# Resource Allocation GUI Manual [*]

Rahul Chandan Dario Paccagnan Jason R. Marden

November 2019

## Contents

## 1 Introduction

In our previous work, we have derived tractable linear programs for computing and optimizing worst-case efficiency bounds of distributed resource-allocation problems, including routing problems, probabilistic-objective problems, and coverage problems. This MATLAB® graphical user-interface (GUI) has been developed as an out-of-the-box option for the interested, but time-sensitive, reader. Our hope is that this tool will make our linear programs more accessible, and will enrich the reader's understanding of our results.

## 2 Installation

Use of this tool requires an installation MATLAB® (R2018a or higher recommended) with the Optimization Toolbox and Symbolic Math Toolbox. Install the application by navigating to the downloaded ‘.mlappinstall’ file in the MATLAB® file browser, and double-clicking the icon.

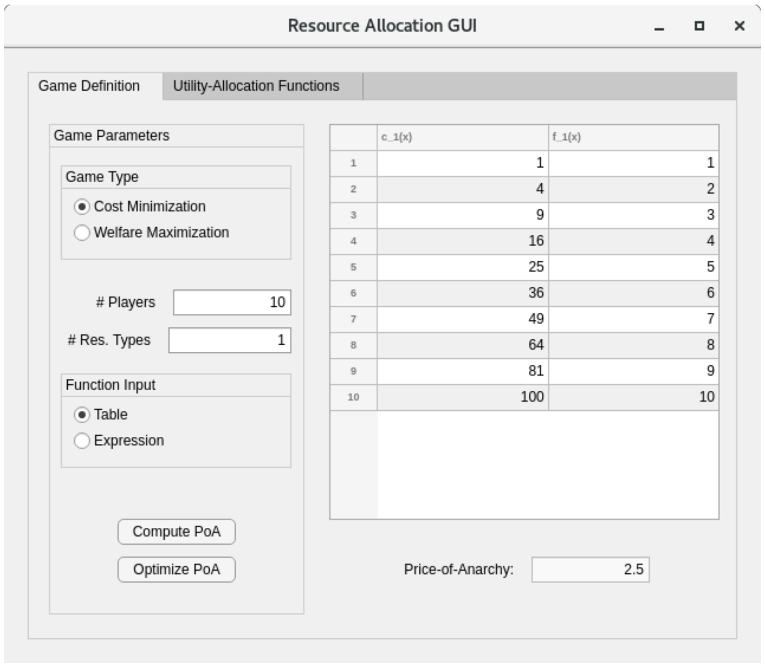
## 3 Operation

Open the application by navigating to the ‘APPS’ tab in MATLAB®, and selecting the ‘Resource Allocation GUI’ icon. This will load the GUI in its default state, as shown in Fig. 1. Observe that, by default, the radio button corresponding to ‘Cost Minimization’ is selected in the ‘Game Type’ button group, the ‘# Players’ field is initialized to 10, the ‘# Res. Types’ is initialized to 1, and the ‘Table’ radio button is selected in the ‘Function Input’ button group. This translates to a ten-player cost-minimization game with one resource type. The local cost function ‘c_1(x)’ and the utility-allocation function ‘f_1(x)’ have been specified in the UITable, and are quadratic and linear in ‘x’, respectively.

[*]The MATLAB® installer for this application is freely available at . The underlying scripts with examples are available for MATLAB® and Python at https://github.com/rahul-chandan/resalloc-poa. Please direct any feedback to Rahul Chandan at rchandan@ucsb.edu.

Clicking ‘Compute PoA’ triggers the code for characterizing the price-of-anarchy of the class of games, as specified by the game type, the number of players and resource types, and the local cost/welfare and utility-allocation functions. The computed price-of-anarchy is displayed in the ‘Price-of-Anarchy’ field, at the bottom right. Observe that the default game has a price-of-anarchy of 2.5. Clicking ‘Optimize PoA’ triggers the code for optimizing the price-of-anarchy of the specified class of games. The computed optimal price-of-anarchy is displayed in the ‘Price-of-Anarchy’ field, and the optimal utility-allocation functions corresponding to the local cost/welfare functions specified are plotted in the UIPlot under the ‘Utility-Allocation Functions’ tab, as in Fig. 2.

Figure 1: The default GUI window, open to the ‘Game Definition’ tab. Currently, a cost minimization game with ten players and one resource type is defined. The functions $c_1(x)$ and $f_1(x)$ have been specified via the ‘Table’ option for function input. The price-of-anarchy of 2.5 is displayed when the ‘Compute PoA’ button is pressed. For this game, after pressing ‘Optimize PoA’, a price-of-anarchy of 2.012 will be displayed, and the optimal utility-allocation function willl be plotted under the ‘Utility-Allocation Functions’ tab.



Game Type: [✓ Cost Minimization] Welfare Maximization

# Players: 10

# Res. Types: 1

Function Input: [✓ Table] Expression

| | c_1(x) | f_1(x) |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 4 | 2 |
| 3 | 9 | 3 |
| 4 | 16 | 4 |
| 5 | 25 | 5 |
| 6 | 36 | 6 |
| 7 | 49 | 7 |
| 8 | 64 | 8 |
| 9 | 81 | 9 |
| 10 | 100 | 10 |

Price-of-Anarchy: 2.5

## 3.1 Function Input

The local cost/welfare and utility-allocation functions can be specified either as vectors under the ‘Table’ setting of the ‘Function Input’ button group, or as functions of ‘x’ under the ‘Expression’ setting.

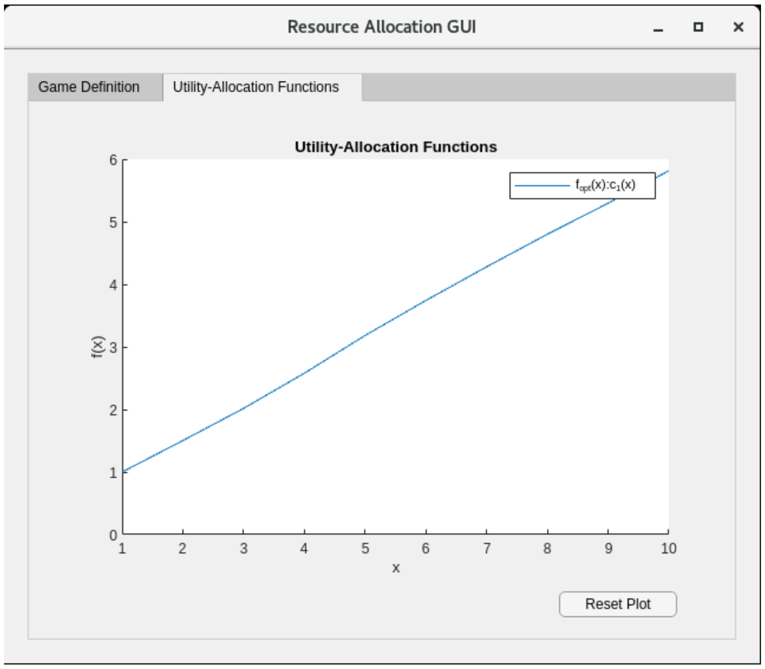
When the ‘Table’ setting is selected, the UITable has ‘# players’ rows, and ‘2(# Res. Types)’ columns. Each of the cells in the UITable can be modified to specify the functions as desired. Under the ‘Expression’ setting, the UITable has one row, and ‘2(# Res. Types)’ columns. The expressions in each cell are parsed using the ‘str2sym’ function provided by the Symbolic Math Toolbox$^{\text{TM}}$.

When switching from the ‘Expression’ to ‘Table’ setting, the GUI will automatically fill the columns of the UITable according to the corresponding symbolic functions specified. Unfortunately, this is practically impossible to do when switching from the ‘Table’ to the ‘Expression’ setting.

## 3.2 The ‘Utility-Allocation Functions’ tab

The UIPlot under this tab is automatically populated with the optimal utility-allocation functions generated when the ‘Optimize PoA‘ button under the ‘Game Definition’ tab is pressed. The labels are generated using the column names in the UITable, under the other tab. To clear the plot, simply click the 'Reset Plot' button, at the bottom right.

Figure 2: The 'Utility-Allocation Functions' tab, with the plot of the optimal utility-allocation function for $c_1(x) = x^2$, and ten players. The plot can be cleared by pressing the 'Reset Plot' button on the bottom-right.



## 4 Copyright Notice

We are by no means experienced programmers, and we welcome any and all feedback concerning this application. Though we are very interested in maintaining and improving this code, we cannot guarantee that your feedback will be addressed in a timely manner. Nevertheless, please feel free to contact us by email at rchandan@ucsb.edu.