A Example games included in Theorem 4.2

In this section, we provide three example level games that are weighted and ordinal potential games and non-potential games (one each).

These examples are specifically and uniquely constructed to demonstrate the existence of level games within each of these classes. As a result, they may not resemble real-world games. Characterising existing games as level games is left for future work.

A.1 Weighted Potential Game

To create the weighted potential game we use the weights $w_a = 1$ and $w_b = 1.5$ and the following potential

$$P = \begin{pmatrix} 6 & 1 & 0 \\ 3 & 2 & 1 \\ 0 & -2 & -3 \end{pmatrix} \tag{3}$$

obtaining,

	a_1	a_2	a_3
b_1	9,6	7.5,1	6,0
b_2	4.5,5	9,4	7.5,3
b_3	0,4	3,2	1.5,1

This is also a level game with the the following partition:

$$A_1 := \{(a_3, b_3)\} \qquad A_2 := \{(a_2, b_3)\}$$
$$A_3 := \{(a_1, b_3)\} \qquad A_4 := \{(a_3, b_1)\}$$
$$A_5 := \{(a_2, b_1)\} \qquad A_6 := \{(a_3, b_2)\}$$
$$A_7 := \{(a_2, b_2)\} \qquad A_8 := \{(a_1, b_2)\}$$
$$A_9 := \{(a_1, b_1)\}.$$

A.2 Ordinal Potential Game

Using the same potential and the same partition we can create the following level game that is also an ordinal potential game:

	a_1	a_2	a_3
b_1	14,6	10,1	6,0
b_2	12,5	12,4	7.5,3
b_3	0,4	4,2	1.5,1

A.3 Non-potential Game

The following is a simple two player game that is a level game and a non-potential game.

	a_1	a_2	a_3
b_1	6,6	4,1	3,2
b_2	3,5	3,2	4,1
b_3	0,4	0,2	0,1

We can see that this is not a potential game because there is a cycle between (a_2, b_1), (a_3, b_1), (a_3, b_2) and (a_2, b_2). In addition the following partition meets the criteria of a level game:

$$A_1 := \{(a_3, b_3)\}$$
$$A_2 := \{(a_2, b_3)\}$$
$$A_3 := \{(a_1, b_3), (a_2, b_1), (a_3, b_1), (a_3, b_2), (a_2, b_2)\}$$
$$A_4 := \{(a_1, b_2)\}$$
$$A_5 := \{(a_1, b_1)\}.$$

B Problem and Algorithm Settings, and Experimental Implementation Details

In this section we will explain the experimental implementation of the algorithms as well as the problem and algorithm settings used in the experiments of the paper. All the experiments were run using CPUs provided by HPC facilities at the University of Birmingham (see http://www.birmingham.ac.uk/bear for more information).

B.1 Singleton Congestion Games

For the experiments we implement a Better Response algorithm (BR) that first chooses a random initial strategy profile, then each iteration it creates a new strategy profile by changing the pure strategy of one player to a new pure strategy picked u. a. r., if the cost of the player decreases we keep the new strategy profile.

The experiments on singleton congestion games consist of 50 independent trials per algorithm on the same random SCG instance [6]. The SCG instance is created by populating a cost matrix $C = I \times k$ (Section 4.1) with random values, $C(r, j) \in [0, 1]$. For NCP-CoEA the mutation rate is $\chi = 0.5$, based on Assumption 2 and a range of population sizes $\lambda = \{10, 25, 50, 100\}$.

B.2 Dynamic Routing Games

For all experiments regarding dynamic routing games we use the implementation from OpenSpiel [7]. The code is provided as supplementary material.

The experiments on dynamic routing games consist of 100 independent trials per algorithm on the same game instance.

We measure the exploitability and runtime with different number of vehicles ($\{1, 2, \ldots, 9, 10, 20, \ldots, 100\}$ for the Pigou and Braess network and 150 for the Augmented Braess network). Note, exploitability is expensive to compute so we only compute it for up to 5 vehicles. Unlike the other algorithms, we let the NCP-CoEA run until a Nash equilibrium is found.

We compare with the following algorithms within the OpenSpiel library:

- Fictitious play
- Counterfactual regret minimisation (CFR)
- Counterfactual regret minimisation with external sampling (ext CFR) [37]

All of these solve the same game and use the default parameter settings and run for 10 iterations each. For NCP-CoEA we did a systematic parameter tuning on the Braess network with 40 vehicles and used the following resulting parameters for all experiments:

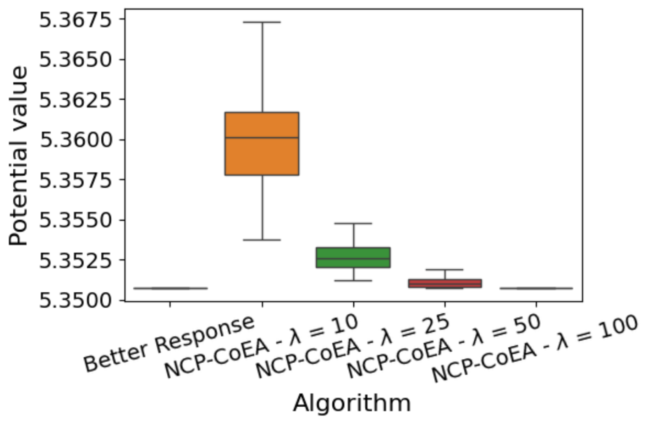
- mutation rate χ: 0.1,

[6]Plots for other instances are shown in Appendix D.
[7]https://github.com/deepmind/open_spiel/tree/master/open_spiel/data/paper_data/routing_game_experiments

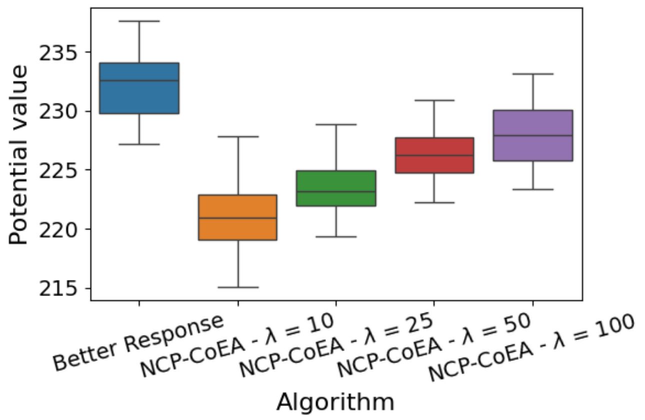
- population size: max (4, ⌈2 ln(# vehicles)⌉).

C Experimental Statistical Analysis

We used a Wilcoxon rank sum test with Bonferroni correction for the analysis of statistical significance. For Monotone SCG, Figure 4a there are significant differences among all algorithm variants (p-value < 0.00001) except for Better Response and NCP-CoEA - λ = 100, p-value = 0.0046 and Bonferroni corrected p-value = 0.0456. For SCG, Figure 4b there are significant differences for all (p-value < 0.00001).



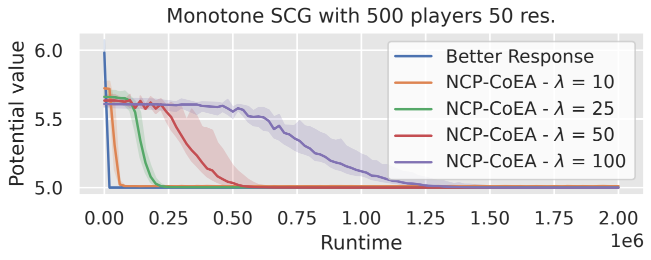
(a) Monotone SCG



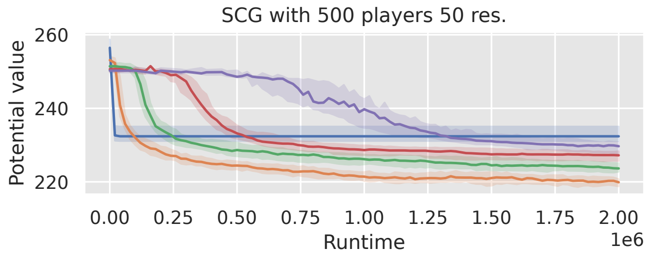
(b) SCG

Figure 4: Potential value boxplots for random SCG instances with $I = 500, k = 50$ at $t = 2e6$. X-axis shows the algorithm variant. Y-axis shows the potential value.

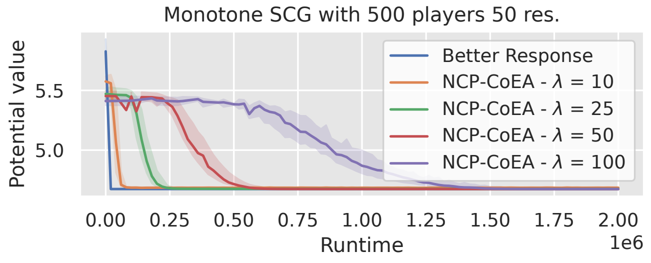
D Experiments on Other Random Instances for SCG



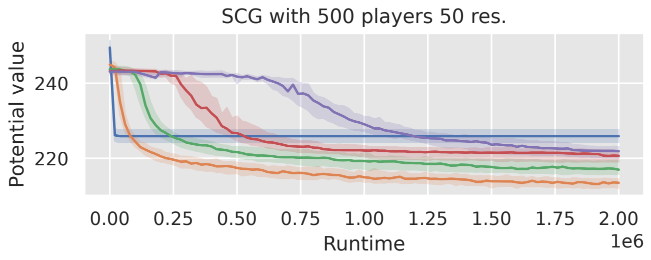
(a) Monotonic SCG



(b) SCG

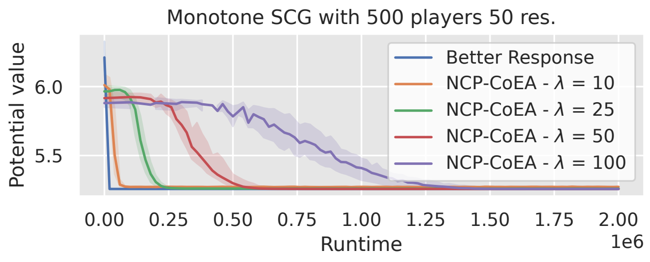


(c) Monotonic SCG

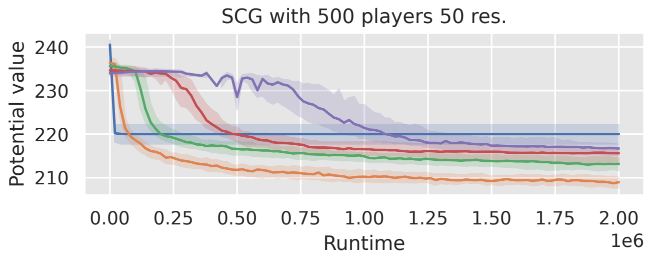


(d) SCG

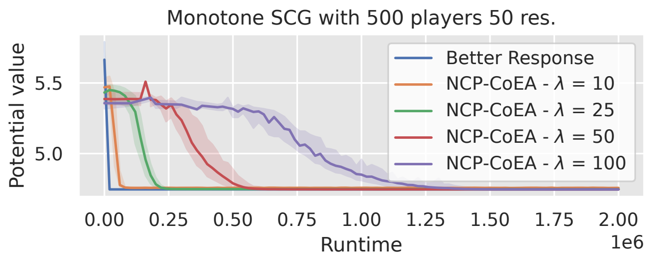
Figure 5: Potential value for random SCG instances with $I = 500, k = 50$. Y-axis shows the potential value. X-axis shows the runtime. Lines show the median best individual with interquartile ranges for the algorithm variants. Different seeds for the creation of the random instances.



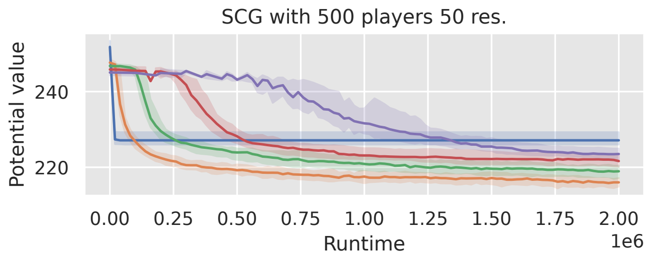
(a) Monotonic SCG



(b) SCG



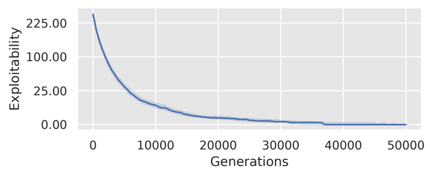
(c) Monotonic SCG



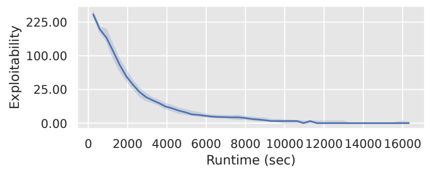
(d) SCG

Figure 6: Potential value for random SCG instances with $I = 500, k = 50$. Y-axis shows the potential value. X-axis shows the runtime. Lines show the median best individual with interquartile ranges for the algorithm variants. Different seeds for the creation of the random instances.

E Other Experiments on Dynamic Routing Games



(a) Exploitability Augmented Braess



(b) Runtime Augmented Braess

Figure 7: Runtime for different dynamic routing game networks. Y-axis shows the exploitability. X-axis shows the number of generations/runtime. Lines show the median best individual with interquartile ranges for the algorithm variants.