LMS J. Comput. Math. 15 (2012) 374–384 © 2012 Author
doi:10.1112/S1461157012001118

Matching simple modules of condensation algebras

Felix Noeske

Abstract

We revise the matching algorithm of Noeske (*LMS J. Comput. Math.* 11 (2008) 213–222) and introduce a new approach via composition series to expedite the calculations. Furthermore, we show how the matching algorithm may be applied in the more general and frequently occurring setting that we are only given subalgebras of the condensed algebras which each contain the separable algebra of one of their Wedderburn–Malcev decompositions.

1. *Introduction*

In [**11**], we aimed to provide an algorithmic solution to the following problem, called the 'matching problem': let A be a finite-dimensional algebra over a finite field F. Condensing an A-module V with two different idempotents e and e', leads to the problem that to compare the composition series of Ve and Ve', we need to match the composition factors of both modules. In other words, if S is a composition factor of V with $Se \neq 0$, then just given the composition factor Se of Ve, we have to find the composition factor of Ve' isomorphic with Se', or prove that $Se' = 0$.

The major obstacle of this task is that we usually cannot compute S explicitly due to constraints on time and computational resources. In fact the whole idea of condensation (see [**12**] for an introduction) is to forego computationally intractable calculations for an A-module V by analysing the *condensed* module Ve which is for suitably chosen e a much smaller-dimensional eAe-module, where eAe is the *condensed* algebra.

The treatment of the matching problem outlined in [**11**] adheres to this fundamental paradigm of condensation, but has two shortcomings: for one, while any match produced by the method of the paper cited is a valid match, it is in general not as easy as stated in [**11**] to prove that a match does not exist. To be precise, in [**11**, Lemma 2.2] the 'if' implication is wrong, as the following example illustrates.

Example 1.1. Let $A = F^{2\times 2}$ be the full matrix ring over a field F, and let e be the idempotent $\left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right]$. Then $v := [0, 1]$ generates the simple A-module $S := F^2$, and we have $Se \neq 0$, even though $0 = veA \neq vA$.

Second, the matching algorithm [**11**, Algorithm 1] is difficult to apply to condensed algebras owing to the generation problem (see, for example, [**10**]): to invoke it as stated in [**11**], we need to have a computationally tractable generating set for the condensed algebras available. In practice, we usually work with just a subalgebra $\mathcal{C}$ of eAe generated by a few elements of eAe. To distinguish $\mathcal{C}$ from eAe, we call it the *condensation* algebra, as all computations are done with respect to this algebra, that is we usually consider the restricted module $Ve{\downarrow_{\mathcal{C}}}$ in lieu of the eAe-module Ve.

The main ingredient to tackle the matching problem successfully are *local* submodules of V. Recall that a module is called *local* if it possesses a unique maximal submodule. If the simple head of a local submodule is isomorphic to S, then we say that the module is S-*local*. The

Received 23 February 2012; revised 13 July 2012.
2010 Mathematics Subject Classification 20C40 (primary).

relevance of local submodules of V to determine structural properties of V has first been demonstrated in [**5**], in which certain elements of A are introduced, so-called *peakwords* (see [**5**, § 3]), which serve to provide generators for local submodules. For all further details on peakwords and their properties, we refer the reader to [**5**].

Note that peakwords are defined with respect to a faithful A-module. But since any A-module V is a faithful module for the homomorphic image of A under the representation it affords, this is not an obstacle: in practice the algebra A is realised as a matrix algebra giving its representation on V, so fixing V we only consider the image of A.

The paper is structured as follows: in § 2, we correct [**11**, Lemma 2.2] to Lemma 2.2. Furthermore, we generalise [**11**, Lemmas 2.1 and 2.3], and revise the results of [**11**] to encompass pathological examples such as Example 1.1. In the process, we revisit the original matching algorithm and improve its runtime considerably by making use of prior computed composition series of Ve, respectively Ve'. With their help, we can replace the costly MeatAxe (see [**3**, Section 7.4]) calculations to construct matrix representations of local submodules by a straightforward membership test of a vector in a subspace. In § 3 we provide a generalisation of the matching algorithm developed in § 2. The algorithm for condensation algebras (Algorithm 4) assumes that these algebras are sufficiently large subalgebras of the condensed algebras such that all simple modules of the latter restrict irreducibly to the former, and no two non-isomorphic simple modules are isomorphic after restriction. In other words, the condensation algebras have the 'same' simple modules as their enveloping condensed algebras, but we allow them to have less glue. Such condensation algebras are a common sight in the Modular Atlas project [**9**], so that Algorithm 4 has immediate applications (see, for example, [**2**]). Section 4 is devoted to illustrating how the matching algorithms proposed in this paper fare when applied to practical examples. We consider tensor products of simple modules for the sporadic simple Harada–Norton group, and discuss how our improvements shorten the expected running times in comparison with the original matching algorithm of [**11**].

Notation-wise, throughout the paper A is a finite-dimensional algebra over a finite field, V is a finite-dimensional, faithful A-module, S is a simple A-module and $e \in A$ an idempotent with $Se \neq 0$. Furthermore, $e' \in A$ is another idempotent.

2. *Match making*

This section mimics [**11**, § 2], but generalises the older results. In particular, the new Lemma 2.2 incorporates Example 1.1 into our matching framework. We formulate an improved matching algorithm with Algorithm 3 to match Se to a composition factor of Ve', and detail the steps necessary to overcome the problems posed by examples such as Example 1.1. As was stated in [**11**] and we illustrate in § 4, in practical cases the runtime of the old algorithm is dominated by the MeatAxe computations involving the construction of an Se'-local submodule of Ve' and the subsequent calculations needed to determine the isomorphism type of its head. Here we are able to provide a new approach which precludes these costly computations.

LEMMA 2.1. *Let $f \in eAe$ be an idempotent. Then f is an S-primitive idempotent in A if and only if it is an Se-primitive idempotent in eAe. Furthermore, if $w \in Ve$ such that it generates an Se-local eAe-submodule of Ve, then w also generates an S-local A-submodule of V.*

Proof. The first claim is immediate: as $f \in eAe$ we have $ef = fe = f$, so fAf is local if and only if $feAef$ is local, and since $Sf = Sef$ the idempotent f is S-primitive in A if and only if it is Se-primitive in eAe.

If $weAe$ is Se-local there is an Se-primitive idempotent $f \in eAe$ such that $weAe = veAe$ for some $v \in Vef = Vf$. As $we = w$ this implies $w \in vA$. By the first claim, vA is an S-local submodule of V. Suppose that $w \in \operatorname{rad}(vA)$. Then wAe is a proper submodule of vAe, as $Se \neq 0$

and condensation is exact. But we have $wAe = weAe = veAe = vAe$, as $v \in Ve$. Therefore, $wA = vA$ and wA is S-local. □

LEMMA 2.2. *Let $v \in V$ such that vA is S-local, and $e' \in A$ an idempotent. Then there is a $w \in vA$ such that $we'A = vA$ if and only if $Se' \neq 0$.*

Proof. The A-module generated by $v + \operatorname{rad} vA$ is isomorphic to S. Therefore it is immediate that if $we'A = vA$ then the A-module generated by $we' + \operatorname{rad} we'A = we' + \operatorname{rad} vA$ is also isomorphic to S, and hence $Se' \neq 0$. For the converse observe that if e' does not annihilate S, then $we' \notin \operatorname{rad} wA$ for some generator w of vA, and thus $we'A = vA$. □

LEMMA 2.3. *Let $e' \in A$ be an idempotent and $v \in Ve$ such that $veAe$ is an Se-local module. There is a $w \in veAe$ such that $we'eAe = veAe$, if and only if $See'e \neq 0$. If $See'e \neq 0$, then $Se' \neq 0$ and $we'Ae'$ is an Se'-local module.*

Proof. Let $we'eAe = veAe$. Then $weAe = veAe$, as by $w \in Ve$ we have $we'eAe = wee'eAe$, so that $veAe = we'eAe$ is contained in $weAe$ which by the hypothesis is a submodule of $veAe$. Consequently, $w \in weAe = veAe \subseteq vA$ and $v \in veAe = weAe \subseteq wA$, such that $wA = vA$, and vA is S-local by Lemma 2.1. In particular, we have $we' \in wA = vA$. Suppose that $we' \in \operatorname{rad}(vA)$. Then $we'A$ is a proper submodule of vA, and applying e we obtain the inequality $(vA)e \gneqq (we'A)e \geq we'eAe = veAe = vAe$, a contradiction. Hence, $we'A = vA$ is an S-local module. Now, the hypothesis $we'eAe = veAe$ also gives that $wee'e \notin \operatorname{rad}(veAe)$, that is $(w + \operatorname{rad}(veAe)) \cdot ee'e \neq 0$, so that $See'e \neq 0$. Obviously, this implies that e' does not annihilate S. So $we'Ae'$ is Se'-local, as claimed. For the converse observe that by $See'e \neq 0$ there is a $w \in veAe$ such that $wee'e = we'e \notin \operatorname{rad}(veAe)$. Hence, $we'eAe = veAe$, as claimed. □

In order to match a composition factor Se of Ve with Lemma 2.3 to a composition factor of Ve', we need to find a generator of an Se-local submodule L such that its image under $e'e$ still generates the Se-local L. As Example 1.1 shows, we may need to test a full basis for the simple head of L to this effect, that is if we are given a basis for the unique maximal submodule of L, and extend this basis to a basis of L, then all of the basis vectors not lying in $\operatorname{rad} L$ will need to be tested.

The method to compute such a complementary basis holds in a broader context, so that we assume $e = 1$ for the moment. The starting point is an ascending composition series

$$0 = U_0 \lneqq U_1 \lneqq \cdots \lneqq U_l = V, \tag{C}$$

and a basis $\mathcal{B}_C$ of V which is adapted to this composition series, that is $\mathcal{B}_C$ is obtained by successively extending a basis $\mathcal{B}_{i-1}$ of U_{i-1} to a basis of U_i for $i = 1, \ldots, l$. A peakword $\omega \in A$ gives rise to the Fitting decomposition $V = \ker_V(\omega^\infty) \oplus \operatorname{im}_V(\omega^\infty)$ (we denote by ω^∞ a suitably large power of ω such that its kernel is maximal) relative to the endomorphism of V induced by ω (see [5, § 3]). We define $\pi_{\ker_V(\omega^\infty)}$ to be the vector space projection of V onto the direct summand $\ker_V(\omega^\infty)$.

Now Algorithm 1 produces the set of generators sought: the sub-basis $\mathcal{B}_i$ of $\mathcal{B}_C$ chosen in line 1 spans a submodule of V which contains S exactly once in its head. Therefore, all v' projected in line 3 lie in the same one-dimensional subspace of $\ker_V(\omega^\infty)$, and a non-zero v' generates an S-local submodule of V. By our choice, $v'A \leq U_i$ and $\operatorname{rad}(v'A) \leq U_{i-1}$, such that a basis for the radical of $v'A$ is contained in $\mathcal{B}_{i-1}$. Line 5 is easily achieved using the MeatAxe by spinning v' modulo the basis $\mathcal{B}_{i-1}$ (see [4, Algorithm 2.1.19] or [3, p. 231]). As a basis of $\operatorname{rad}(v'A)$ is contained in $\mathcal{B}_{i-1}$, this way we obtain its extension to $v'A$, giving a full set of representatives of a basis for the factor $v'A/\operatorname{rad}(v'A)$ in the process.

Applying Algorithm 1 yields an additional benefit. By working with an ascending composition series of V, we have some control over the size of the S-local submodule whose

Algorithm 1 Computing generators of an S-local submodule

Input: The kernel $\ker_V(\omega^\infty)$ of an S-peakword ω, a basis $\mathcal{B}_C$ adapted to a composition series (C).

Output: A full set of representatives $\mathcal{B}_S$ of a basis of the radical factor of an S-local submodule of V.

```
1: Choose a member $U_i$ of (C) of minimal dimension such that $U_i/U_{i-1} \cong S$.
2: repeat
3:     Compute $v' := \pi_{\ker \omega^\infty}(v)$ for $v \in \mathcal{B}_i \setminus \mathcal{B}_{i-1}$
4: until $v' \neq 0$
5: Extend $\mathcal{B}_{i-1}$ to a basis $\mathcal{B}'$ of $v'A + U_{i-1}$.
6: return  $\mathcal{B}_S := \mathcal{B}' \setminus \mathcal{B}_{i-1}$
```

generators we determine: in choosing U_i minimal in line 1 we attempt to choose the S-local module contained therein as small as possible.

The problem of projecting onto $\ker_V(\omega^\infty)$ in line 3 of Algorithm 1 can also be solved with the MeatAxe, and the program `clean` in particular (see [4, Algorithm 2.1.13] or [3, p. 223]). Given a semi-echelonised basis of a subspace W of a vector space V and a vector $v \in V$, the program `clean` performs Gaussian elimination on v with respect to the pivots of the semi-echelonised basis. In particular, `clean` can test whether v lies in W, as in this case v will be reduced to the zero vector, and can then also express v in the given basis of W by returning the vector of coefficients used in the Gaussian elimination. Given $V = U \oplus W$ a decomposition into direct sums of vector spaces, we can project v onto U by repeated calls of `clean` as stated in Algorithm 2. The correctness of Algorithm 2 follows immediately from the fact that the coefficient vector $\kappa_{\mathcal{B}_U}(\pi_U(v))$ expressing $\pi_U(v)$ with respect to the basis $\mathcal{B}_U$ is identical to the coefficient vector $\kappa_{\mathcal{B}'_U}(v')$ expressing the cleaned v with respect to the cleaned basis $\mathcal{B}'_U$.

Algorithm 2 Projecting onto a direct summand

Input: Semi-echelonised bases $\mathcal{B}_U = \{u_1, \dots, u_d\}$ of U and $\mathcal{B}_W$ of W with $V = U \oplus W$, a vector $v \in V$.

Output: The projection of v onto U, i.e. $\pi_U(v)$.

```
1: Clean $\mathcal{B}_U$ with $\mathcal{B}_W$ to obtain $\mathcal{B}'_U$.
2: Clean $v$ with $\mathcal{B}_W$ to obtain $v'$.
3: Clean $v'$ with $\mathcal{B}'_U$ to obtain $\kappa = \kappa_{\mathcal{B}'_U}(v')$.
4: Return $\sum_{i=1}^{d} \kappa_i u_i$.
```

With Algorithm 3 we formulate a revised version of the original matching algorithm [11, Algorithm 1] which addresses the need to consider more than just one generator of an Se-local submodule, if we are trying to match Se. To prepare the input for Algorithm 3 a precomputation finds an Se-peakword $\omega \in eAe$ (see [5] for details on a peakword search) and semi-echelonised bases for the direct summands of the fitting decomposition $Ve = \ker_{Ve}(\omega^\infty) \oplus \operatorname{im}_{Ve}(\omega^\infty)$ it induces. Then a run of Algorithm 1 provides us with the input basis for our matching algorithm.

The steps of Algorithm 3 given in the lines 2–5 mirror the approach of [11, Algorithm 1]: we zig-zag between the two module categories of eAe and $e'Ae'$ by projecting a generator v of our implicitly chosen Se-local submodule first with e' onto Ve' and then with e back onto Ve. Once we obtain $ve'e \in veAe$ in line 5, we have to check whether $ve'e$ still generates $veAe$. Instead of determining the dimension of the eAe-module $ve'eAe$, as is suggested in [11, Algorithm 1],

Algorithm 3 Matching a composition factor

Input: A complementary basis $\mathcal{B}_{Se}$ of generators of an Se-local submodule of Ve output by Algorithm 1, a database `DB` of representatives of the isomorphism types of composition factors of Ve'.

Output: The Se' in `DB` which matches Se, or `fail` if the match fails.

```
1: for $v \in \mathcal{B}_{Se}$ do
2:     Embed $v$ into $V$.
3:     Project $v$ onto $Ve'$, obtaining $ve' \in Ve'$.
4:     Embed $ve'$ into $V$.
5:     Project $ve'$ onto $Ve$, giving $ve'e \in Ve$.
6:     if $ve'e \notin \mathrm{rad}(veAe)$ then
7:         Determine the module $Se'$ in DB isomorphic to the head of $ve'Ae'$.
8:         return $Se'$.
9:     end if
10: end for
11: return fail.
```

we can achieve the same effect more efficiently as follows: recall that in order to produce $\mathcal{B}_{Se}$ we have worked with a basis of Ve which is adapted to a composition series (C) found by the MeatAxe. As we have presented in Algorithm 1, and using the notation in place, we have $veAe$ contained in U_i and $\mathrm{rad}(veAe)$ contained in U_{i-1} for some $i \in \{1, \ldots, l\}$. Therefore, cleaning $ve'e$ with the sub-basis $\mathcal{B}_{i-1}$ answers this question. The vector $ve'e$ lies in the radical of $veAe$ if and only if the cleaned vector is zero.

The original algorithm next determined the head of $ve'Ae'$ as detailed in [8], and tested it for isomorphism with the entries of a database of isomorphism types of composition factors of Ve'. However, we can determine the head of $ve'Ae'$ with much less effort: consider more generally an A-module V with composition factor S, and the composition series (C) of V. Then given an S-local submodule L of V, there is a minimal $i \in \{1, \ldots, l\}$ such that $L \leq U_i$. As the head of L is isomorphic to S, we have $U_i/U_{i-1} \cong S$. So, taking a basis adapted to a composition series of Ve', allows us to determine the isomorphism type of the simple head of $ve'A'e'$ by working up along the ascending composition series of Ve' and cleaning ve' in each step. Once ve' reduces to zero, we can immediately derive the isomorphism type of the head of $ve'Ae'$ from our current position in the composition series.

If the algorithm has run through all vectors of the basis $\mathcal{B}_{Se}$, and we were unable to find a match, it returns `fail`. This can happen for two reasons: there is no match for the simple module considered, because the idempotent e' annihilates S. Or we also have $Se' \neq 0$, but $See'e = 0$, that is our test sample of vectors used to find a match is too small. We are unable to detect, for example, whether e and e' act *orthogonally* on S in the sense that $0 \neq Se \subseteq S(1-e')$ (or, alternatively, $0 \neq Se' \subseteq S(1-e)$).

Unfortunately, our approach renders these two cases indistinguishable: by adhering to the paradigm of condensation to avoid costly and usually infeasible computations in the original module V, the only vectors which we project with e' lie in Ve. Hence, if $ee'e$ annihilates Se, then we cannot find a match of Se among the composition factors of Ve', even though it might exist. In fact, as Example 1.1 indicates, to decide ultimately whether or not $Se' = 0$, we would have to determine a basis for the head of an S-local submodule of V, and project it entirely with e'. Of course, this renders the usage of condensation *ad absurdum*.

A couple of remarks on the implementation and behaviour of Algorithm 3 are in order. Experience shows that in applications such as the Modular Atlas project [9] in whose context the matching problem arose, the original algorithm [11, Algorithm 1] performs very well.

We found that it is sufficient to just consider a single element of $\ker_{Ve}(\omega)$ to produce a valid match. So in order to expedite the matching process, an implementation of Algorithm 3 should first follow the strategy of [**11**, Algorithm 1], and only if a match cannot be made, follow up by producing and checking the basis $\mathcal{B}_{Se}$. Of course, the improvements we have made which allow for a quicker calculation in lines 6 and 7 of Algorithm 3 should also be used if we are following the strategy of [**11**, Algorithm 1]. If we are unable to find a match because $See'e = 0$ it may still be that $Se'ee' \neq 0$, so that if `fail` is returned we can rerun the algorithm on the unmatched composition factors of Ve' to find a match in Ve. If this still does not produce a match we have to use further information such as the multiplicities of the composition factors, for example, to prove $Se' = 0$ or match Se.

3. *Matching in condensation algebras*

In practice, it is difficult to apply the matching algorithm, as we seldom have a generating system for the condensed algebra at hand, but instead are forced to work with the condensation algebra. Recall that given an algebra A and an idempotent $e \in A$, we call eAe the condensed algebra of A, and given $a_1, \ldots, a_r \in A$ we call $\mathcal{C} := \langle ea_1e, \ldots, ea_re \rangle$ the condensation algebra, that is the unital subalgebra of eAe generated by the condensed elements.

Of course, in general $\mathcal{C}$ differs greatly from eAe to the extent that little to no information regarding V or Ve may be gained by considering $Ve{\downarrow_{\mathcal{C}}}$. In practice, however, experience shows that usually a small set of elements chosen arbitrarily in eAe gives rise to a condensation algebra which if not equal to the condensed algebra contains the separable algebra of a Wedderburn–Malcev decomposition of eAe (see [**1**, Thoerem 72.19]). In other words, $\mathcal{C}$ is *sufficiently large* in the sense that all simple eAe-modules restrict irreducibly to $\mathcal{C}$, and two non-isomorphic simple eAe modules remain non-isomorphic after restriction.

In fact, an *inert* generating $\mathcal{E}$ set as stated in [**10**, Theorem 2.7], which is too large to be of immediate computational use, yields such a condensation algebra: we choose a small subset $\mathcal{E}' \subseteq \mathcal{E}$ and let $\mathcal{C} := \langle \mathcal{E}' \rangle$. Then we calculate an ascending composition series of $Ve{\downarrow_{\mathcal{C}}}$ with the MeatAxe, and record a basis of Ve which exhibits the composition series found: with respect to this basis matrices giving the action of elements of $\mathcal{C}$ on Ve are block diagonal lower triangular such that the blocks are matrix representations of the composition factors in standard basis, that is isomorphic composition factors give identical block matrices. Then to verify that $\mathcal{C}$ is sufficiently large we determine the action of the remaining generators of eAe on Ve in the basis computed, and check that the block-diagonal structure is preserved, as well as that non-isomorphic composition factors remain non-isomorphic.

Therefore, we assume that our condensation algebras contain the separable algebra of a Wedderburn–Malcev decomposition of the respective condensed algebras. The purpose of this section is to formulate a matching algorithm for such condensation algebras. We maintain our general setup and let $T := Se{\downarrow_{\mathcal{C}}}$ be the simple $\mathcal{C}$-module corresponding to Se via restriction. Furthermore, let $\mathcal{C}' \leq e'Ae'$ be a sufficiently large condensation algebra.

We begin by introducing the theoretical underpinning which allows us to refer to our results on condensed algebras from § 2. As restriction from eAe to $\mathcal{C}$ is exact, the first statement is immediate.

Lemma 3.1. *Let $veAe$ be an Se-local module for some simple A-module S. Then if $v\mathcal{C}$ is a local module, it is T-local.*

We obtain the following analogy to Lemma 2.1.

Lemma 3.2. *Let $f \in \mathcal{C}$ be an idempotent. Then f is T-primitive if and only if it is Se-primitive as an idempotent in eAe. Hence, if $\omega \in \mathcal{C}$ is a T-peakword it is an Se-peakword in A.*

Proof. The first claim follows from adjointness: for any simple eAe-module $S'e$ we have $\mathrm{Hom}_{eAe}(feAe, S'e) \cong \mathrm{Hom}_{\mathcal{C}}(f\mathcal{C}, T')$ if T' is the simple $\mathcal{C}$-module isomorphic to $S'e{\downarrow_{\mathcal{C}}}$. So by our hypothesis f is T-primitive if and only if it is Se-primitive in eAe. By [**5**, Theorem 3.4], the stable kernel of a T-peakword on a $\mathcal{C}$-module U is Uf for some T-primitive idempotent $f \in \mathcal{C}$, which by the preceding is Se-primitive as an idempotent in eAe. □

In order for Algorithm 3 to work with condensation algebras, we need to replace the equality condition of Lemma 2.3 which is based on condensed algebras. Since condensation algebras are in general proper subalgebras of the condensed algebras, we have to take into account that the element $ee'e \in eAe$ maps v to an element $w := ve'e \in veAe$ not necessarily lying in $v\mathcal{C}$. Since we have a composition series of $Ve{\downarrow_{\mathcal{C}}}$ at our disposal, we exploit the information it provides.

To this end let $0 = U_0 < U_1 < \ldots < U_l = Ve{\downarrow_{\mathcal{C}}}$ be a composition series of $\mathcal{C}$-modules. Furthermore, let $i \in \{1, \ldots, l\}$ be minimal such that $U_i/U_{i-1} \cong T$. Take $v \in U_i$ such that $v\mathcal{C}$ is a T-local submodule by using Algorithm 2 to project U_i onto a T-peakword kernel. We now work up along the composition series and clean w with a basis for every submodule to determine the submodule U_j, $j \in \{1, \ldots, l\}$, containing w. In this notation three possible cases arise.

Remark 3.3. (1) If $w \in U_j$ for some $j < i$, then by the minimality of i the $\mathcal{C}$-module $w\mathcal{C}$ has no composition factor isomorphic to T, so in particular there is no homomorphism mapping $w\mathcal{C}$ onto T. By adjointness, Se does not appear in the head of $w\mathcal{C} \otimes_{\mathcal{C}} eAe$. Hence, $weAe$ cannot be Se-local, so $w \in \mathrm{rad}(veAe)$.

(2) If $w \in U_i$, but not in U_{i-1}, then $w\mathcal{C}$ maps onto T. Let $f \in \mathcal{C}$ be a T-primitive idempotent. There is a $c \in \mathcal{C}$ such that the element vcf generates the simple $f\mathcal{C}f$-module U_if, and therefore $wc'f = vcf$ for some $c' \in \mathcal{C}$. From this $v\mathcal{C} = vcf\mathcal{C} \leq w\mathcal{C}$ follows, so $veAe = weAe$.

(3) If $w \in U_j$ for some $j > i$, then $v\mathcal{C}$ may be a submodule of $w\mathcal{C}$, giving $veAe = weAe$. Otherwise $v\mathcal{C} \not\leq w\mathcal{C}$ and $w\mathcal{C} \not\leq v\mathcal{C}$ as $w \notin U_i$. We can check whether $v\mathcal{C} \leq w\mathcal{C}$ by the following: we spin up w modulo U_{i-1} and clean v with $U_{i-1} + w\mathcal{C}$. This yields an exhaustive search for elements in $U_i \setminus U_{i-1}$. If we find such an element we are done, as this generates a submodule of $w\mathcal{C}$ containing $v\mathcal{C}$.

Remark 3.3 enables us to verify $veAe = weAe$ by testing for the sufficient condition whether or not $v\mathcal{C}$ is a submodule of $w\mathcal{C}$ without computing an explicit basis for either $v\mathcal{C}$ or $w\mathcal{C}$. Furthermore, since we allow $\mathcal{C}$ to be a proper subalgebra of eAe this condition is also less restrictive than to impose the condition $v\mathcal{C} = w\mathcal{C}$. Note that the special treatment of case (3) is necessary, as $w\mathcal{C}$ may not be a local module. This is in contrast to the uniform approach just entailing successive cleanings we were able to use for condensed algebras in § 2. If we find in case (3) that neither $v\mathcal{C} \leq w\mathcal{C}$ nor $w\mathcal{C} \leq v\mathcal{C}$, then the information provided is inconclusive, as $\mathcal{C}$ differs too greatly from eAe.

We now propose Algorithm 4 to match a composition factor of $Ve{\downarrow_{\mathcal{C}}}$ to a composition factor of $Ve{\downarrow_{\mathcal{C}'}}$. The strategy employed is the same as in Algorithm 3, only this time we are calculating within $\mathcal{C}$- and $\mathcal{C}'$-modules. The difference to Algorithm 3 occurs in lines 6–18: by Remark 3.3 the condition if true gives $veAe = ve'eAe$, which by Lemma 2.3 gives that $Se' \neq 0$, and therefore also $T' := Se'{\downarrow_{\mathcal{C}'}} \neq 0$. Also, $ve'Ae'$ is an Se'-local submodule of Ve', and by Lemma 3.1, if $ve'\mathcal{C}'$ is a local module, then it is T'-local. In this case in lines 7–10 we can match T to T'. But, owing to the fact that $ve'\mathcal{C}'$ may not be a local module, we cannot simply clean ve' using a $\mathcal{C}'$-composition series of $Ve'{\downarrow_{\mathcal{C}'}}$ to determine the head of $ve'\mathcal{C}'$. So in line 9 of Algorithm 4 we need to construct the module $ve'\mathcal{C}'$ and compute its head as detailed in [**8**].

A smooth run of this matching process may be hindered in two ways: if we arrive in the exceptional case (3) of Remark 3.3 and find $v\mathcal{C} \not\leq w\mathcal{C}$ and $w\mathcal{C} \not\leq v\mathcal{C}$, we need to restart the matching program with additional generators for $\mathcal{C}$. Also, in the scenario that $ve'\mathcal{C}'$ is not local, we have found that $\mathcal{C}' \neq e'Ae'$, and need to provide additional generators for $\mathcal{C}'$.

Algorithm 4 Matching a composition factor for condensation algebras

Input: A complementary basis $\mathcal{B}_T$ of generators of a T-local submodule of $Ve{\downarrow_{\mathcal{C}}}$ output by Algorithm 1, a database `DB` of representatives of the isomorphism types of composition factors of $Ve'{\downarrow_{\mathcal{C}'}}$.

Output: The T' in `DB` which matches T, or `fail` if the match fails.

```
1:  for $v \in \mathcal{B}_T$ do
2:    Embed $v$ into $V$.
3:    Project $v$ onto $Ve'$, obtaining $ve' \in Ve'$.
4:    Embed $ve'$ into $V$.
5:    Project $ve'$ onto $Ve$, giving $ve'e \in Ve$.
6:    if $v\mathcal{C} \leq ve'e\mathcal{C}$ then
7:      Compute the head $H$ of $ve'\mathcal{C}'$.
8:      if $H$ is simple then
9:        Determine the module $T'$ in DB isomorphic with $H$.
10:       return $T'$.
11:     else
12:       print More generators for $\mathcal{C}'$ are needed.
13:       return fail
14:     end if
15:   else
16:     print More generators for $\mathcal{C}$ are needed.
17:     return fail
18:   end if
19: end for
20: return fail.
```

4. *Practical performance*

Condensation finds a major application in computational representation theory of finite groups and in the Modular Atlas project [9] in particular. For this reason we focus on the condensation of tensor products $V \otimes_F W$ of modules V and W for some group algebra $A = FG$, where G is a finite group. There are efficient algorithms available to achieve this goal for specific idempotents e and e' (see [6, 7]), that is algorithms which determine the action of an element in eAe on $(V \otimes_F W)e$ while adhering to the paradigm of condensation to avoid explicit calculations in $V \otimes_F W$. In particular, these algorithms entail methods to project any vector $v \in V \otimes_F W$ onto ve without evaluating e, and hence facilitate projection with e and e' at the heart of our matching algorithm. As for these efficient projections the modules have to be given with respect to a special basis of $V \otimes_F W$, we need to be able to change these bases from one to the other before projection is possible. By the very nature of this basis change, we are forced to perform calculations in the uncondensed module and cannot conform to the paradigm of condensation. In our case of tensor products, however, we are in a benign situation: using the natural isomorphism of $V \otimes_F W$ and $\mathrm{Hom}_F(V^*, W)$ the necessary basis changes can be realised as basis changes in the tensor factors. Thus, working under the basic assumption that the tensor factors are computationally manageable the same holds true for the basis changes. See also [11] for details how this fundamental problem of our algorithm may be circumvented for permutation modules.

We choose not to discuss the peakword search part of the precomputations necessary to enable our matching algorithms. Current peakword search methods are randomised in the sense that a word generator produces pseudo-random elements in the algebra which are then checked for peakword properties. Therefore different runs of a peakword search may produce different peakwords and the outcome depends very much on the strategy of the word generator used. Thus, while a peakword search may contribute substantially to the total running time of the matching process, we only consider the computations in Algorithms 3 and 4 in comparison to their corresponding steps in [**11**, Algorithm 1].

We return to our old example from [**11**]: the tensor product $V := 133_1 \otimes 8778_1$ for the simple sporadic Harada–Norton group in characteristic 3, so let $G = \mathsf{HN}$ and $F = GF(9)$. The idempotents e and e' used are the trace idempotents of the subgroups $2^3.2^2.2^6$ and $5^2.5.5^2$, which are normal in the ninth, respectively tenth, maximal subgroup of G (see [**2**] for details). Condensation gives a 567-dimensional $eFGe$ module Ve which possesses the composition factors

$$1_1, 1_1, 1_1, 1_1, 1_1, 1_2, 8, 8, 9, 9, 26_1, 26_1, 26_2, 138, 138, 173$$

and a 438-dimension $e'FGe'$-module Ve' with composition factors

$$1', 3', 3', 7', 7', 8'_1, 8'_1, 8'_1, 8'_2, 8'_2, 8'_3, 8'_3, 8'_3, 8'_4, 16', 16', 16', 16', 89', 96', 96'.$$

Running the matching algorithms on a modern (at the time of writing) machine with an Intel Core i7 870 processor at 2.93 GHz finds the matches

$$1_2 \leftrightarrow 1', 8 \leftrightarrow 7', 26_1 \leftrightarrow 8'_2, 26_2 \leftrightarrow 8'_4, 138 \leftrightarrow 96', 173 \leftrightarrow 89'.$$

Using the strategy of [**11**, Algorithm 1] augmented with a computation of a set of generators for the local modules occurring (Algorithm 1) this takes about 5–7.5 s either way. Most of the time is spent on performing the basis changes to facilitate an easy projection with the idempotents. This makes up for about 3.5 s of running time for one match. The remaining time goes into the standard MeatAxe calculations within the condensed modules, namely determining dimensions of submodules, computing the heads of local modules, and their matrix representations. The calculation of an exhaustive set of generators for the chosen local submodules is almost negligible: even for the larger composition factors it clocks in at most 3/10 of a second. However, in the case that we cannot find a match (and for which we know that a match does not exist, see [**2**]), the algorithm loops over as many generators for a local module as is given by the dimension of its simple head. For each of these vectors basis changes are conducted, thus giving us the output `fail` after about $d \times 3.5$ s, where d is the dimension of the composition factor. Applying the algorithms proposed in this paper, eliminates one spinning procedure in the case of condensation algebras, and even all further head and action calculations in the case that we know to work with the full condensed algebras. In the latter case the algorithm's running time is almost exclusively spent on the basis changes.

The dominance of the basis changes in the matching calculation of the preceding example is misleading: the condensations of the tensor product $133_1 \times 8778_1$ give comparatively low-dimensional modules (438, respectively 567) in relation to the dimension of one of the composition factors (8778). Hence, it is not surprising that calculations within the condensed modules proceed much quicker than calculations within the tensor factors. However, the future applications of the matching algorithms in the Modular Atlas project cannot be expected to follow this example. In fact, as condensation is applied to push the boundary of the possible in computational representation theory, the goal is to invoke a matching algorithm for a large

TABLE 1. *Comparison of matching times.*

S	Gens	eAe old	eAe	$\mathcal{C}$
10	6.5	17 208.8	22.8	132.1
34	13.1	163.0	29.4	139.4
231	51.1	716.4	67.5	574.0
702	94.4	2 348.5	110.7	1 666.9
1 333	596.5	4 619.0	612.9	3 795.0

condensed tensor product whose dimension exceeds the dimension of its factors. In this case, the calculations conducted in the condensed modules become the bottleneck of matching, so our improvements aim to mitigate this problem.

For a demonstration, we consider the tensor product $V := 3344 \otimes 3344$ for G in characteristic 3. As a condensation subgroup we choose a subgroup of index 2 in the normal subgroup $2^3.2^2.2^6$ of the ninth maximal subgroup of G. Now we obtain a condensed module Ve of dimension 11 248. For illustrative purposes, we match from Ve to itself. Since the basis changes in the projection steps of the matching process become trivial this way, we conduct arbitrary basis changes with random matrices in their place to still obtain a good estimate of the general running time. Thus, the time spent on changing bases to project one vector back and forth is in the worst case about 15.8 s in this example.

We exemplify the different algorithms' running times by matching the composition factors 10, 34, 231, 702, and 1333 of Ve. The computation times needed in seconds are given in Table 1. In this table the column labelled by 'Gens' gives the time taken to compute a set of generators for an Se-local module, the column 'eAe old' gives the times of the match using the strategy of [**11**, Algorithm 1], the column 'eAe' gives the running time of Algorithm 3 and the column '$\mathcal{C}$' gives the times of Algorithm 4.

From Table 1 we see that as the dimension of the composition factor to be matched increases, the proportion of time the algorithms spend on performing basis changes is reduced. In fact, while basis changes are performed at the constant time of about 16 s, the remaining calculations quickly dominate the overall running time. If we are working with condensed algebras, so that Algorithm 3 may be applied, we avoid the majority of the MeatAxe calculations in the condensed module, explaining the speed-up we see when comparing columns 'eAe old' and 'eAe'. Not having full condensed algebras, but only condensation algebras, allows us to avoid some of the calculations performed in the old algorithm. As the table indicates, these savings will become more substantial as the dimension of the composition factors and also the condensed module increases.

Note that not only the size of the composition factors contributes to the overall running time, but also the structure of the module as a whole: if the matching of a composition factor is enabled through a large local submodule a long running time will reflect this. The excessively long time the original algorithm spends in matching 10 exhibits this fact, and also demonstrates the benefit of our new approach that we choose a local submodule of small dimension via a composition series (see Algorithm 1). There are 10 occurrences of 10 as a factor in a composition series of Ve, and the old algorithm simply takes the first vector in the kernel of a peakword for 10. As Table 1 shows this leads to a 10-local submodule which is larger than necessary and therefore increases the duration of all subsequent calculations drastically.

Acknowledgement. The author is indebted to Jürgen Müller for bringing Example 1.1 to his attention.

References

1. C. W. Curtis and I. Reiner, 'Representation theory of finite groups and associative algebras', *Pure and Applied Mathematics,* vol. XI (Interscience Publishers, a division of John Wiley & Sons, New York, London, 1962).
2. G. Hiss, J. Müller, F. Noeske and J. Thackray, 'The Brauer characters of the sporadic simple Harada–Norton group and its automorphism group in characteristics 2 and 3', *LMS J. Comput. Math.* 15 (2012) 257–280.
3. D. F. Holt, B. Eick and E. A. O'Brien, *Handbook of computational group theory*, Discrete Mathematics and its Applications (Chapman & Hall/CRC, Boca Raton, FL, 2005).
4. K. Lux, 'Algorithmic methods in modular representation theory'. http://math.arizona.edu/∼klux/habil.dvi.gz, 1997.
5. K. Lux, J. Müller and M. Ringe, 'Peakword condensation and submodule lattices: an application of the MEAT-AXE', *J. Symbolic Comput.* 17 (1994) no. 6, 529–544.

6. K. Lux, M. Neunhöffer and F. Noeske, 'Condensation of homomorphism spaces', *LMS J. Comput. Math.* 15 (2012) 140–157.
7. K. Lux and M. Wiegelmann, 'Condensing tensor product modules', *The atlas of finite groups: ten years on (Birmingham, 1995)*, London Mathematical Society Lecture Note Series 249 (Cambridge University Press, Cambridge, 1998) 174–190.
8. K. Lux and M. Wiegelmann, 'Determination of socle series using the condensation method', *J. Symbolic Comput.* 31 (2001) no. 1–2, 163–178; *Computational algebra and number theory* (Milwaukee, WI, 1996).
9. The modular atlas hompage. http://www.math.rwth-aachen.de/∼MOC/.
10. F. Noeske, 'Tackling the generation problem in condensation', *J. Algebra* 309 (2007) no. 2, 711–722.
11. F. Noeske, 'Matching simple modules of condensed algebras', *LMS J. Comput. Math.* 11 (2008) 213–222.
12. A. J. E. Ryba, 'Computer condensation of modular representations', *J. Symbolic Comput.* 9 (1990) no. 5–6, 591–600; *Computational group theory*, Part 1.

Felix Noeske
Lehrstuhl D für Mathematik
RWTH Aachen University
52056 Aachen
Germany

felix.noeske@math.rwth-aachen.de