Where do I find that program?

Evaluating the effectiveness of the Metropolis-Hastings algorithm for the problem of program synthesis.

Author:
Victor van Wieringen,
V.J.vanWieringen@student.tudelft.nl

Supervisor:
Sebastijan Dumančić

CSE3000 Research Project

1. Background

In this research multiple different search algorithms are applied to and evaluated for the problem of program synthesis.

Program synthesis:
Search for programs in a Domain specific language that solve a task which is specified using input-output examples.

Example:
remove the exclamation mark at the end of a string.
Input examples:
“abcd!” ➙ “abcd”
“Hey you!” ➙ “Hey you”
Expected program:
[LoopWhile(NotAtEnd,[moveRight]), Drop]

Cost
Finding the right program is essentially a search problem. To aid in the search, a cost function is defined, it compares the output of a program to the expected output.
This cost is used as a minimizing heuristic in the search.

2. Method

Domains

Domain	Description	Cost function
String	Manipulate strings in specific ways.	Levenshtein
Pixel	Paint the right pixels in a grid.	Hamming
Robot	Pickup and move a ball to the right spot.	#Steps

Search algorithms
The following algorithms were evaluated:

- Brute[2] *(Best-first search)*
- Metropolis-Hastings
- Very large neighborhood search
- Monte Carlo tree search
- A-star search

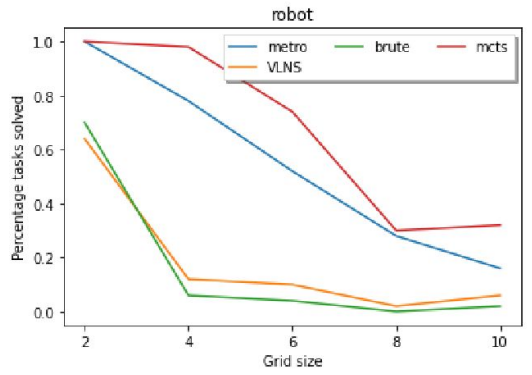
Metropolis-Hastings
The Metropolis-Hastings algorithm can be used to search the program space.

It traverses the program space by starting at a program and repeatedly mutating it in such a way that the cost of the resulting program is minimized.

Proposed mutations are either accepted or rejected, this probability *A(x,y)* is calculated by looking at the induced change in cost:

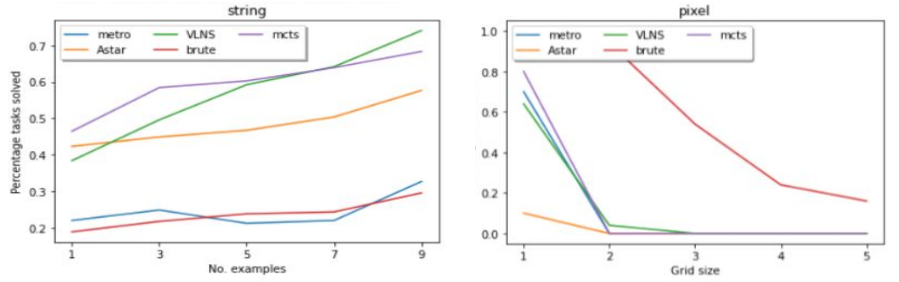
1. $A(x,y) = \frac{\pi(y)}{\pi(x)}$ 2. $\pi(x) = e^{-Cost(x)}$

Mutations that increase the cost have a non-zero chance of still being accepted!

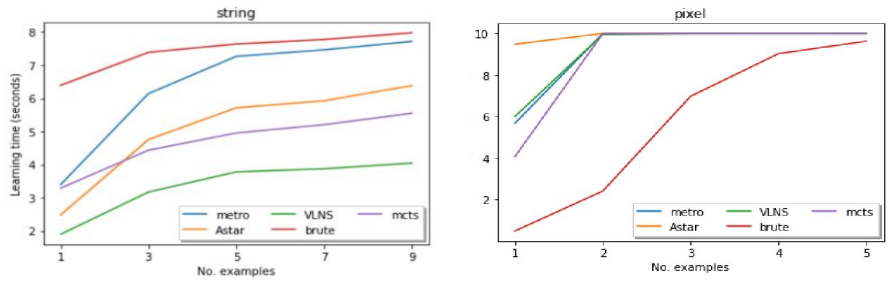


Performance for Robot domain with the original cost function[2]

3. Results



Performance of different search methods for String and Pixel domains



Learning times of different search methods for String and Pixel domains

4. Conclusion

String domain
Performing similar to Brute, the Metropolis-Hastings algorithm pales in comparison to the other algorithms that were implemented, both in terms of number of examples solved and running time.

Pixel domain
Brute outperforms all other algorithms here.
This is due to the fact that in this domain, a large minimal number of ‘steps’ have to be taken before any decrease in cost can be achieved.
It seems simple enumerative approaches work best for domains with this characteristic.

Cost function
The exact definition of the cost function greatly influences the performance of all search algorithms: after altering the cost function for the robot domain, all algorithms were able to completely solve it.

Relevant work:
[1] “Introduction to program synthesis,”MIT lecture series (6.S084/6.887 2020).
[2] A. Cropper and S. Dumancic, “Learning large logic programs by going beyond entailment,”Pro-ceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Jun 2019.
[3] E. Schkufza, R. Sharma, and A. Aiken, “Stochastic superoptimization,” in Architectural Support for Programming Languages and Operating Systems, ASPLOS ’13, Houston, TX, USA - March 16- 20, 2013, pp. 305–316, 2013

TU Delft