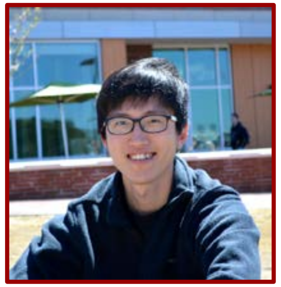
A Comparison of the Quality of Data-driven Programming Hint Generation Algorithms

Thomas W. Price[1] Yihuan Dong[1] Rui Zhi[1] Benjamin Paaßen[2] Nicholas Lytle[1] Veronica Cateté[1] Tiffany Barnes[1]

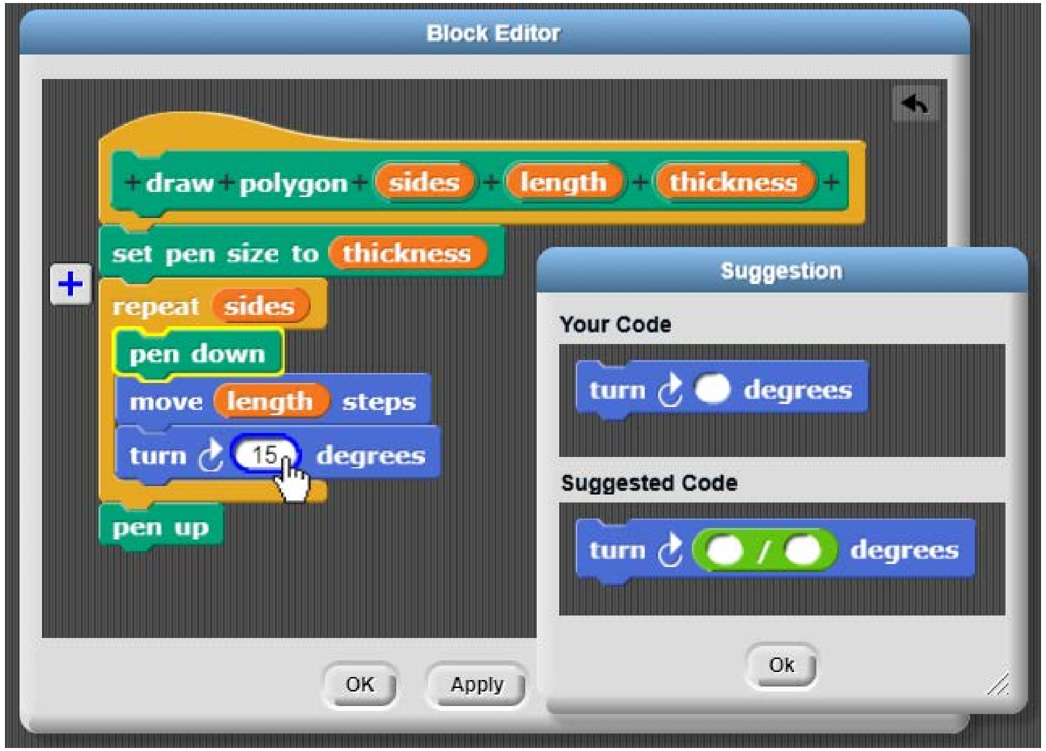






[1]North Carolina State University [2]Bielefeld University

June 27th, 2019 - AIED

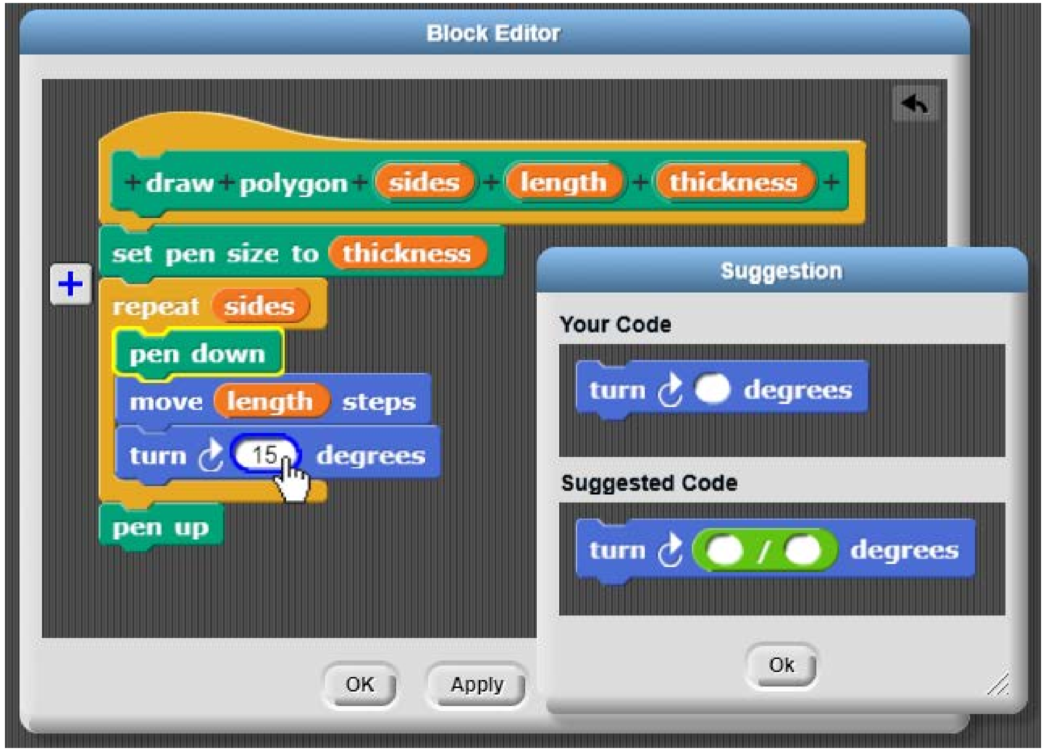
Programming Hints



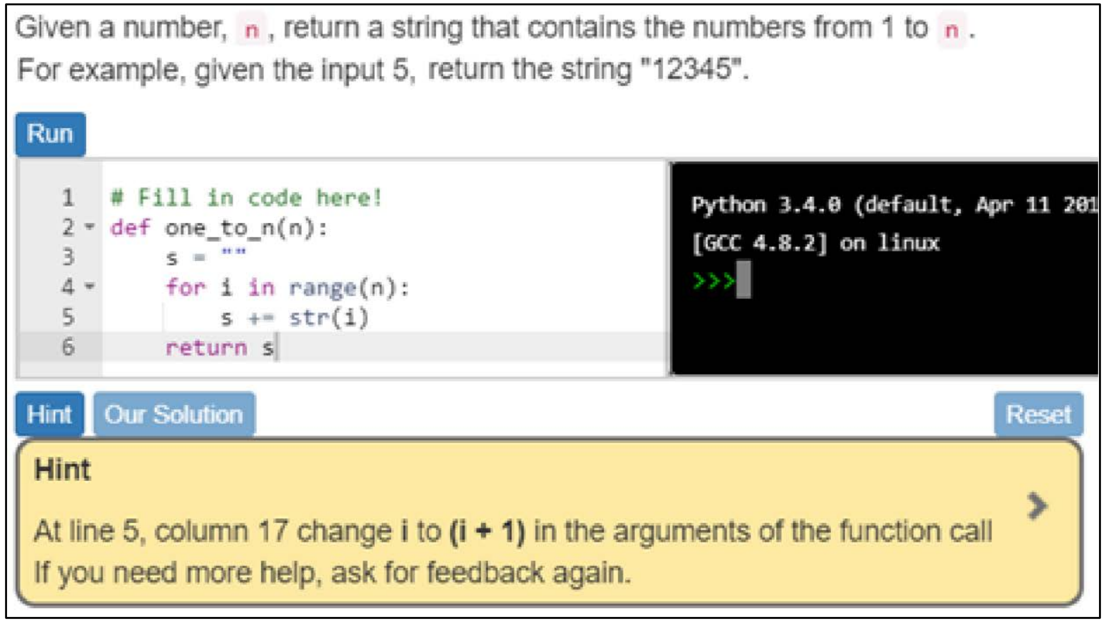
iSnap (Price 2017)

1. **On-demand**
2. **Next-step, edit-based**
3. **Data-driven**

Programming Hints



iSnap (Price 2017)



ITAP (Rivers 2017)

Programming Hints

In the dom

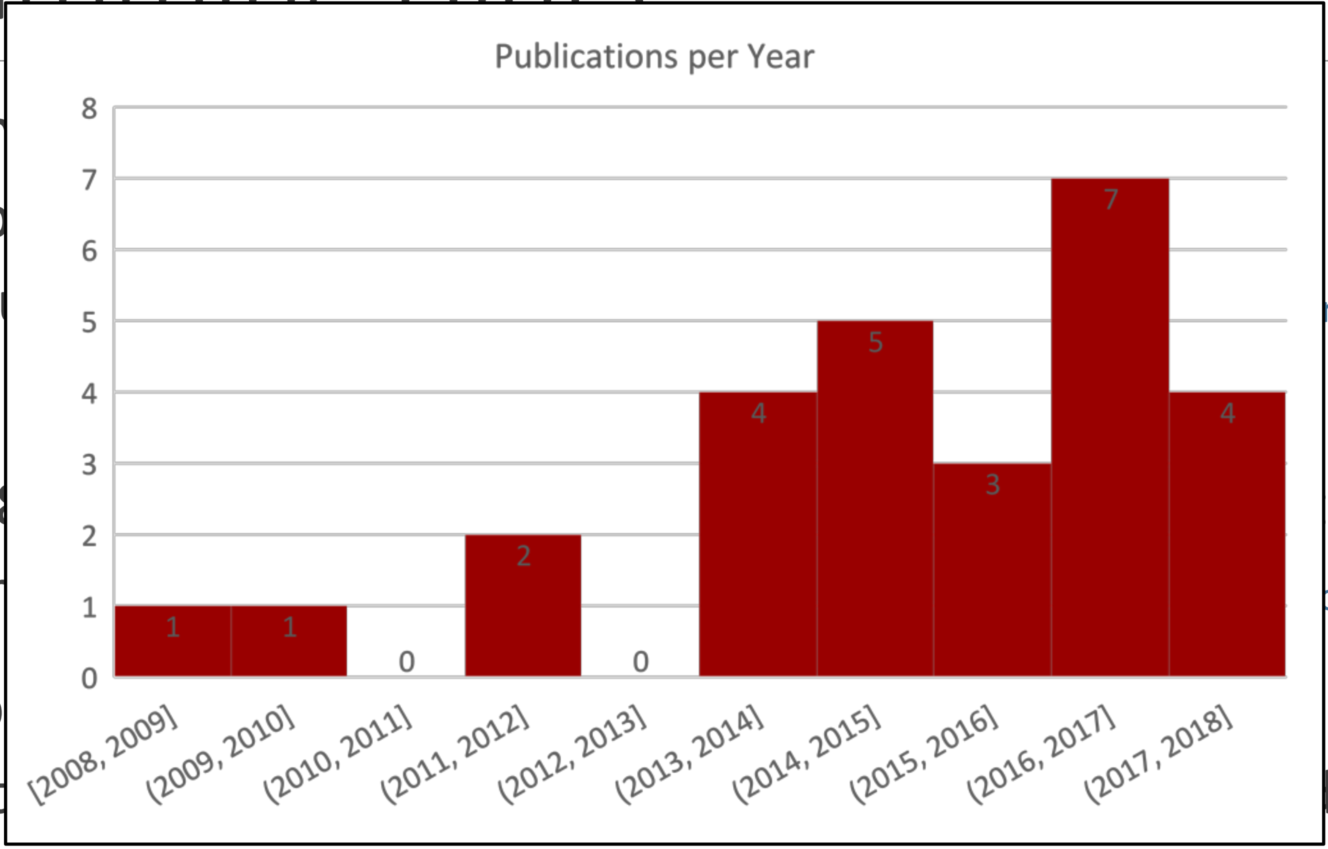
- Improve p
- Improve fu
rwan 2019, forthcoming)

Data-drive ptive

- Since 2008 s
- Evaluation
ord 2014; Rivers 2017)

Not all pro

- The qualit lerably
- Even one low-quality hint can deter students from requesting future hints



Proposed Contributions

1. **Methods**: QUALITYSCORE: A procedure for comparing the quality of hint generation approaches, that is *validated* and *reusable*
2. **Results:**
   a) An evaluation of *six* hint generation algorithms on *multiple datasets* and multiple programming languages.
   b) Insight into current strengths and limitations of these algorithms.
3. **Data**: All data and code needed to rate a new algorithm available at: go.ncsu.edu/hint-quality-data

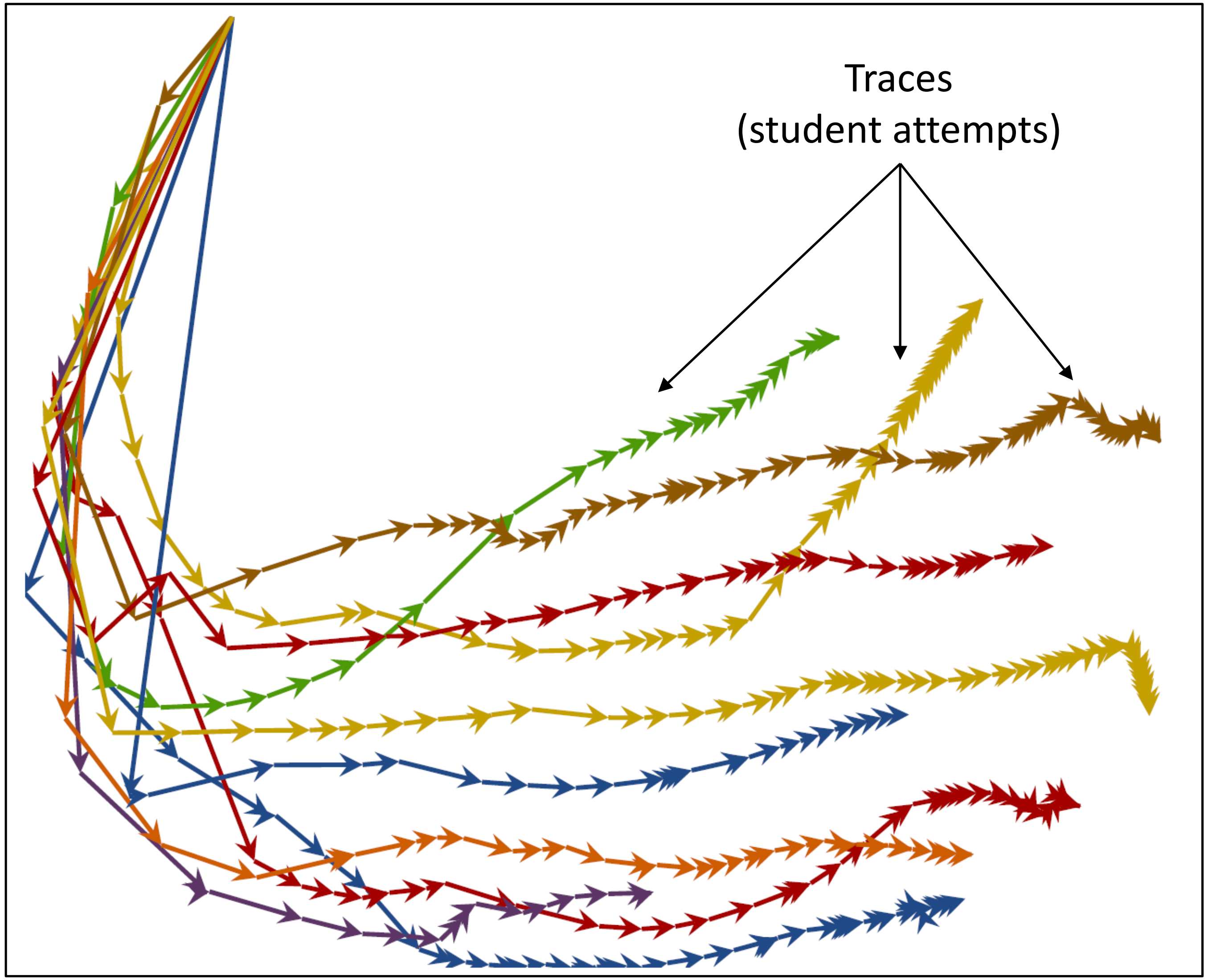
Data-Driven Hints Generation Algorithms

OVERVIEW OF THE SIX ALGORITHMS COMPARED

Data-driven Hint Generation

Inputs:

- Correct Solutions (training data)



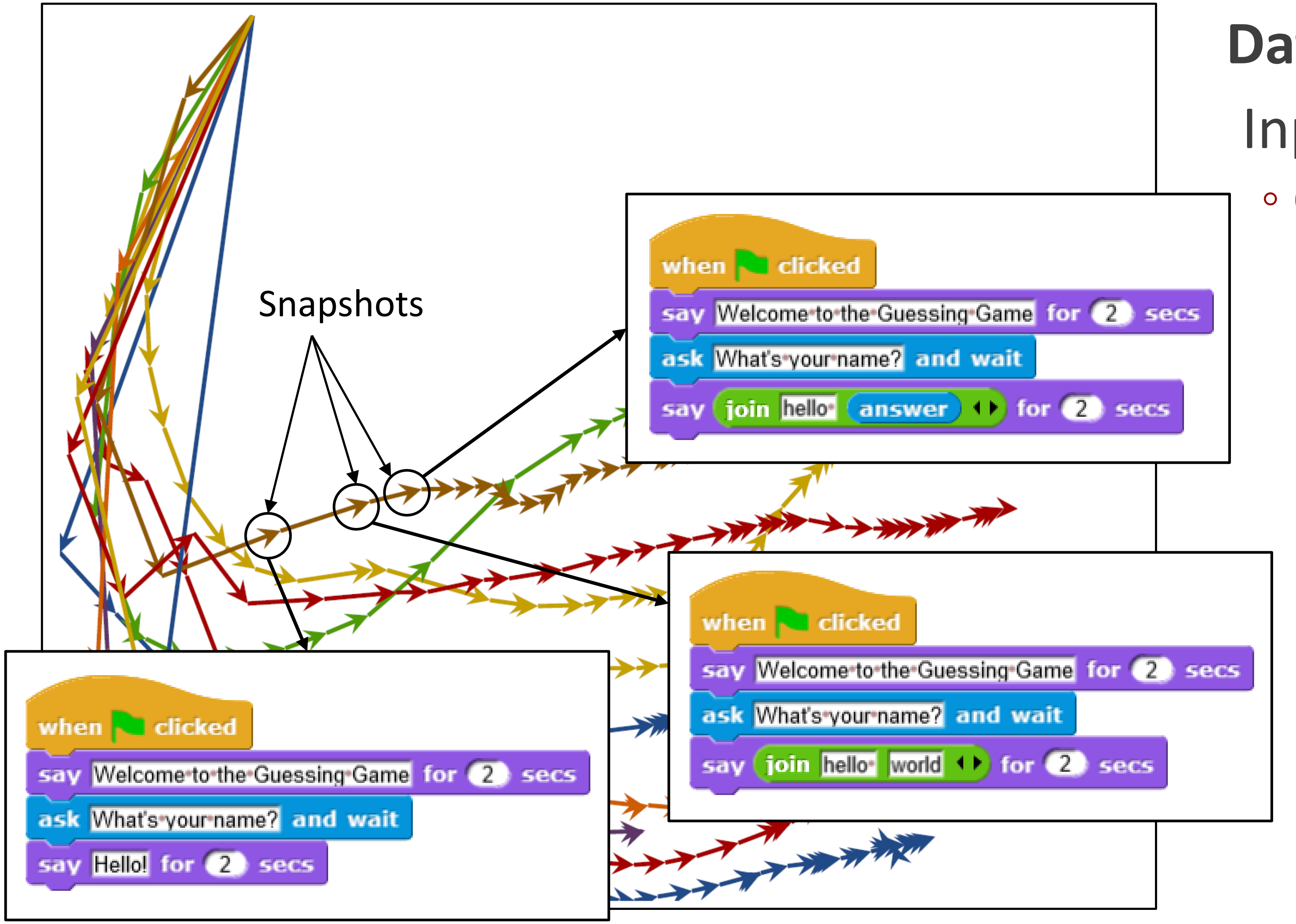
Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

Data-driven Hint Generation

Inputs:

- Correct Solutions (training data)



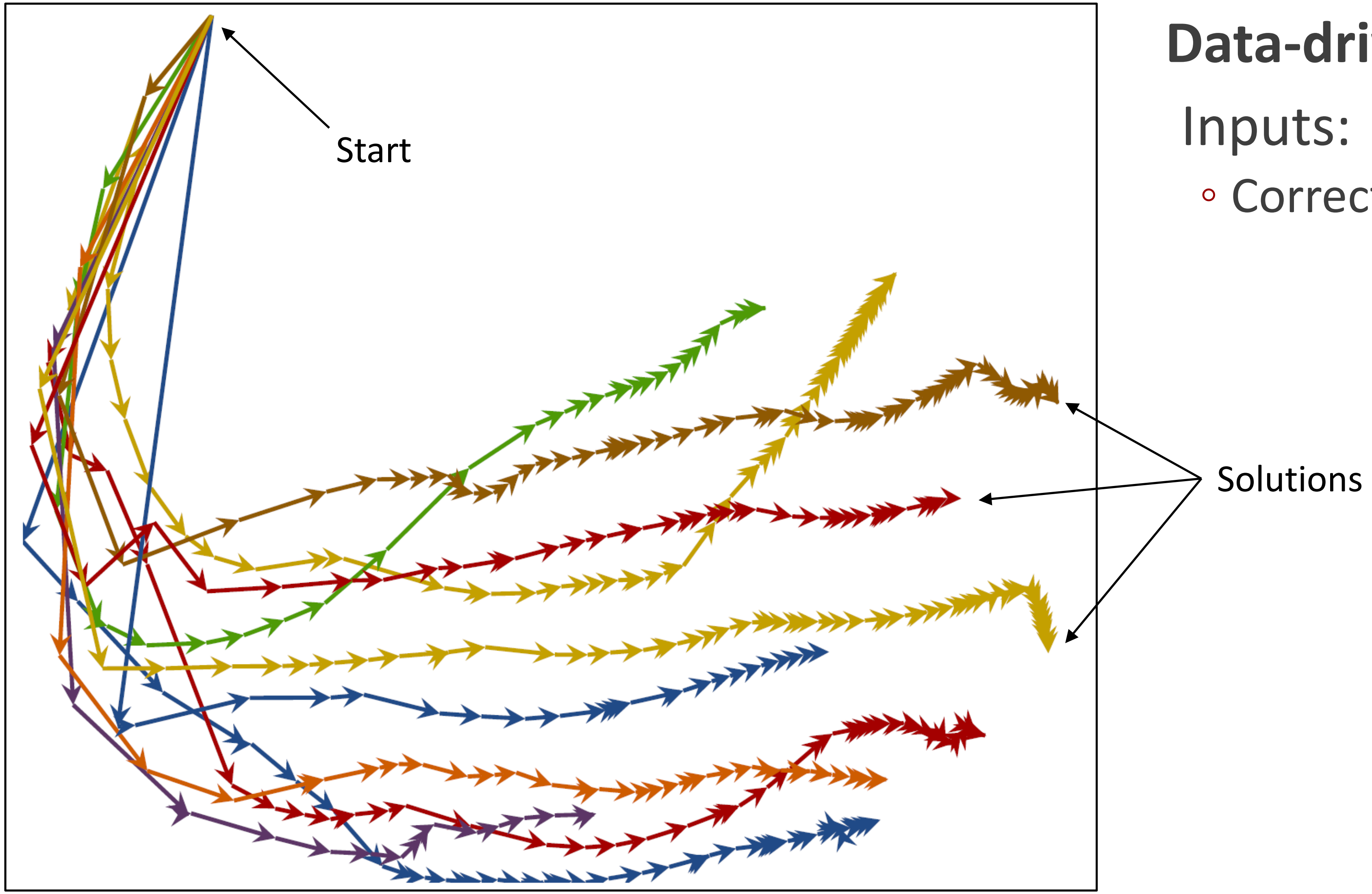
Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

Data-driven Hint Generation

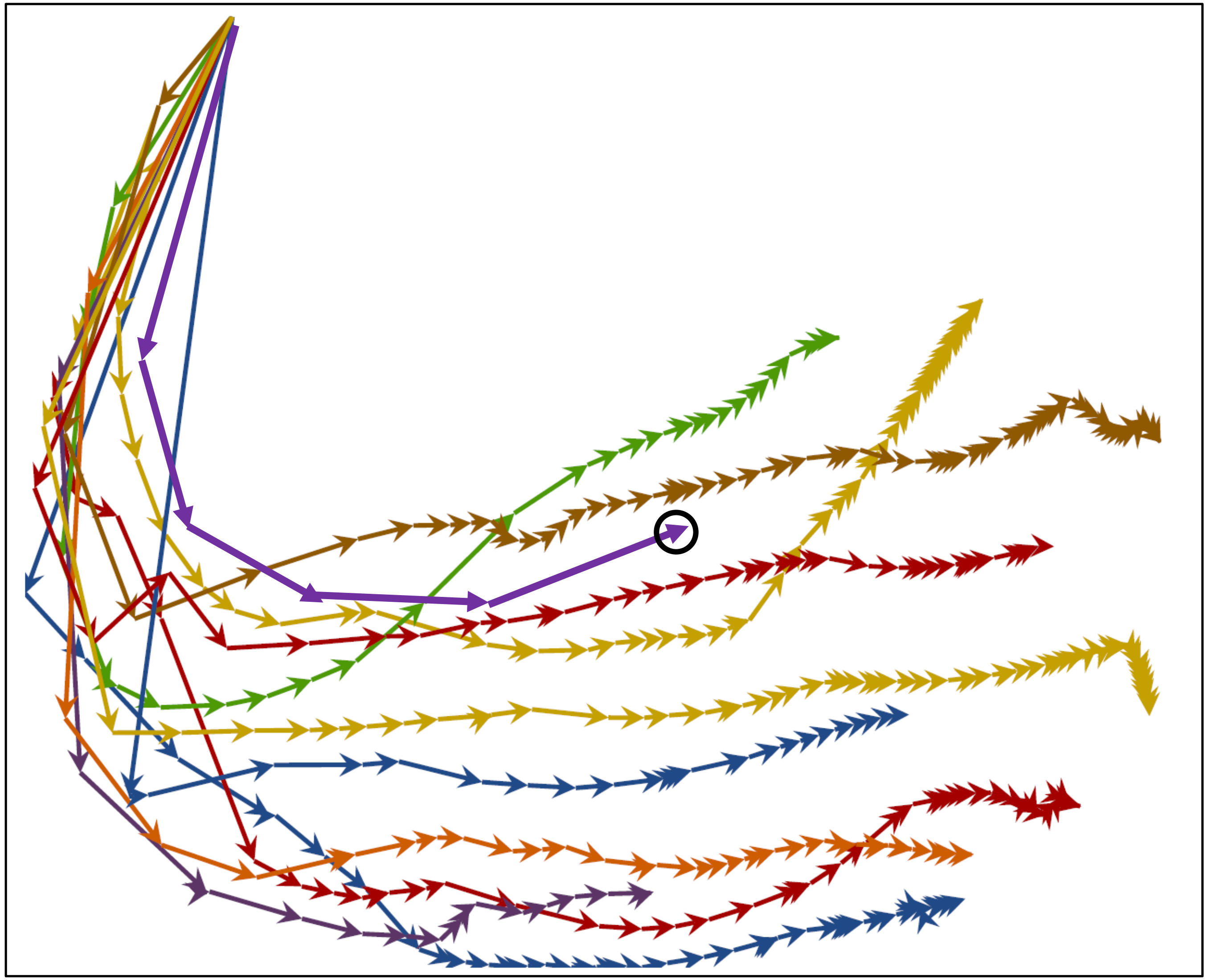
Inputs:

- Correct Solutions (training data)



Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)



Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

Data-driven Hint Generation

Inputs:

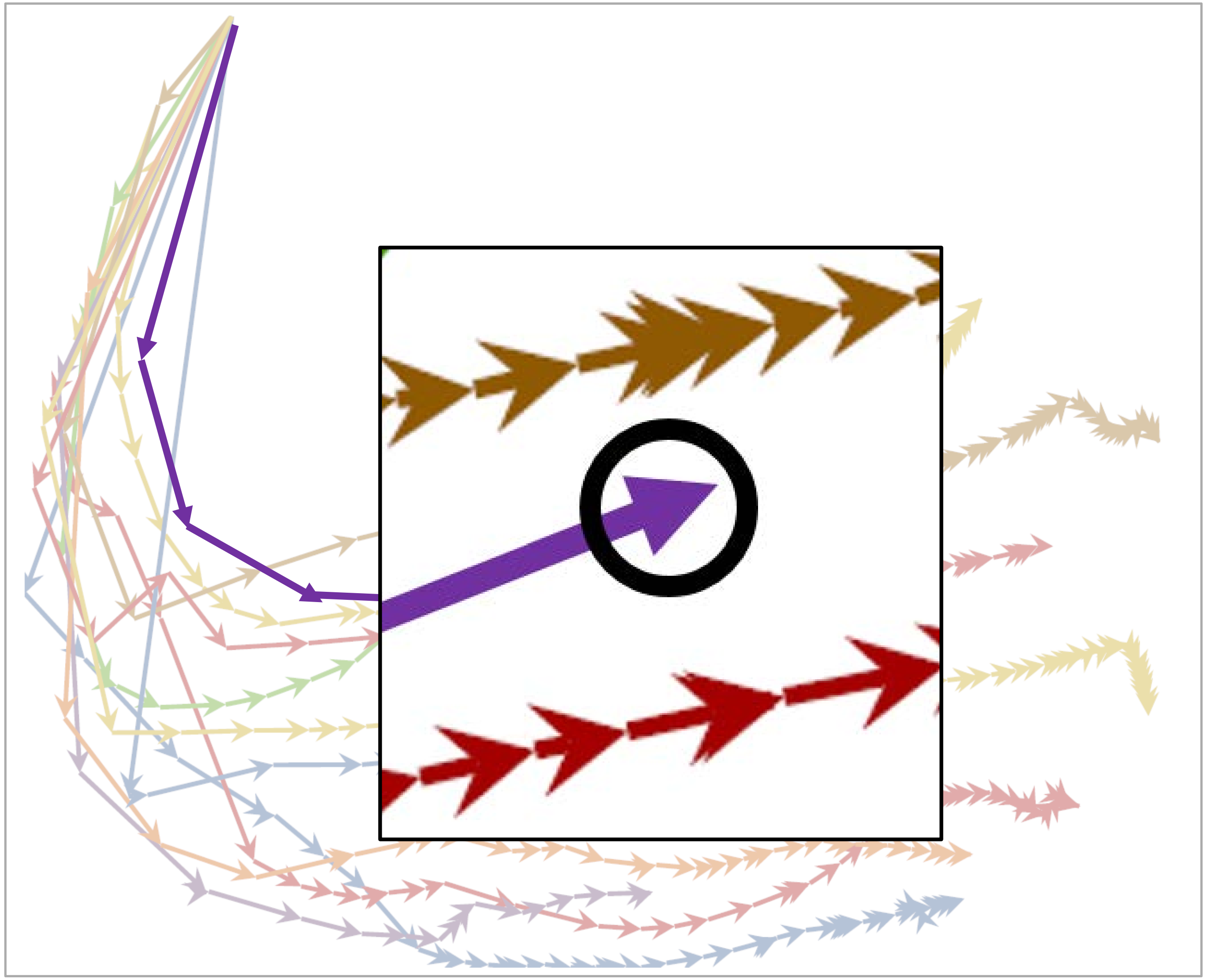
- Correct Solutions (training data)
- Hint Request (purple)

Outputs:

- Next suggested snapshot/edit

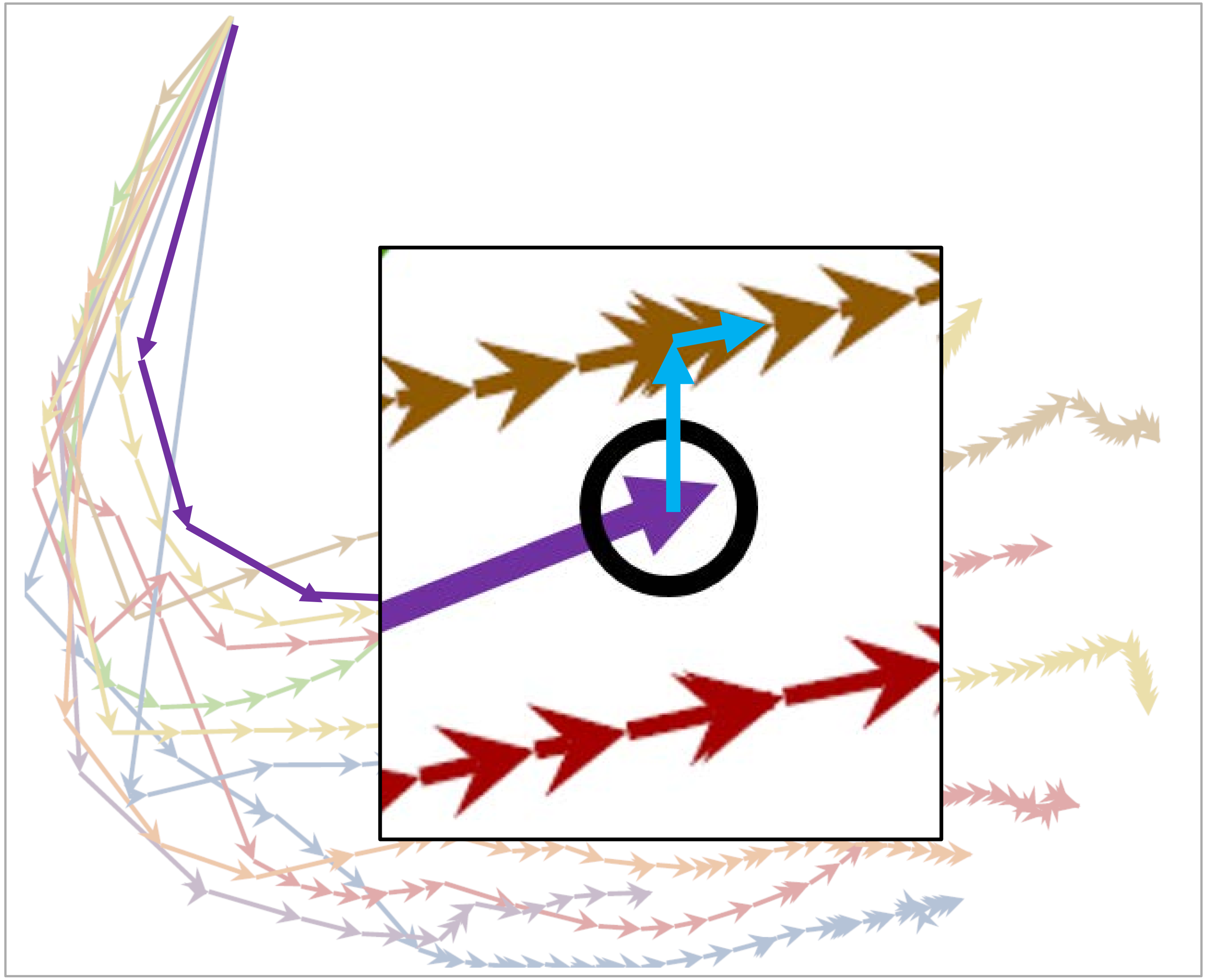
Graph-based Approaches:

- Follow prior students' paths to a solution



Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

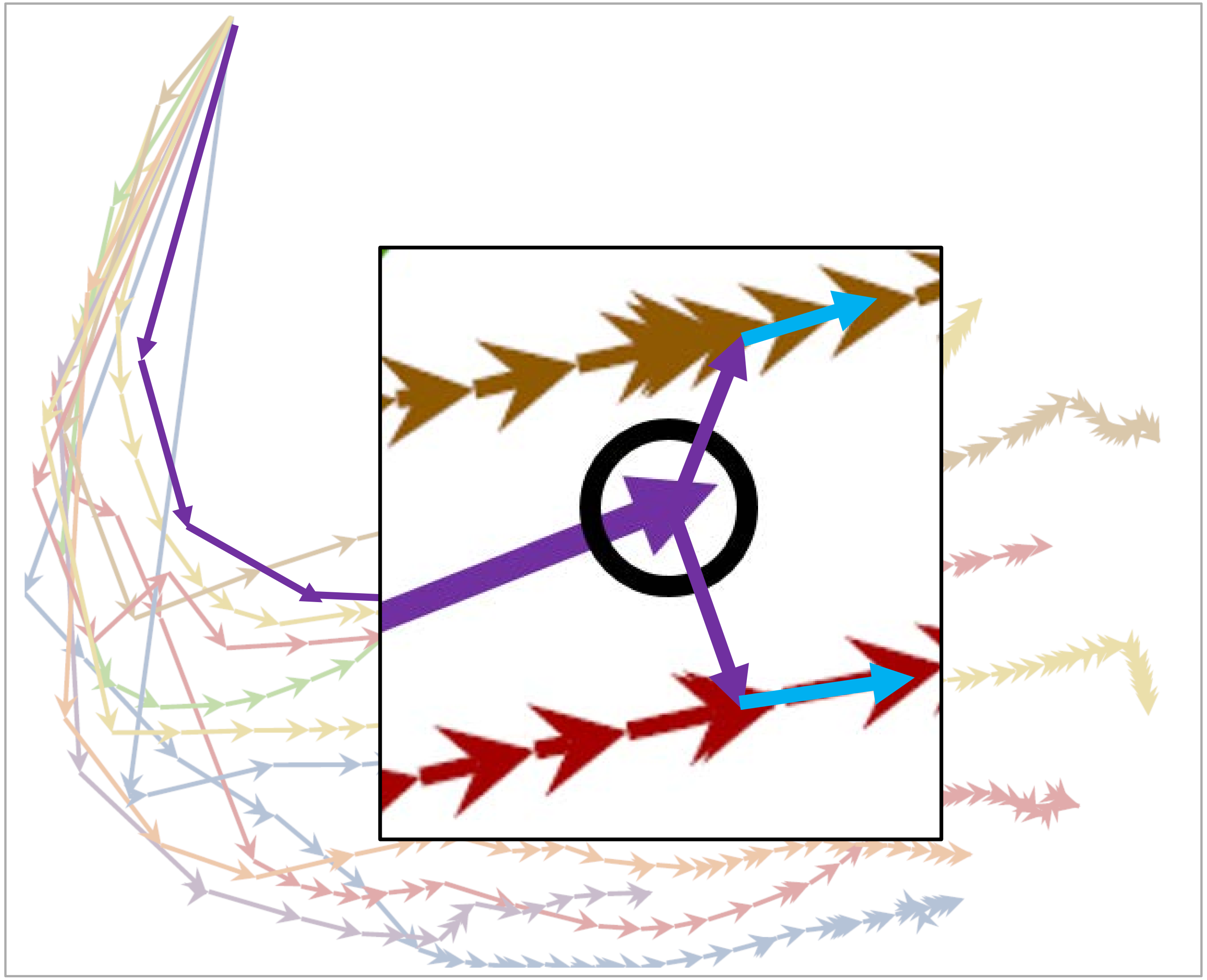


Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

Graph-based Approaches:

1. NSNLS: Next Step of Nearest Learner Solution (Gross 2014)
   a) Find the closest partial student solution
   b) Suggest the next step

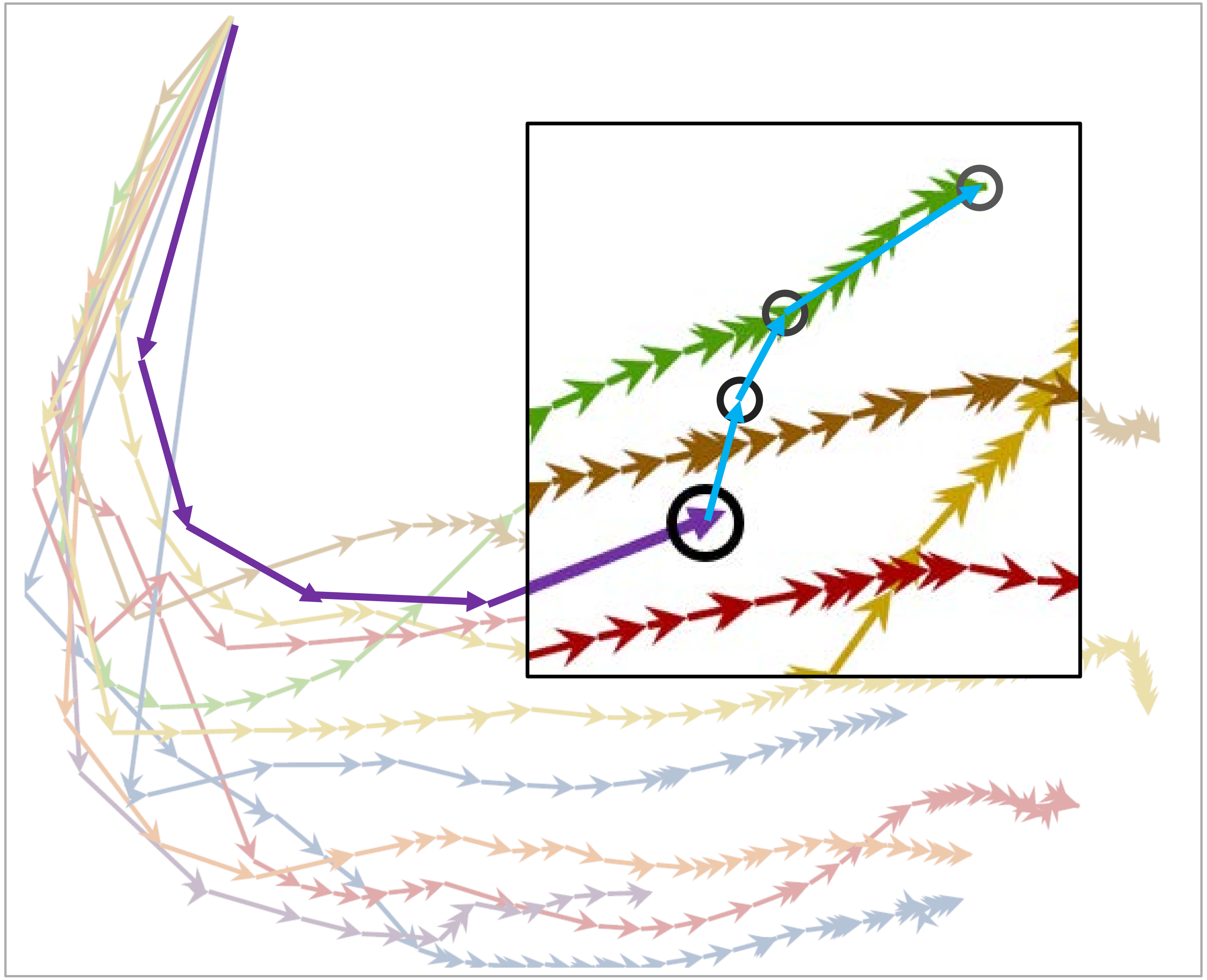


Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

Graph-based Approaches:

1. NSNLS (Gross 2014)
2. CTD: Contextual Tree Decomposition (Price 2016)
   a) Decompose the source code into *subtrees*
      - E.g. All code inside a given if-statement
   b) For each subtree, construct the solution space; suggest an edit

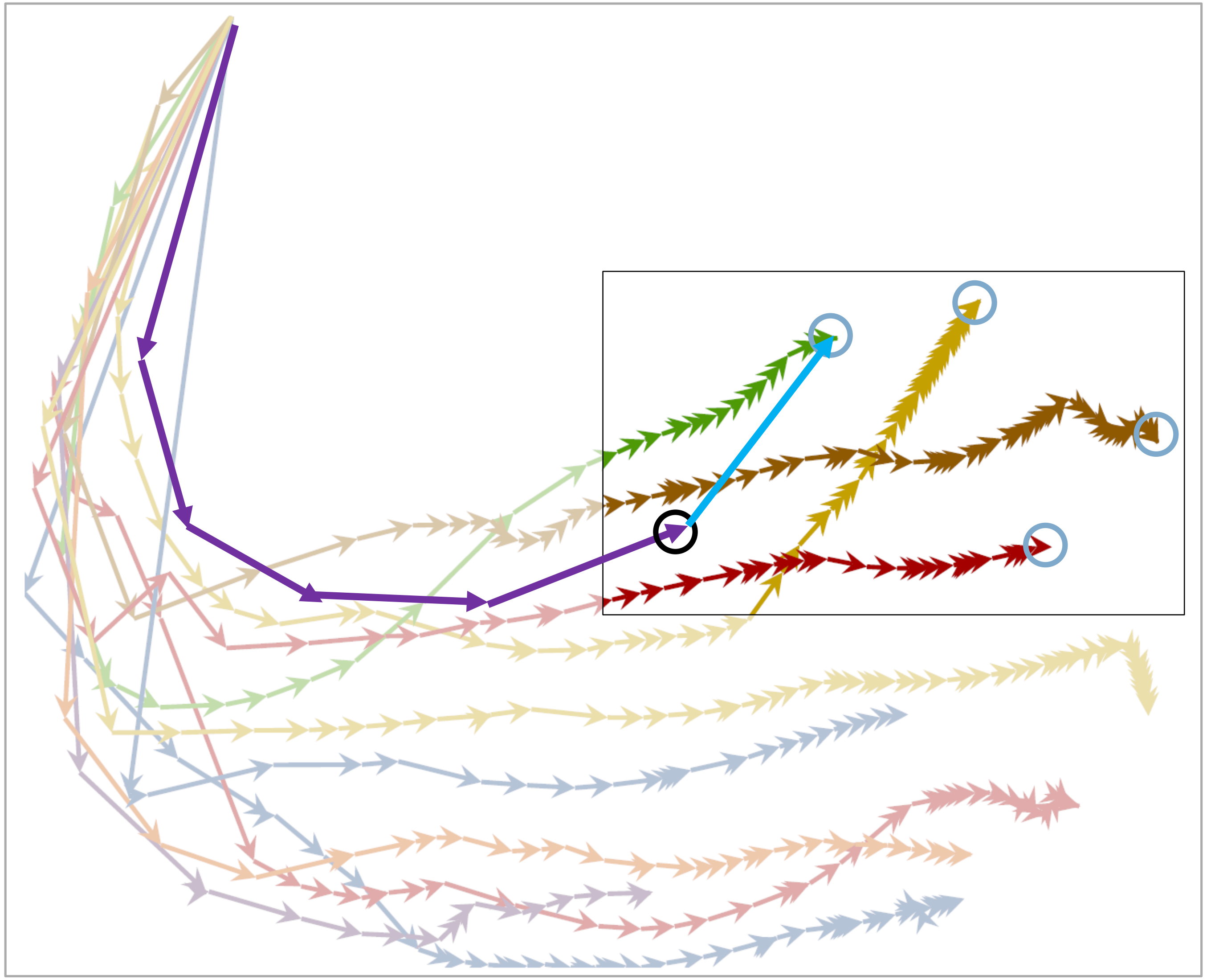


Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

Graph-based Approaches:

1. NSNLS (Gross 2014)
2. CTD (Price 2016)
3. ITAP (Rivers 2017)
   a) Identify the closest solution
   b) Select a target state
   c) Suggest a single edit



Solution Space (one problem)

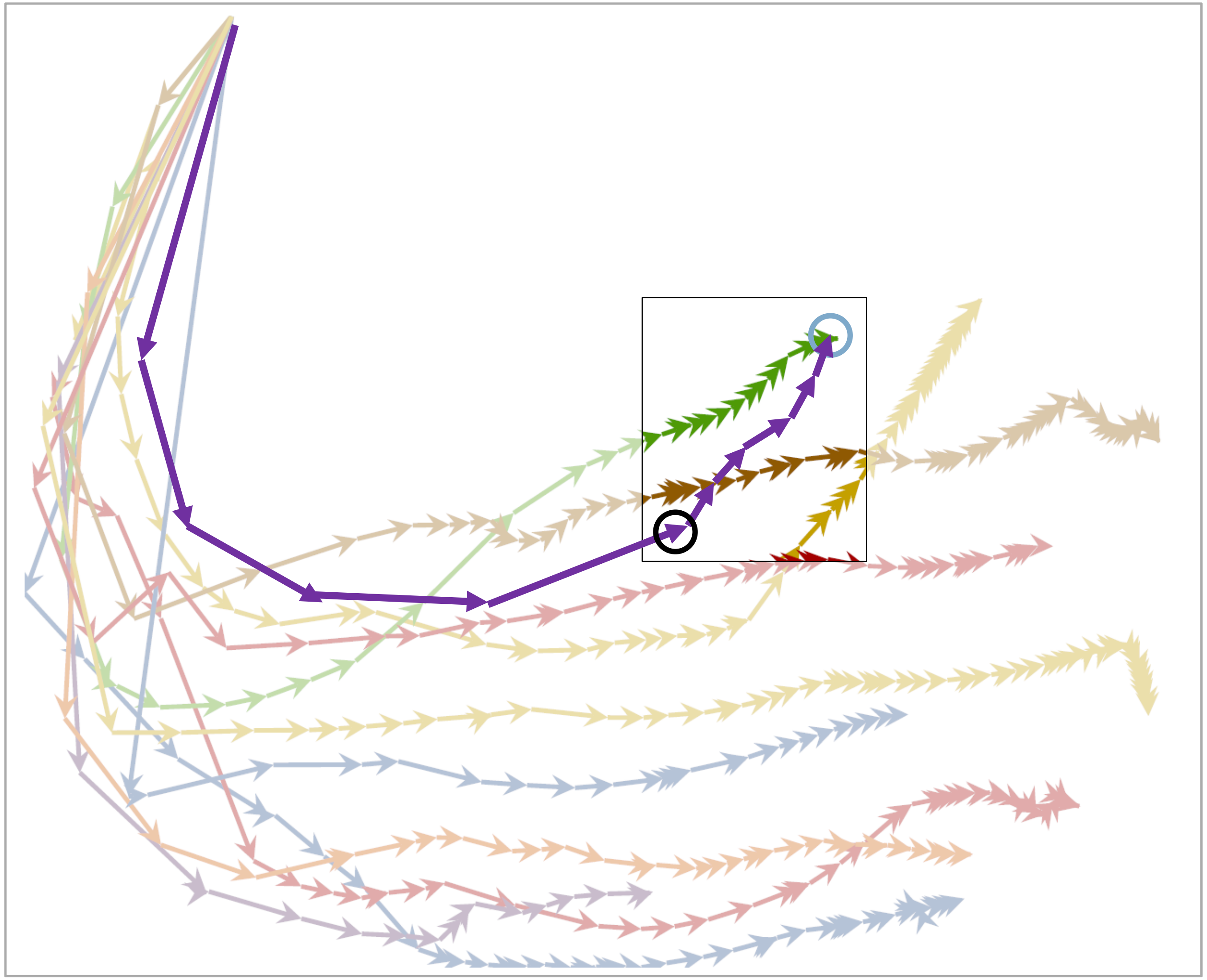
T-SNE embedding of iSnap data (Paaßen 2018)

Graph-based Approaches:

1. NSNLS (Gross 2014)
2. CTD (Price 2016)
3. ITAP (Rivers 2017)

Solution-based Approaches:

4. TR-ER (Zimmerman 2015)
5. SourceCheck (Price 2017)
   a) Identify the closest solution
   b) Suggest edits to get closer to that solution



Solution Space (one problem)

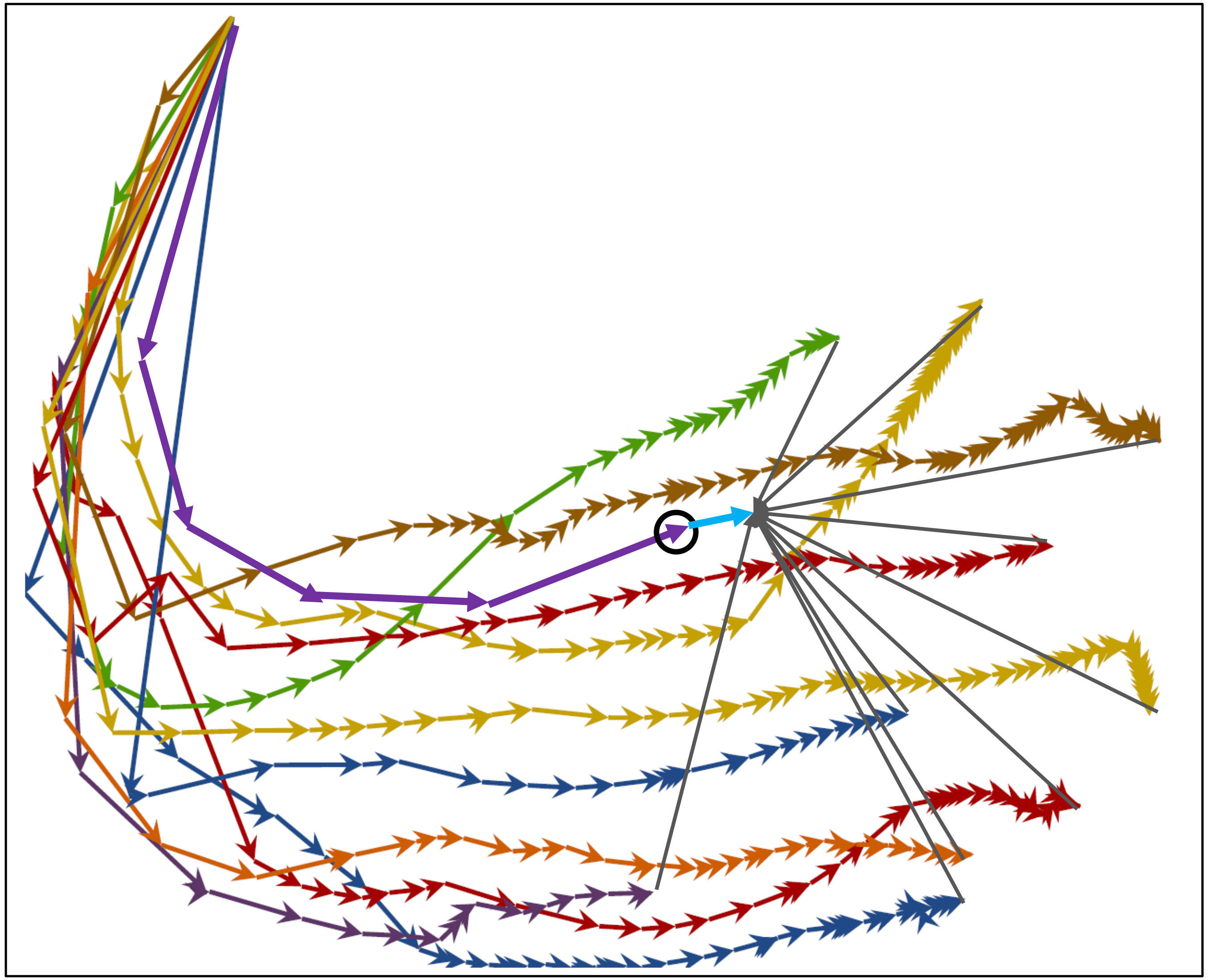
T-SNE embedding of iSnap data (Paaßen 2018)

Graph-based Approaches:

1. NSNLS (Gross 2014)
2. CTD (Price 2016)
3. ITAP (Rivers 2017)

Solution-based Approaches:

4. TR-ER (Zimmerman 2015)
5. SourceCheck (Price 2017)
   a) Identify the closest solution
   b) Suggest edits to get closer to that solution



Solution Space (one problem)

T-SNE embedding of iSnap data (Paaßen 2018)

Graph-based Approaches:

1. NSNLS (Gross 2014)
2. CTD (Price 2015)
3. ITAP (Rivers 2017)

Solution-based Approaches:

4. TR-ER (Zimmerman 2015)
5. SourceCheck (Price 2016)

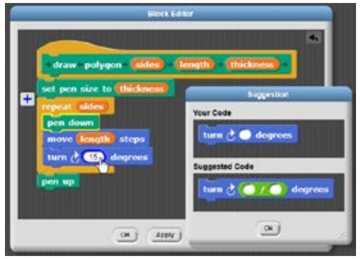
Machine Learning Approaches:

6. Continuous Hint Factory (Paaßen 2018)
   a. Predicts how successful students *would* edit their code

Method: QualityScore

REUSABLE QUALITY METRIC FOR DATA-DRIVEN HINT GENERATION

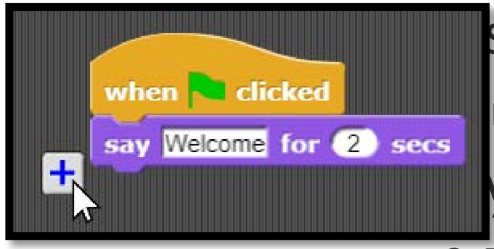
Data



```
def isWeekend(day):
    return bool(day=='sunday'
     or day=='saturday')
```

iSnap (Price 2017)

- Novice programming environment
- On-demand data-driven hints
- 120 non-CS majors
  - Fall 2016 and Spring 2017
- 2 iSnap assignments
  - 10-13 lines of code
  - Loops, conditionals, variables, procedures
- Extracted 47 hint requests
  - One per student per problem
  - 23-24 per problem



ITAP (Rivers 2017)

- ITS for Python programming
- On-demand data-driven hints
- students in introductory CS
  - ring 2017
- ython assignments
  - 2-5 lines of code
  - Loops, variables, string operations, arithmetic
- Extracted 51 hint requests
  - Up to two per student per problem
  - 7-14 per problem

QUALITYSCORE Calculation

1. 3 tutors independently generated Gold Standard hints for each hint request (e.g. Piech 2015)
   - Any hint voted valid by 2 out of 3 tutors included in G.S.
2. An algorithm generates hints for each hint request
   - It assigns a confidence weight to each hint it generates, summing to 1
3. Keep only hints which match a Gold Standard hint
4. QUALITYSCORE is the sum of the weights of the remaining hints

def firstAndLast(s):
    s[~~1~~0] + s[]

```
def firstAndLast(s):
    return s[1] + s[]
```

Hints: A B

Partial Matches

A hint is a *partial match* to the gold standard when:

1. The hint suggests a *subset* of the edits of a gold standard hint
2. At least one of these edits adds code

Examples (Gold Standard vs Generated Hint):

```
return 'Hello World'   vs   return __ 'Hello World'
repeat(x * 4)          vs   repeat(x__ * __)
return __ + __         vs   return __ BinOp __
```

Validating the QUALITYSCORE

Why not just have the tutors rate hints directly (e.g. Price 2017)?

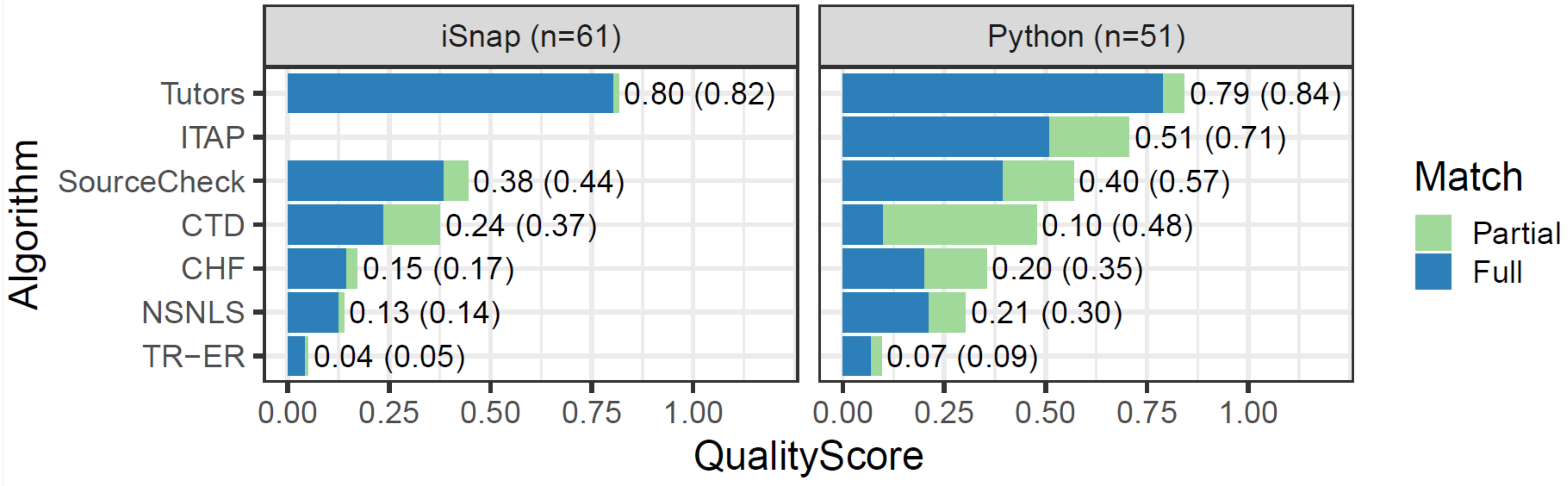
- **Advantage of QUALITYSCORE**: We can scale this approach to any number of hint generation algorithms
- **Concern**: Does the QUALITYSCORE reflect human quality judgements?

Validation: Used QUALITYSCORE to rate 252 hints on the iSnap dataset, and asked 3 human tutors to do the same, come to consensus:

- Agreement (Cohen's kappa) between QUALITYSCORE and consensus: 0.78
- Agreement each human tutor and consensus: 0.76, 0.78, 0.85
- **Conclusion**: QUALITYSCORE is as valid as a single human rater

Results

COMPARISON OF HINT GENERATION ALGORITHM QUALITY

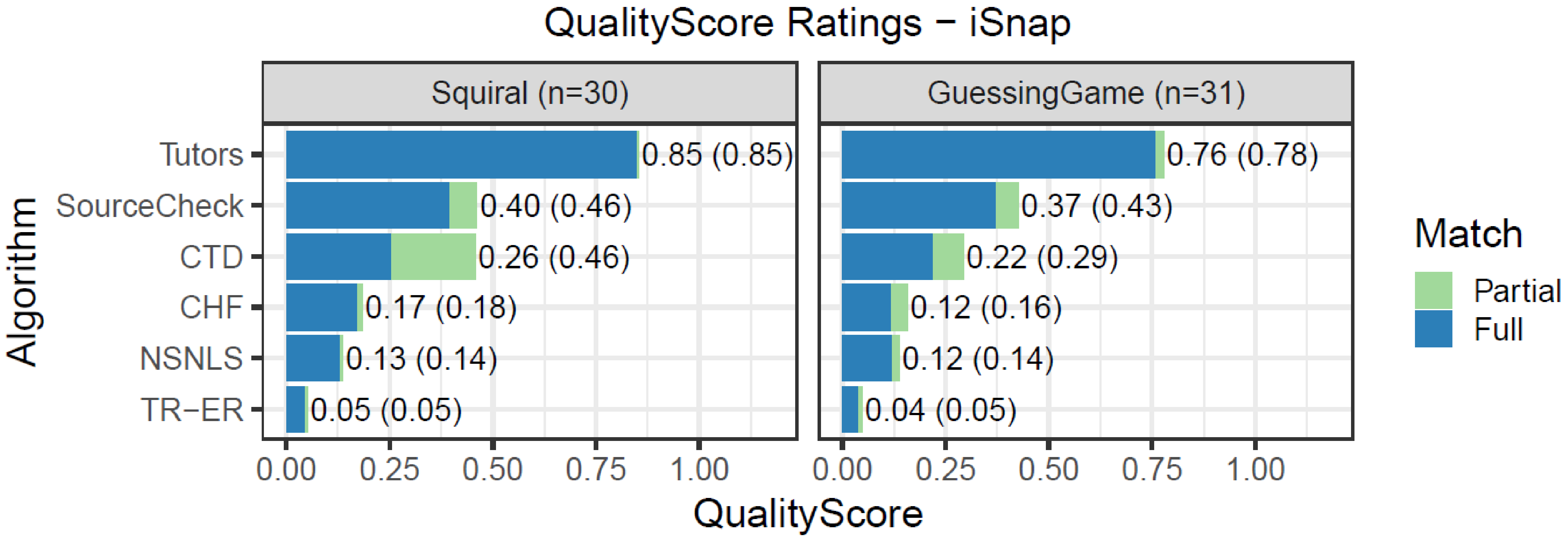


Significant differences in ratings across algorithms ($p < 0.001$, both datasets):

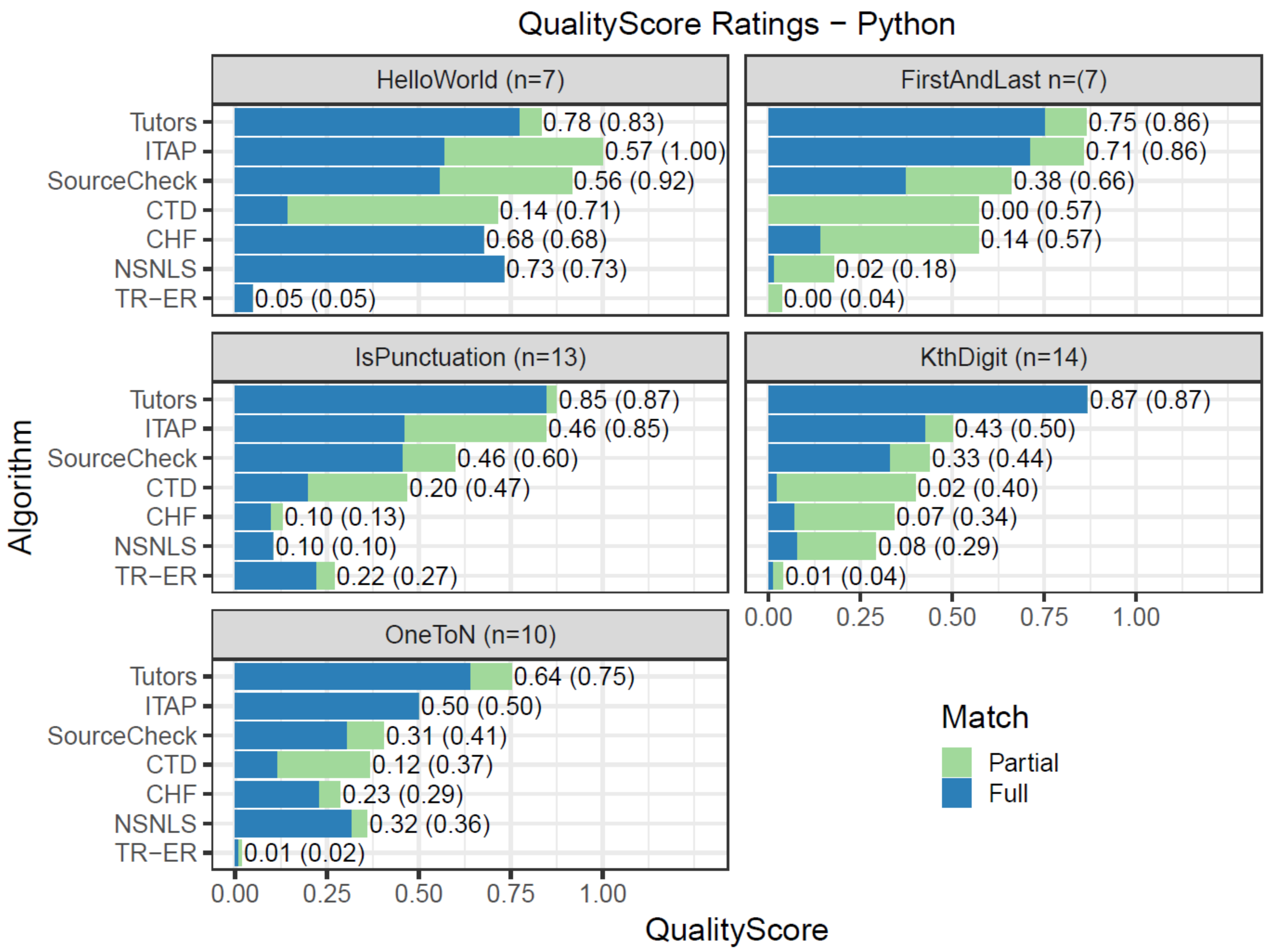
iSnap (full or partial): TR-ER < NSNLS, CHF < CTD < SourceCheck < Tutors

Python (full matches): TR-ER, CTD < CHF, NSNLS < SourceCheck, ITAP < Tutors

Python (partial matches): TR-ER < NSNLS, CHF < CTD, SourceCheck < ITAP, Tutors



Performance is consistent across the two problems in the iSnap dataset.



Performance is **not** consistent across problems in the ITAP dataset.

What makes hint generation hard?

Some hint requests had lower-quality hints *across* algorithms. Why?

Hypotheses: Hint generation is more difficulty for…

- Large Code: The more code a student has written
  - ✓ Supported: $r_s = 0.376$ (iSnap) and 0.389 (ITAP); $p < 0.01$
- Divergent Code: The more unique a student's code is compared to others'
  - ✓ Supported: $r_s = 0.356$ (iSnap) and 0.432 (ITAP); $p < 0.01$
- Few Correct Hints: The fewer Gold Standard hints there are
  - ✗ Not supported: No significant correlation

What makes algorithms perform poorly?

Some *algorithms* performed worse across hint requests. Why?

Hypotheses: Algorithms perform worse due to…

- Unfiltered Hints: Algorithms suggest too many hints
  - ✅ Supported: $r_s = 0.437$ (iSnap) and 0.487 (ITAP); $p < 0.001$
  - Algorithms generated more hints for larger code; humans did not
- Incorrect or Unhelpful Deletions: Many hints suggest deleting code only
  - ✅ Supported: Only 2.8% of generated deletion hints matched the gold standard
  - The best-performing algorithms did not suggest deletions (SourceCheck, ITAP)

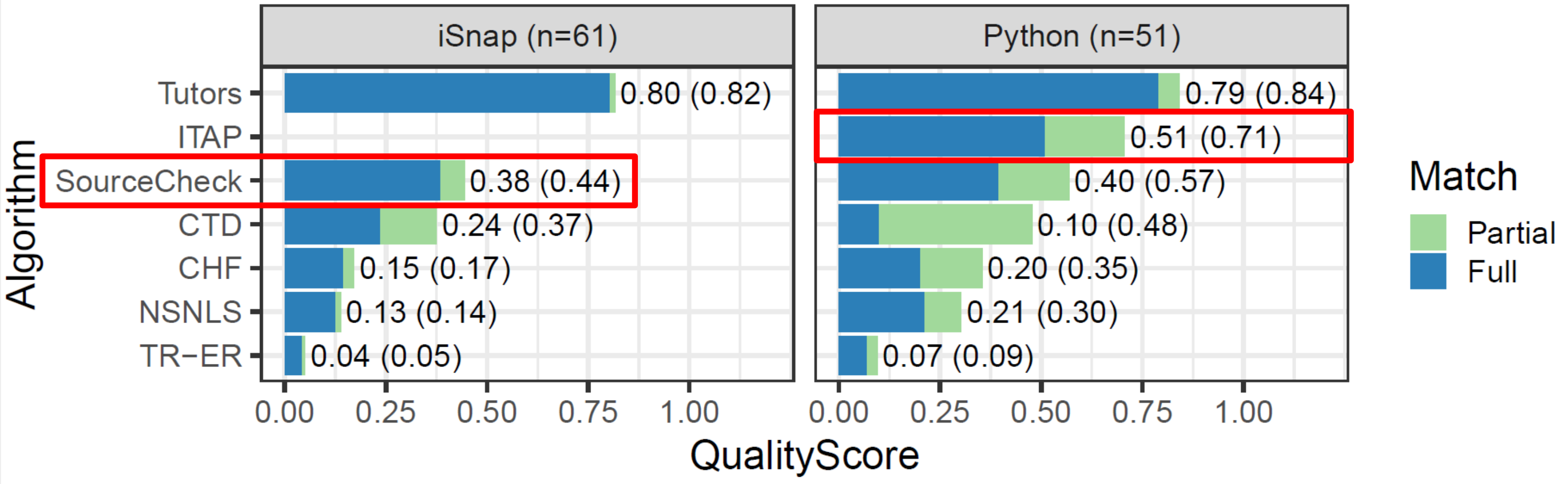
Discussion

Top-performing Algorithms

SourceCheck (iSnap) and ITAP (Python) performed the best

- These algorithms were designed for their respective datasets
  - However, SourceCheck still performs well on Python, outperforms its predecessor CTD

The ranking of the algorithms is consistent across datasets



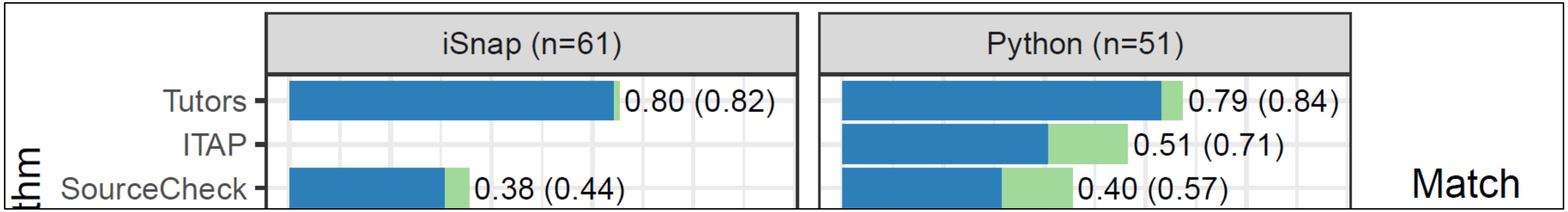
Algorithms vs Human Tutors

Algorithms are beginning to approach human-quality hints

- ITAP performed 84% as well as human tutors on the Python dataset
- However, this is only for the *simpler* dataset, counting *partial matches*

More complex assignments remain difficult

- SourceCheck performed only half as well as human tutors on the iSnap dataset
- These assignments were longer (10-13 LOC vs 2-4) and more complex



Improving Hint Quality

Address current weaknesses:

- More emphasis on selecting the *right* hint when multiple can be generated
  - Also suggested in prior work (Price 2017)
- Avoid hints to delete without adding code

Recognize when a hint is unlikely to be high quality

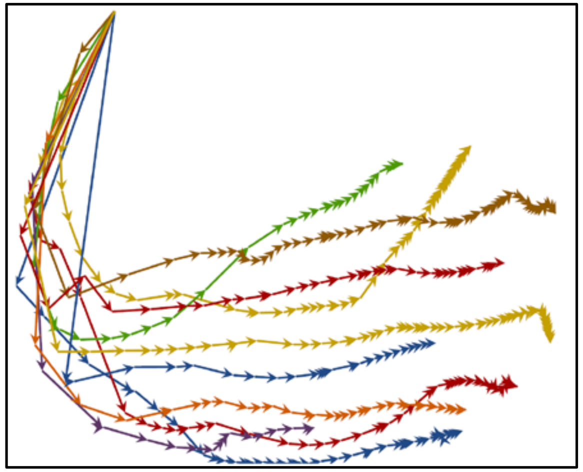
- E.g., when the student's code it unique

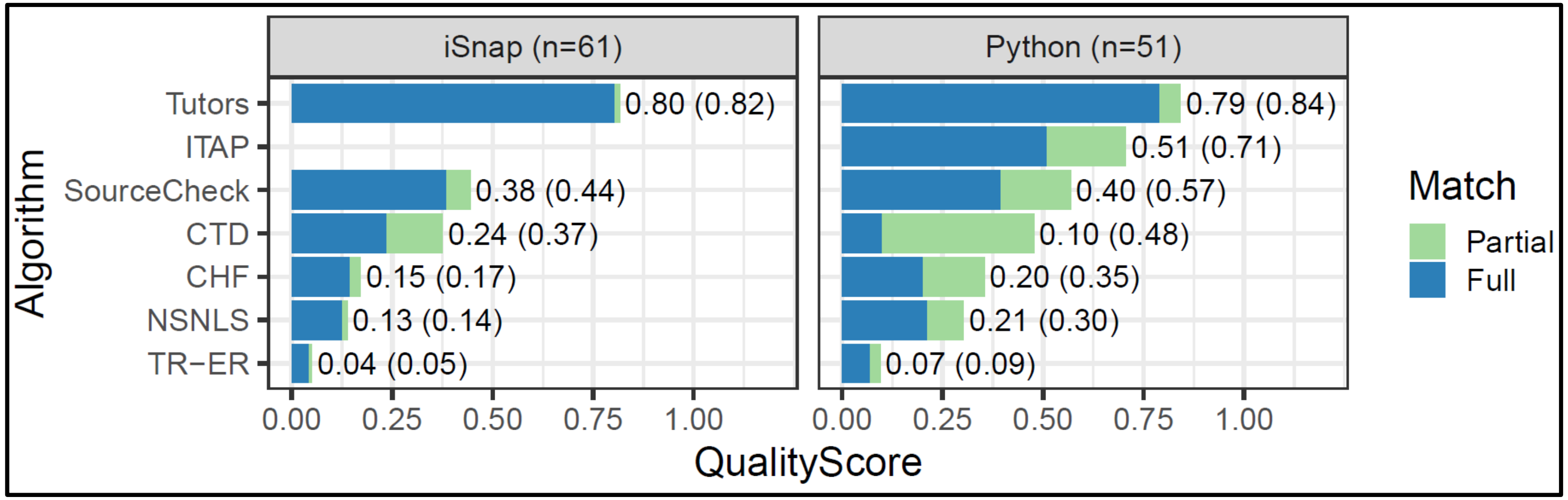
Evaluate the quality of *new* and *existing* algorithms

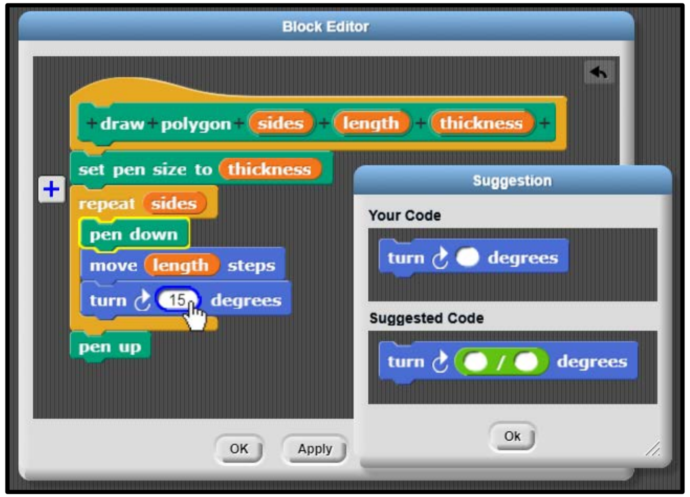
Thank You! Questions?

Contact: twprice@ncsu.edu

- Have a programming dataset with hint requests?
- Have a hint generation algorithm you would like to evaluate?
- **Data Available**: go.ncsu.edu/hint-quality-data

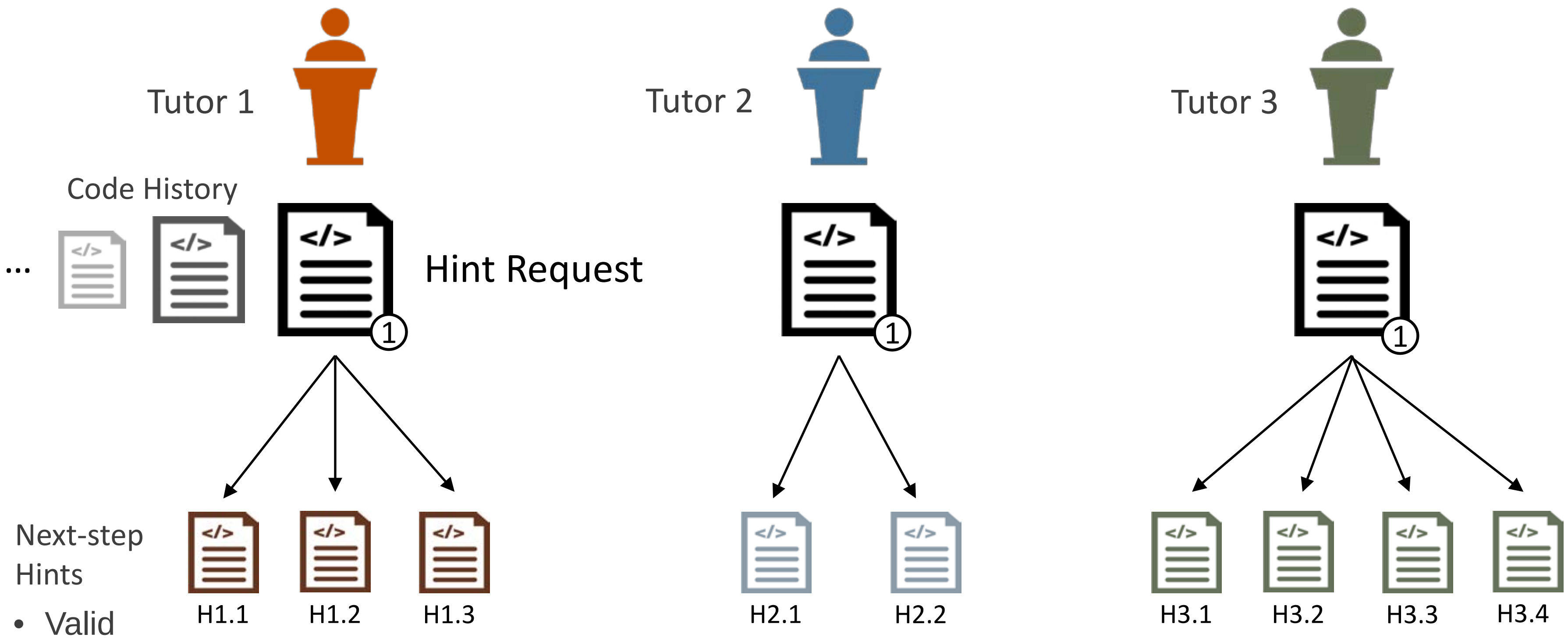






Secret Bonus Slides™

Gold Standard Hints



- Valid
- Useful
- Not confusing
- One edit (if possible)

Gold Standard Hints

Each tutor rates each other tutor's hints:

Any hint with at least 2 votes part of the gold standard: