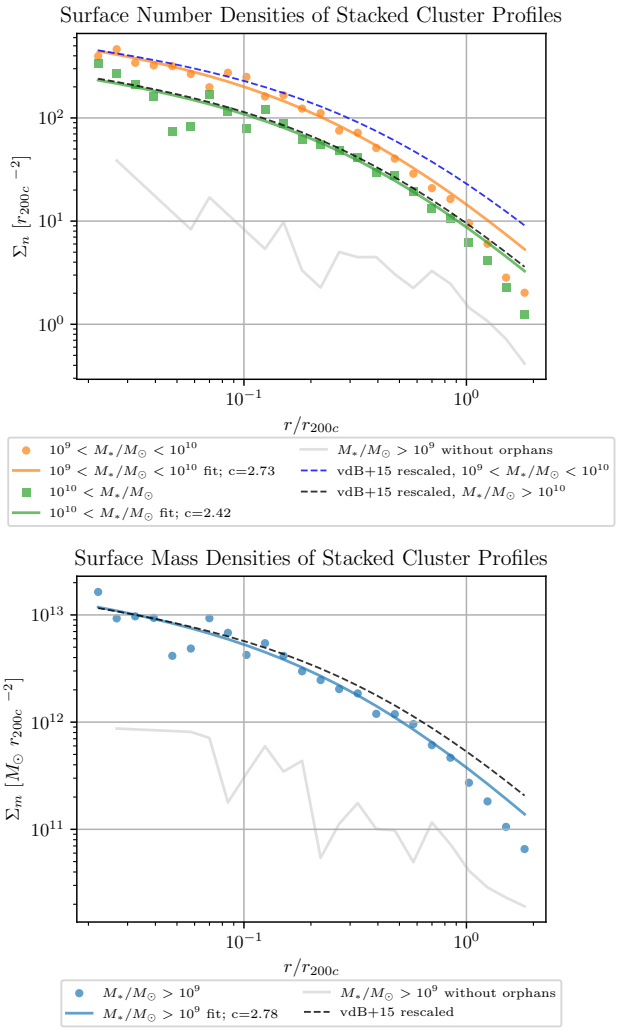

Figure 12. Upper panel: the light grey solid line shows the satellite number density profile averaged over the 10 largest haloes in our simulation *including only satellite galaxies with a detected parent clump with masses above* $10^9 M_\odot$. Satellites are then grouped in two mass bins indicated in the legend. The colored symbols show the average number density profile *including also orphan satellite galaxies*. The colored solid lines show the corresponding best fit projected NFW profiles, while the dashed lines show the best fit projected NFW profiles of the observed sample in van der Burg et al. (2015). Note that for the latter, we have renormalized the surface density to match the lower median mass of the simulated sample (see text for details). The lower panel shows the stellar mass surface density averaged over our simulated sample. Here again the light grey solid line shows the mass density profile *only for satellite galaxies within a detected clump* while the blue symbols shows the same quantity *also including orphan satellite galaxies*. The blue solid line corresponds to our best fit projected NFW profile while the dashed line shows the best fit NFW profiles of the observed sample in van der Burg et al. (2015). Here again this last profile has been renormalized to account for the difference in median mass between the simulated and the observed cluster samples.

log radius in units of r_{200c}. We then average the 30 profiles (three projections for 10 clusters) to obtain the final average radial surface density profile. Each profile is then fitted to a projected NFW profile (Navarro et al. 1996) using a standard least square fitting procedure. We obtain in particular the concentration parameter that we can compare to the observed value.

The resulting profiles are shown in Figure 12, again including and excluding orphan galaxies. Orphan galaxies play here also a crucial role, as the profiles without orphans underestimate the true value by an order of magnitude. Following van der Burg et al. (2015), we adapt the same galaxy stellar mass M_* thresholds of $10^9 M_\odot < M_* < 10^{10} M_\odot$ and $M_* > 10^{10} M_\odot$ for the surface number density profile, and a stellar mass threshold of $M_* > 10^9 M_\odot$ for the surface mass density profile. It is worth stressing that this lower mass threshold correspond exactly to the mass resolution limit of our mock galaxy catalogue.

As already noted above, the median mass of the simulated and observed catalogues widely differ. In order to facilitate a meaningful comparison, we assume that the total stellar mass in satellites roughly scales with M_{200c} in the halo mass range of interest here. We then adopt the following simple scaling relation

$$\Sigma \propto \frac{M_{200c}}{R_{200c}^2} \propto M_{200c}^{1/3} \tag{14}$$

and rescale the observed average profile found by van der Burg et al. (2015) using this scaling relation and the ratio of the two median masses. We plot in Figure 12 the corresponding best fit projected NFW profiles, showing excellent agreement with our simulated results.

Interestingly, the observed surface density profiles have a slightly smaller concentration than the simulated ones. Although we find for the number densities of satellite galaxies above $10^{10} M_\odot$ a concentration $c = 2.4$, in strikingly good agreement with $c = 2.3$ found by van der Burg et al. (2015) for the same mass range, our results differ for the mass range $10^9 M_\odot < M_* < 10^{10} M_\odot$: we find $c = 2.7$ while van der Burg et al. (2015) found $c = 1.8$. The same mismatch is found using the mass density profile. we find $c = 2.8$, while van der Burg et al. (2015) found $c = 2.0$.

We believe that this mismatch in the concentration parameter is consistent with the difference we have in the median sample mass. Indeed, the theory predicts a larger concentration for smaller mass haloes, roughly in the amplitude observed here (Zhao et al. 2009). We therefore could in principle improve the agreement between our simulation results and the observations of van der Burg et al. (2015) by also rescaling the radial direction according to the theoretical expectations. We believe this is beyond the scope of this paper to try and fit exactly the data.

In addition, we believe there is much more to the story. van der Burg et al. (2015) found that larger mass satellite galaxies have a significantly larger concentration parameter than the low mass bin ($c = 2.3$ compared to $c = 1.8$). Moreover, they have also found a strong excess of satellite galaxies compared to the best fit NFW profile in the centre ($r < 0.1 R_{200c}$). These observations are consistent with the effect of dynamical friction bringing the more massive galaxies faster to the central regions of the cluster. Dynamical friction is expected to be sufficiently efficient for the most massive subhaloes, with a mass larger than a few percent of that of the host halos (e.g. Binney & Tremaine 2008; Mo et al. 2011). In our case, this translates into subhaloes more massive than a few $10^{12} M_\odot$ and satellite galaxies more massive than a few $10^{10} M_\odot$. Our simulation clearly suffers from numerical overmerging in this mass range (van den Bosch et al. 2018), as highlighted by the importance of including orphan galaxies in our methodology. Moreover, our pure DMO parent simulation cannot follow precisely the many baryonic effects that are needed to predict accurately the individual trajectory of these higher mass satellite galaxies.

With these caveats in mind, we conclude that using our merger tree code and a state-of-the-art SHAM method, we can model reasonably well the cluster satellite galaxy number density and mass density profiles.

7 CONCLUSION

We presented `ACACIA`, a new algorithm to identify dark matter halo merger trees, which is designed to work *on-the-fly* on systems with distributed memory architectures, together with the adaptive mesh refinement code `RAMSES` with its *on-the-fly* clump finder `PHEW`. Clumps of dark matter are tracked across snapshots through a user-defined maximum number of most bound particles of the clump n_{mb}. We found that using $n_{mb} \simeq 200$ tracer particles is a safe choice to obtain robust results for our merger tree algorithm, while not being computationally unrealistically expensive. We also recommend adopting a conservative mass threshold of 200 particles per clump to get rid of a few rare spurious dead branches that would need to be pruned from the halo catalogue anyway.

Additionally, we examined the influence of various definitions of substructure properties on the resulting merger trees. Whether we define substructures to contain their respective substructures' masses or not had negligible effect on the merger trees. However defining particles to be strictly gravitationally bound to their parent substructure (by requiring that particles can't leave the spatial extent of that substructure) leads to better results, with much less extreme mass growths and extreme mass growth fluctuations of dark matter clumps. We recommend to use this strictly bound definition as the preferred definition for robust merger trees. The resulting merger trees are in agreement with the bottom-up hierarchical structure formation picture for dark matter haloes. The merger trees of massive haloes at $z = 0$ have more branches than their lower mass counterparts. Their formation history can often be traced to very high redshifts.

Once a progenitor clump is merged into a descendant, `ACACIA` keeps track of the progenitor's most strongly bound particle, called the "orphan particle". It is possible for a temporarily merged sub-halo to re-emerge from its host halo at a later snapshot because it hasn't actually dissolved or merged completely, but only because it wasn't detected by the clump finder as a separate density peak. Such a situation is illustrated in Figure 4. In these cases, orphan particles are used to establish a link between progenitor and descendant clumps across non-adjacent snapshots. By default, `ACACIA` will track orphans until the end of the simulation, and orphans are only removed after they have indeed established

a link between a progenitor and descendant and thus have served their purpose. Nonetheless, the current implementation offers the option to remove orphan particles after a user defined number of snapshots has passed. Keeping track of orphan particles indefinitely might lead to misidentifications of progenitor-descendant pairs and therefore to wrong formation histories. Our analysis shows however that matches between progenitor-descendant pairs over an interval greater than 10 snapshots are quite rare, so we expect this type of misidentifications to be a negligible issue.

Additionally, orphan particles also serve a second purpose. If we also want to produce a mock galaxy catalogue on-the-fly using a dark matter only simulation, the orphan particles are also used to track orphan *galaxies*. Those are galaxies that don't have an associated dark matter clump any longer because of numerical overmerging. If we interpret orphan particles as orphan galaxies, there could be additional reasons to consider stopping tracking them. For example, the effects of dynamical friction makes them fall towards the central galaxies. Once the orphan galaxies have lost enough energy, they may find themselves in close proximity to the central galaxies, even below the resolution limit. In these cases, it makes little sense to keep track of these orphans as individual galaxies. They should rather be regarded as merged into the central galaxy, and for that reason removed from the list of tracked orphans. Given that the model we employ doesn't provide us with the galaxy radii, this approach requires some form of galaxy-galaxy merging cross-sections to compute the probability of a collision between galaxies that will result in a galaxy merger. A different approach that other models use is to estimate the time for orphan galaxies to merge into the parent structure. This estimate could be e.g. the dynamical friction time (as is done in Moster et al. (2013)), or the fitting formula for the merger timescale of galaxies in cold dark matter models by Jiang et al. (2008). These physically motivated approaches to remove orphans will be the subject of future work.

Finally, as a proof of concept and using the known formation history of dark matter clumps from the merger trees and a widely adopted stellar-mass-to-halo-mass relation (Behroozi et al. 2013c), we generate a mock galaxy catalogue from a dark matter only simulation. The influence of the merger trees on the quality of the galaxy catalogues is twofold. First, while the stellar-mass-to-halo-mass relation can be directly applied to central galaxies associated to main haloes, using the peak clump mass for sub-haloes is a better approach for satellite galaxies. The reason is that tidal stripping of galaxies inside a dark matter halo sets in much later than for their host sub-halo (Nagai & Kravtsov 2005). Second, without properly keeping track of all merging events, no orphan galaxies can be traced, nor can their stellar mass be estimated through the stellar-mass-to-halo-mass relation unless it's known from which halo the orphan galaxy originated from, and what properties this halo had in the past.

To highlight the impact of the merger trees, we compute observables from our mock galaxy catalogue, both including and excluding orphan galaxies. Specifically, we compute the stellar mass two-point correlation functions and radial profiles of projected number densities and projected stellar mass densities in galaxy clusters. When orphan galaxies are included in the analysis, we obtain correlation functions and radial profiles in good agreement with observations, validating the different steps in our overall methodology.

The `RAMSES` code is publicly available and can be downloaded from `https://bitbucket.org/rteyssie/ramses/`. Instructions on how to use `ACACIA` and `PHEW` during a simulation can be found under `https://bitbucket.org/rteyssie/ramses/wiki/Content`.

DATA AVAILABILITY

The data underlying this article will be shared on reasonable request to the corresponding author.

ACKNOWLEDGEMENTS

MI would like to thank B. Roukema for helpful suggestions concerning the history of the application of merger trees in the astrophysical context, S. Avila for discussions on details of the Sussing Merger Tree Comparison project, and Y. Revaz for his support in many ways. This work was supported by the Swiss National Supercomputing Center (CSCS) under projects s1006 and uzh5 and by the Swiss National Science Foundation (SNF) under project 72535 "Multi-scale multi-physics models of galaxy formation". This work made use of the NUMPY (Harris et al. 2020) and SCIPY (Virtanen et al. 2020) python libraries for the data analysis and the MATPLOTLIB (Hunter 2007) python library for plotting tools.

REFERENCES

Avila S., et al., 2014, MNRAS, 441, 3488
Behroozi P. S., Wechsler R. H., Wu H.-Y., 2013a, The Astrophysical Journal, 762, 109
Behroozi P. S., Wechsler R. H., Wu H.-Y., Busha M. T., Klypin A. A., Primack J. R., 2013b, ApJ, 763, 18
Behroozi P. S., Wechsler R. H., Conroy C., 2013c, ApJ, 770, 57
Benson A. J., Cole S., Frenk C. S., Baugh C. M., Lacey C. G., 2000, Monthly Notices of the Royal Astronomical Society, 311, 793
Berlind A. A., Weinberg D. H., 2002, ApJ, 575, 587
Binney J., Tremaine S., 2008, Galactic Dynamics: Second Edition
Bleuler A., Teyssier R., Carassou S., Martizzi D., 2015, Computational Astrophysics and Cosmology, 2, 1
Bower R. G., Benson A. J., Malbon R., Helly J. C., Frenk C. S., Baugh C. M., Cole S., Lacey C. G., 2006, MNRAS, 370, 645
Bullock J. S., Wechsler R. H., Somerville R. S., 2002, Monthly Notices of the Royal Astronomical Society, 329, 246
Campbell D., van den Bosch F. C., Padmanabhan N., Mao Y.-Y., Zentner A. R., Lange J. U., Jiang F., Villarreal A., 2018, MNRAS, 477, 359
Chaves-Montero J., Angulo R. E., Schaye J., Schaller M., Crain R. A., Furlong M., Theuns T., 2016, Monthly Notices of the Royal Astronomical Society, 460, 3100
Conroy C., Wechsler R. H., Kravtsov A. V., 2006, The Astrophysical Journal, 647, 201
Croton D. J., et al., 2006, Monthly Notices of the Royal Astronomical Society, 365, 11
Davis M., Efstathiou G., Frenk C. S., White S. D. M., 1985, ApJ, 292, 371
Dubois Y., et al., 2014, Monthly Notices of the Royal Astronomical Society, 444, 1453

Elahi P. J., Thacker R. J., Widrow L. M., 2011, Mon. Not. R. Astron. Soc., 418, 320
Frigo M., Johnson S. G., 2005, Proceedings of the IEEE, 93, 216
Guo Q., White S., Li C., Boylan-Kolchin M., 2010, Monthly Notices of the Royal Astronomical Society, 404, 1111
Guo Q., et al., 2011, Monthly Notices of the Royal Astronomical Society, 413, 101
Hahn O., Abel T., 2011, MNRAS, 415, 2101
Harris C. R., et al., 2020, Nature, 585, 357
Hunter J. D., 2007, Computing in Science Engineering, 9, 90
Jiang C. Y., Jing Y. P., Faltenbacher A., Lin W. P., Li C., 2008, ApJ, 675, 1095
Jiang L., Helly J. C., Cole S., Frenk C. S., 2014, MNRAS, 440, 2115
Jung I., Lee J., Yi S. K., 2014, ApJ, 794, 74
Kang X., Jing Y. P., Mo H. J., Boerner G., 2005, Astrophys. J., 631, 21
Kauffmann G., White S. D. M., Guiderdoni B., 1993, MNRAS, 264, 201
Khandai N., Di Matteo T., Croft R., Wilkins S., Feng Y., Tucker E., DeGraf C., Liu M.-S., 2015, Mon Not R Astron Soc, 450, 1349
Knebe A., Libeskind N. I., Knollmann S. R., Yepes G., Gottlöber S., Hoffman Y., 2010, Mon. Not. R. Astron. Soc.
Knebe A., et al., 2011, MNRAS, 415, 2293
Knollmann S. R., Knebe A., 2009, ApJ, 182, 608
Kravtsov A. V., Berlind A. A., Wechsler R. H., Klypin A. A., Gottlöber S., Allgood B., Primack J. R., 2004, ApJ, 609, 35
Lacey C., Cole S., 1993, Monthly Notices of the Royal Astronomical Society, 262, 627
Lee J., et al., 2014, Mon. Not. R. Astron. Soc., 445, 4197
Li C., White S. D. M., 2009, MNRAS, 398, 2177
Lu Y., Kereš D., Katz N., Mo H. J., Fardal M., Weinberg M. D., 2011, Mon. Not. R. Astron. Soc., pp no–no
Mo H., Van den Bosch F., White S., 2011, Galaxy formation and evolution. Cambridge University Press
Moster B. P., Somerville R. S., Maulbetsch C., van den Bosch F. C., Macciò A. V., Naab T., Oser L., 2010, ApJ, 710, 903
Moster B. P., Naab T., White S. D. M., 2013, MNRAS, 428, 3121
Nagai D., Kravtsov A. V., 2005, ApJ, 618, 557
Navarro J. F., Frenk C. S., White S. D. M., 1996, The Astrophysical Journal, 462, 563
Nuza S. E., et al., 2013, Monthly Notices of the Royal Astronomical Society, 432, 743
Onions J., et al., 2012, Monthly Notices of the Royal Astronomical Society, 423, 1200
Peacock J. A., Smith R. E., 2000, Monthly Notices of the Royal Astronomical Society, 318, 1144
Perret V., 2016, DICE: Disk Initial Conditions Environment, Astrophysics Source Code Library (ascl:1607.002)
Planck Collaboration et al., 2016, A&A, 594, A13
Potter D., Stadel J., Teyssier R., 2017, Computational Astrophysics and Cosmology, 4, 2
Press W. H., Schechter P., 1974, ApJ, 187, 425
Reddick R. M., Wechsler R. H., Tinker J. L., Behroozi P. S., 2013, The Astrophysical Journal, 771, 30
Rodriguez-Gomez V., et al., 2015, Mon. Not. R. Astron. Soc., 449, 49
Roukema B. F., Yoshii Y., 1993, The Astrophysical Journal Letters, 418, L1
Roukema B. F., Quinn P. J., Peterson B. A., 1993, in Observational Cosmology. p. 51
Schaye J., et al., 2015, Monthly Notices of the Royal Astronomical Society, 446, 521
Scoccimarro R., Sheth R. K., Hui L., Jain B., 2001, The Astrophysical Journal, 546, 20
Seljak U., 2000, MNRAS, 318, 203
Shankar F., Lapi A., Salucci P., De Zotti G., Danese L., 2006, The Astrophysical Journal, 643, 14
Somerville R. S., Primack J. R., 1999, MNRAS, 310, 1087
Somerville R. S., Hopkins P. F., Cox T. J., Robertson B. E., Hernquist L., 2008, Monthly Notices of the Royal Astronomical Society, 391, 481
Springel V., White S. D. M., Tormen G., Kauffmann G., 2001, MNRAS, 328, 726
Springel V., et al., 2005, Nature, 435, 629
Springel V., Pakmor R., Zier O., Reinecke M., 2020, arXiv:2010.03567 [astro-ph]
Srisawat C., et al., 2013, MNRAS, 436, 150
Stadel J. G., 2001, PhD thesis, University of Washington
Teyssier, R. 2002, A&A, 385, 337
Trujillo-Gomez S., Klypin A., Primack J., Romanowsky A. J., 2011, The Astrophysical Journal, 742, 16
Tweed D., Devriendt J., Blaizot J., Colombi S., Slyz A., 2009, Astronomy and Astrophysics, 506, 647
Vale A., Ostriker J. P., 2004a, Monthly Notices of the Royal Astronomical Society, 353, 189
Vale A., Ostriker J. P., 2004b, MNRAS, 353, 189
Vale A., Ostriker J. P., 2006b, Monthly Notices of the Royal Astronomical Society, 371, 1173
Vale A., Ostriker J. P., 2006a, MNRAS, 371, 1173
Van Den Bosch F. C., Yang X., Mo H. J., 2003, Mon Not R Astron Soc, 340, 771
Virtanen P., et al., 2020, Nature Methods, 17, 261
Vogelsberger M., et al., 2014, Nature, 509, 177
Wang Y., et al., 2016, MNRAS, 459, 1554
Wechsler R. H., Somerville R. S., Bullock J. S., Kolatt T. S., Primack J. R., Blumenthal G. R., Dekel A., 2001, The Astrophysical Journal, 554, 85
Wetzel A. R., White M., 2010, Monthly Notices of the Royal Astronomical Society, 403, 1072
White S. D. M., Frenk C. S., 1991, ApJ, 379, 52
Yamamoto M., Masaki S., Hikage C., 2015, arXiv:1503.03973 [astro-ph]
Yang X., Mo H. J., van den Bosch F. C., 2003, Monthly Notices of the Royal Astronomical Society, 339, 1057
Yang X., Mo H. J., van den Bosch F. C., 2009, ApJ, 693, 830
Yang X., Mo H. J., van den Bosch F. C., Zhang Y., Han J., 2012, The Astrophysical Journal, 752, 41
Zehavi I., et al., 2004, ApJ, 608, 16
Zentner A. R., Hearin A. P., van den Bosch F. C., 2014, Monthly Notices of the Royal Astronomical Society, 443, 3044
Zhao D. H., Jing Y. P., Mo H. J., Boerner G., 2009, The Astrophysical Journal, 707, 354
van den Bosch F. C., Ogiya G., Hahn O., Burkert A., 2018, Monthly Notices of the Royal Astronomical Society, 474, 3043
van der Burg R. F. J., Hoekstra H., Muzzin A., Sifón C., Balogh M. L., McGee S. L., 2015, Astronomy & Astrophysics, 577, A19

APPENDIX A: DETAILED DESCRIPTION OF THE MERGER TREE ALGORITHM

The merger tree code starts once clumps have been identified and the particle unbinding is completed. The algorithm is designed to work in parallel on distributed memory architectures and makes use of the MPI standard. When `RAMSES` is executed, each processing unit involved in the execution is typically referred to by a unique index, called the "MPI rank", and given a unique domain, i.e. part of the volume of the simulated space, to work on. In what follows, we assume that the code is being executed in parallel over multiple MPI ranks. The code then proceeds as follows:

(i) Write the current clump data to file which is to be read in as progenitor data in the subsequent snapshot: For every clump, identify up to n_{mb} tracer particles with minimal binding energy. If a clump consists of less than n_{mb} particles, then take the maximally available number of particles. Then write the tracer particles of all ranks into a single shared file. This file will be read in by every rank in once the next snapshot is being processed. If past merged progenitors exist, each rank writes these to (a different) shared file for later use.

(ii) Every rank reads in the progenitor data from the shared file of the previous snapshot.

(iii) The read in progenitor data is processed: First we find which progenitor tracer particles are on each rank's domain by comparing the tracer particles' unique ID to the IDs of the particles currently in this rank's domain. Each rank needs to know which tracer particles are currently on its own domain. Then we find and communicate globally which rank is the "owner" of which progenitor (and past merged progenitor): The owner of any progenitor is defined as the rank which has the most strongly bound particle of that progenitor within its domain. This particle is referred to as the "galaxy particle" of this progenitor. Each rank henceforth only keeps track of the tracer particles that are on its domain. The rest are removed from memory.

(iv) Find links between progenitors and descendants, that is find "which tracer particle ended up where":

After clump finding, the clump to which any particle belongs is known, and after reading in progenitor data, the progenitor clump to which any tracer particle belonged to is known. Each rank now loops through all its local tracer particles. Using these two informations (in which clump the particle was and in which clump the particle is now) for every tracer particle, all descendant candidates for all progenitors are found and stored in a sparse matrix, where the rows of the matrix correspond to progenitors and the columns are the descendants. The exact number of particle matches between a progenitor-descendant candidate pair is kept.

With the sparse matrices populated, they are now communicated across ranks where they are needed. First every rank that has data on progenitors that it doesn't own itself sends this data (specifically, the sparse matrix data) to the owner of that progenitor. The owners then gather and sum up all the matches found in the previous linking step for the progenitors that they own and then send them back to any rank that has at least one particle of that progenitor on their domain. (These are the same ranks that sent data to the owner of the progenitor in the first place.)

After communications are done, a transverse sparse matrix is created, where the rows are descendants and the columns are progenitors. These matrices will be used to loop through progenitor or descendant candidates.

(v) Make trees: We first obtain an initial guess for the main progenitors of every descendant and for the main descendant of every progenitor by finding the candidate that maximises the merit function given by Equation (6).

Then we loop to establish matches:

A main progenitor-descendant pair is established when the main progenitor of a descendant is the main descendant of said progenitor, or in pseudocode:

```
match = (main_prog(idesc)==iprog) &&
        (main_desc(iprog)==idesc)
```

While there are still descendants without a match and still progenitor candidates left for these descendants:

- For progenitors without a match: Loop through all descendant candidates. If you find a match, stop there and mark this descendant candidate as the main descendant for this progenitor.
- Then for all descendants still without a match: Switch to the next best progenitor candidate as current best guess.

The loop ends either when all descendants have a match, or if descendants run out of candidates. If a progenitor hasn't found a match at this point, we assume that it merged into its best descendant candidate, i.e. the one that maximises the merit function. The merged progenitors are added to the list of past merged progenitors by adding their "galaxy particle" to the list.

If there are descendants that still have no main progenitor, we now try finding a progenitor from an older, non-consecutive snapshot. Past merged progenitors are tracked by one particle, their former "galaxy particle", which we now refer to as the "orphan particle". All particles of the descendant under investigation are checked for being an orphan particle of a past merged progenitor. The most strongly bound orphan particle will be considered the main progenitor of the descendant under consideration. If a match is found, the past merged progenitor is removed from the list of past merged progenitors.

Finally, descendants that still haven't found a progenitor at this point are deemed to be newly formed.

(vi) The results are written to file.

Table B1. Comparison of simulation and evaluation parameters used in this work and of A14, where the parameters of the latter have been converted using $h = 0.704$. m_m is the mass threshold for main haloes, m_s is the mass threshold for sub-haloes.

	This work	A14
particle mass $[10^9 M_\odot]$	1.55	1.32
particles used	256^3	270^3
box size [Mpc]	88.6	88.8
snapshots until $z = 0$	67	62
m_m $[10^{12} M_\odot]$	1.51	1.12 - 1.37
m_s $[10^{11} M_\odot]$	3.91	4.26 - 9.72

APPENDIX B: TREE STATISTICS USING Avila et al. (2014) SELECTION CRITERIA

In this section, the merger tree statistics introduced in Section 5 when following the selection criteria that are used in Avila et al. (2014) (A14 from here on) are presented. Ideally, `ACACIA` should be tested on the same datasets and halo catalogues used in the Comparison Project to enable a direct comparison to the performance of other merger tree codes. However, since `ACACIA` was designed to work on the fly, using it as a post-processing utility would defeat its purpose. Furthermore, `ACACIA` is not necessarily compatible with other definitions of haloes and sub-haloes. But most importantly, we also want to demonstrate that the halo finder `PHEW` can

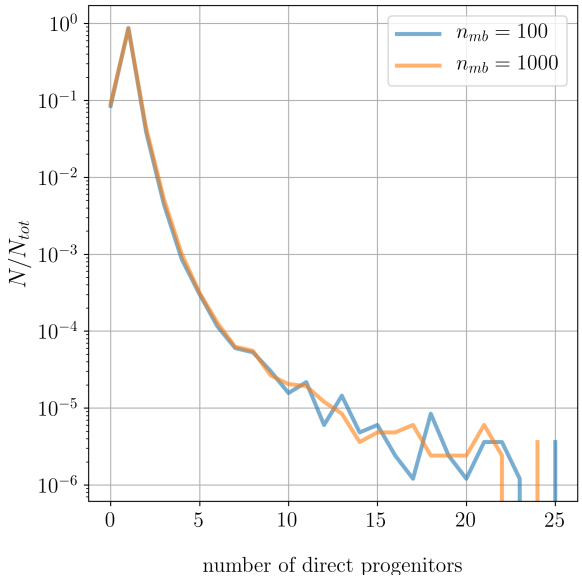

Figure B1. Histogram of the number of direct progenitors for all clumps from $z = 0$ to $z = 2$ for $n_{mb} = 100$ and 1000 tracer particles per clump. The histogram is normalized by the total number of events found.

be used to produce reliable merger trees. So instead, the tests are performed on our own datasets and halo catalogues, which are described in section 5.1. A comparison of the used parameters of our simulations and the ones used in A14 is given in Table B1. In the following, the results for $n_{mb} = 100$ and $n_{mb} = 1000$ are shown. Like before, when the influence of the number of tracer particles was investigated, the `strictly bound` parameter and the `exclusive` mass definition were used.

The difference to the results presented in Section 5 is that the mass thresholds are set such that only the 1000 most massive main haloes and only the 200 most massive sub-haloes at $z = 0$ are included. This gives effective mass threshold $m_m = 1.51 \times 10^{12} M_\odot$ and $m_s = 3.9 \times 10^{11} M_\odot$, which are on one hand comparable to the mass thresholds applied in A14 (Table B1), but already show differences in the resulting halo catalogue. `PHEW` finds a higher mass threshold for main haloes, but a lower mass threshold for sub-haloes. This is consistent with the fact that the `strictly bound` parameter was used: Unbound particles are passed on to substructure that is higher up in the hierarchy, and the unbinding is repeated until the top level, which are the main haloes, is reached. The more strict unbinding criterion tends to assign more particles to the main haloes and remove them from sub-haloes, which is reflected in the mass thresholds. Indeed, using the `loosely bound` parameter instead leads to $m_m = 1.39 \times 10^{12} M_\odot$ and $m_s = 2.00 \times 10^{12} M_\odot$.

The length of the main branches for haloes and sub-haloes individually are shown in Figure B2. For the halo population, two noticeable differences compared to Figure 3 of A14 appear:

(i) `ACACIA` finds some main haloes with short (< 10) main branches. In A14, his only happens for the `JMerge` tree maker, `TreeMaker` for `Rockstar` haloes, and `SubLink` for `AHF` haloes.

(ii) Most clumps satisfying the mass thresholds have very

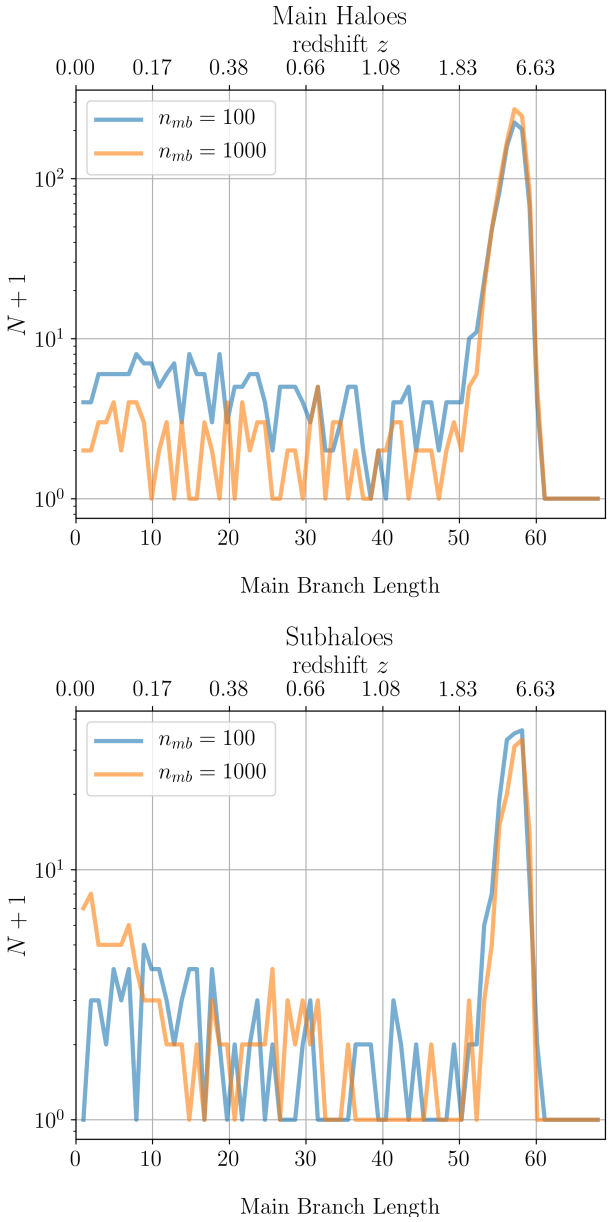

Figure B2. Histograms of the length of the main branch. The left plot shows the length of the 1000 most massive haloes at $z = 0$, the right plot shows the length of the 200 most massive sub-haloes for $n_{mb} = 100$ and 1000 tracer particles per clump.

large main branch lengths that are in a narrow range (~ 10) of snapshots, while the high main branch length distribution found in A14 is much wider (> 20).

This indicates that on one hand, `ACACIA` probably makes more misidentifications, hence the short main branches, but simultaneously is able to track clumps to higher redshifts.

In Figure B1 the number of direct progenitors for all clumps between $z = 0$ and $z = 2$ are shown. Comparing to Figure 5 of A14, `ACACIA` gives very comparable results: $\sim 10^{-1}$ haloes have no direct progenitor, almost all have one, and the distribution follows an exponential decay with the maximal number of direct progenitors lying around 20-25. Many tree makers and halo finders in A14 exhibit the

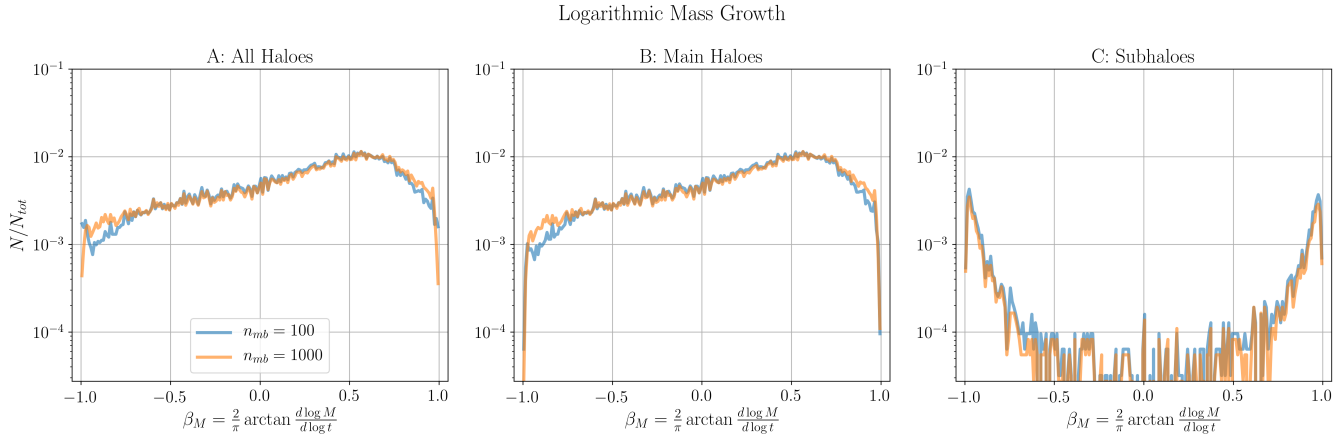

Figure B3. Logarithmic mass growth for haloes and sub-haloes satisfying the mass thresholds. Group A contains clumps that are either haloes or sub-haloes in consecutive snapshots k and $k+1$ with masses $m \geq m_m$. Group B contains clumps that are only haloes in two consecutive snapshots with mass above m_m, group C contains only clumps that were sub-haloes in two consecutive snapshots with mass greater than m_s. The histogram is normalized by the total number of events found for group A.

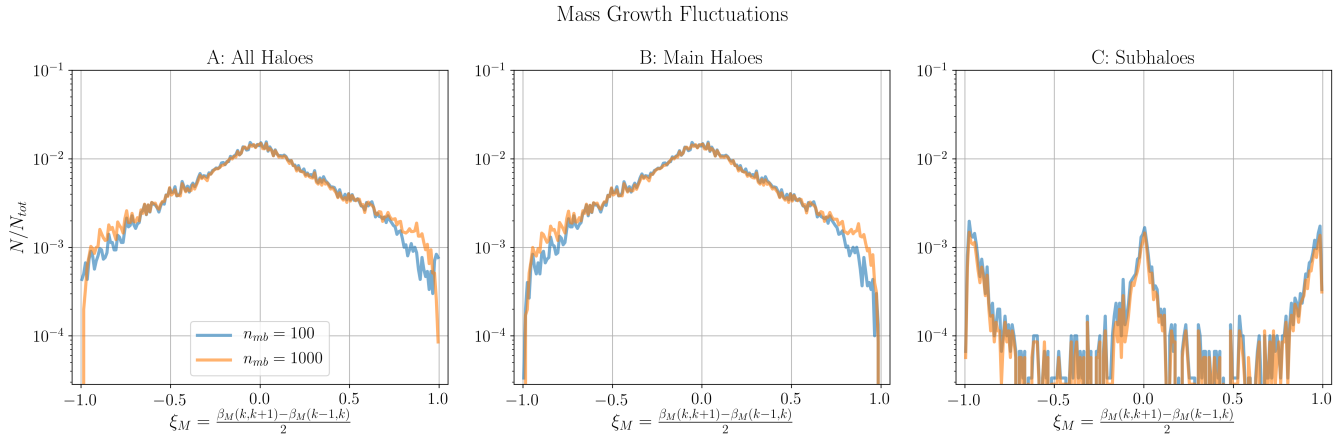

Figure B4. Histogram of mass growth fluctuations for haloes and sub-haloes satisfying the mass thresholds. Group A contains clumps that are either haloes or sub-haloes in three consecutive snapshots with masses $m \geq m_m$. Group B contains clumps that are only haloes in three consecutive snapshots with mass above m_m, group C contains only clumps that were sub-haloes in three consecutive snapshots with mass greater than m_s. The histogram is normalized by the total number of events found for group A.

same kind of behaviour, particularly so for the `AHF`, `Subfind`, and `Rockstar all` halo finders in Figure 5 of A14.

For the logarithmic mass growth (Figure B3) and the mass growth fluctuations (Figure B4), the statistics are separated into three groups. Group A contains clumps that are either haloes or sub-haloes in *consecutive* snapshots k and $k+1$ with masses $m \geq m_m$. Group B contains clumps that are exclusively main haloes in two consecutive snapshots with mass above m_m, group C contains only clumps that were sub-haloes in two consecutive snapshots with mass greater than m_s. Except for the branching ratio statistic, we follow clumps of the $z = 0$ snapshot along the main branch only.

The logarithmic mass growth resulting from `ACACIA` follows the general trend that the tree makers in A14 exhibit too. The growth for groups A and B increases steadily and peaks around $\beta_M \sim 0.5$, where the peak is $\sim 10^{-2}$. For $n_{mb} = 100$, the extreme mass loss with $\beta_M = -1$ increases for group A, which is an undesirable property, but is also exhibited by `Sublink` in A14. For $n_{mb} = 1000$, it drops below 10^{-3} (10^{-4} for group B), which is comparable behaviour to `MergerTree`, `Sublink`, and `VELOCIraptor`, particularly so in combination with `AHF` and `Subfind` halo finders. Group C, containing only mass growths of clumps that have been sub-haloes in two consecutive snapshots, shows a distribution peaking around extreme mass growths $\beta_M \to \pm 1$ at $\sim 5 \times 10^{-3}$, which can again be seen in A14 for almost all tree makers, albeit not for all halo finders. More noticeably, almost no sub-haloes are found with $-0.5 < \beta_M < 0.5$ with `PHEW` and `ACACIA` in Figure B3. This is most likely due to the fact that once a halo is merged into another, it quickly looses its outer mass due to to the strict unbinding method used here.

The mass growth fluctuations (Figure B4) of `ACACIA` share the general trend with the ones from Figure 8 in A14, in that they peak around $\xi_M = 0$ and decrease outwards towards $\xi_M = \pm 1$. In A14, in all cases groups A and B peak just below 10^{-1}, while our results peak just above 10^{-2}.

However, similarly to the results of e.g. `Sublink`, `TreeMaker`, and `VELOCIraptor` with the `AHF` or `Subfind` halo finders, the distribution around $\xi_M \sim \pm 0.5$ drops to $\sim 5 \times 10^{-3}$, and then continues dropping below $10^{-4} - 10^{-3}$ at $\xi_M \sim \pm 1$. For $n_{mb} = 100$, the group A distribution rises again for $\xi_M \sim 1$, as it does for `TreeMaker` and `Sublink` tree makers with `AHF`. Group B shows a steeper drop around the extreme values $\xi_M \sim \pm 1$ compared to group A, dropping below 10^{-4} at these values, similarly to the behaviour of many tree makers and halo finders in A14. The sub-halo group C of this work shows three main peaks, around -1, 0, and 1. These peaks also appear in the A14 results. However, the peaks at the extreme values in A14 are lower than the ones of this work, while the peaks around 0 is higher. The missing values around $\xi_M \sim \pm 0.5$ that were also seen in the mass growth in Figure B3 remain unsurprisingly. Similar distributions are obtained by `Sublink` and `VELOCIraptor` in combination with the `AHF` halo finder.

In summary:

(i) `ACACIA` finds more massive haloes with short main branch lengths, but the main branch length distribution peaks at very high numbers and is narrower

(ii) The distribution of the number of direct progenitors for all haloes within $0 < z < 2$ is consistent with the results from A14

(iii) The logarithmic mass growth is similar to what is obtained with some other halo finders and tree makers, in particular the `Sublink` and `VELOCIraptor` tree makers in combination with the `AHF` and `Subfind` halo finders.

(iv) Apart of the higher peak value of the distribution in A14, the mass growth fluctuations are similar to the results of the `Sublink` and `VELOCIraptor` tree makers in combination with the `AHF` halo finder.

Hence we conclude that our tree maker gives comparable results with respect to other state of the art merger tree and halo finding codes.

This paper has been typeset from a TEX/LATEX file prepared by the author.