The Optimality of the Greedy Algorithm in Carpool Problem

Mentor: Chai Wah Wu [1]
João Pedro Boavida[2] Vikram Kamat[3]
Darshana Nakum[4] Hung (Ryan) Nong[5]
Xinyi Zhang[6]

[1]TJ Watson Research Center, IBM

[2]University of Minnesota [3]Arizona State University

[4]University of Nevada [5]Rice University

[6]University of Delaware

Outline

Outline

Outline

Outline

Outline

Outline

The Carpool Problem

Carpool Problem

Suppose some of the *n* people in a neighborhood get together to carpool to work every morning, what is the fairest way to choose the driver each day?

Criterion for fairness?

Fairness

Fagin & Williams, 83: At each time t, Let $x_i(t)$ be the # of times that carpool member i has driven so far, and $y_i(t)$ be his/her ideal number of drives, then an algorithm is fair if for all schedules

$$|x_i(t) - y_i(t)| \leq P, \quad \forall i \in [n]$$

for some finite P.

Definition (Ideal Number of Drives)

If b_k is the number of times that person i participated in the carpool when a total of k people participated in the carpool, then his ideal number of drives is $\sum_k \frac{b_k}{k}$.

Algorithms

Different algorithms

- Simple rotation (fair)
- Simple tokens (not fair)
- Subsets (fair)
- Greedy carpool scheduling (fair)
  - Assign initial values $E = (0, \ldots, 0)$ to the n people
  - If there are k people appear on that day, than pick the person who has the highest value to drive and update E:
    - Add $(-1 + \frac{1}{k})$ to driver's value
    - Add $\frac{1}{k}$ to each passenger

  Note : At each time t, $E_i = x_i - y_i$, and $\sum_i E_i = 0$.

Outline

Conjecture: Greedy Algorithm is Optimal among all Online Algorithms

An online algorithm $\mathbb{A}$ depends only on the past and present schedules of arrivals.

Notations

Start from $E(0) = (0, ..., 0)$. At each time t, the cumulative error vector is

$$E(t) = (E_1(t), ..., E_n(t)), t \in \mathbb{N}$$

Define the error bound of carpool problem as

$$B = \min_{\mathbb{A}} B_{\mathbb{A}}, \quad \text{where } B_{\mathbb{A}} = \max_{schedule} (\max_{i,t} |E_i(t)|)$$

If $B_{\mathbb{A}}$ is finite, then the algorithm $\mathbb{A}$ is fair.

Conjecture: Greedy Algorithm is Optimal among all Online Algorithms

Theorem (Upper and Lower Bound)

If $n > 2$, then

$$B \leq B_{greedy} \leq \frac{n-1}{2} - \frac{1}{lcm(2,3,...,n)}$$

and for every n,

$$B > \frac{3(n-1)}{8} - \frac{1}{4}$$

Conjecture

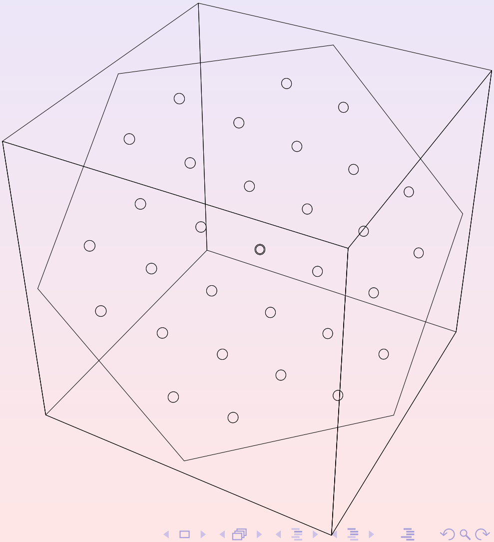
The Greedy algorithm is optimal among all online algorithms, the lower bound = the upper bound

Outline

Lower Bound

Recall: $\sum E_i = 0$.

Consider the error vector E_{max} maximizing the second moment $\sum E_j^2$, i.e. farthest from the origin .



Look for Small Gaps

For the maximum vector E_{max}, the gap g between two successive E_j is at least $\frac{3}{6}$. Similarly, we can exclude some 2-gaps and 3-gaps. e.g. $(\frac{3}{6}, \frac{3}{6})$, $(\frac{3}{6}, \frac{6}{6}, \frac{3}{6})$.

Idea of the Proof

Let $\bar{g}$ be the average of gaps, then

$$\exists \text{ a constant } \alpha > 0, \quad \bar{g} \geq \alpha,$$

Combining with the fact that $\sum_i E_i = 0$, roughly $B \geq E_n > \frac{\alpha}{2}(n-1)$.

If we only allow $k \leq 3$ people to appear each day , then $\alpha = \frac{3}{4}$.
To increase the lower bound, increase α.

Outline

Approach 1

Idea: Try to eliminate more gaps, by allowing more people to carpool everyday, for example, $k \leq 4$.

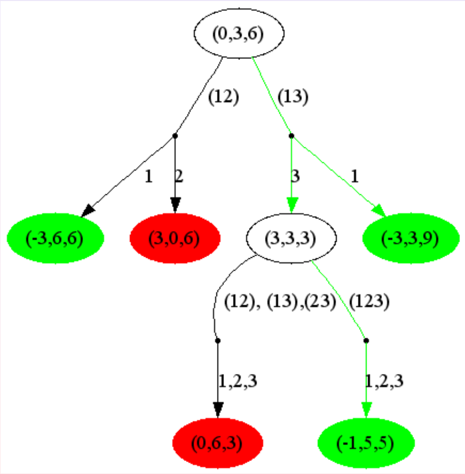
Algorithm

- Given an initial error vector E, with a 2nd moment m.
- Search for all possible schedules,
- If there is a tree of schedule that we will always have a new error vector with a 2nd moment higher than m, no matther how we choose the driver,
- Then E is not a maximum vector.

Note: Recursion.

Example

How did we exclude the gap $(\frac{3}{6}, \frac{3}{6})$? The corresponding error vector is $(0, \frac{3}{6}, \frac{6}{6})$, For simple notation, we abbreviate as $(0, 3, 6)$. Let the recursion limit to be 3.



Difficulties

As the search tree grows exponentially, the computer's running time becomes an important issue.

1. Special data structures should be employed to avoid repetitions while searching through all subsets of arrivals.
2. Subsets of arrivals may be chosen in a random or well-organized fashion to increase the likelihood of meeting the ultimate subset of arrivals early in the search.

Approach 2

Random Arrivals

- Other than search for all possible arrivals, we select a percentage of the schedules randomly. However it is even more time consuming.
- Start from the origin, we applied the greedy algorithm to the random arrivals to check the upper bound.

For $n = 6$ and 40 million consecutive arrivals, the biggest value of E_i we have is $\frac{88}{60}$, where the upper bound is $\frac{149}{60}$.

Our Results

- Our algorithms successfully excluded more gaps than prior literature.
- However, to improve the lower bound, we still need to exclude at least 3 more cases,

  $(\frac{8}{12}, \frac{10}{12}, \frac{9}{12})$, $(\frac{7}{12}, \frac{11}{12}, \frac{9}{12})$, and $(\frac{9}{12}, \frac{9}{12}, \frac{9}{12})$.

  We tested these cases for 10 recursions, it costs more than 2 days for the code in Matlab.

Our Results

Excluded Gaps

$(\frac{0}{12})$
$(\frac{4}{12})$
$(\frac{6}{12}, \frac{8}{12})$
$(\frac{7}{12}, \frac{6}{12})$
$(\frac{8}{12}, \frac{6}{12})$
$(\frac{10}{12}, \frac{6}{12})$
$(\frac{6}{12}, \frac{12}{12}, \frac{8}{12})$
$(\frac{7}{12}, \frac{11}{12}, \frac{7}{12})$
$(\frac{8}{12}, \frac{8}{12}, \frac{9}{12})$

$(\frac{1}{12})$
$(\frac{5}{12})$
$(\frac{6}{12}, \frac{9}{12})$
$(\frac{7}{12}, \frac{7}{12})$
$(\frac{8}{12}, \frac{7}{12})$
$(\frac{11}{12}, \frac{6}{12})$
$(\frac{6}{12}, \frac{12}{12}, \frac{9}{12})$
$(\frac{7}{12}, \frac{11}{12}, \frac{8}{12})$
$(\frac{8}{12}, \frac{9}{12}, \frac{8}{12})$

$(\frac{2}{12})$
$(\frac{6}{12}, \frac{6}{12})$
$(\frac{6}{12}, \frac{10}{12})$
$(\frac{7}{12}, \frac{8}{12})$
$(\frac{9}{12}, \frac{6}{12})$
$(\frac{6}{12}, \frac{12}{12}, \frac{6}{12})$
$(\frac{7}{12}, \frac{10}{12}, \frac{7}{12})$
$(\frac{7}{12}, \frac{12}{12}, \frac{6}{12})$
$(\frac{9}{12}, \frac{8}{12}, \frac{8}{12})$

$(\frac{3}{12})$
$(\frac{6}{12}, \frac{7}{12})$
$(\frac{6}{12}, \frac{11}{12})$
$(\frac{7}{12}, \frac{9}{12})$
$(\frac{9}{12}, \frac{7}{12})$
$(\frac{6}{12}, \frac{12}{12}, \frac{7}{12})$
$(\frac{7}{12}, \frac{10}{12}, \frac{8}{12})$
$(\frac{8}{12}, \frac{8}{12}, \frac{8}{12})$

Several Observations

- There are some gaps that can not be eliminated at all. e.g. $(\frac{4}{6}, \frac{4}{6})$.
  Since when implementing the algorithm, we come out a loop for every branch. (We have an algorithm for testing, but again takes long time).
  Conjecture: The three cases above can not be excluded.
- At each stage, using greedy algorithm always achieves a vector has a lower 2nd moment than any other algorithms.
- For small number of n, the greedy algorithm is optimal.
  - $n = 2, B = 1$.
  - $n = 3, B = \frac{5}{6}$.

Future Work

- Try to optimize our algorithms about testing whether a given initial vector will have a higher 2nd moment. (kill more loops)
- Apply our algorithms to 4-gaps.
- Try to optimize the algorithm that showed there are some cases can not be excluded.
- Research on whether all vector inside the bounded box can be achievable by greedy algorithm.

References

- R.Fagin and J.H.Williams, *A fair carpool scheduling algorithm*, IBM Journal of Research and development **27**(2) (1983), 133–139.
- M.Ajtai, J.Aspnes, M.Naor, Y.Rabani, L.J.Schulman and O.Waarts, *Fairness in Scheduling*, Journal of Algorithms **29**(2) (1998), 306–357.
- Moni Naor, *On fairness in the carpool problem*, Journal of Algorithms **55**(1) (2005), 93–98.
- Don Coppersmith, Tomasz J.Nowicki, Guiseppe A.Paleologo, Charles Tresser, Chai Wah Wu, *The Optimality of the On-line greedy algorithm in carpool and chairman assignment problems* , IBM Research Report, (2005)

Our Algorithm

$$\text{TestPt}(x) \equiv \text{TestMoment}(x, ||x||^2).$$

$$\text{TestMoment}(x, m) \equiv \bigvee_{s \text{ set of arrivals}} \text{TestSet}(x, m, s).$$

$$\text{TestSet}(x, m, s) \equiv \bigwedge_{i \in s} \text{TestDriver}(x, m, s, i).$$

$$\text{TestDriver}(x, m, s, i) \equiv x' \leftarrow x + \delta(s, i);$$

if x' was visited before **return** *false*;

if $||x||^2 > m$ **return** *true*;

if iteration too deep **return** *false*;

return $\text{TestMoment}(x', m)$.