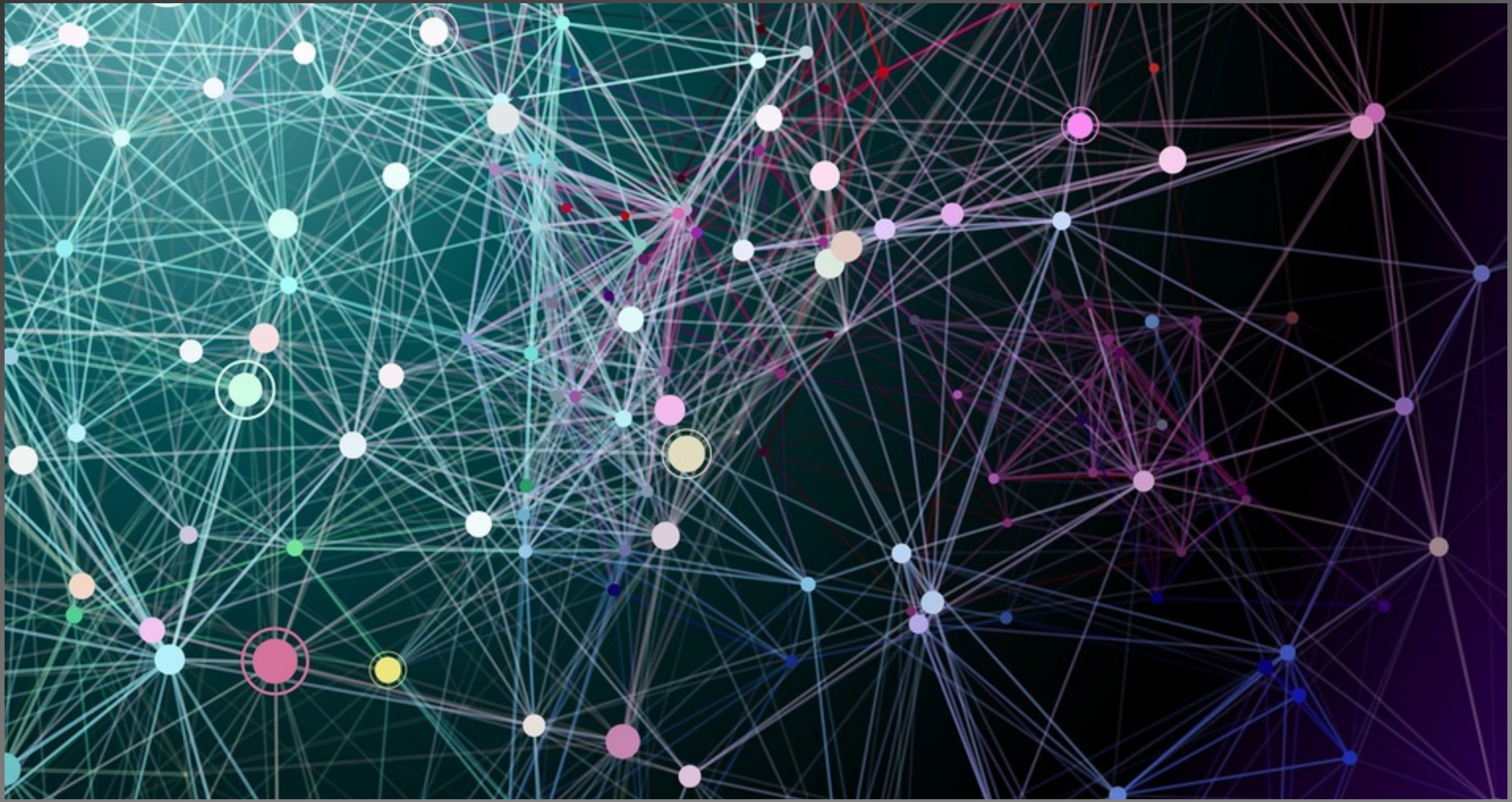
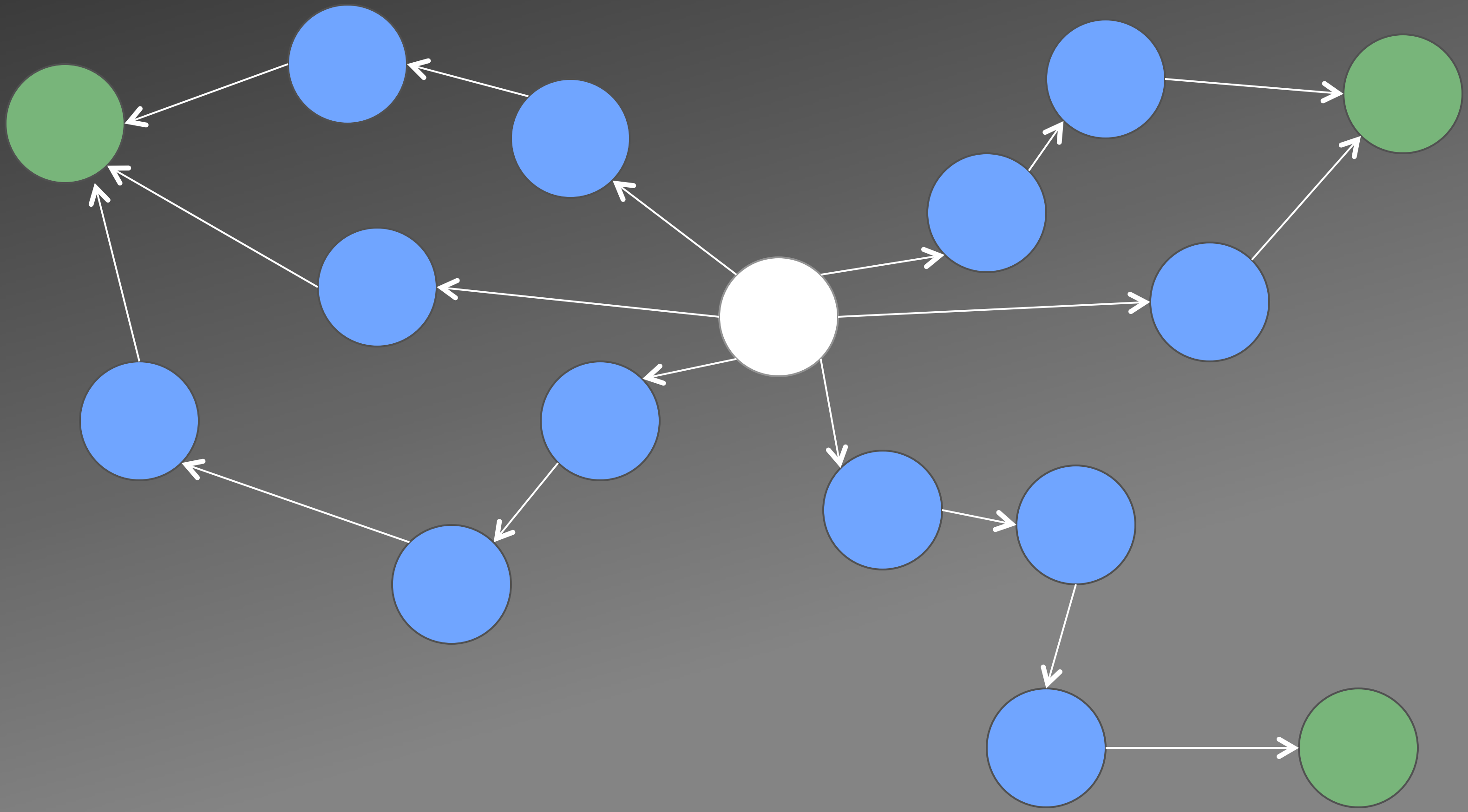
Automating Hint Generation with Solution Space Path Construction

Kelly Rivers and Ken Koedinger

The Rise of Big Data in Open-Ended Learning Domains



Solution Space



Approaches to Hinting

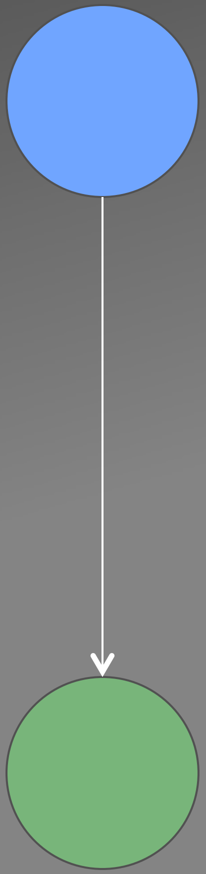
- Hint Factory
  - Barnes & Stamper, 2008
- Cluster-based approaches
  - Gross, Mokbel, Hammer, & Pinkwart, 2012
  - Huang, Piech, Nguyen, & Guibas, 2013

- Shared feature: reliance on *existing states*

Solution:
Path Construction

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

```
def automatedReadabilityIndex(text):
    return 4.71*(float(charCount(text))/ wordCount(text)) +
(0.5*(float(wordCount(text))/sentenceCount(text))) - 21.43
#has to be float
```

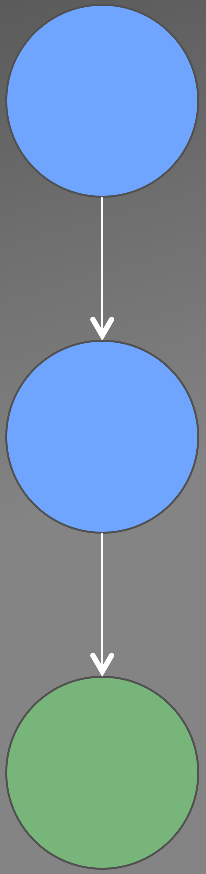


Solution:
Path Construction

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

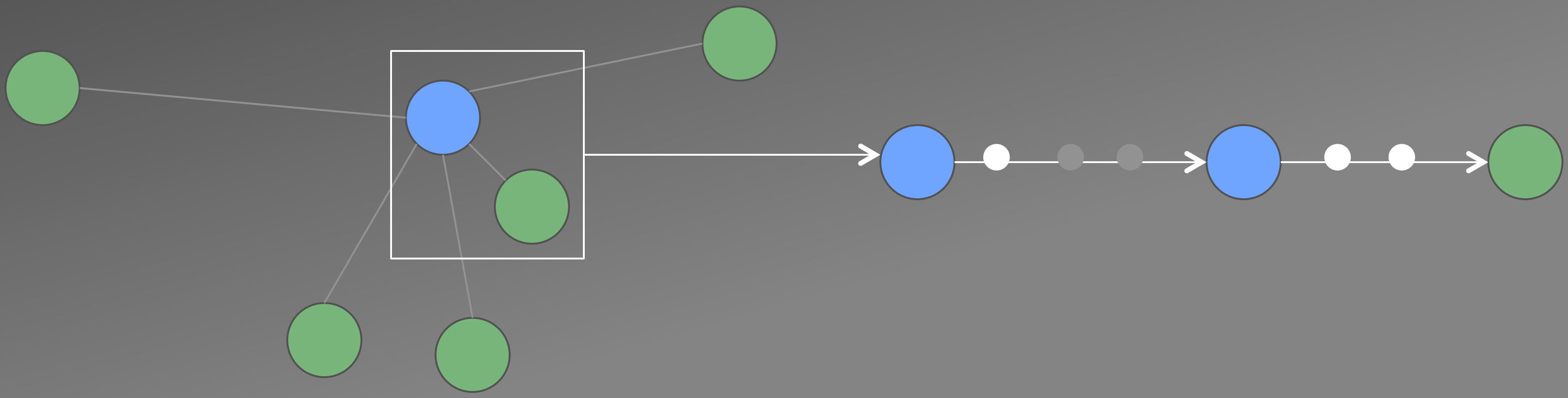
```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.71 * ((charCount(text)) / (wordCount(text)))  + \
            0.5 * ((wordCount(text)) / (sentenceCount(text))) -
21.43)
```

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return 4.71*(float(charCount(text))/ wordCount(text)) +
(0.5*(float(wordCount(text))/sentenceCount(text))) - 21.43
```



Path Construction

1. Identify Optimal Goal
2. Generate Valid Intermediate States
3. Choose an Optimal Change Path

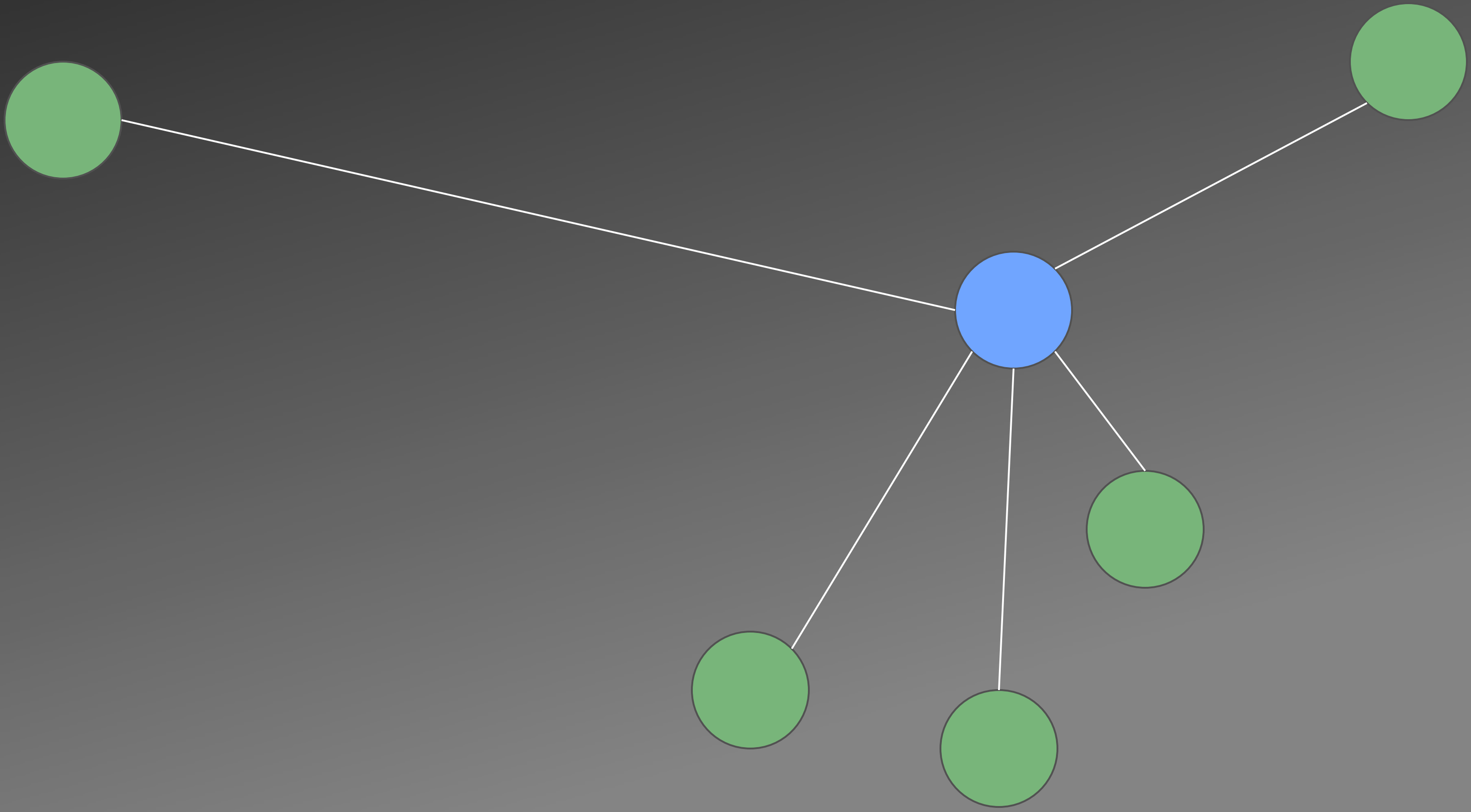


Disclaimers

- Required ingredients:

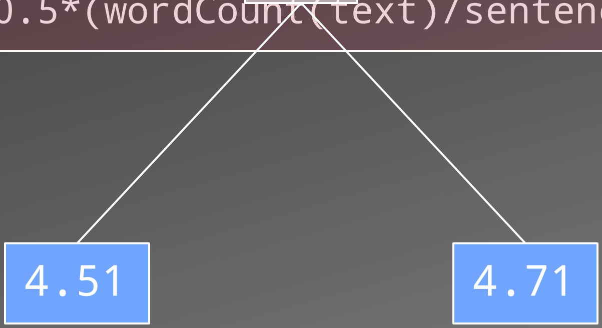
– S	A data set of solution states
– test(s)	returns a score in [0,1]
– diff(s,t)	returns a distance in [0,1]

Step 1: Identify Optimal Goal

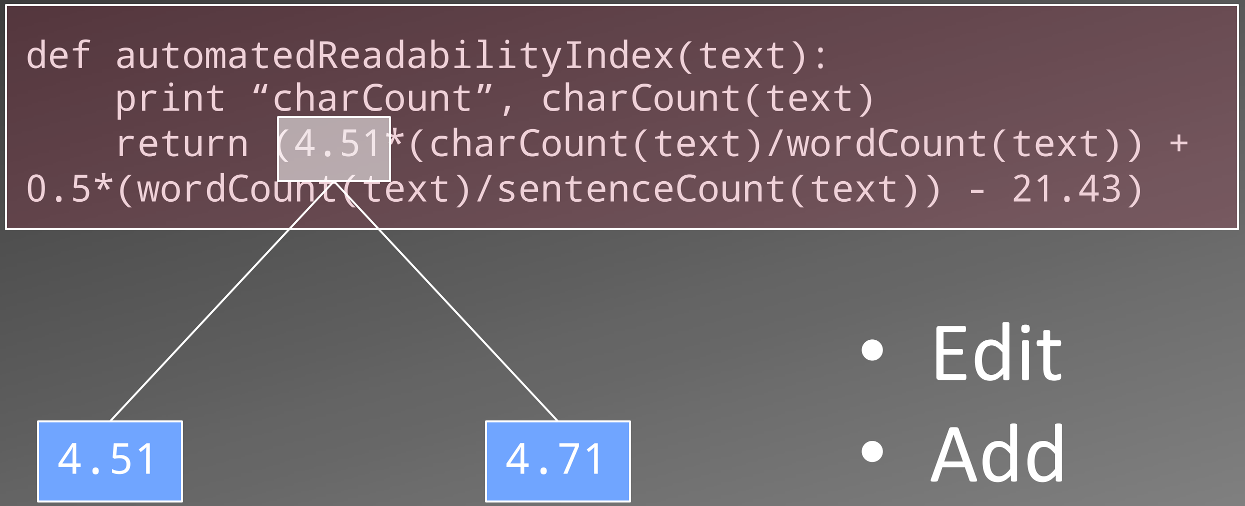


Change Vectors

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



Change Vectors

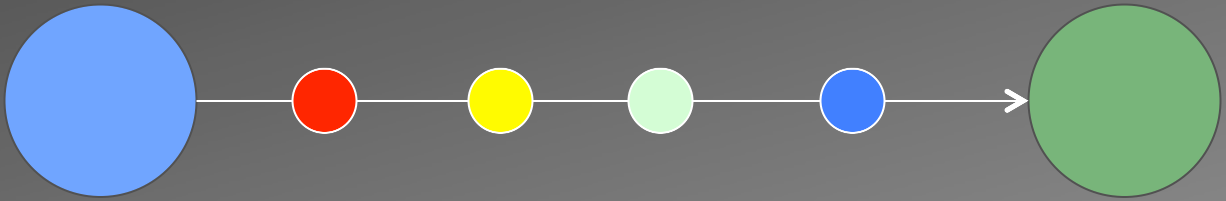


```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

- Edit
- Add
- Delete
- Swap
- Move

Step 1: Identify Optimal Goal

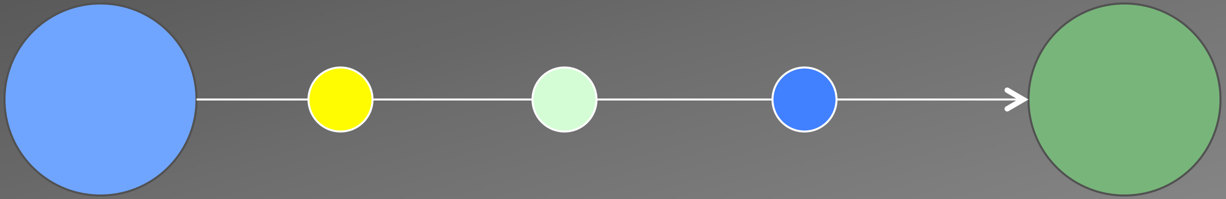
```
def automatedReadabilityIndex(text):
    print "charCount", charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



```
def automatedReadabilityIndex(text):
    return (4.71*(float(charCount(text))/wordCount(text)) +
0.5*(float(wordCount(text))/sentenceCount(text)) - 21.43)
```

Step 1: Identify Optimal Goal

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

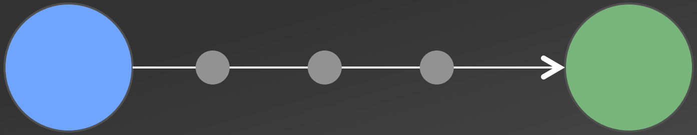


```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.71*(float(charCount(text))/wordCount(text)) +
0.5*(float(wordCount(text))/sentenceCount(text)) - 21.43)
```

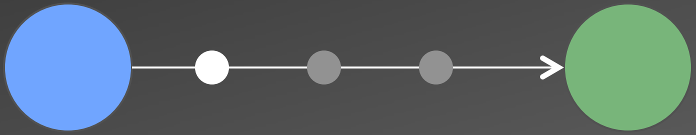
Step 2: Identify Valid Intermediate States

- A valid state must be:
  - Well-formed
  - Closer to the goal than the original state
  - No worse in score than the original state

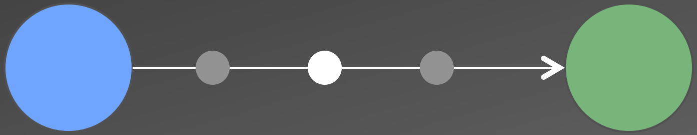
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



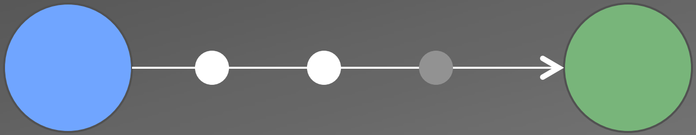
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



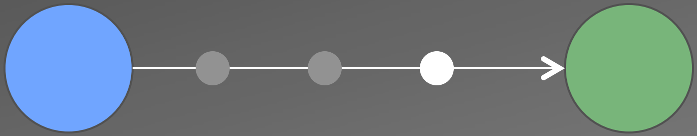
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



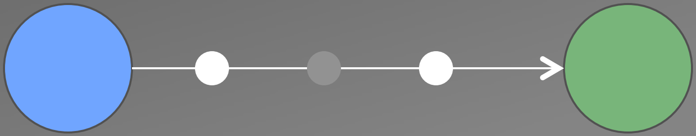
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



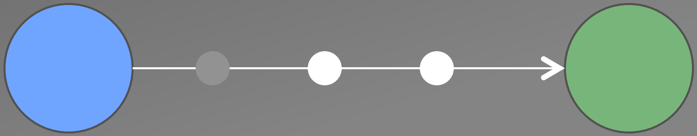
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



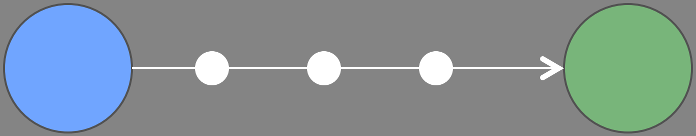
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



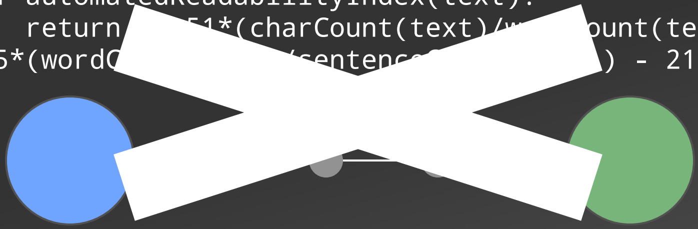
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



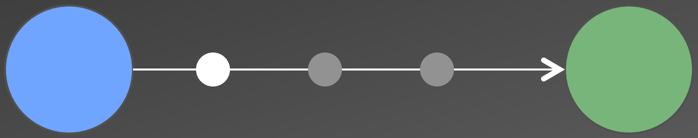
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



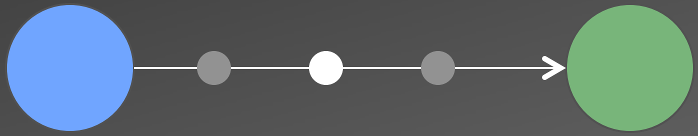
```
def automatedReadabilityIndex(text):
    return (  .51*(charCount(text)/w      ount(text)) +
0.5*(wordC        /sentenc           ) - 21.43)
```



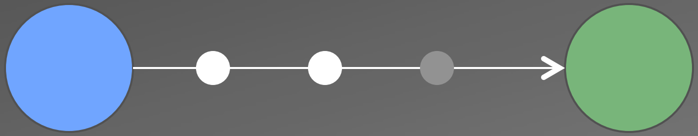
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



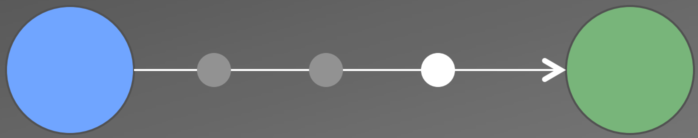
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



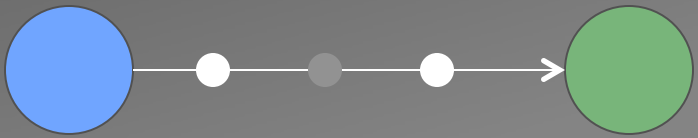
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



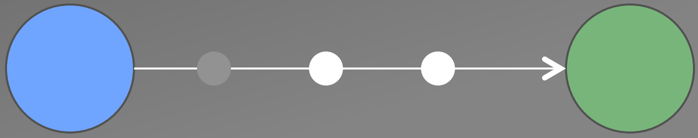
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



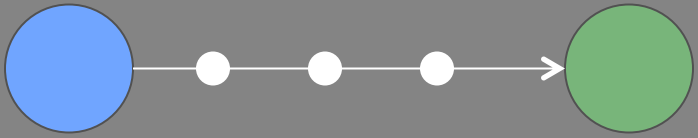
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



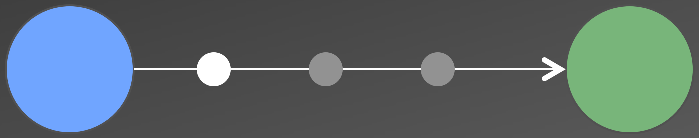
```
def automatedReadabilityIndex(text):
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



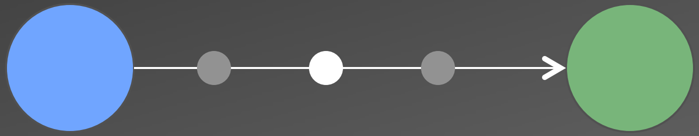
Step 3: Create an Optimal Change Path

- Desirable Properties:
  - Seen before (SB)
  - Near current state (NS)
  - Well-performing (WP)
  - Near goal (NG)

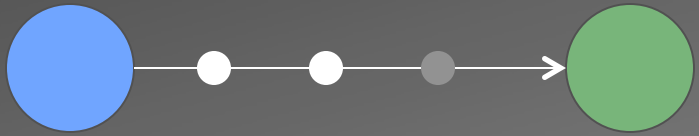
SB: 1.0 NS: 0.95 WP: 0.5 NG: 0.73



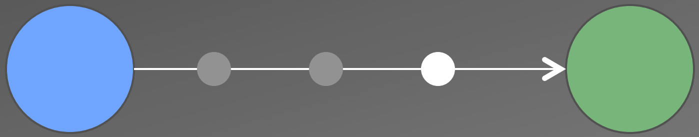
SB: 0.0 NS: 0.86 WP: 0.0 NG: 0.82



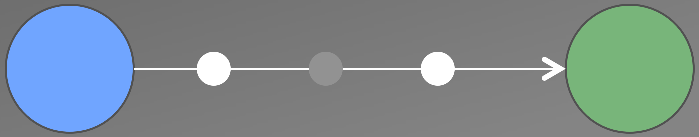
SB: 1.0 NS: 0.81 WP: 0.5 NG: 0.86



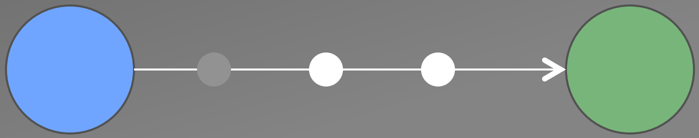
SB: 0.0 NS: 0.86 WP: 0.0 NG: 0.82



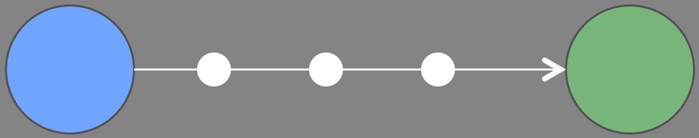
SB: 0.0 NS: 0.81 WP: 0.5 NG: 0.86



SB: 0.0 NS: 0.72 WP: 0.0 NG: 0.95

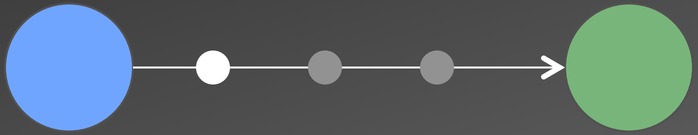


SB: 1.0 NS: 0.68 WP: 1.0 NG: 1.00

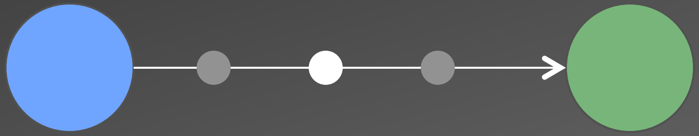


$$\frac{2 \cdot SB + 4 \cdot NS + 1 \cdot WP + 2 \cdot NG}{9}$$

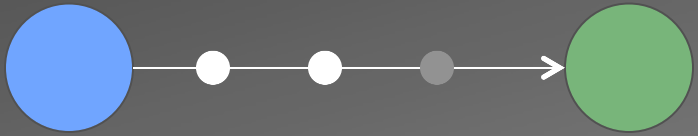
SB: 1.0 NS: 0.95 WP: 0.5 NG: 0.73 -> 0.86



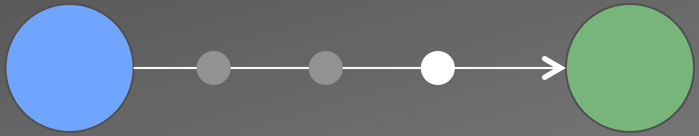
SB: 0.0 NS: 0.86 WP: 0.0 NG: 0.82 -> 0.56



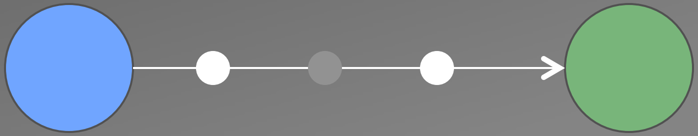
SB: 1.0 NS: 0.81 WP: 0.5 NG: 0.86 -> 0.83



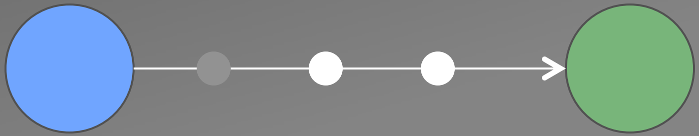
SB: 0.0 NS: 0.86 WP: 0.0 NG: 0.82 -> 0.56



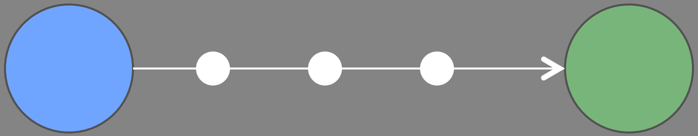
SB: 0.0 NS: 0.81 WP: 0.5 NG: 0.86 -> 0.61



SB: 0.0 NS: 0.72 WP: 0.0 NG: 0.95 -> 0.53

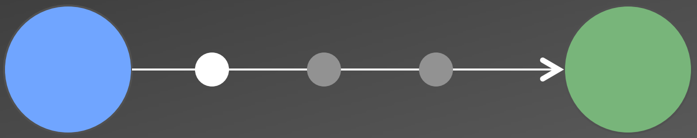


SB: 1.0 NS: 0.68 WP: 1.0 NG: 1.00 -> 0.85

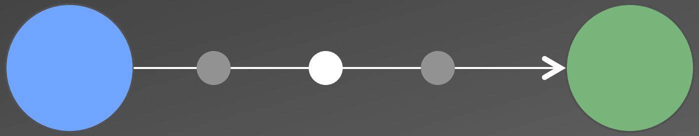


$$\frac{2*SB + 4*NS + 1*WP + 2*NG}{9}$$

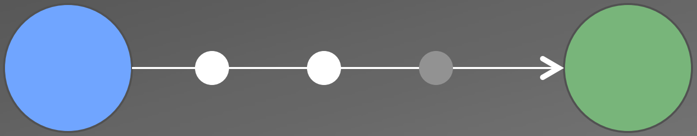
SB: 1.0 NS: 0.95 WP: 0.5 NG: 0.73 -> 0.86



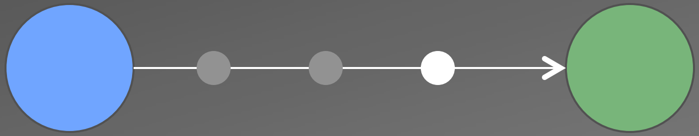
SB: 0.0 NS: 0.86 WP: 0.0 NG: 0.82 -> 0.56



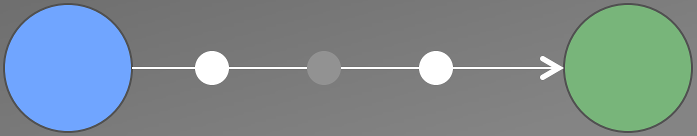
SB: 1.0 NS: 0.81 WP: 0.5 NG: 0.86 -> 0.83



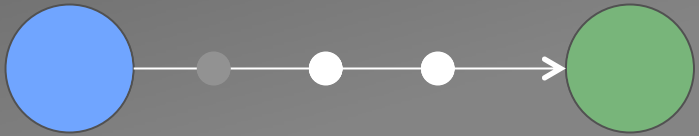
SB: 0.0 NS: 0.86 WP: 0.0 NG: 0.82 -> 0.56



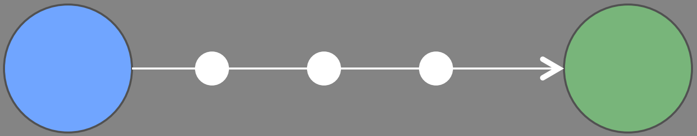
SB: 0.0 NS: 0.81 WP: 0.5 NG: 0.86 -> 0.61



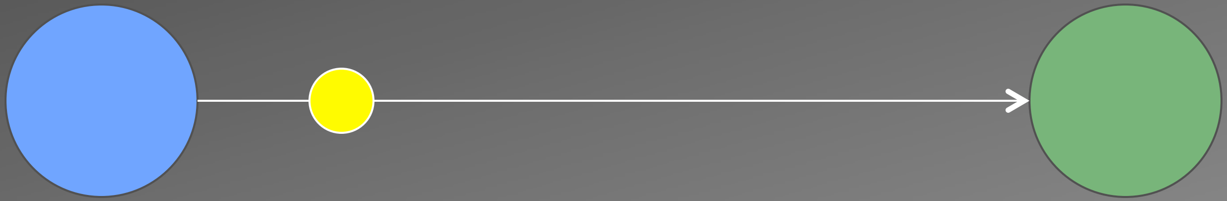
SB: 0.0 NS: 0.72 WP: 0.0 NG: 0.95 -> 0.53



SB: 1.0 NS: 0.68 WP: 1.0 NG: 1.00 -> 0.85



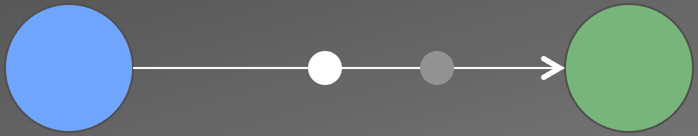
```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```



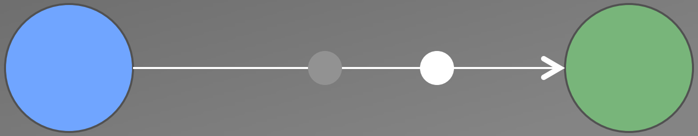
```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.71*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

$$\boxed{\frac{2 \cdot SB + 4 \cdot NS + 1 \cdot WP + 2 \cdot NG}{9}}$$

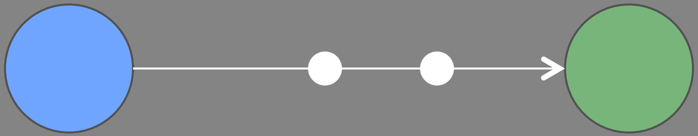
SB: 1.0 NS: 0.86 WP: 0.5 NG: 0.86



SB: 0.0 NS: 0.86 WP: 0.5 NG: 0.86

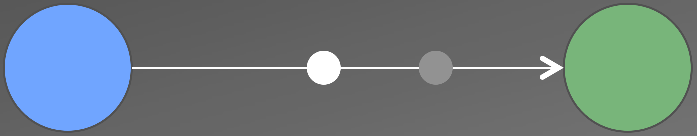


SB: 1.0 NS: 0.72 WP: 1.0 NG: 1.00

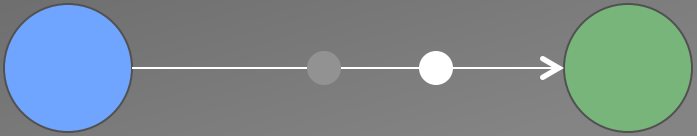


$$\frac{(2*SB + 4*NS + 1*WP + 2*NG)}{9}$$

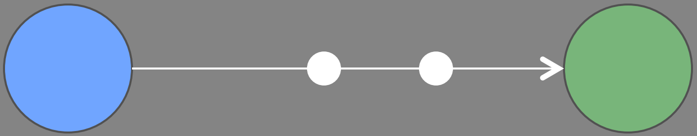
SB: 1.0 NS: 0.86 WP: 0.5 NG: 0.86 -> 0.85



SB: 0.0 NS: 0.86 WP: 0.5 NG: 0.86 -> 0.63

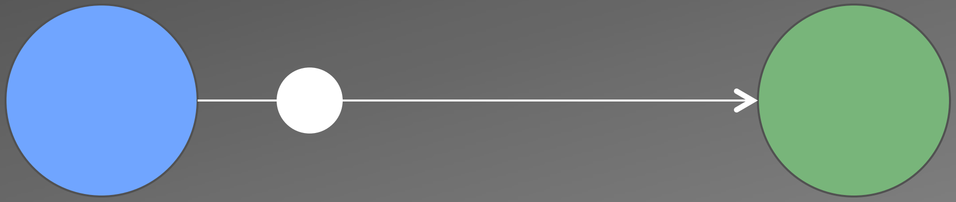


SB: 1.0 NS: 0.72 WP: 1.0 NG: 1.00 -> 0.88



Making Messages Human-Readable

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

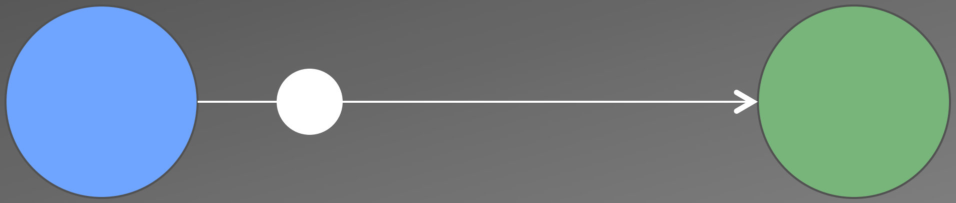


Position

In line 3 replace **4.51** with **4.71** in the left side of the binary operation.

Making Messages Human-Readable

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

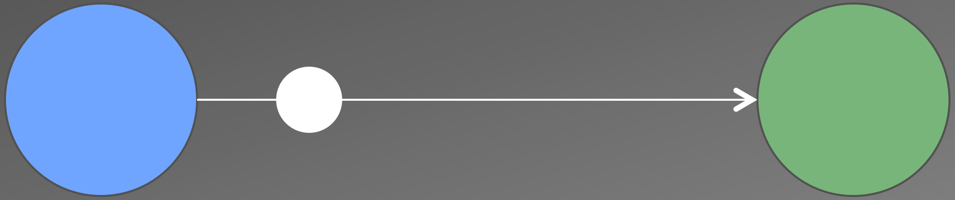


Edit type

In line 3 replace **4.51** with **4.71** in the left side of the binary operation.

Making Messages Human-Readable

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

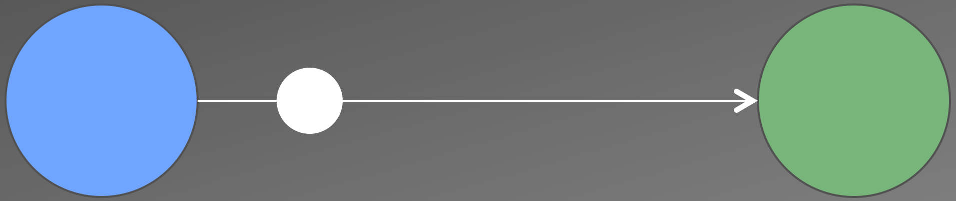


Old value

In line 3 replace **4.51** with **4.71** in the left side of the binary operation.

Making Messages Human-Readable

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) + 
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

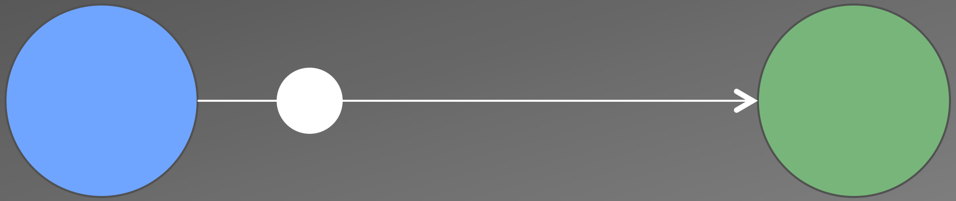


New value

In line 3 replace **4.51** with **4.71** in the left side of the binary operation.

Making Messages Human-Readable

```
def automatedReadabilityIndex(text):
    print “charCount”, charCount(text)
    return (4.51*(charCount(text)/wordCount(text)) +
0.5*(wordCount(text)/sentenceCount(text)) - 21.43)
```

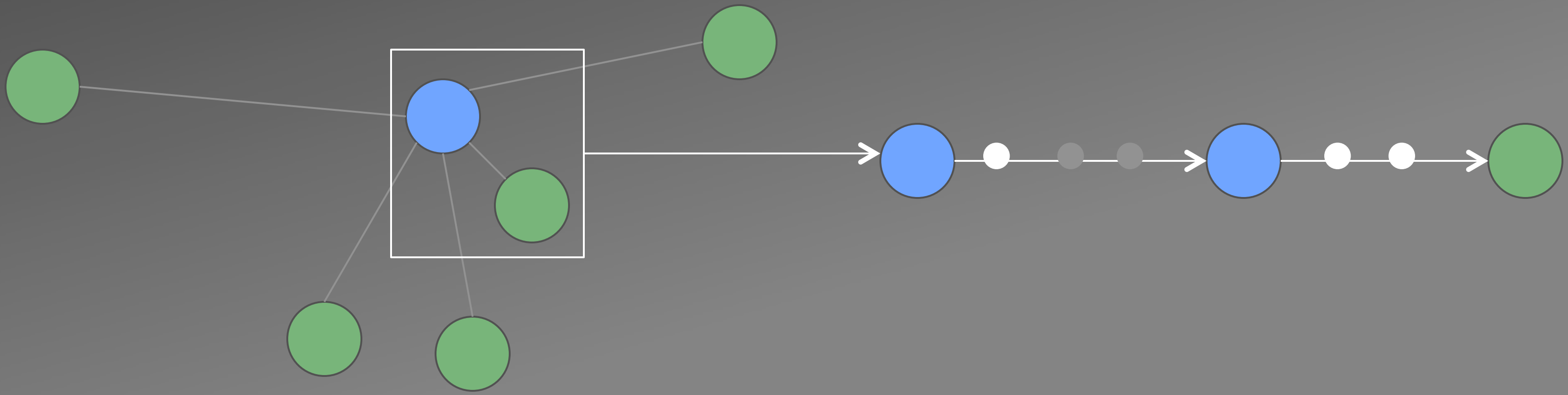


Context

In line 3 replace **4.51** with **4.71** in the left side of the binary operation.

Path Construction

1. Identify Optimal Goal
2. Generate Valid Intermediate States
3. Choose an Optimal Change Path



Evaluation

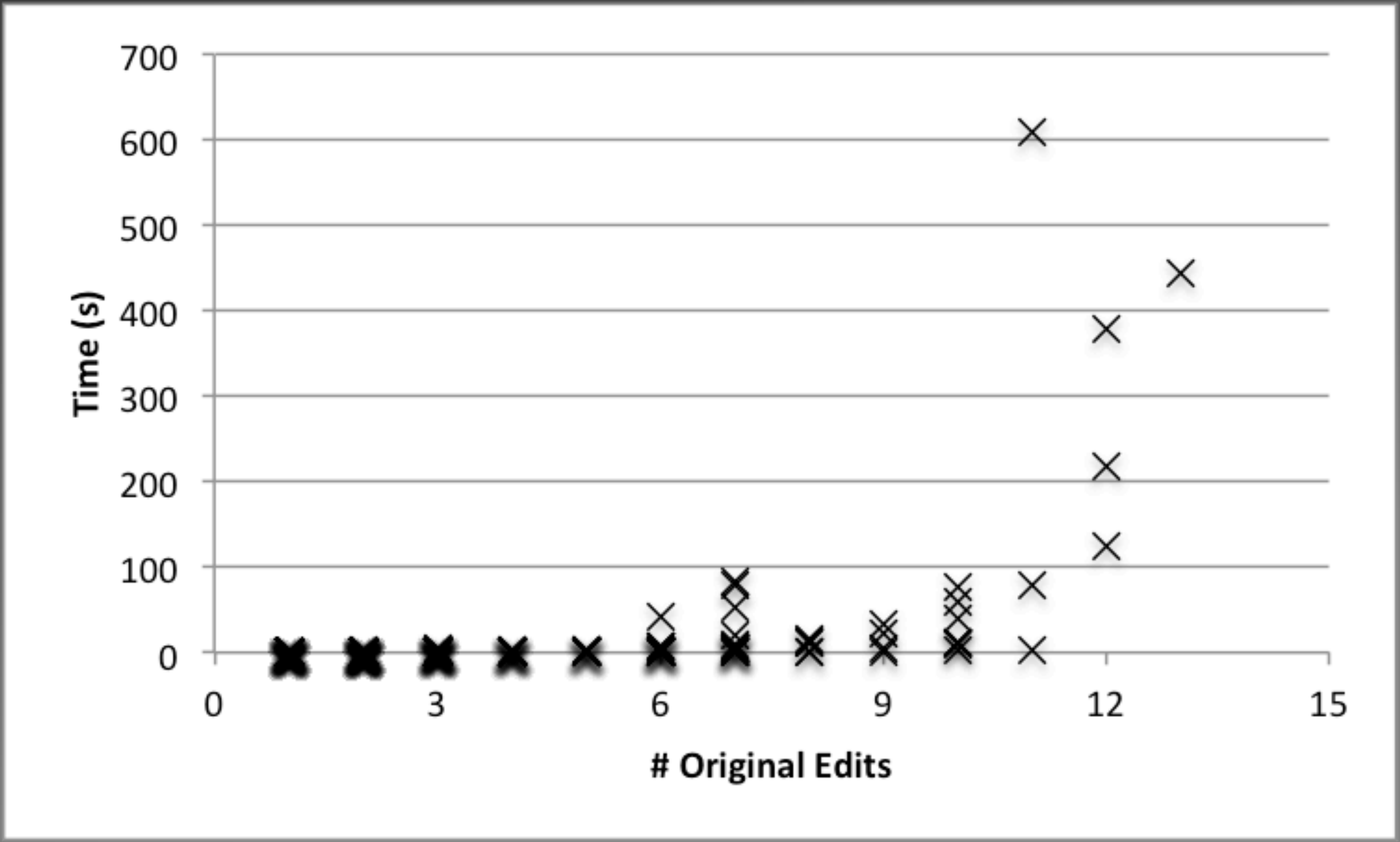
- Q: Do the generated hints help students?
- A: We don't know yet… but we can analyze previous data.
  - 9 problems
  - 248 incorrect states (including generated)

Q1: When Are Hints Possible?

- Bare Minimum: Always
  - If there's at least one correct solution in the space

- Personalized: 91.9%
  - On average, anything less than 10 edits will take less than 1 minute to run
  - 95.6 for < 10 minutes

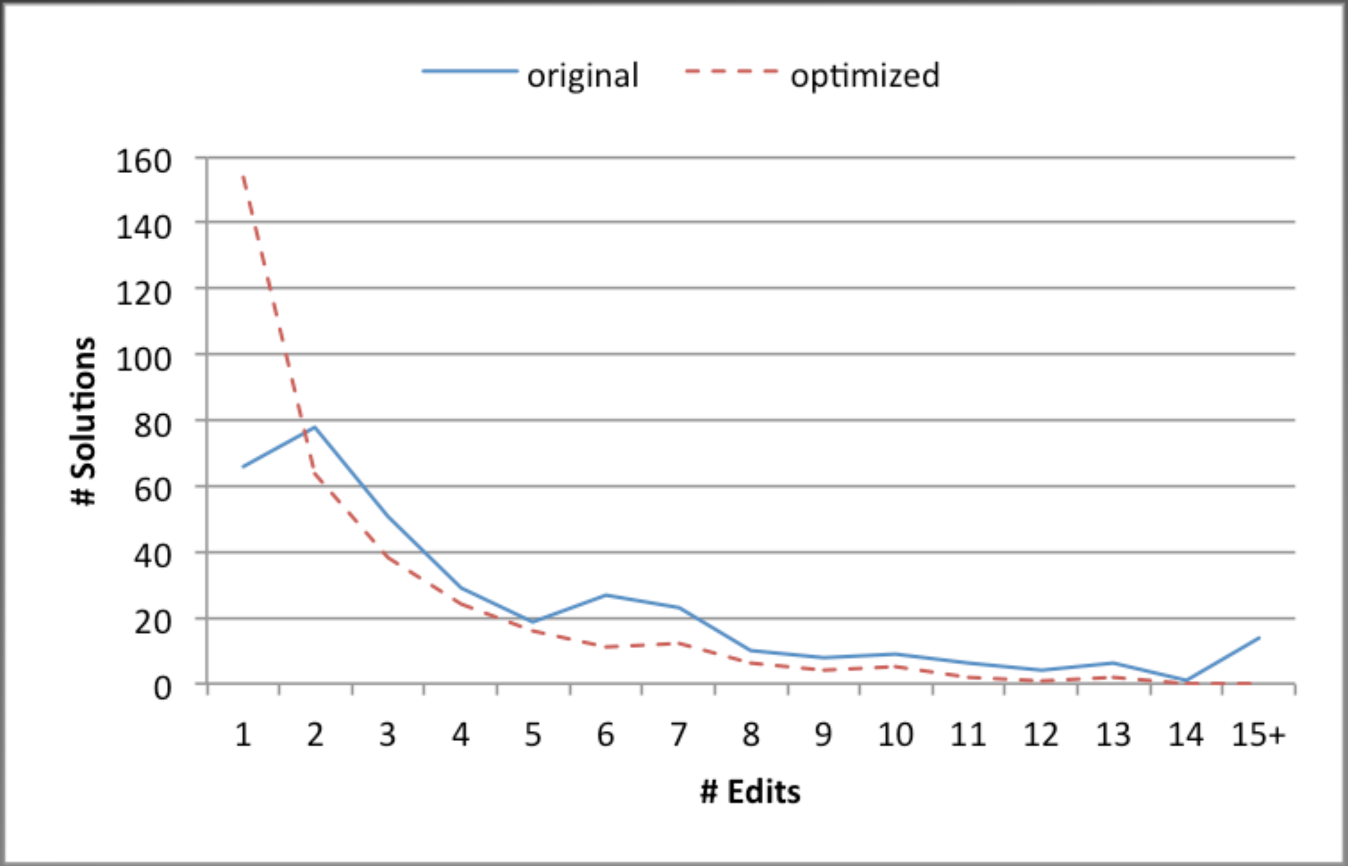
Q2: How Long Do Hints Take?



Q3: Are Hints Well-Aligned?

- True Measure: Only known by student
- Approximate Measure: Distance to optimal solution – 64%
- Is personalization working?

Q3: Are Hints Well-Aligned?

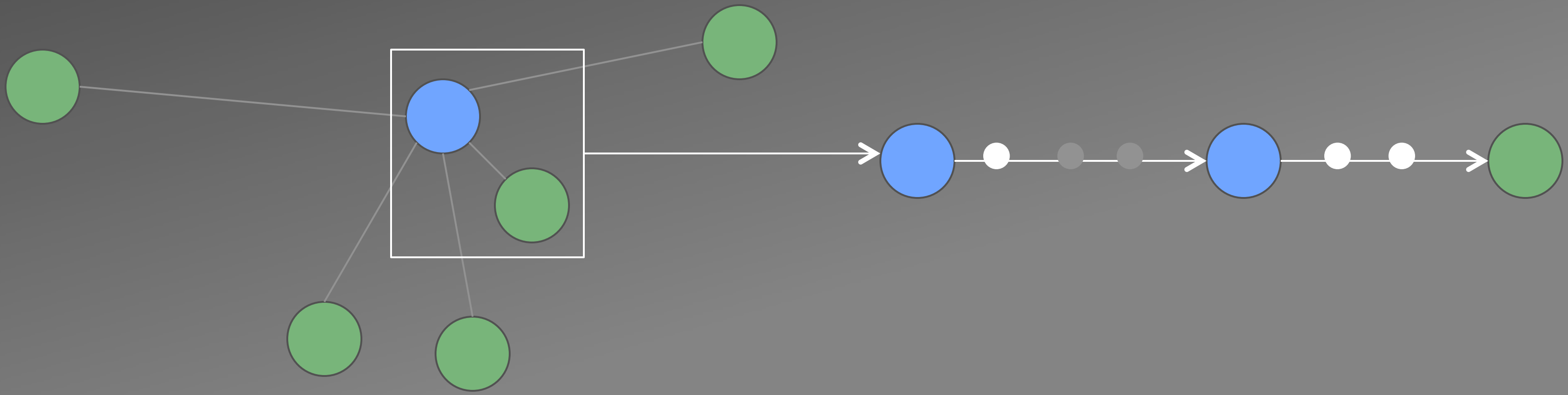


Limitations

- Lack of empirical evaluation
- Current data set is mostly composed of final solutions
- Far-away states more difficult to handle than nearby states.

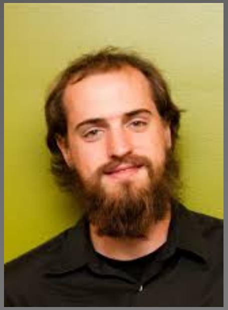
Conclusions

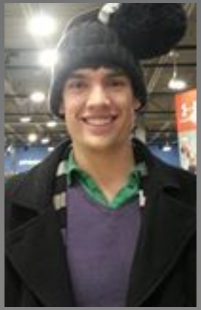
- New hint generation method: path construction



Questions?

Thanks to…





This work was supported in part by Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (# R305B090023).

Demo

- krivers.net/cloudcoder.html