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

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

† 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}
‡ Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 {rus(at)csail.mit.edu}

**L**TL **O**ptimal **M**ulti-Robot **P**lanner (LOMP) is a software package for optimal path planning of a team of robots subject to high-level specifications. Robots are modeled as transition systems and specifications are expressed in linear temporal logic (LTL). The LTL specification is a conjunction of an arbitrary LTL formula and a optimizing proposition. The runs obtained using our approach are optimal in the sense that the maximum time between any satisfying instances of the optimizing proposition is minimized. LOMP is an implementation of the approach presented in:
[[1]](https://sites.bu.edu/hyness/lomp/#1) A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus (2011) “Optimal Multi-Robot Path Planning with Temporal Logic Constraints,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, San Fransisco, CA, USA, 2011.

|  |  |
| --- | --- |
| How to Install   | LOMP |

You can run the binaries (.app files) of LOMP RAC and LOMP Simulator from anywhere in your computer by double-cicking on them. The Matlab implementation of the Optimal-Run algorithm [[2]](https://sites.bu.edu/hyness/lomp/#2) provided below automatically adds relevant paths so you don’t need to do anything. You can also download the source code of LOMP RAC and LOMP Simulator along with the Xcode project file if you want to examine and/or modify the code. Make sure that your system meets the requirements given below before attempting to run any of the programs.

**Requirements**

LOMP Region Automaton Constructor (LOMP 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 LOMP RAC (there is a space between the two words) on the desktop.

LOMP Simulator requires:

* Mac OS X 10.6 or higher.

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

[LOMP RAC 0.1a (Xcode project)](https://calinbelta.com/wp-content/uploads/2023/12/lompRac.zip)

[LOMP Simulator 0.1a (Xcode project)](https://calinbelta.com/wp-content/uploads/2023/12/lompSimulator.zip)

[Optimal-Run Algorithm Implementation (Matlab)](https://calinbelta.com/wp-content/uploads/2023/12/optRun.zip)

Binaries

[LOMP Region Automaton Constructor (RAC) 0.1a](https://calinbelta.com/wp-content/uploads/2023/12/LOMP_RAC.app_.zip)

[LOMP Simulator 0.1a](https://calinbelta.com/wp-content/uploads/2023/12/LOMP_Simulator.app_.zip)

How to Use

LOMP software package currently consists of three components:

* **LOMP Region Automaton Constructor (LOMP 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 LOMP RAC.
* **LOMP Simulator** for simulating the satisfying optimal run.

A typical usage of our software comprises three steps:

1. The user defines the transition systems that model each robot in LOMP RAC (currently hardcoded in lompRacAppDelegate+robotDefs.m). Then, the application creates the region automaton following the approach detailed in [[1]](https://sites.bu.edu/hyness/lomp/#1) and exports an M-file under ~/Desktop/LOMP RAC/ to define the region automaton in Matlab.
2. 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.
3. Finally, the resulting motion of the team is demonstrated in LOMP Simulator (trajectories are currently hardcoded in robot.m).

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

**LOMP Region Automaton Constructor (LOMP RAC)**



LOMP Region Autmaton Constructor (LOMP RAC)

Using LOMP RAC, a user can obtain the region automaton R 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/lomp/#1), LOMP 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. LOMP RAC utilizes the dot tool [[3]](https://sites.bu.edu/hyness/lomp/#3) to visualize transition systems and automata. Once finished, LOMP RAC creates an M-file under ~/Desktop/LOMP RAC/ which defines the resulting region automaton in Matlab.

**Optimal-Run Algorithm**

The Optimal-Run algorithm implementation expects an automaton (output of LOMP RAC) and an LTL specification (hardcoded in main.m) and outputs an optimal run that also satisfies the LTL specification, if one exists. Once the output of LOMP 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 optimal path is displayed once the algorithm finishes. This optimal path can be translated to Objective-C code using the results\_parser.m script, and then can be embedded in the LOMP Simulator to simulate the satisfying optimal run of the robots. Our Optimal-Run algorithm implementation uses the LTL2BA software [[4]](https://sites.bu.edu/hyness/lomp/#4) to convert LTL specifications to Buchi automata.

**LOMP Simulator**



LOMP Simulator

The output of the result\_parser.m script can be used to define the robot trajectories to be simulated in LOMP Simulator (currently hardcoded in robot.m). Currently, our simulator supports only two robots traveling on a particular environment, but can be easily expanded to simulate an arbitrary number of robots on an arbitrary map.

Acknowledgements

We would like to thank Jana Tumova for her help with the Matlab implementation of the Optimal-Run Algorithm.

References

[1] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus (2011) “Optimal 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] “Graphviz – graph visualization software,” <http://www.graphviz.org/>.

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