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 of 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

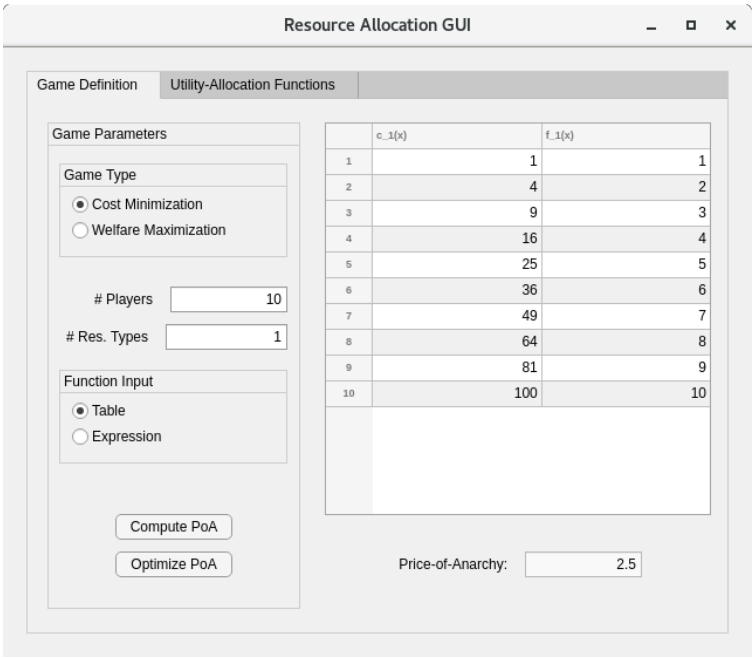
Open the application by navigating to the 'APPS' tab in MATLAB®, and selecting the 'Resource Allocation GUI' icon. This will load the GUI in its default state, as shown in Fig. 1. Observe that, by default, the radio button corresponding to 'Cost Minimization' is selected in the 'Game Type' button group, the '# Players' field is initialized to 10, the '# Res. Types' is initialized to 1, and the 'Table' radio button is selected in the

*The MATLAB® installer for this application is freely available at https://github.com/rahul-chandan/resalloc-gui. 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.

'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.

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.



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.

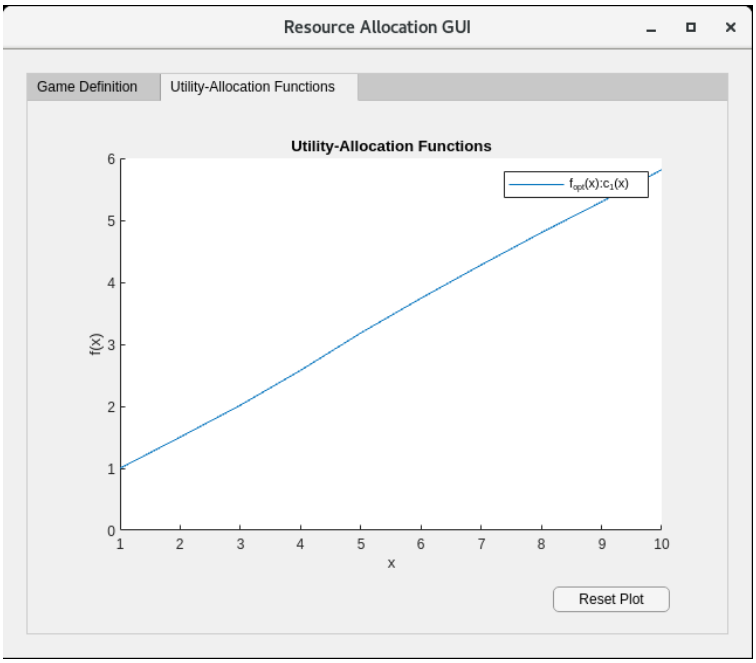
When the 'Table' setting is selected, the UITable has '# players' rows, and '2(# Res. Types)' columns. Each of the cells in the UITable can be modified to specify the functions as desired. Under the 'Expression' setting, the UITable has one row, and '2(# Res. Types)' columns. The expressions in each 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 Examples

Here we demonstrate how one can reproduce existing results in three well-studied classes of games; congestion games, covering games, and probabilistic-objective games. These correspond to the classes of games used in [1] as illustrative examples.

4.1 Congestion games

Here, we consider the class of cost-minimization games called *congestion games*, as investigated in [2, 3, 4]. The class of polynomial congestion games of order $d \geq 1$ is equivalent to the scalable class of local cost-sharing games with basis set $B = \{(x, 1), \ldots, (x^{d+1}, x^d)\}$, i.e. all edges $e \in E$ of a polynomial congestion game of order d have latency functions of the form $\ell_e(x) = \sum_{k=1}^{d} \alpha_{e,k} x^k$, and $c_e(x) = \ell_e(x)x$, where all coefficients $\alpha_{e,k} \geq 0$. This class of games can be encoded into the GUI as shown in Fig. 3. In this particular case, we examine the class of polynomial congestion games of order $d = 2$ with 25 players, and enter the definitions for the latency and cost functions using the 'Expression' option for function input. After clicking 'Compute PoA', the price of anarchy of 9.583 will appear in the 'Price-of-Anarchy' field.

4.2 Covering games

Consider the κ-coverage problem, introduced in [5]. As we define in [1], a κ-coverage game is a welfare maximization game with $w^{\ell}(x) = \max\{x, \ell\}$, where $\kappa \geq 1$ is a positive integer. Observe in Fig. 4 that we have

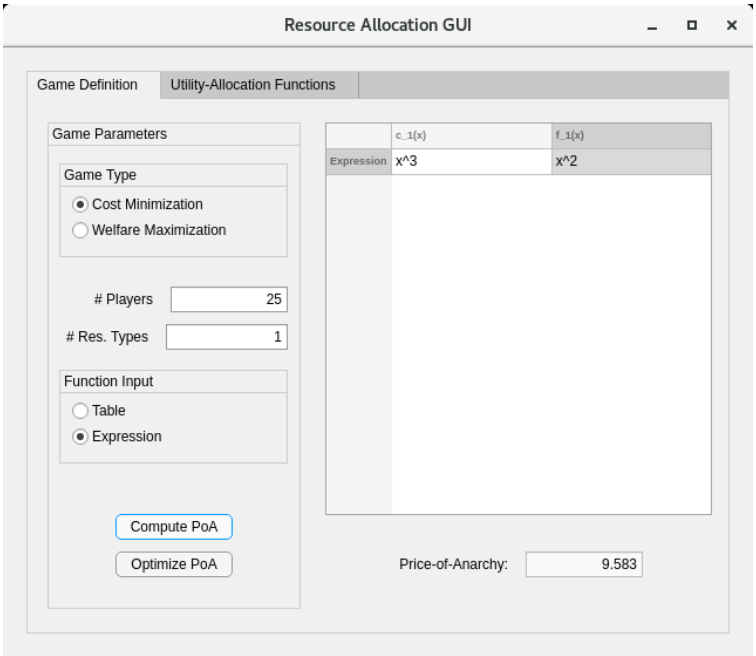


Figure 3: Configuration of the GUI to compute the price of anarchy of a 25-player polynomial congestion game with order $d = 2$.

encoded the κ-coverage game with $\kappa = 5$ and 25 players into the GUI, using the 'Table' option for function input. After clicking 'Optimize PoA', the price of anarchy of 1.213 will appear in the 'Price-of-Anarchy' field, and the optimal utility-allocation function is plotted on the UIPlot under the 'Utility-Allocation Functions' tab. The price of anarchy of classes of games with multiple values for $\kappa \geq 1$ can be computed by defining further columns.

4.3 Probabilistic-objective games

For the final example, we consider the class of probabilistic-objective games, which includes vehicle-target assignment, animal dispersal, and credit assignment games. As defined in [1], a probabilistic-objective game is a welfare maximization game with $w^q(x) = 1 - q^x$, where $q \in [0, 1]$ is the probability of failure. In Fig. 5, we have encoded the class of probabilistic-objective games with $q = 0.5$ and 25 players into the GUI, using the 'Expression' option for function input. We have defined the function f_1(x) as the equal-shares function, i.e., $f_1(x) = w^q(x)/x$. Pressing 'Compute PoA', the price of anarchy of 1.778 corresponding to the equal-shares utility-allocation function will appear in the 'Price-of-Anarchy' field. After clicking the 'Optimize PoA' button, the optimal price of anarchy of 1.287 will appear in the 'Price-of-Anarchy' field, and the optimal utility-allocation function is plotted on the UIPlot under the 'Utility-Allocation Functions' tab. The price of anarchy of classes of games with multiple values for $q \in [0, 1]$ can be computed by defining further columns.

5 Copyright Notice

This MATLAB® application has been developed as a companion for our papers on local resource-allocation games (e.g., Optimal mechanisms for distributed resource-allocation). This application is freely available, and is covered by the GNU Standard License, Version 3 (GPLv3). You are encouraged to download, develop and redistribute this application so long as you abide by the conditions of the license agreement (e.g., do not commercialize any software developed from this code).

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

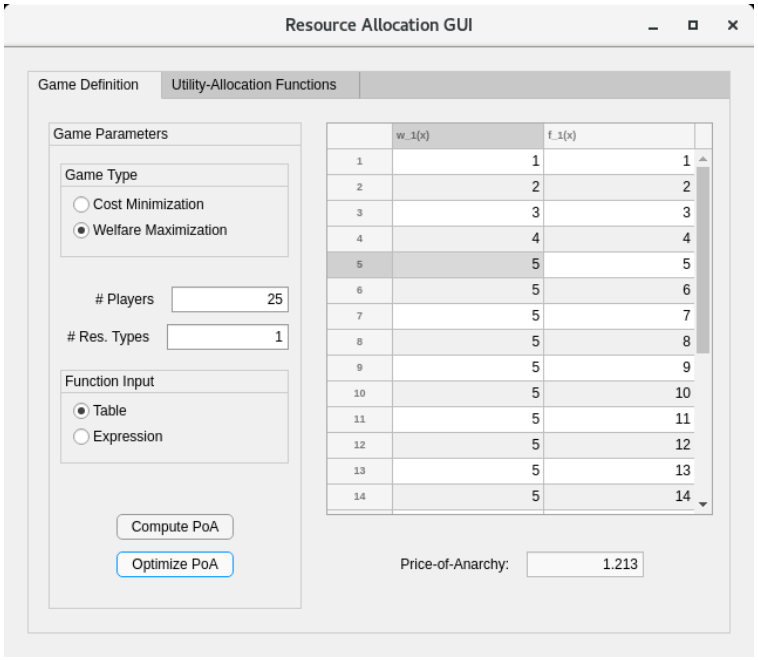


Figure 4: Configuration of the GUI for optimizing the price of anarchy of a 25-player, κ-coverage game for $\kappa = 5$. The optimal utility-allocation function is plotted under the 'Utility-Allocation Functions' tab.

that your feedback will be addressed in a timely manner. Nevertheless, please feel free to contact us by email at rchandan@ucsb.edu.

References

[1] R. Chandan, D. Paccagnan, and J. R. Marden, "Optimal mechanisms for utility design," *Submitted*, 2019.

[2] B. Awerbuch, Y. Azar, and A. Epstein, "The price of routing unsplittable flow," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pp. 57–66, ACM, 2005.

[3] G. Christodoulou and E. Koutsoupias, "The price of anarchy of finite congestion games," in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pp. 67–73, ACM, 2005.

[4] S. Aland, D. Dumrauf, M. Gairing, B. Monien, and F. Schoppmann, "Exact price of anarchy for polynomial congestion games," *SIAM Journal on Computing*, vol. 40, no. 5, pp. 1211–1233, 2011.

[5] S. Barman, O. Fawzi, S. Ghoshal, and E. Gürpınar, "Tight approximation bounds for maximum multi-coverage," *arXiv preprint arXiv:1905.00640*, 2019.

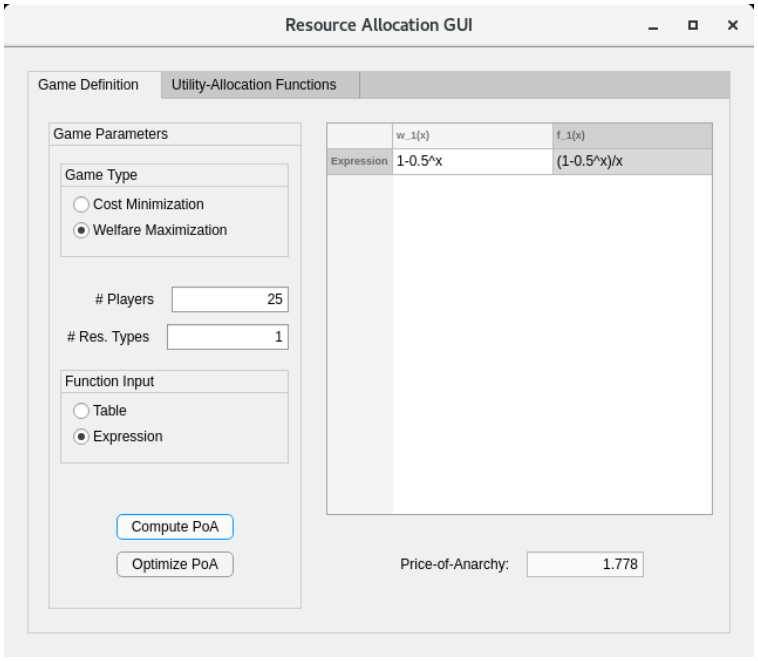


Figure 5: Configuration of the GUI for computing and optimizing the price of anarchy of a 25-player, probabilistic-objective game for $q = 0.5$. The price of anarchy is computed for the equal-shares utility-allocation function, $f(x) = w^q(x)/x$.