Problem Statement
Computing Nash equilibria using GPUs

- Application Constraints: Real world strategic interactions usually require the modeling of large number of agents having a large number of choices or actions.
- Processing Constraints: Computing all Nash equilibria for large bimatrix games using single-processor computers is computationally expensive.

Nash Equilibria in Bimatrix Games
Bimatrix Game \( \Gamma(A, B) \)
- A set of two players: \{Player 1, Player 2\}
- A finite set of actions for each player
  - Player 1’s actions: \( M = \{s_1, s_2, \ldots, s_m\} \)
  - Player 2’s actions: \( N = \{t_1, t_2, \ldots, t_n\} \)
- Player payoff matrices \( A, B \in \mathbb{R}^{m \times n} \)

Strategies \( \{x, y\} \)
- Strategies are probabilities representing player’s choice of actions.
- \( x = \{x_1, \ldots, x_m\} \) s.t. \( \Pr\{(\text{Player 1}) \leftarrow s_i\} = x_i \)
- \( y = \{y_1, \ldots, y_n\} \) s.t. \( \Pr\{(\text{Player 2}) \leftarrow t_j\} = y_j \)

Support Key Set & Block Distribution
- Set of Support keys \( \mathcal{S}^x \)
- Support size \( i \)
- Co-lexicographical order.
- \( B_k \) blocks select strategies for Player 1
- \( i \) support size
- \( j \) order of permutation

Thread Distribution
- \( \text{Threads} \) \( t_j \) selects strategies for Player 2
- \( \text{Processes} \) strategy pairs
- \( \text{Computes} \) \( (x, y) \) probabilities
- \( \text{Support} \) size \( i \)
- \( \text{Thread} \) index \( k \) in block \( j \)

Computing Nash Equilibria

- Pure \((A, B, q, \phi)\): Computes pure Nash Equilibria in \( \Gamma(A, B) \).
  A pure strategy \( x \) is Nash equilibrium strategy where players choose a single action with probability 1.
- Mixed \((A, B, k, q, \theta)\): Computes mixed Nash Equilibria in \( \Gamma(A, B) \).
  A mixed strategy \( x \) is Nash equilibrium where players choose actions according to a probability distribution over their pure actions.

Computing Nash Equilibria in Bimatrix Games
GPU-based Parallel Support Enumeration
Input: Player 1 payoff, Player 2 payoff \((A, B)\)
Output: Set of equilibria \( \phi \)
1: \( \phi = \emptyset \)
2: \( q = \min(m, n) \)
3: \( \Theta = \text{Generate}(1, q) \)
4: \( \phi = \text{Pure}(A, B, q, \Theta) \)
5: \( \text{for } k = 2, \ldots, q \)
6: \( \Theta = \text{Generate}(k, q) \)
7: \( \phi = \phi \cup \text{Mixed}(A, B, k, q, \Theta) \)
8: output \( \phi \)

Conclusion
- GPU processing outperforms OpenMP implementations for computing equilibria in larger games.
- GPU speedups range from 144.07 to 1013.53 against OpenMP configurations from 1 to 16 CPUs.

Acknowledgment
This research was partially supported by NSF grants DGE-0854014 and CNS-1116787.