LROMP: A Toolset for Robust Optimal Multi-Robot Path Planning Subject to LTL Specifications

Alphan Ulusoy†      Stephen L. Smith\*      Xu Chu Ding†      Calin Belta†

† Hybrid and Networked Systems Laboratory, Boston University, Boston, MA 02215 {alphan(at)bu.edu, xcding(at)bu.edu, cbelta(at)bu.edu}  
\* Dept. of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada {stephen.smith(at)uwaterloo.ca}

**L**TL **R**obust **O**ptimal **M**ulti-Robot **P**lanner (LROMP) is a software package for automatically planning robust optimal paths for a team of robots subject to high level mission specifications. Robots are modeled as weighted transition systems and the mission is given as a linear temporal logic (LTL) formula over a set of propositions satisfied by the regions of the environment. The LTL formula, which is assumed to be trace-closed, is the conjunction of an arbitrary LTL formula, an optimizing proposition and a synchronizing proposition. The paths obtained using our approach are guaranteed to satisfy the LTL formula even with uncertainty in the robots’ traveling times. And, in the absence of such timing errors they are optimal in the sense that the maximum time between any satisfying instances of the optimizing proposition is minimized. LROMP is an implementation of the approach presented in:

A. Ulusoy, S. L. Smith, X. C. Ding and C. Belta (2011) “Robust Multi-Robot Path Planning with Temporal Logic Constraints,” *submitted*.

|  |  |
| --- | --- |
| How to Install | LROMP |

You can download the source code of LROMP RAC along with the Xcode project file if you want to examine and/or modify the code. The Matlab implementation of the Optimal-Run algorithm provided below automatically adds relevant paths so you don’t need to do anything. In order to check whether a given LTL formula is trace-closed or not we use an algorithm adapted from. Make sure that your system meets the requirements given below before attempting to run any of the programs.

**Requirements**

LROMP Region Automaton Constructor (LROMP RAC) requires:

* Mac OS X 10.6 or higher,
* For Xcode projects: Xcode 4,
* For automaton diagrams: ps2pdf and dot tool (comes with the graphviz package, version 2.26.3 or higher), both at /usr/local/bin, and
* For outputs: A folder named LROMP RAC (there is a space between the two words) on the desktop.

The Optimal-Run algorithm implementation requires:

* Matlab 2010b or later,
* A properly configured MEX compiler with openmp support (GCC 4.2.1 or higher), and
* Parallel computation toolbox.

**Sources**  
[LROMP RAC 0.1a (Xcode project)](https://calinbelta.com/wp-content/uploads/2024/01/lromp_rac.zip)

[Optimal-Run Algorithm Implementation (Matlab)](https://calinbelta.com/wp-content/uploads/2024/01/opt_run.zip)

[Trace-Closedness Checker Implementation (Matlab)](https://calinbelta.com/wp-content/uploads/2024/01/trace_closedness.zip)

How to Use

LROMP software package currently consists of three componenets:

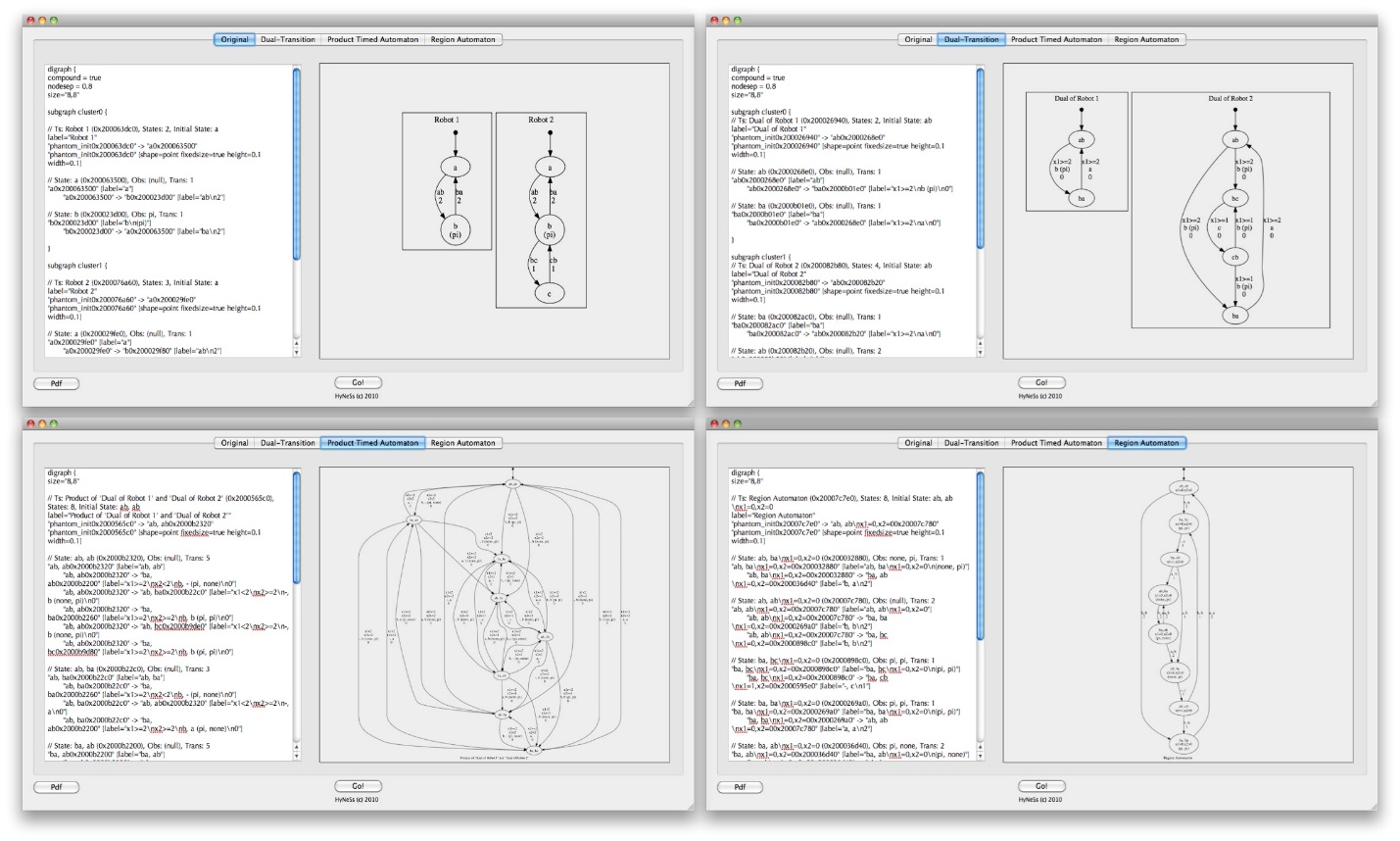
* **LROMP Region Automaton Constructor (LROMP RAC)** for constructing the finite state region automaton that captures the joint behavior of the team.
* **Optimal-Run algorithm implementation** for obtaining a satisfying optimal run on the region automaton constructed by LROMP RAC.
* **Trace-closedness checker**for checking whether a given formula is trace-closed or not.

A typical usage of our software comprises three steps:

1. The user defines the transition systems that model each robot in LROMP RAC (currently hardcoded in lompRacAppDelegate+robotDefs.m). Then, the application creates the region automaton following the approach detailed in and exports an M-file under ~/Desktop/LROMP RAC/ to define the region automaton in Matlab.
2. The user makes sure that the LTL formula to be satisfied by the robotic team is trace-closed using the provided trace-closedness checker
3. Optimal-Run algorithm is executed in Matlab to find the satisfying optimal run on the region automaton, which is then projected onto transition systems of robots to obtain the corresponding run of each robot.

In the following, you can find more details on how to use each individual program.

**LROMP Region Automaton Constructor (LROMP RAC)**



**LROMP Region Autmaton Constructor (LROMP RAC)**

Using LROMP RAC, a user can obtain the region automaton with synchronization states that corresponds to the robots modeled by transition systems by clicking on the Go! button. Following the approach given in [[1]](https://sites.bu.edu/hyness/lromp/#1), LROMP RAC first converts each transition system to a timed automaton, then takes the product of these timed automata to obtain the product timed automaton, after which the region automaton is constructed using a recursive depth-first search on the product timed automaton. The resulting region automaton is a finite bisimulation quotient of the product timed automaton that captures the joint behavior of the team of robots. The transition systems that model each robot are currently hardcoded in lompRacAppDelegate+robotDefs.m.

The user can examine each step of this process by selecting the appropriate tab of the GUI. Each tab features a textual description (in dot language) on the left and an auto-generated diagram on the right. LROMP RAC utilizes the dot tool [[4]](https://sites.bu.edu/hyness/lromp/#4) to visualize transition systems and automata. Once finished, LROMP RAC creates an M-file under ~/Desktop/LROMP RAC/ which defines the resulting region automaton in Matlab.

**Trace-Closedness Checker**[**[3]**](https://sites.bu.edu/hyness/lromp/#2)

The provided implementation checks if an LTL formula is trace-closed for a given distribution. The user defines the distribution in the distribution.txt file and runs the check\_LTL\_traceclosed.m script. Once executed, the script prompts for an LTL formula over the set of propositions defined in the distribution.txt file and outputs whether the given LTL formula is trace-closed or not over the given distribution.

**Optimal-Run Algorithm**[**[2]**](https://sites.bu.edu/hyness/lromp/#2)

The Optimal-Run algorithm implementation expects an automaton (output of LROMP RAC) and an LTL specification (hardcoded in main.m) and outputs a robust optimal run that also satisfies the LTL specification, if one exists. Once the output of LROMP RAC is copied under the ./transition\_systems/ directory and relevant changes are made in the main.m file, our implementation can be run by executing the main.m script. The robust optimal path is displayed once the algorithm finishes. Our Optimal-Run algorithm implementation uses the LTL2BA software [[5]](https://sites.bu.edu/hyness/lromp/#5) to convert LTL specifications to Buchi automata.

Acknowledgements

We would like to thank Jana Tumova for her help with the Matlab implementation of the Optimal-Run Algorithm and Yushan Chen for her help with the implementation of the trace-closedness checker.

**References**

[1] A. Ulusoy, S. L. Smith, X. C. Ding and C. Belta (2011) “Robust Multi-Robot Path Planning with Temporal Logic Constraints,” *submitted*.

[2] S. L. Smith, Jana Tumova, C. Belta, and D. Rus (2010) “Optimal Path Planning under Temporal Logic Constraints,” in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, Taipei, Taiwan, Oct. 2010, pp. 3288-3293.

[3] D. Peled, T. Wilke, and P. Wolper , “Analgorithmic approach for checking closure properties of temporal logic specifications and omega-regular languages,” in *Theor. Comput. Sci.*, vol. 195, no. 2, pp. 183-203, 1998.

[4] “Graphviz – graph visualization software,” <http://www.graphviz.org/>.

[5] “LTL2BA,” [http://www.lsv.ens-cachan.fr/˜gastin/ltl2ba/index.php](http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/index.php).