# Data-Driven Discovery of Anchor Points for PDC Content

Matthew McQuaigue[1], **Erik Saule**[1], Kalpathi Subramanian[1], Jamie Payton[2]
esaule@uncc.edu

[1]UNC Charlotte, [2]Temple University

EduHPC 2023

# Towards HPC in early CS

## Improving PDC education in undergrad education

- We've provided curriculum guidelines for PDC
- We have inspired ACM/IEEE CS guidelines.
- We train instructors
- We have produced Peachy assignments

## We have a matching problem

- We have people who may want to teach PDC in early CS
- We have people with PDC expertise.
- Yet we are running into early CS instructors who don't know what to teach
- And PDC experts who don't know how to help

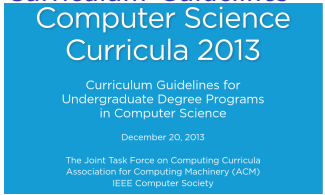
# The core problem

Not every "CS1" lecture/assignment will fit in every "CS1" class

- Instructors have a lack of understanding of how to find materials that work for them.
- PDC experts have have a lack of understanding of how to develop materials that will integrate well.
- We need to understand how CS is being taught to both find materials that work and develop PDC content **with a target** of where it could be adopted.

# CS Materials: https://cs-materials.herokuapp.com

## Curriculum Guidelines




memory" across all higher-level topics — in addition to the hours allocated to related topics such as "SPMD," "tasks and threads," and "synchronization." In contrast, the hours allocated to Algorithms topics represent our estimates of the effort required to achieve the desired level of competence solely within the context of Algorithms instruction. This decision reflects our recognition that many Algorithms topics develop concepts and tools that will pervade the coverage of many disparate non-Algorithms topics — the specific list of topics varying from institution to institution. The cumulative number of hours to master a topic is, therefore, impossible to estimate in isolation.

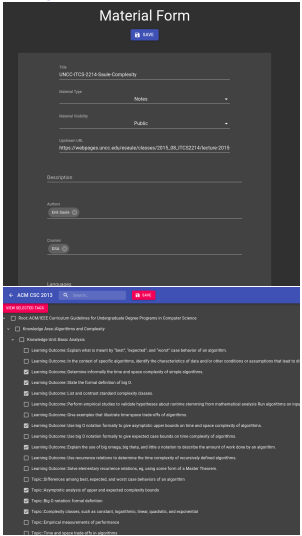
**8.2 Architecture Topics**

**Table 1: Architecture**

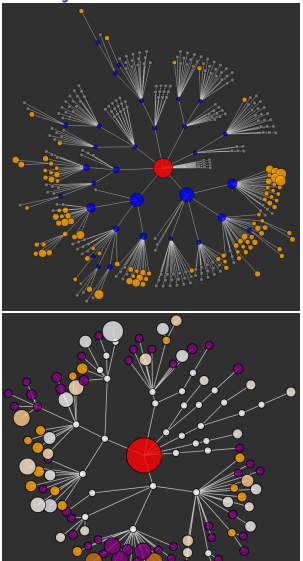
| Topics | ELO-M | HOURS | Where Covered | Learning Outcome |
|---|---|---|---|---|
| **Classes** | | | | |
| Taxonomy | C | 0.5 | Systems | Flynn's taxonomy, data vs. control parallelism, shared/distributed memory |
| Data vs. control parallelism | | | | |
| • Superscalar (ILP) | K | 0.25 to 1 based on level | Systems | Describe opportunities for multiple instruction issue and execution (different instructions on different data) |
| SIMD/Vector (e.g., SSE, Cray) | K | 0.1 to 0.5 | Systems | Describe uses of SIMD/Vector (same operation on multiple data items) (e.g., accelerating graphics for games. |
| *Pipelines* | | | | |
| • Single vs multicycle | K | 1 to 2 | Systems | Describe basic pipelining process (multiple instructions can execute at the same time), describe stages of instruction execution |
| • Data and control hazards | | | Compilers (A), Arch 2 (C) | Understand how one pipe stage can depend on a result from another, or delayed branch resolution can start the wrong instructions in a |

17

## Map classes to curriculum




## Analyze classes



# Data collection

## Workshops

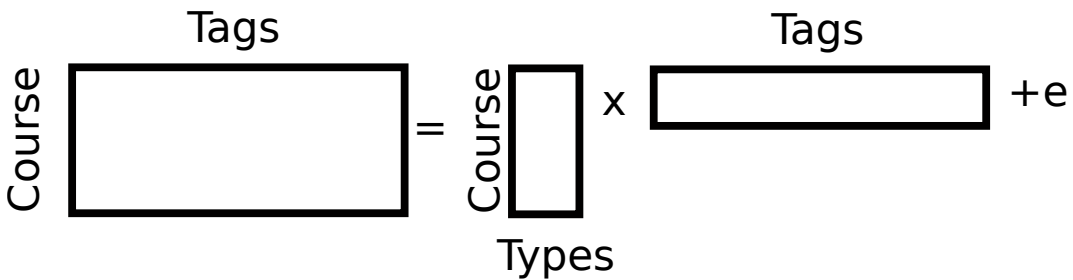
- Conducted Workshops online during COVID
- Conducted 2 workshops summer 2023 to train instructors in analyzing their courses using CS Materials
- 31 courses
- About 1700 materials (lectures, assignments, quizzes, etc.) in CS Materials system

## Dataset

- Retained 20 courses in total for analysis
- Courses were tagged with course type based on course names

# Discovering Types of Courses

## Non Negative Matrix Factorization




By setting the number of types, you can investigate different clustering of courses.

# Results and Conclusions

## What we saw

- Confirmed that NNMF enables to recover the type of courses
- Identified 3 types of CS1 courses: Object Oriented, Imperative, Algorithmic Thinking
- Identified 3 types of Data Structure courses: Focus on interfaces and OOP, focus on application, cover combinatorial algorithms.

## What it means for PDC in early CS courses

- Object Oriented CS1 may not be able to integrate loop-based parallelism, but may support promise style concurrency.
- Only Imperative CS1 talks about number representation, so representation based discussion of parallel reduction probably only makes sense there.
- OOP style Data structure probably can support thread safe DS discussions.
- Dependency extraction and PTG is probably easier in DS with combinatorial alg.

# Thank You!

## Learn more

- `https://cs-materials.herokuapp.com`
- Goncharow et al. CS-Materials: A system for classifying and analyzing pedagogical materials to improve adoption of parallel and distributed computing topics in early cs courses. JPDC 2021.
- Goncharow et al. Mapping materials to curriculum standards for design, alignment, audit, and search. SIGCSE 21.
- Goncharow et al. Classifying pedagogical material to improve adoption of parallel and distributed computing topics. EduPar 19.
- We will be running CS Materials workshops Summer 24.
- Contact us: esaule@uncc.edu

## We are recruiting!

UNC Charlotte is recruiting: PhD Students, Faculty, Chaired Faculty.


## Acknowledgement

This work was supported by grants from the NSF: OAC-1924057, CCF-1652442.