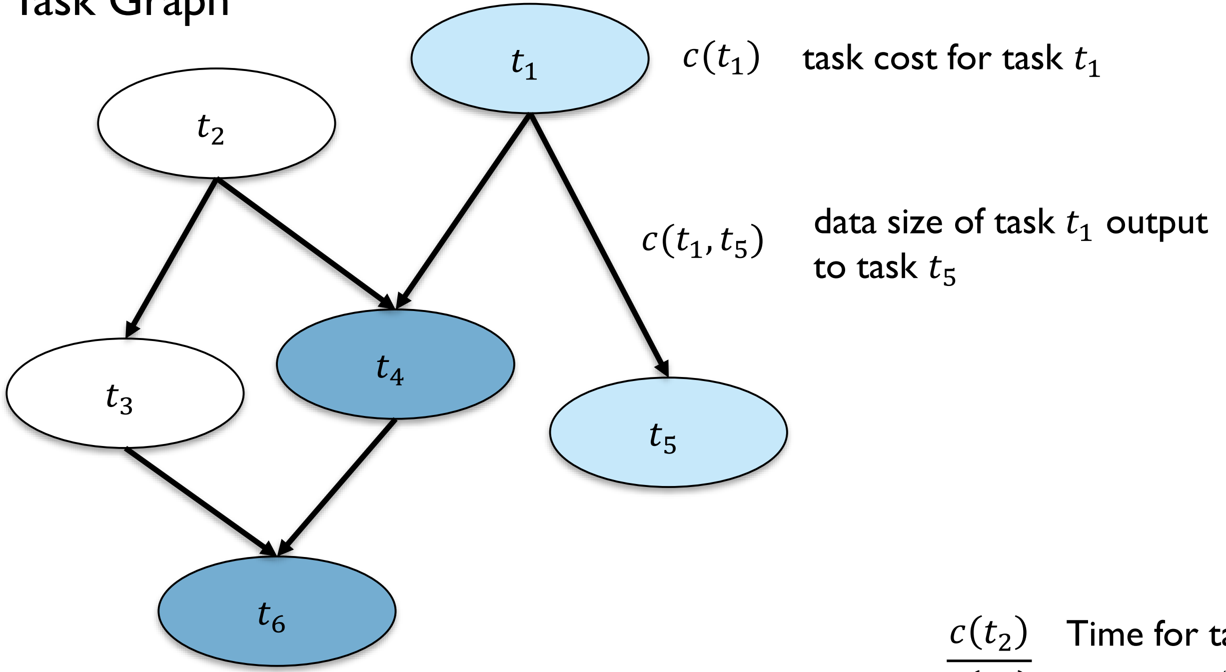
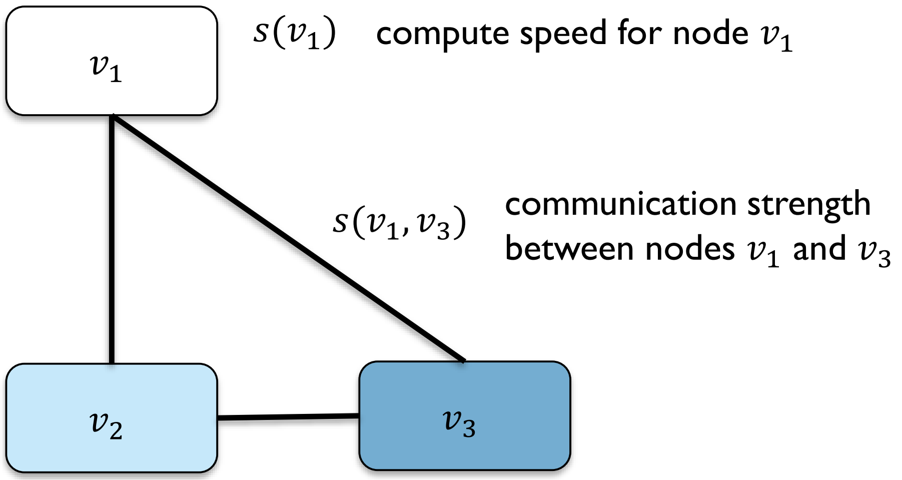
PARAMETERIZED TASK GRAPH SCHEDULING ALGORITHM FOR COMPARING ALGORITHMIC COMPONENTS

Jared Coleman	Loyola Marymount University
Bhaskar Krishnamachari	University of Southern California
Ravi Agrawal	University of Southern California
Ebrahim Hirani	University of Southern California

Task Graph



Compute Network



Schedule

$\frac{c(t_2)}{s(v_1)}$ Time for task t_2 to execute on compute node v_1

compute node v_1	task t_2	task t_3				
compute node v_2	task t_1	task t_5				
compute node v_3		task t_4		task t_6		

$\frac{c(t_2, t_4)}{s(v_1, v_3)}$ Time for compute node v_1 to send task t_2 outputs to compute node v_3

A BRIEF HISTORY

- Task scheduling work started in the 1950s
- A survey in 1979 by Graham classified many different variants
  - Did not consider heterogeneous communication
- Hwang et. al. were the first to study heterogeneous communication in 1989
  - Consider homogeneous computation
- Su et. al address fully heterogeneous case in 2019
- Distributed computing goes mainstream – tons of heuristic algorithms are proposed
  - BIL, CPoP, ETF, FCP, FLB, GDL, HEFT, MinMin, MaxMin, MCT, MET, OLB, SMT, WBA, …
- Modern use cases (ML/data science workflows, IoT, etc.) drive benchmarking efforts and theoretical advances
- Bazzi et. al. show scheduling is not approximable within a constant factor in 2015

COMPARING SCHEDULING ALGORITHMS

- Scheduling is NP-Hard and not approximable, but what about for practical situations?
  - Scheduling an ML workflow
  - Running a scientific workflow on a supercomputing system
  - Collecting and processing data from an IoT system
  - Running analyses in a tactical edge environment
- Many tasks scheduling heuristic algorithms have been proposed
  - HEFT, CPoP, BIL, ETF, FCP, FLB, GDL, MinMin, MaxMin, MCT, MET, OLB, SMT, WBA, …
- SAGA: Scheduling Algorithms Gathered
  - 17 Algorithm Implementations
  - 15 Dataset Generators
  - Tools for Benchmarking
  - Tools for Adversarial Analysis

COMPARING SCHEDULING ALGORITHMS

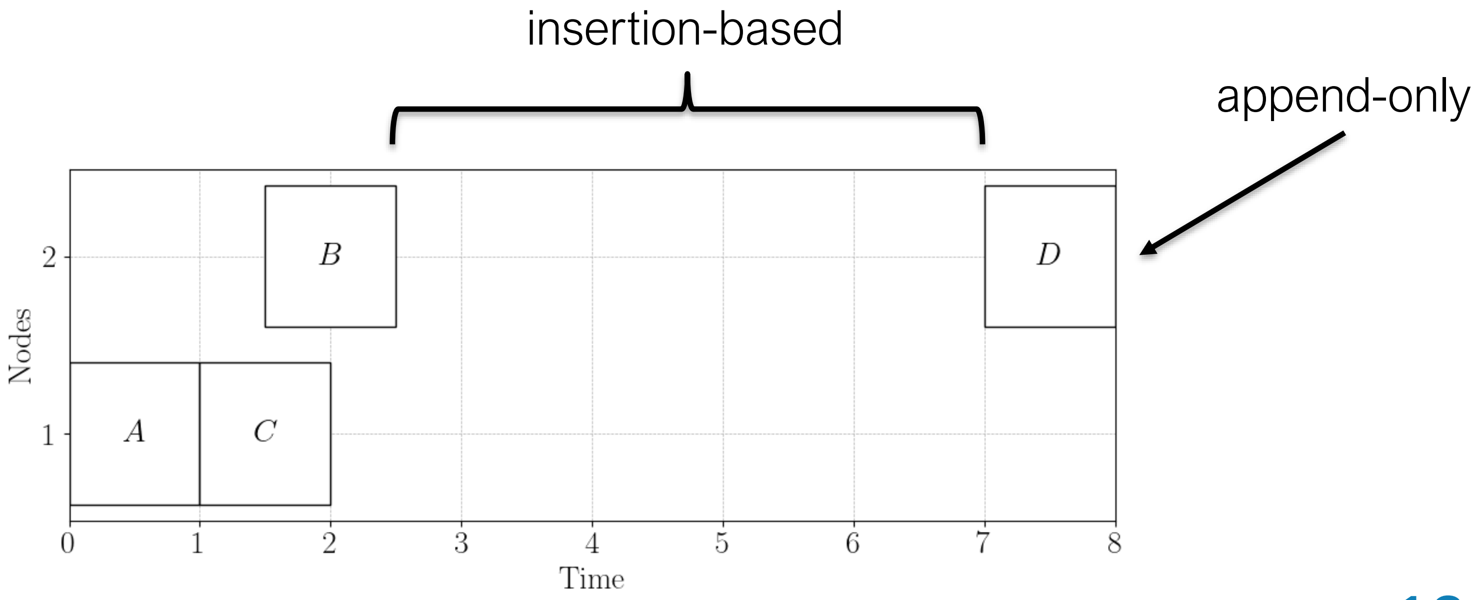
- Many of the heuristic algorithms are similar to each other
- List Scheduling Algorithms (HEFT, CPoP, ETF, etc.):
  1. Compute priorities for tasks
  2. Greedily schedule tasks to node that minimizes/maximizes a cost function
- **Question:** How do individual algorithmic differences affect algorithm performance and runtime?

COMPARING ALGORITHMIC COMPONENTS

- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage

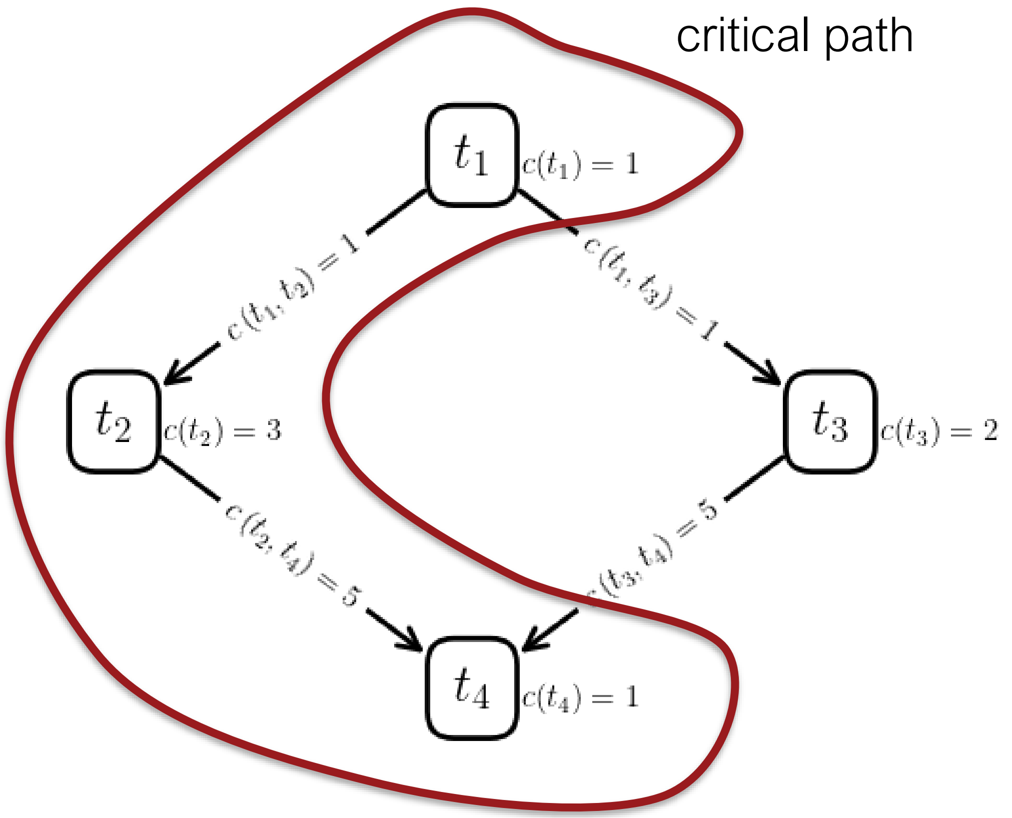
COMPARING ALGORITHMIC COMPONENTS

- » Algorithmic components:
  - » Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - » Comparison Function: EFT, EST, Quickest
  - » Insertion-based vs. append-only scheduling
  - » Critical path reservation vs no reservation
  - » Sufferage vs no sufferage



COMPARING ALGORITHMIC COMPONENTS

- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage



COMPARING ALGORITHMIC COMPONENTS

- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage

COMPARING ALGORITHMIC COMPONENTS

- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage
- 72 possible scheduling algorithms total

COMPARING ALGORITHMIC COMPONENTS

- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage
- 72 possible scheduling algorithms total

HEFT

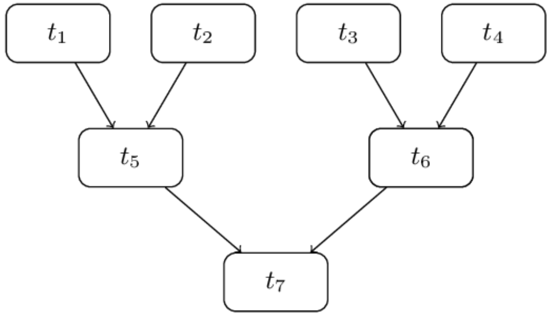
COMPARING ALGORITHMIC COMPONENTS

- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage
- 72 possible scheduling algorithms total

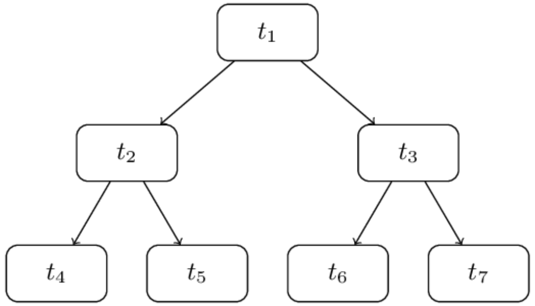
CPoP

COMPARING ALGORITHMIC COMPONENTS

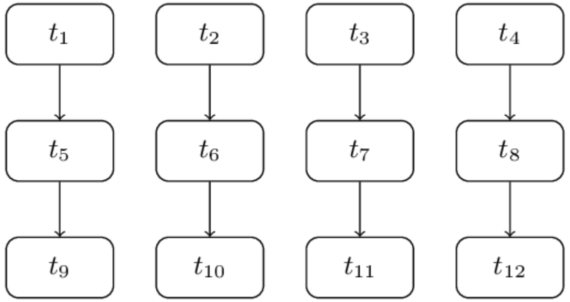
- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage
- 72 possible scheduling algorithms total
- 20 datasets
  - 4 types: **in-trees, out-trees, chains**, cycles
  - 5 CCRs: $\left\{\frac{1}{5}, \frac{1}{2}, 1, 2, 5\right\}$
  - 100 problem instances in each dataset
- 144,000 evaluations



in-trees



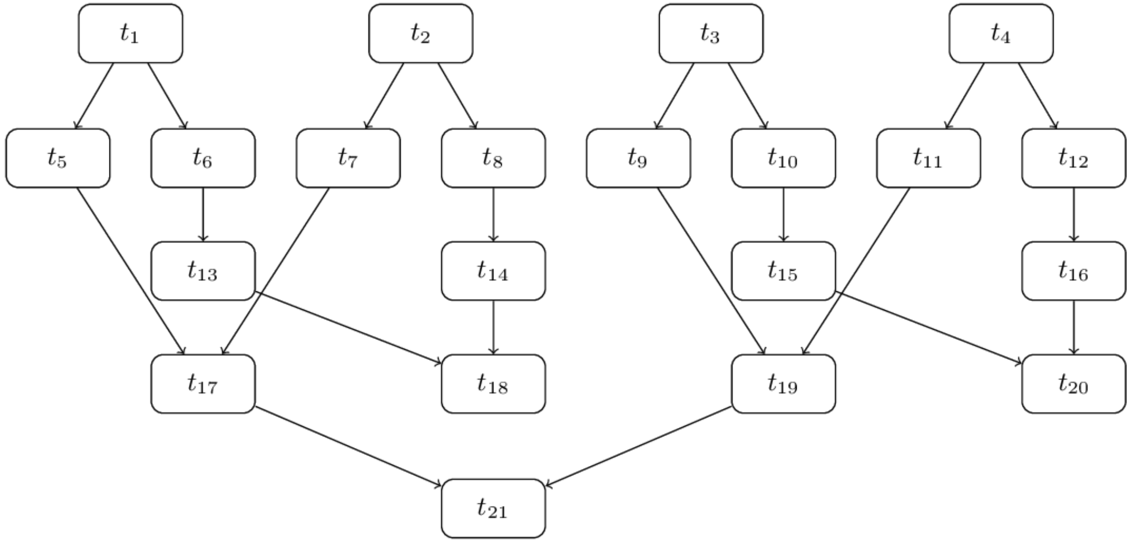
out-trees



chains

COMPARING ALGORITHMIC COMPONENTS

- Algorithmic components:
  - Priority Function: UpwardRank, CPoPRank, ArbitraryTopological
  - Comparison Function: EFT, EST, Quickest
  - Insertion-based vs. append-only scheduling
  - Critical path reservation vs no reservation
  - Sufferage vs no sufferage
- 72 possible scheduling algorithms total
- 20 datasets
  - 4 types: chains, in-trees, out-trees, **cycles**
  - 5 CCRs: $\left\{\frac{1}{5}, \frac{1}{2}, 1, 2, 5\right\}$
  - 100 problem instances in each dataset
- 144,000 evaluations



cycles

MAKESPAN/RUNTIME RATIO

» For a network N and task graph G

» Let $M_A(N,G)$ denote the **makespan** of algorithm A for the problem instance (N,G)

» The **makespan ratio** of algorithm A against algorithm B is

$$MR_{A,B}(N,G) = \frac{M_A(N,G)}{M_B(N,G)}$$

* how many times worse A is than B on a particular instance

» The **runtime ratio** of algorithm A against algorithm B is

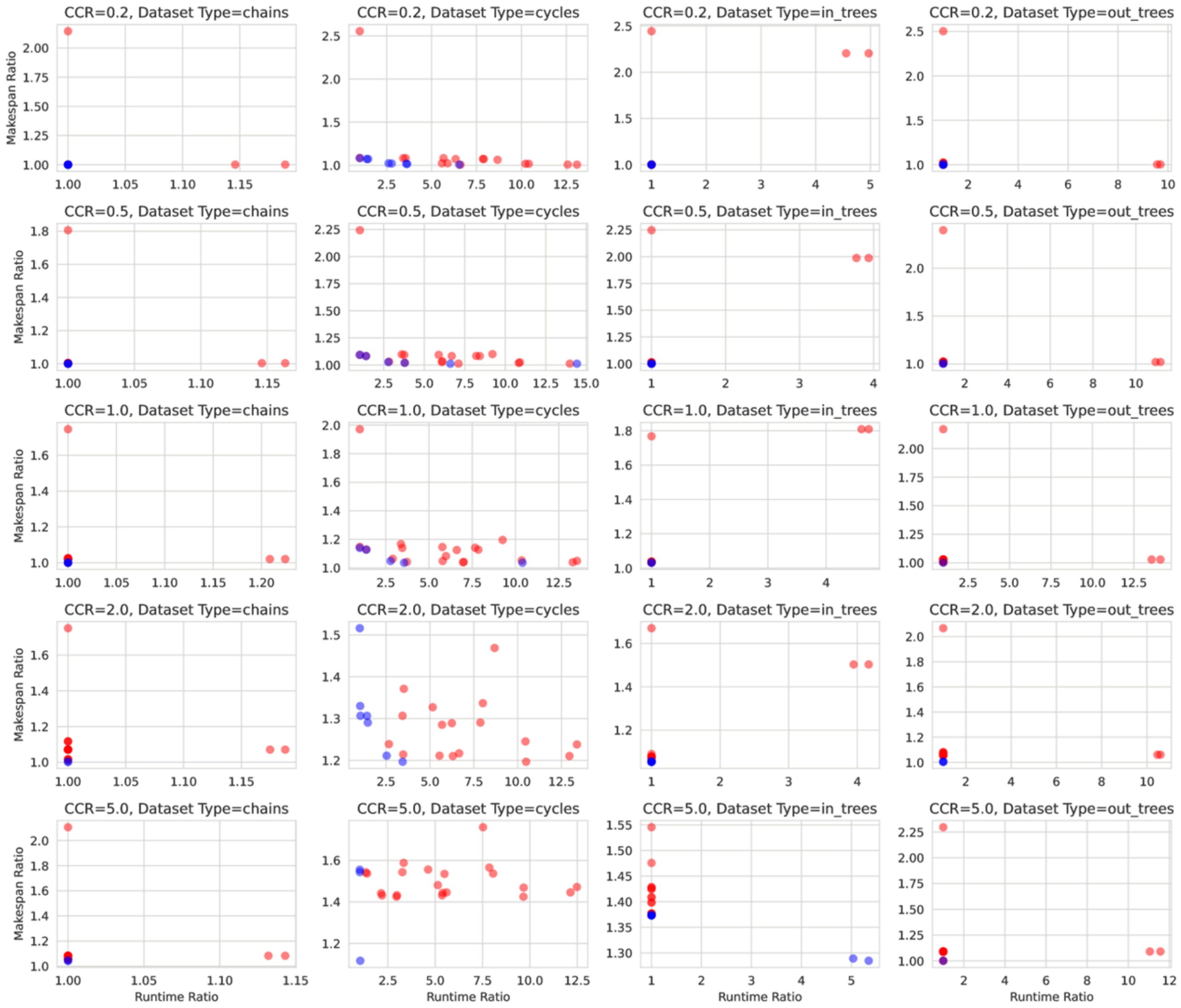
$$RR_{A,B}(N,G) = \frac{T_A(N,G)}{T_B(N,G)}$$

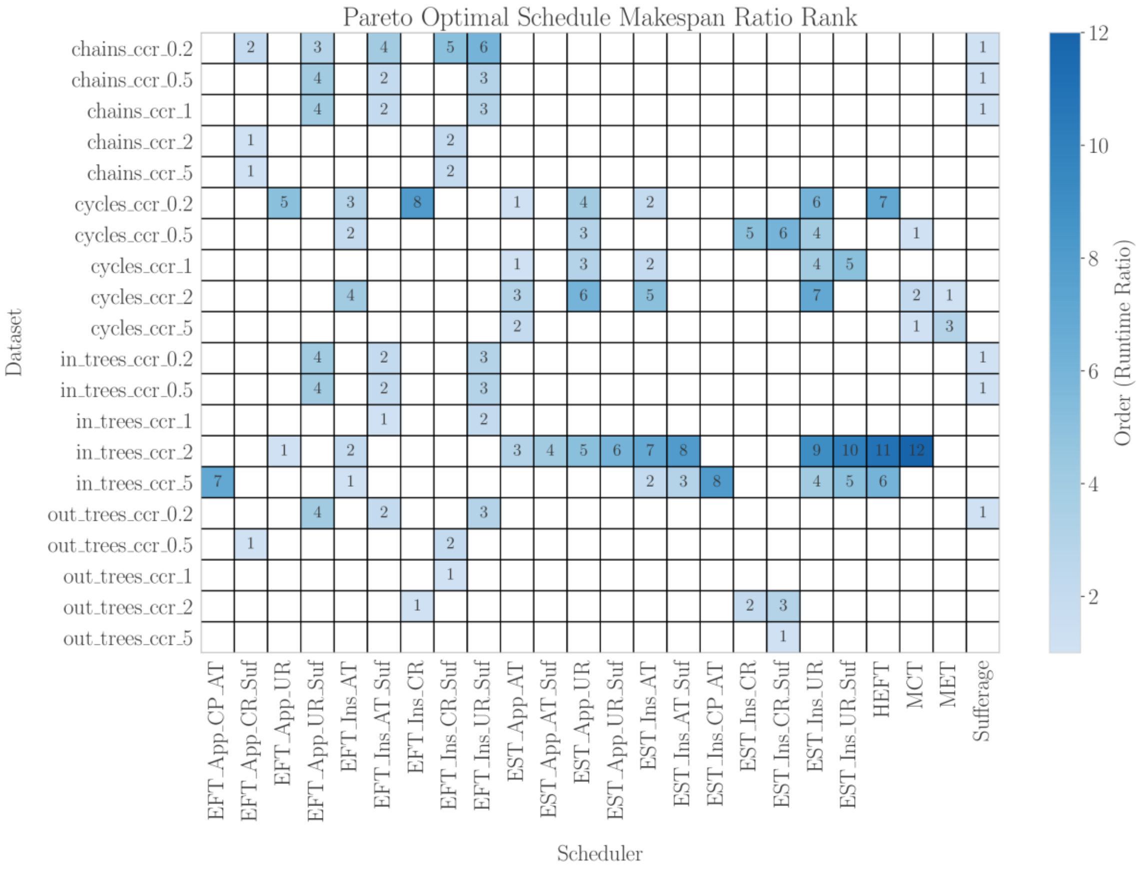
* how many times longer does A take to run than B on a particular instance

PARETO-OPTIMAL SCHEDULERS

- Pareto-optimal: for at least one dataset, no other algorithm exists that has better average makespan ratio **and** average runtime ratio
- 20/24 pareto-optimal schedulers have never been studied before

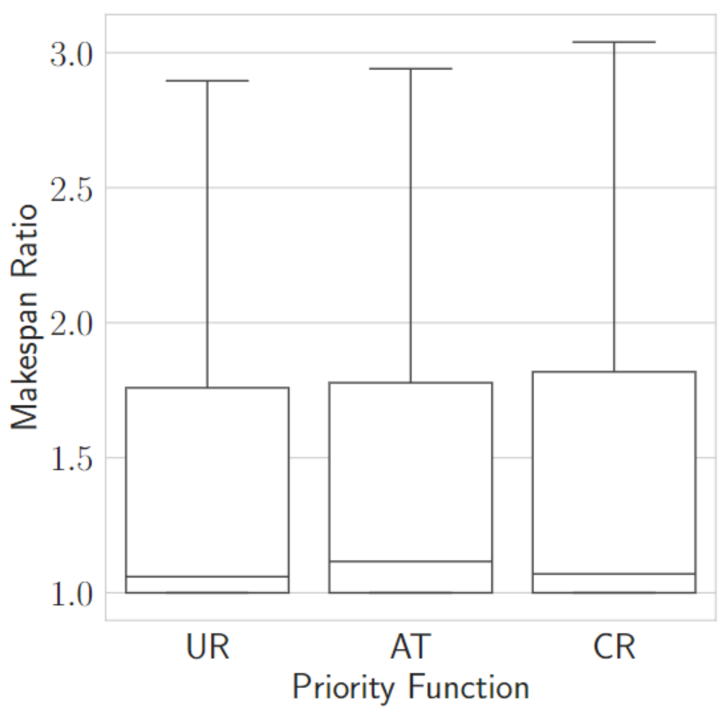
component	initial_priority	append_only	compare	critical_path	sufferage
scheduler					
EFT_Ins_AT	ArbitraryTopological	False	EFT	False	False
EFT_Ins_AT_Suf	ArbitraryTopological	False	EFT	False	True
EST_Ins_AT	ArbitraryTopological	False	EST	False	False
EST_Ins_AT_Suf	ArbitraryTopological	False	EST	False	True
EST_Ins_CP_AT	ArbitraryTopological	False	EST	True	False
MCT [10]	ArbitraryTopological	True	EFT	False	False
Sufferage [12]	ArbitraryTopological	True	EFT	False	True
EFT_App_CP_AT	ArbitraryTopological	True	EFT	True	False
EST_App_AT	ArbitraryTopological	True	EST	False	False
EST_App_AT_Suf	ArbitraryTopological	True	EST	False	True
MET [10]	ArbitraryTopological	True	Quickest	False	False
EFT_Ins_CR	CPoPRanking	False	EFT	False	False
EFT_Ins_CR_Suf	CPoPRanking	False	EFT	False	True
EST_Ins_CR	CPoPRanking	False	EST	False	False
EST_Ins_CR_Suf	CPoPRanking	False	EST	False	True
EFT_App_CR_Suf	CPoPRanking	True	EFT	False	True
HEFT [6]	UpwardRanking	False	EFT	False	False
EFT_Ins_UR_Suf	UpwardRanking	False	EFT	False	True
EST_Ins_UR	UpwardRanking	False	EST	False	False
EST_Ins_UR_Suf	UpwardRanking	False	EST	False	True
EFT_App_UR	UpwardRanking	True	EFT	False	False
EFT_App_UR_Suf	UpwardRanking	True	EFT	False	True
EST_App_UR	UpwardRanking	True	EST	False	False
EST_App_UR_Suf	UpwardRanking	True	EST	False	True

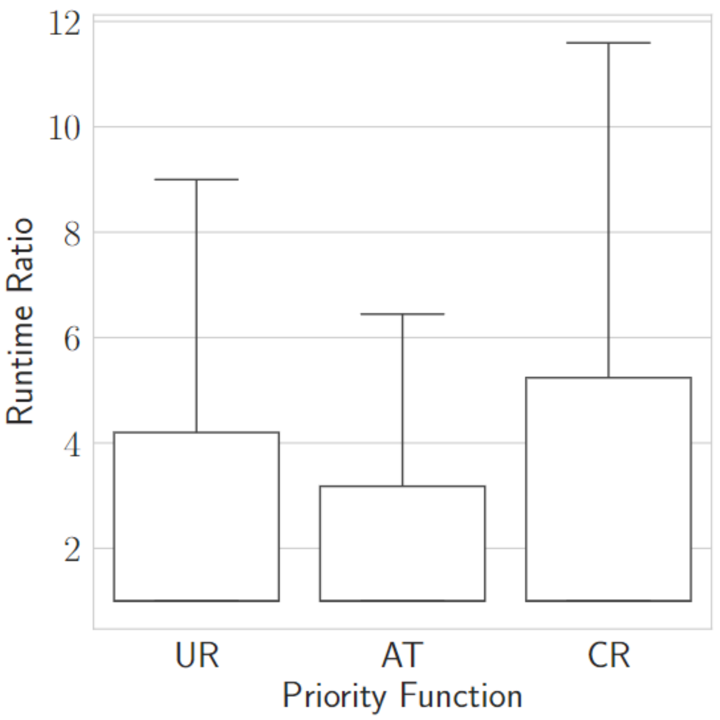

CCR=0.2, Dataset Type=chains
CCR=0.2, Dataset Type=cycles
CCR=0.2, Dataset Type=in_trees
CCR=0.2, Dataset Type=out_trees
CCR=0.5, Dataset Type=chains
CCR=0.5, Dataset Type=cycles
CCR=0.5, Dataset Type=in_trees
CCR=0.5, Dataset Type=out_trees
CCR=1.0, Dataset Type=chains
CCR=1.0, Dataset Type=cycles
CCR=1.0, Dataset Type=in_trees
CCR=1.0, Dataset Type=out_trees
CCR=2.0, Dataset Type=chains
CCR=2.0, Dataset Type=cycles
CCR=2.0, Dataset Type=in_trees
CCR=2.0, Dataset Type=out_trees
CCR=5.0, Dataset Type=chains
CCR=5.0, Dataset Type=cycles
CCR=5.0, Dataset Type=in_trees
CCR=5.0, Dataset Type=out_trees
Makespan Ratio
Runtime Ratio


Pareto Optimal Schedule Makespan Ratio Rank
Dataset
chains_ccr_0.2
chains_ccr_0.5
chains_ccr_1
chains_ccr_2
chains_ccr_5
cycles_ccr_0.2
cycles_ccr_0.5
cycles_ccr_1
cycles_ccr_2
cycles_ccr_5
in_trees_ccr_0.2
in_trees_ccr_0.5
in_trees_ccr_1
in_trees_ccr_2
in_trees_ccr_5
out_trees_ccr_0.2
out_trees_ccr_0.5
out_trees_ccr_1
out_trees_ccr_2
out_trees_ccr_5
Scheduler
EFT_App_CP_AT
EFT_App_CR_Suf
EFT_App_UR
EFT_App_UR_Suf
EFT_Ins_AT
EFT_Ins_AT_Suf
EFT_Ins_CR
EFT_Ins_CR_Suf
EFT_Ins_UR_Suf
EST_App_AT
EST_App_AT_Suf
EST_App_UR
EST_App_UR_Suf
EST_Ins_AT
EST_Ins_AT_Suf
EST_Ins_CP_AT
EST_Ins_CR
EST_Ins_CR_Suf
EST_Ins_UR
EST_Ins_UR_Suf
HEFT
MCT
MET
Sufferage
Order (Runtime Ratio)

EFFECTS OF INDIVIDUAL ALGORITHMIC COMPONENTS

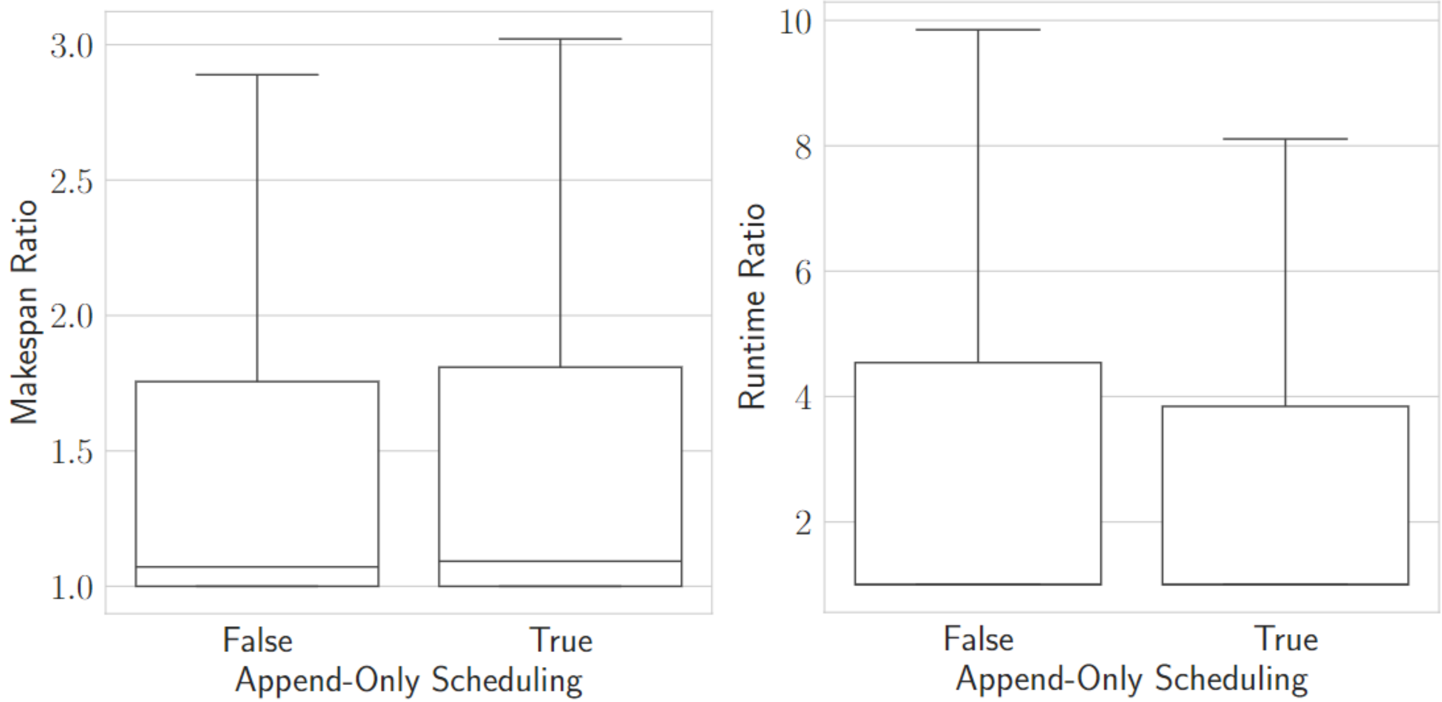
- Priority Function
  - UpwardRank (UR)
  - ArbitraryTopological (AT)
  - CPoPRank (CR)





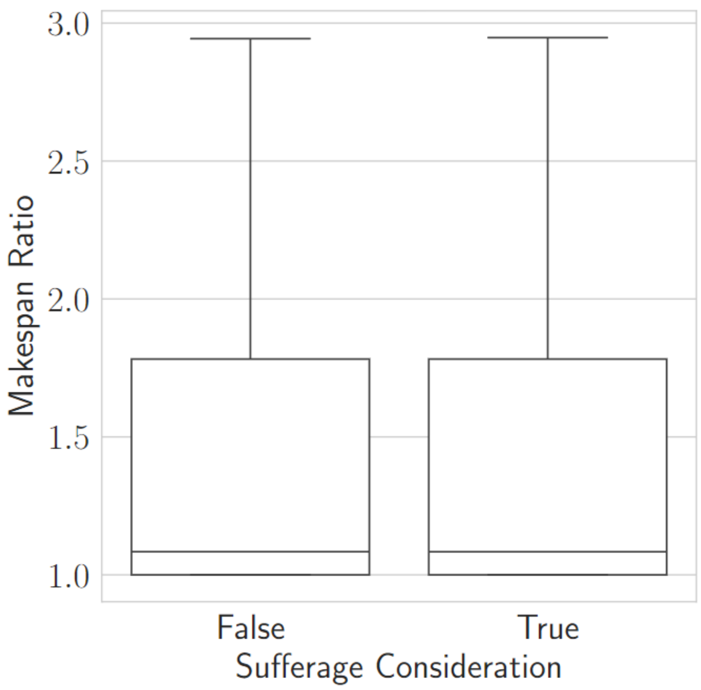
EFFECTS OF INDIVIDUAL ALGORITHMIC COMPONENTS

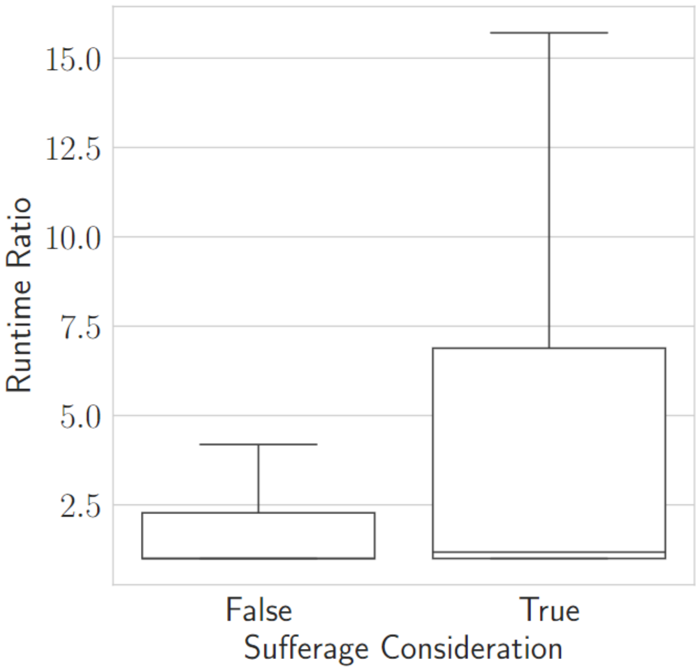
- Priority Function
- Append-only vs Insertion-based



EFFECTS OF INDIVIDUAL ALGORITHMIC COMPONENTS

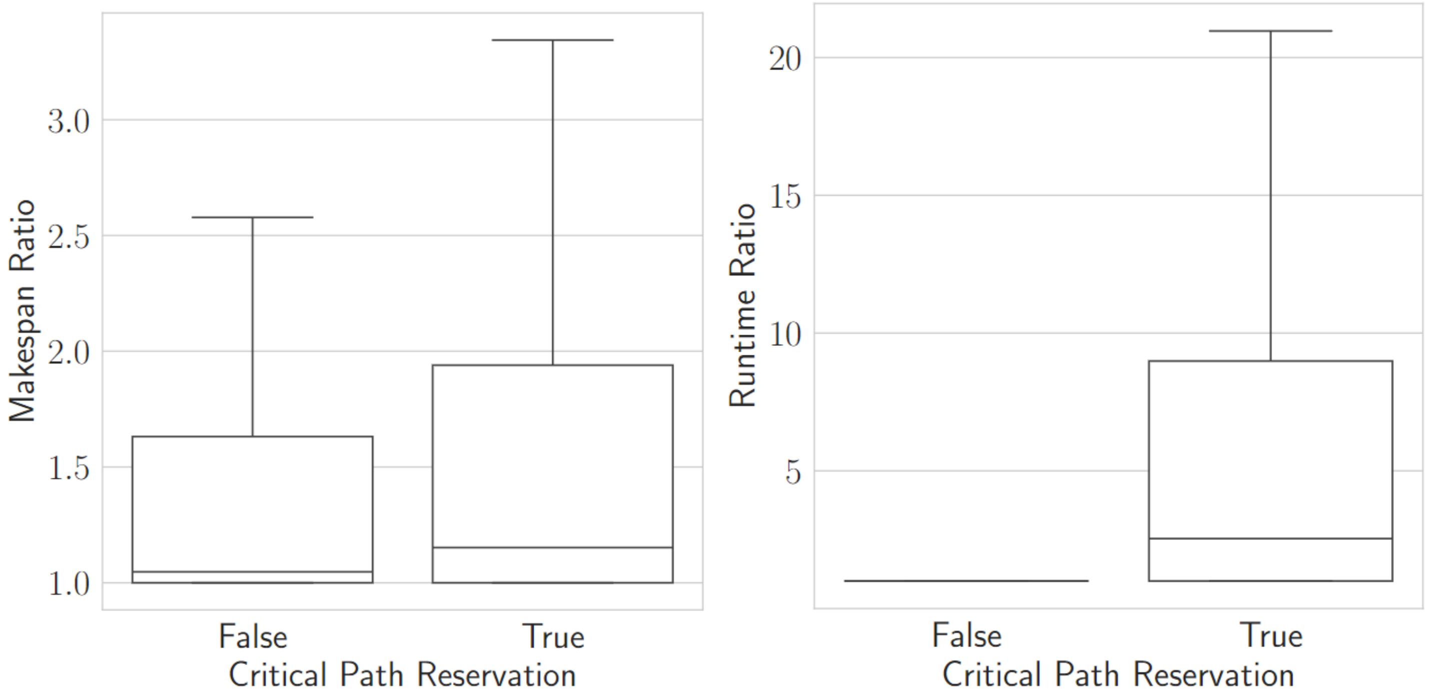
- Priority Function
- Append-only vs Insertion-based
- Sufferage





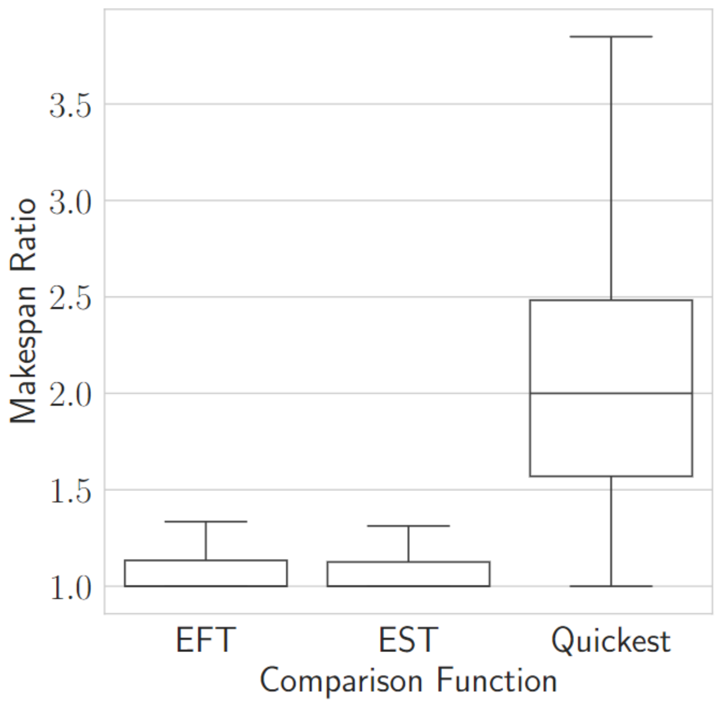
EFFECTS OF INDIVIDUAL ALGORITHMIC COMPONENTS

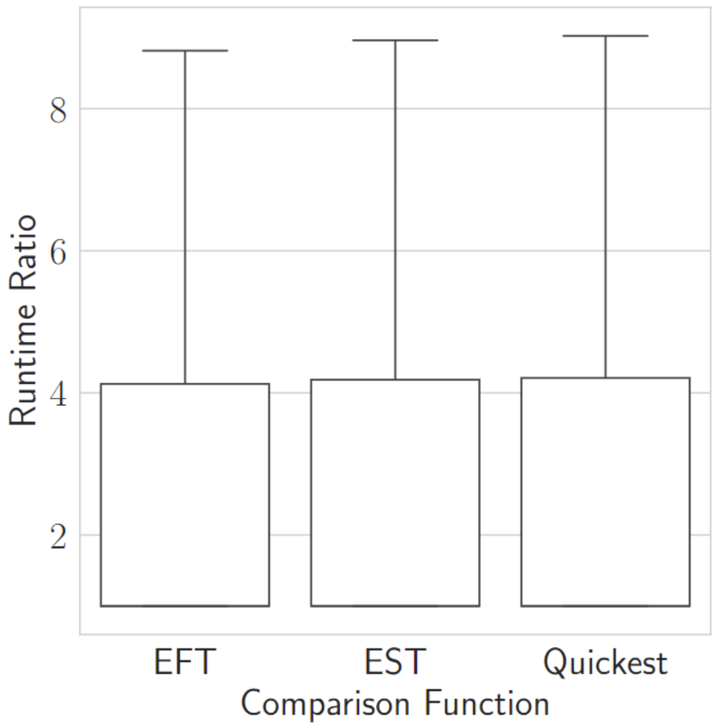
- Priority Function
- Append-only vs Insertion-based
- Sufferage
- Critical path reservation



EFFECTS OF INDIVIDUAL ALGORITHMIC COMPONENTS

- Priority Function
- Append-only vs Insertion-based
- Sufferage
- Critical path reservation
- Comparison Function





EFFECTS OF INDIVIDUAL ALGORITHMIC COMPONENTS

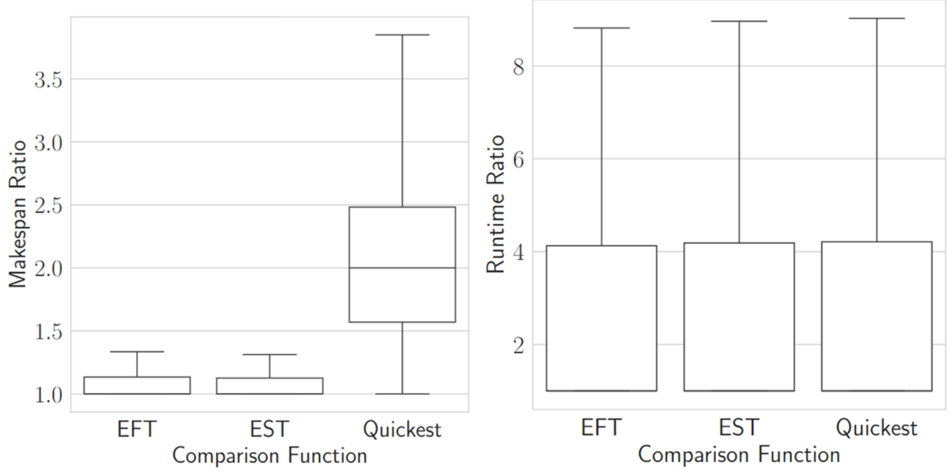
- Priority Function
- Append-only vs Insertion-based
- Sufferage
- Critical path reservation
- Comparison Function

component scheduler	initial_priority	append_only	compare	critical_path	sufferage
EFT_Ins_AT	ArbitraryTopological	False	EFT	False	False
EFT_Ins_AT_Suf	ArbitraryTopological	False	EFT	False	True
EST_Ins_AT	ArbitraryTopological	False	EST	False	False
EST_Ins_AT_Suf	ArbitraryTopological	False	EST	False	True
EST_Ins_CP_AT	ArbitraryTopological	False	EST	True	False
MCT [10]	ArbitraryTopological	True	EFT	False	False
Sufferage [12]	ArbitraryTopological	True	EFT	False	True
EFT_App_CP_AT	ArbitraryTopological	True	EFT	True	False
EST_App_AT	ArbitraryTopological	True	EST	False	False
EST_App_AT_Suf	ArbitraryTopological	True	EST	False	True
MET [10]	ArbitraryTopological	True	Quickest	False	False
EFT_Ins_CR	CPoPRanking	False	EFT	False	False
EFT_Ins_CR_Suf	CPoPRanking	False	EFT	False	True
EST_Ins_CR	CPoPRanking	False	EST	False	False
EST_Ins_CR_Suf	CPoPRanking	False	EST	False	True
EFT_App_CR_Suf	CPoPRanking	True	EFT	False	True
HEFT [6]	UpwardRanking	False	EFT	False	False
EFT_Ins_UR_Suf	UpwardRanking	False	EFT	False	True
EST_Ins_UR	UpwardRanking	False	EST	False	False
EST_Ins_UR_Suf	UpwardRanking	False	EST	False	True
EFT_App_UR	UpwardRanking	True	EFT	False	False
EFT_App_UR_Suf	UpwardRanking	True	EFT	False	True
EST_App_UR	UpwardRanking	True	EST	False	False
EST_App_UR_Suf	UpwardRanking	True	EST	False	True

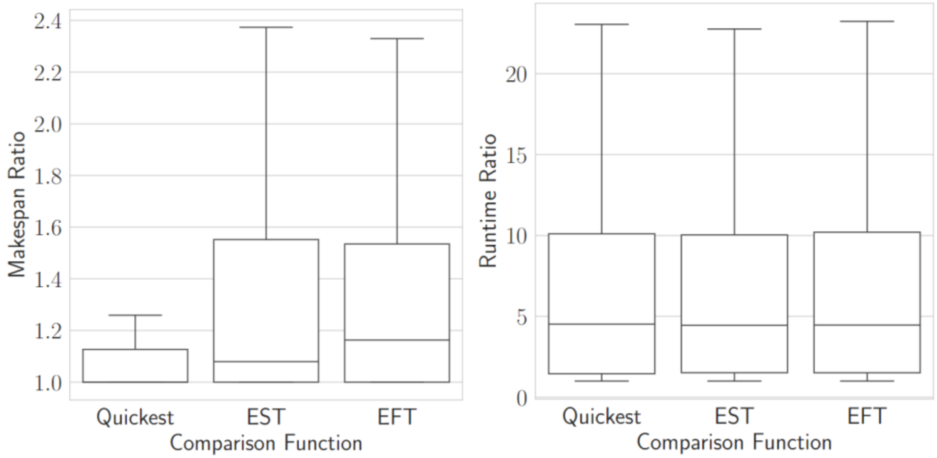
EFFECTS OF INDIVIDUAL ALGORITHMIC COMPONENTS

- » Priority Function
- » Append-only vs Insertion-based
- » Sufferage
- » Critical path reservation
- » Comparison Function
  - » Results look different for specific datasets!

Overall

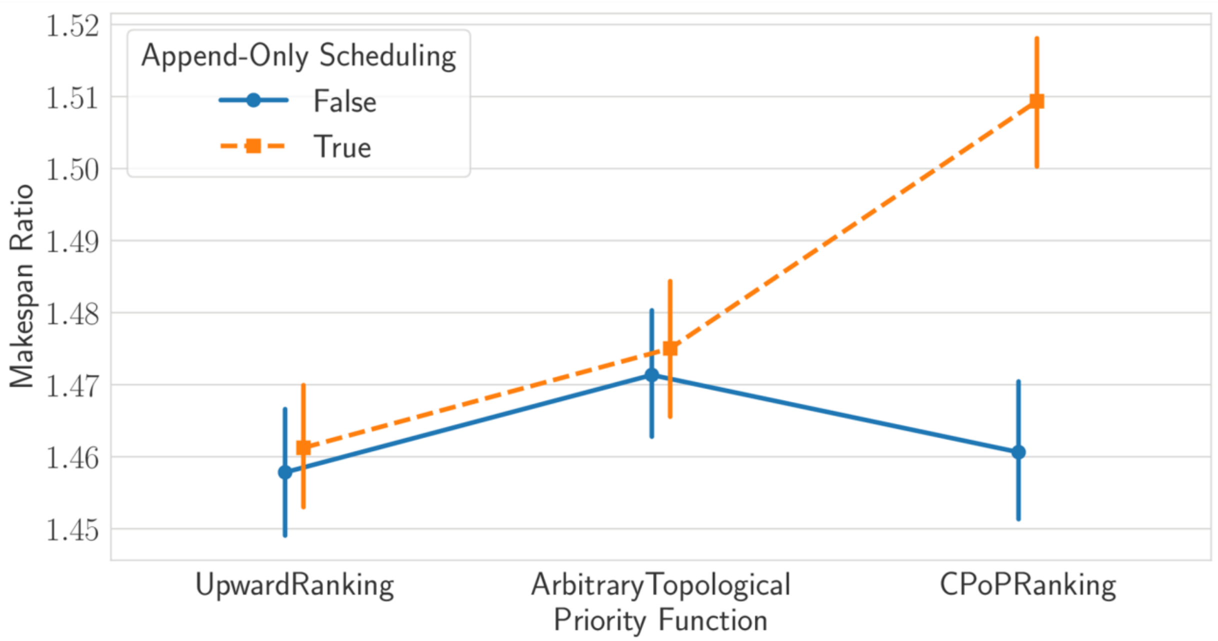


cycles dataset w/ CCR=5



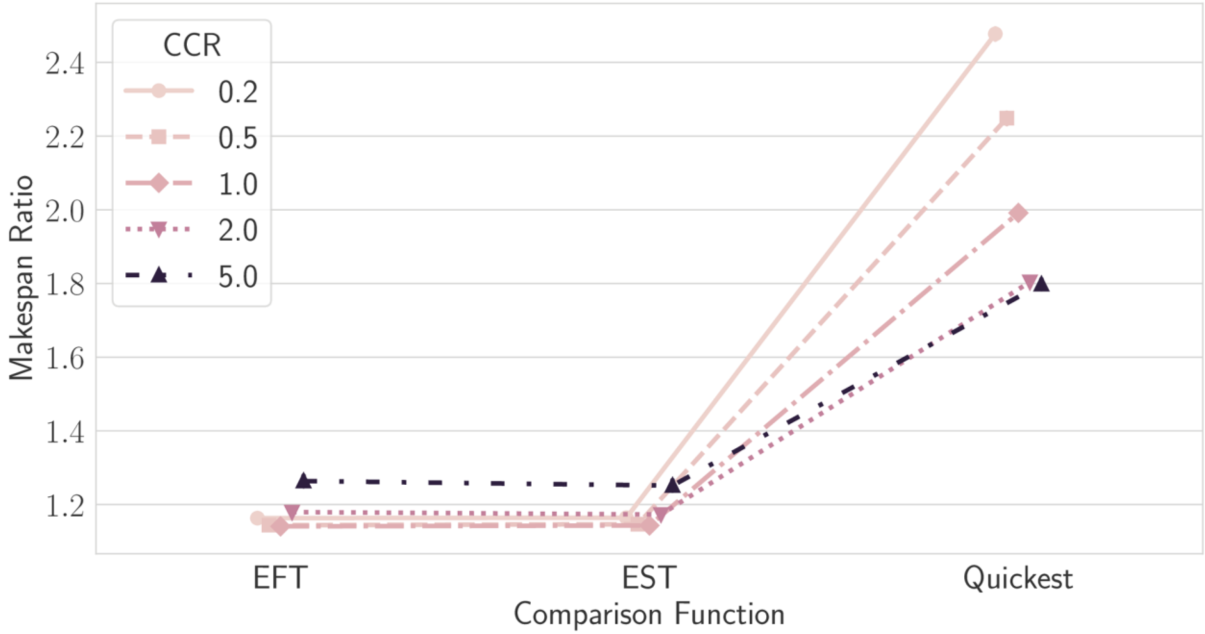
INTERACTIONS

- Append-only Strategy and Priority Function
  - Append-only strategy is *particularly* bad with CPoP ranking



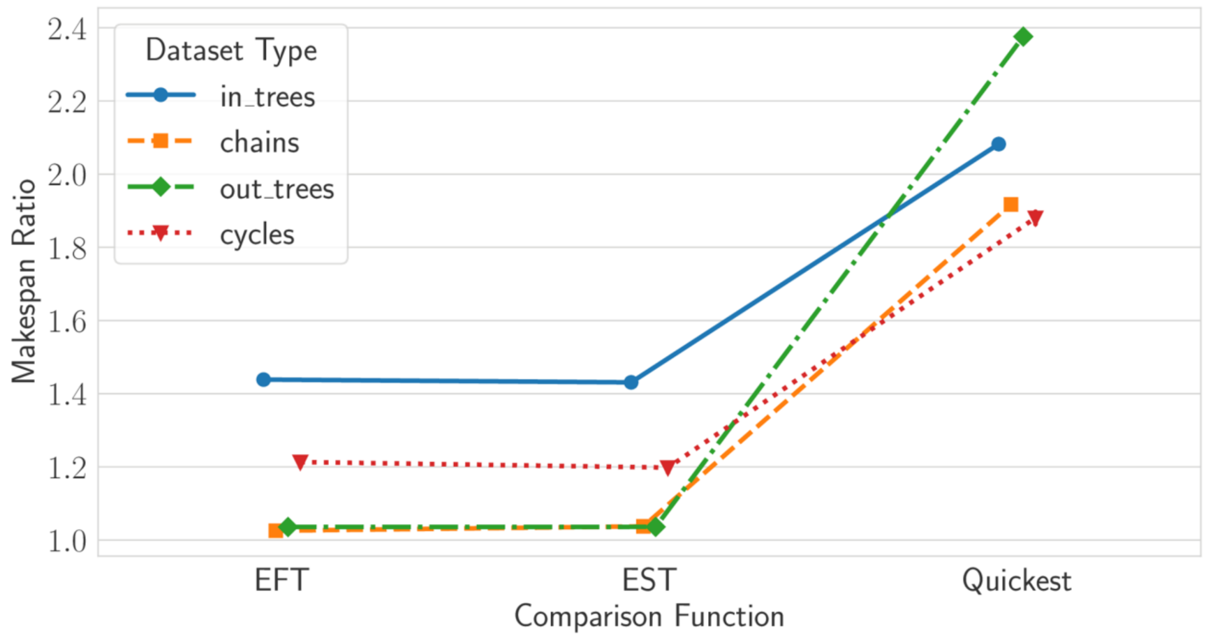
INTERACTIONS

- Append-only Strategy and Priority Function
  - Append-only strategy is *particularly* bad with CPoP ranking
- CCR and Comparison Function
  - Quickest strategy is particularly bad for compute-heavy task graphs



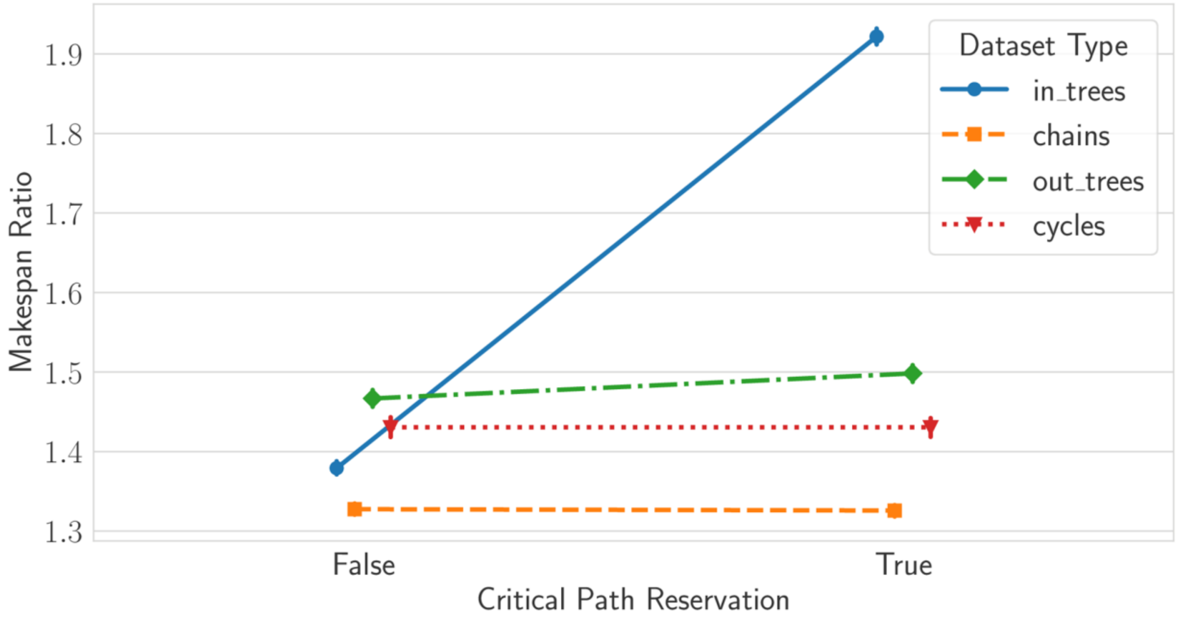
INTERACTIONS

- Append-only Strategy and Priority Function
  - Append-only strategy is *particularly* bad with CPoP ranking
- CCR and Comparison Function
  - Quickest strategy is particularly bad for compute-heavy task graphs
- Dataset Type and Comparison Function
  - The quickest strategy is particularly bad for the out-tree task graph structure



INTERACTIONS

- Append-only Strategy and Priority Function
  - Append-only strategy is *particularly* bad with CPoP ranking
- CCR and Comparison Function
  - Quickest strategy is particularly bad for compute-heavy task graphs
- Dataset Type and Comparison Function
  - The quickest strategy is particularly bad for the out-tree task graph structure
- Dataset Type and Critical Path
  - Critical path reservation is particularly bad for in-tree task graph structures



CONCLUSION

- **Generalized Parametric Scheduler:** Proposed a modular list-scheduling algorithm that enables flexible combination of five key algorithmic components.
- **Benchmarking**: Evaluated **72 unique schedulers** generated from all component combinations on **20 datasets**, encompassing diverse graph structures and CCRs.
- **Component-level Insights**: Analyzed both **individual** and **combined effects** of components on makespan and runtime across datasets.
- **Dataset-specific Behavior**: Revealed how component effectiveness is **problem-dependent**, highlighting complex interactions between components and dataset characteristics.
- **Future Work**:
  - Implement new algorithmic components (e.g., k-depth lookahead)
  - Incorporate additional datasets
  - Leverage findings to design adaptive or hybrid schedulers tuned to application characteristics

SCAN ME

ACKNOWLEDGEMENTS

- This work was supported in part by Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196.
- The authors acknowledge the Center for Advanced Research Computing (CARC) at the University of Southern California for providing computing resources that have contributed to the research results reported within this publication. URL: https://carc.usc.edu

SCAN ME