Project Parlay

James Tran, Jason Bohlinger, Andrew Broek,
Max Hartel

Concept

This application is meant to take user input for the capital they'd like place down on what games and tell them the best way to split that bet (called a Parlay) to win back the most capital.

Background and Literature

- What is a parlay?
  - a single bet that links together two or more individual wagers and is dependent on all of those wagers winning together
- What is simulated annealing?
  - A more advanced hill climbing algorithm that aims to find an approximate maximum of a function by comparing neighbor-states and progressively moving to higher values within an error threshold

Implementation (Front-end and Back-end)

- Front-End has two options
  - Tkinter
  - HTML
- SQLite for backend
  - Database full of game info that is queried by our program
- Supported by simulated annealing algorithm
- Data Structures:
  - Lists
  - 2D Lists

Life cycle of a user input:

The user is asked to enter games they would like to bet on, as well as the games odds:

Game 1:	
Team:	Implied Win Percentage:
Rams	36%
Chargers	64%

Game 2:	
Team:	Implied Win Percentage:
Cardinals	40%
Seahawks	60%

Game 2:	
Team:	Implied Win Percentage:
Packers	71%
Bears	39%

Game 1:	
Team:	Implied Win Percentage:
Patriots	49%
Dolphins	51%

Game 1:	
Team:	Implied Win Percentage:
Vikings	74%
Cowboys	26%

Life cycle of a user input:

The User Input is then transformed into objects that are can be used by the program:

Game 1:	
Team:	Implied Win Percentage:
Rams	36%
Chargers	64%

Life cycle of a user input:

Game 1:	
Team:	Implied Win Percentage:
Rams	36%
Chargers	64%

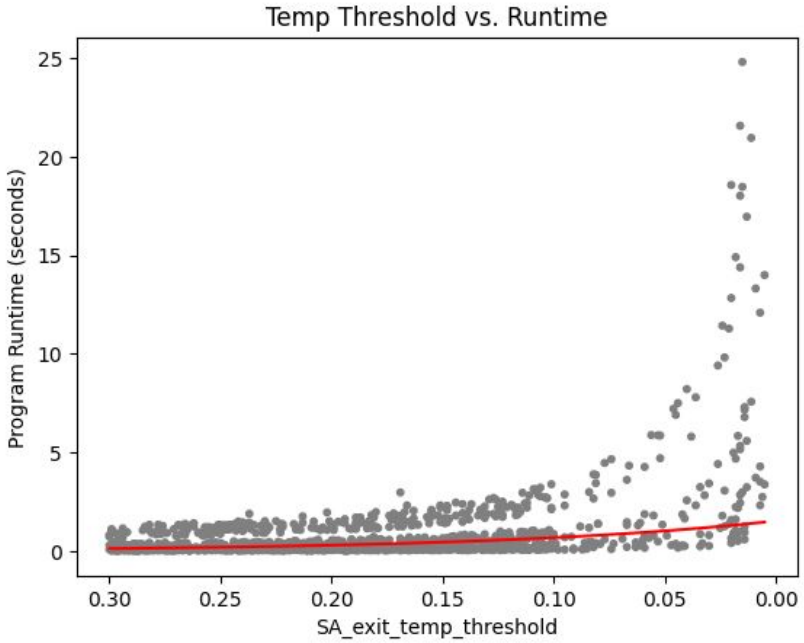
```
game(favoredTeam, underdog, favWinP, gameID)
```

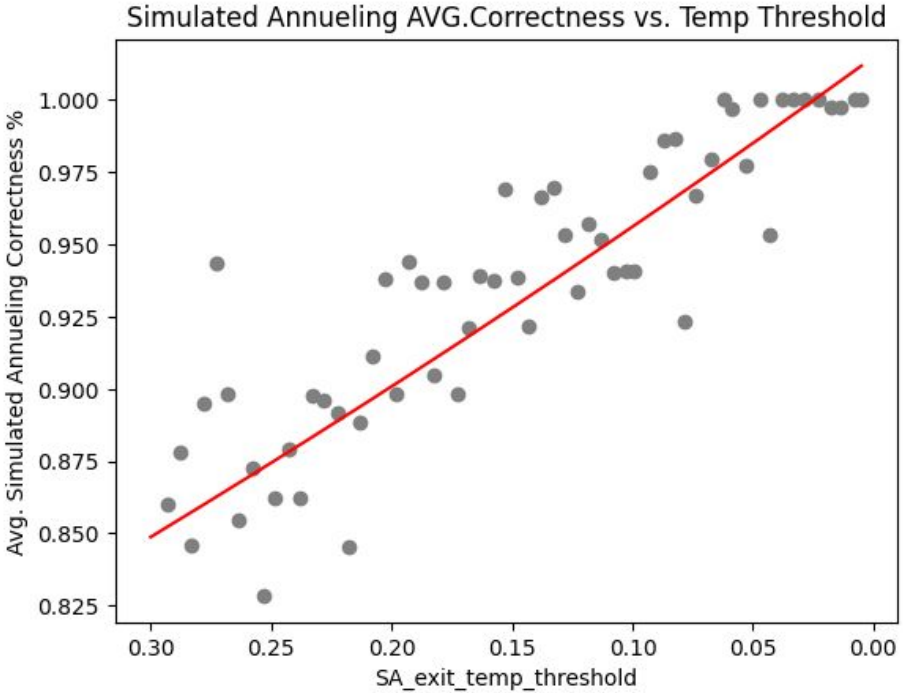
```
outcome(favWinP, favoredTeam, gameID)
outcome((100 - favWinP), underdog, gameID)
```

`OutcomeList[]` → `Simulated_Annealing()`

Simulated Annealing:

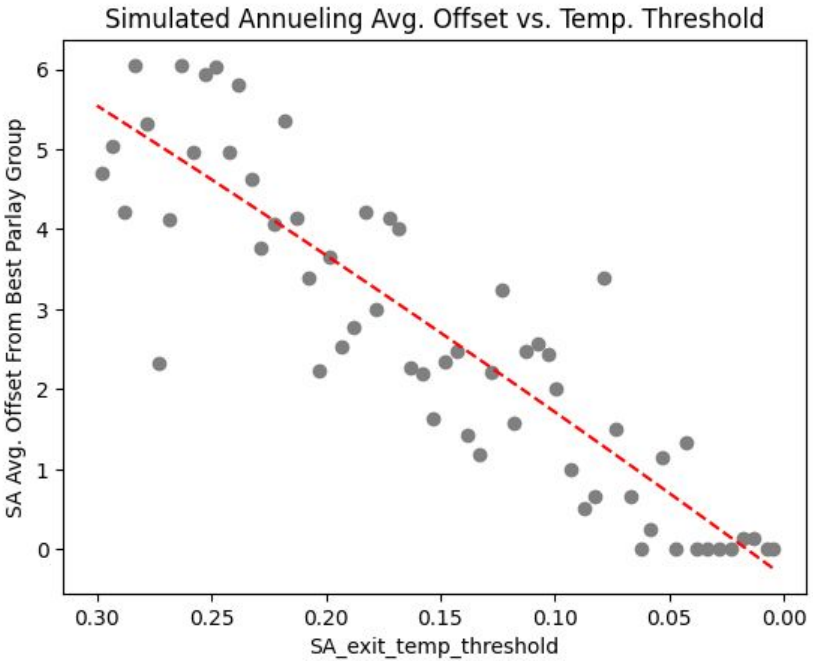
Testing and Analysis:



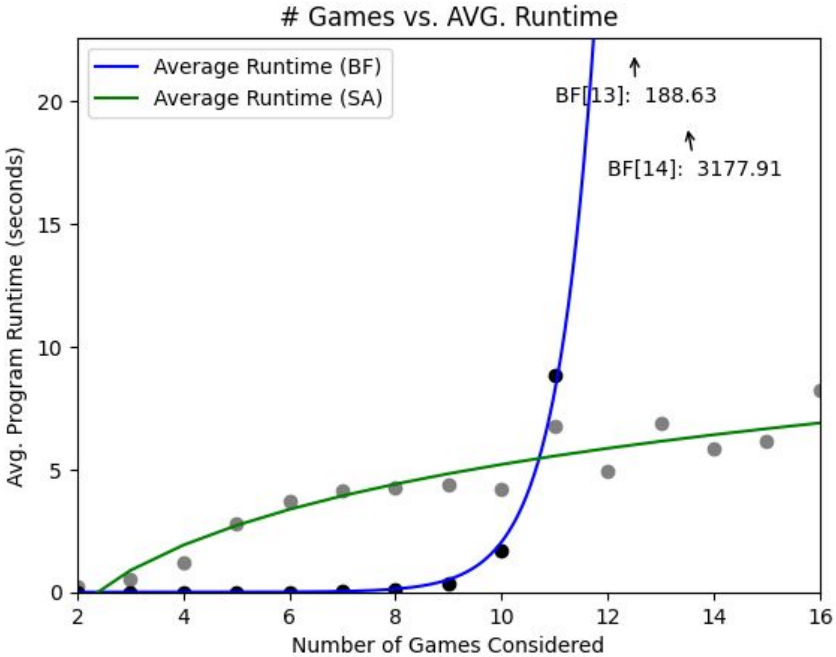


We found optimal temp threshold to be 0.03, with an Average Correctness of 98%, and average runtime of 1.6 seconds

Testing and Analysis:



With a Threshold of 0.03, there was an average of >1 Offset from the best possible result



Our implementation of SA has logarithmic runtime, meaning that scaling to a larger pool of games to consider does not significantly affect performance

It was impossible to test SA for correctness after the number of games made the brute force method no longer viable, however there is no indication SA would perform worse with larger game pool

Demo

Results and Achievements

- We were able to accomplish the main feature of the program of calculating the best parlays for the user based on their input
- We were not able to get it directly ported to a web app but the results can still be accessed in an HTML opened in a browser or a tkinter window

Contributions

Max Hartel - Research, Simulated Annealing Implementation, Testing, Data Analysis

James Tran - Research, Database Cleaning and Management, Simulated Annealing Implementation, Front-End (HTML and Tkinter), Testing

Jason Bohlinger - Research, Retrieval and Management of the SQL Database and historical data, Simulated Annealing Implementation, Testing

Andrew Broek - Research, SQL Testing, and Simulated Annealing implementation

Future Directions

- Database can be expanded to include more sports and their relevant stats
  - Horse racing would be a large target audience, but almost any sport can work
- Further optimization with more specialized algorithms
- Calling an API to get live results from games and active parlays
- Creating a more sophisticated front-end
- Compare a single parlay to thousands of other historical parlays to find the best value without maximizing risk
- Replace Lists with Hash Tables
- Dynamically free up memory

Questions?