dtControl: Decision Tree Learning Algorithms for Controller Representation

Repeatability Evaluation README

dtControl is a tool for representing strategies/controllers (given as list of state-action-pairs) as decision trees.
To evaluate dtControl, download `dtcontrol_rep.zip`. We assume it was downloaded to the default downloads folder `~/Downloads`.
Extract the contents to your home folder by issuing the following commands in your console:

```
$ cd ~
$ unzip ~/Downloads/dtcontrol_rep.zip -d ~
```

If you did not download to the default folder, replace `~/Downloads/dtcontrol_rep.zip` with that location in the above command.
You should now have the following directory structure:

```
/home/YOUR_USERNAME
|-- dtcontrol_rep
|   |-- developer_manual.pdf
|   |-- dtcontrol_tum-1.0.0rc1-py3-none-any.whl
|   |-- dtcontrol-tum-1.0.0rc1.tar.gz
|   |-- examples
|   |   |-- 10rooms.scs.zip
|   |   |-- aircraft.scs.zip
|   |   |--       .
|   |   |--       .
|   |-- experiments_all.sh
|   |-- experiments_quick.sh
|   |-- experiments_many.sh
|   |-- HSCC_2020_paper_69.pdf
|   |-- LICENSE
|   |-- README.html
|   |-- user_manual.pdf
```

On extracting our archive, you will find the following contents:

- The `README.html` file which gives you all the instructions to get started with this Repeatability Evaluation. It is the same as the RE Instructions PDF.
- The python package `dtcontrol_tum-1.0.0rc1-py3-none-any.whl`, which you can use to install dtcontrol, given that you have python 3.6 or higher.
- The folder `examples`, which contains zip-archives for all the case studies we use in the paper.
- The scripts `experiments_quick.sh`, `experiments_many.sh` and `experiments_all.sh` which run a quick subset of experiments (~10min), many (those that are not prone to getting a memout, ~4 hours) or all the experiments of the paper (~12 hours).
- The `user_manual.pdf` which describes how to use dtControl.

- The `developer_manual.pdf` which describes how you can extend dtControl.
- The file `dtcontrol-tum-1.0.0rc1.tar.gz`, which contains the source code for dtControl. You need this only if you are interested in looking at the code, it is not necessary for repeating the experiments.
- The PDF file of our paper `HSCC_2020_paper_69.pdf`.

Elements of the paper included in the REP

You can run our tool and verify that the input and output specifications we describe in Section 2 “Tool” of the paper are correct. For that, it makes sense to run all (or part of) the experiments, the results of which are aggregated in Table 1 on page 5 of the paper. This means running up to 8 algorithms on up to 10 case studies. Additionally, to test robustness, you could also run `dtcontrol` on other controllers given as UPPAAL (`*.dump`), SCOTS (`*.scs`) or generic (`*.csv`) files.

System requirements

Operating Systems

1. Linux (tested on Ubuntu 18.04, Manjaro): all methods supported
2. MacOS (tested on Catalina 10.15): all methods,**except OC1**, supported
3. Windows was not tested and is not described in this README. An advanced user should be able to transfer the commands given in the README and make it work on Windows.

Major requirements

1. **Python**: dtControl was built and tested with Python 3.6.8 and Python 3.7.3. However it is compatible with any Python >= 3.6.
2. **gcc** (Optional for OC1): Any current version of the GNU C compiler needs to be located at `/usr/bin/gcc`.
3. Remaining dependencies (pandas, scikit-learn, jinja2 and tqdm) will be automatically installed when installing dtcontrol.

Hardware Requirements

We recommend that dtControl is run on a machine with atleast 4GB RAM and 5GB of free space.

Original Environment

The results shown in the paper were generated on a machine with the following specifications

- OS: Manjaro Linux 18.0.4
- Processor: Intel Xeon W-2123 CPU @ 3.60GHz
- RAM: 64GB, HDD: 1TB
- Dependencies: python 3.7.3, scikit-learn 0.21.3, pandas 0.25.1, numpy 1.16.4, jinja2 2.10.3 and tqdm 4.36.1

The timeout for each experiment was set to 3 hours.

Installing dtControl on your machine

Note: In case of difficulty when following any of the instructions in this section, please check the section ‘Common Installation Issues’ below

1. Make sure you have `python3.6.8` (or newer) and `pip3`

Ubuntu 16.10 or newer:

```
$ sudo apt-get install python3.6 python3-pip
```

MacOS:

```
$ brew install python3
```

2. We use `virtualenv` to make sure that the installation is clean and easy, and does not interfere with the python packages installed in your system. Install `virtualenv` by running

```
$ sudo pip3 install virtualenv
```

Try running `virtualenv` in the console.

3. Then run

```
$ virtualenv -p python3 ~/dtcontrol-venv
```

to create a virtual environment for *dtControl*. This will create the folder `dtcontrol-venv` in your home directoy. After evaluating our artifact, you can delete this folder and thereby all traces of the python packages you installed for the REP.

4. Run

```
$ source ~/dtcontrol-venv/bin/activate
```

to enter the virtual environment. Run `python` and check that the displayed version is greater than 3.6.8 (if not, see (3) in the next section). Press Ctrl+D to exit the python console again.

5. With the virtual environment activated, run

```
$ pip install ~/dtcontrol_rep/dtcontrol_tum-1.0.0rc1-py3-none-any.whl
```

This should install *dtControl* and all its dependencies. Try running *dtControl* by typing `dtcontrol` in the console. It should print the help text.

Common Installation Issues

1. If `sudo apt-get install python3.6` does not work, this might help you: https://askubuntu.com/questions/865554/how-do-i-install-python-3-6-using-apt-get.
2. In case of errors when trying to run `virtualenv`, check that it is located in a directory that is included in your path; this might help you: https://stackoverflow.com/questions/31133050/virtualenv-command-not-found.
3. When trying to run our scripts, if you get the error `dtcontrol: not found`, then make sure you have activated the virtual environment by running step 4 of the previous section.
4. If you don't see what went wrong, leave the virtual environment (run `deactivate`), delete the folder `rm -rf ~/dtcontrol-venv` and go through all the installation steps again. If errors still occur, look at section *Fail safe: Virtual machine*.

Fail safe: Virtual machine

In case you cannot install *dtControl* on you machine, you can use a virtual machine we prepared with *dtControl* pre-installed (i.e. we already executed all five steps of the installation and unpacked the examples). The VM image was tested with VirtualBox 5.1.38.

1. Download the virtual machine from https://syncandshare.lrz.de/getlink/fiNAMxi683APQd8mm78UjCza/dtcontrol_vm.ova (It is based on the virtual machine of the TACAS20 artifact evaluation, as that is very well tested and stable, hence the user name and password is tacas20ae.)
2. Install VirtualBox by downloading it here https://www.virtualbox.org/ and following the installation instructions here https://www.virtualbox.org/manual/ch02.html
3. Open VirtualBox.
4. In the top toolbar, click “File” and select “Import Appliance”. Then select the dtcontrol_vm.ova that you just downloaded.
5. You can change the memory and CPU of the virtual machine as follows:
   Select it, click on settings, and go to the system card. Then you can change RAM by moving the slider for “Base Memory” and CPU by clicking on the Processor tab and then moving the slider for “Processor(s)”. We gave it 4GB RAM and 2 CPUs of a 4 core 3.2 GHz CPU. Make sure to choose something that your host system can handle, i.e. probably do not push the slider into the red region.
6. Start the virtual machine by selecting it and clicking “Start”.
7. After it booted, open a terminal and run `source ~/dtcontrol-venv/bin/activate` to enter the virtual environment where everything is already installed.
8. Then you should be able to run dtControl just by typing `dtcontrol`.

Running the experiments

This section assumes you have installed *dtControl* so that upon entering `dtcontrol` in your command line, the help text is displayed. Before you can run the experiments, you need to unzip all the examples. For this, go to the examples folder and unzip all the files. Note that this might take upto 4GB of space.

```
$ cd ~/dtcontrol_rep/examples
$ unzip '*.zip'
```

The following table gives a mapping between the names of the case studies we use in the paper and the name of the file in the `dtcontrol_rep/examples` directory.

Case Study Name	Controller Filename
cartpole	`cartpole.scs`
2D Thermal	`tworooms-noneuler-latest.dump`
helicopter	`helicopter.scs`
cruise	`cruise-latest.dump`
dcdc	`dcdc.scs`
10D Thermal	`10rooms.scs`
truck_trailer	`truck_trailer.scs`
traffic	`traffic_30m.scs`

Case Study Name	Controller Filename
vehicle	`vehicle.scs`
aircraft	`aircraft.scs`

To continue running the experiments, change your working directory to the `dtcontrol_rep` directory.

```
$ cd ~/dtcontrol_rep
```

Note that running all experiments may take several hours, or possibly run out of memory. A sensible subset of case studies which run quickly (less than 5 minutes per case study/algorithm combination) is: `cartpole`, `tworooms`, `10rooms` and `vehicle` (vehicle not for the linsvm method).

To execute a single algorithm on a single model, run a command like

```
$ dtcontrol -i ~/dtcontrol_rep/examples/cartpole.scs -m linsvm -t 30m --artifact
```

where you can replace `cartpole.scs` and `linsvm` with any example and method respectively.
The following table shows the mapping from the names in the paper to the commandline parameters for dtControl.

Method Name	Method Switch
CART	`-m cart -d none`
LinSVM	`-m linsvm -d none`
LogReg	`-m logreg -d none`
OC1	`-m oc1 -d none`
MaxFreq	`-m cart -d maxfreq`
MaxFreqLC	`-m logreg -d maxfreq`
MinNorm	`-m cart -d minnorm`
MinNormLC	`-m logreg -d minnorm`

Reproducing Table 1

We prepared three shell scripts to make your job simpler.

1. Use

```
$ cd ~/dtcontrol_rep
$ sh experiments_quick.sh
```

to run all the case study/algorithm combinations which take less than 5 minutes; the whole script should not take more than 10 minutes.

2. Use

```
$ cd ~/dtcontrol_rep
$ sh experiments_many.sh
```

to run most of the experiments, but exclude those which need a lot of memory or which need more than 30 minutes. This gives you almost the complete table, but needs ~4 hours.

3. Use

```
$ cd ~/dtcontrol_rep
$ sh experiments_all.sh
```

to run all the experiments that we used to get Table 1 of the paper.
This script uses the timeout of 3 hours that we used in the paper, hence it might take very long, approximately 20 hours.
It also might produce out-of-memory-errors.

In case you want to run any other subset of the experiments, please read the help output of dtControl (`dtcontrol --help`), the User Manual (`user_manual.pdf`) or look at the contents of the above shell scripts to understand how to cal `dtcontrol`.

Common Runtime Issues

If any experiment with OC1 is executed, dtcontrol automatically tries to compile the OC1 C library. In cas `gcc` is not available on your system, this might lead to errors. In such cases, we recommend using our Virtual Machine image (section above).

Reading the output

To get an overview of the results, the file `benchmark.html` is created in the directory from which you call `dtcontrol`. You can open it in any browser. It has the same structure as Table 1 in the paper, but the cells contain more information. The relevant number (which is reported in Table 1) is the number of decision paths (labeled 'paths'). As the algorithms involve some random choices, the number of decision paths can vary, but it should be in the correct order of magnitude. The DOT and C-Code representations of the decision trees are saved in the folder named `decision_trees` located in the folder from which you executed `dtcontrol`. You can also view them from the `benchmark.html` page by clicking "view dot/C file" in any cell. See the User Manual `user_manual.pdf`) for more information on our output.

Note that in the paper, we report several "-" under the OC1 column due to failure to produce a result. Since then, we introduced a fallback mechanism that applies CART when OC1 fails to fit the data, so now in the OC1 column there may be results while in the paper we had "-".

Reproducing Figure 1 and Figure 2a-c

This section assumes that you have activated the virtual environment (Step 4 of Installing dtControl on your machine).

In order to generate the decision trees in Figure 1 and Figure 2a, first run

```
$ cd ~/dtcontrol_rep
$ dtcontrol -i ~/dtcontrol_rep/examples/cartpole.scs -m cart -d maxfreq -t 30m --artifact
$ dtcontrol -i ~/dtcontrol_rep/examples/10rooms.scs -m cart -d maxfreq -t 30m --artifact
```

Then you need to process the DOT output as follows.

1. Install graphviz to visualize the DOT files.

Ubuntu 16.10 or newer:

```
$ sudo apt-get install graphviz
```

MacOS:

```
$ brew install graphviz
```

2. Generate PDFs from the DOT-files

```
$ dot -Tpdf ~/dtcontrol_rep/decision_trees/MaxFreqDT/10rooms/MaxFreqDT.dot -o
~/dtcontrol_rep/Figure1.pdf
$ dot -Tpdf ~/dtcontrol_rep/decision_trees/MaxFreqDT/cartpole/MaxFreqDT.dot -o
~/dtcontrol_rep/Figure2a.pdf
```

Reading the output

You can now view the decision trees by opening Figure1.pdf or Figure2a.pdf in the `~/dtcontrol_rep` directory with your favourite PDF-viewer (e.g. by going there with a file browser and double-clicking on it).

Figure 2b and 2c were generated by manually extracting the information from the decision tree in Figure 2a.
In Figure 2b, there is a box for every leaf node of the decision tree, the boundaries of which are defined by the conditions on the path to the leaf node. Figure 2c contains the mapping of the symbols introduced in Figure 2b to the label of the respective leaf node.

Cleaning Up

You can remove all traces of dtControl by deleting the virtual environment and the extracted folder.

```
$ rm -rf ~/dtcontrol-venv
$ rm -rf ~/dtcontrol_rep
```