

Rideshare Archival Record

Yoni Ben-Meshulam
Garrett Cooper
Derrick Huhn
Patrick Lowry

Outline

- ▶ The Carpooling problem
- ▶ How Rideshare solves it
 - Design choices
- ▶ What is involved?
 - Optimizer
 - User Interface
 - Controller
- ▶ How successful is it at solving the problem?
 - Testing
 - Evaluation
- ▶ Project management – Trac



Problem Statement

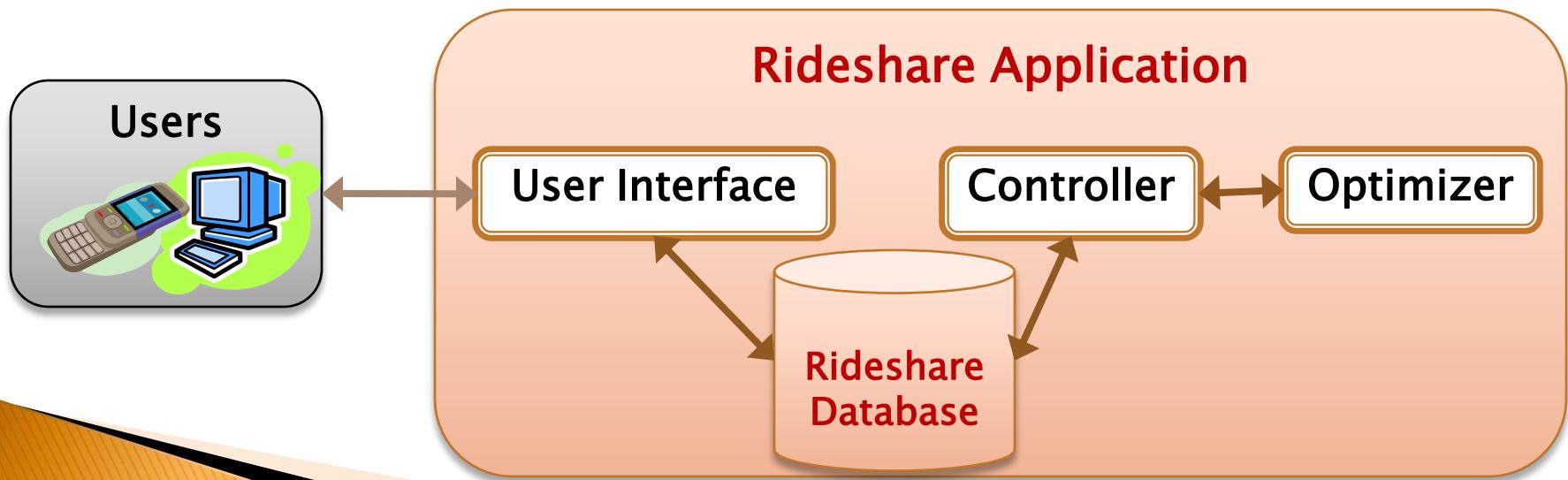
- ▶ Carpooling Problem = Vehicle Routing Problem
- ▶ But Really:
 - Capacitated Multi-Depot Vehicle Routing Problem with Time-Windows
 - Other Preferences (age, gender, rating)
 - NP-Hard
- ▶ Realistic solution times
- ▶ Globally optimized
- ▶ Reduce consumption, ecological footprint, cost, and energy dependency.

Choices

- ▶ User Interface
 - Maps Integration
 - Google Maps, MapQuest, Yahoo! Maps
- ▶ Optimization
 - Controller Library
 - Maps Integration
 - Database Interaction
 - Build System
 - Optimizers
 - Brute Force, Genetic, Bipartite Graph Matching

Architecture

- ▶ Separated user interface from optimizers
 - Communicate information through database
 - Allows data to be gathered while optimizers are running



Optimizer Challenges

- ▶ Framework to build on
 - Usable by all optimizers
 - Assessing Compatibility between a Rideshare and an unmatched rider
 - Simple preference checks
 - Scheduling
 - Deviation
 - Optimizing the order of points in a route
 - Common Scoring Function

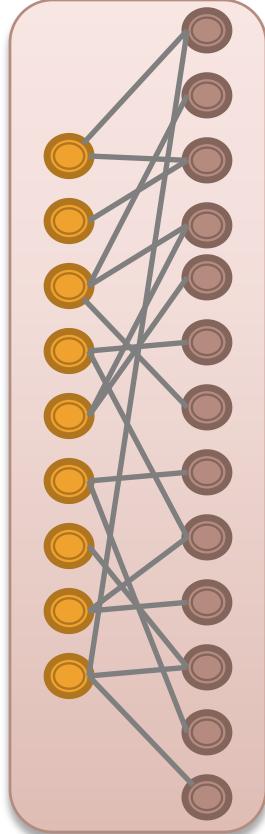
Scoring Function

$$Score = \text{Users Matched} + \frac{1}{\text{Distance}}$$

- ▶ Allows us to compare different solutions
- ▶ Users Matched (Whole Number)
 - The number of matched users (drivers and riders)
 - Bias toward spreading riders
- ▶ Distance (Fraction)
 - Total distance all drivers will take (whether they are matched or not)
 - Minimizes route deviation

Bipartite Optimizer

1. Get seed solution

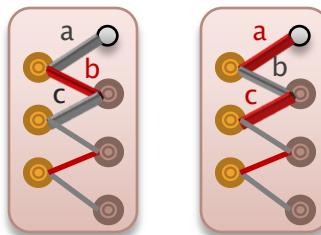


2. Connect compatible riders and rideshares with unmatched edges.

Rideshares
 Unmatched riders
— Unmatched edges
— Matched edges

3. For each unmatched rider, check if there is an augmenting path leading from that rider.

Such a path allows us to add one edge to the matching, since the number of unmatched edges is one greater than the number of matched edges.



The path **a->b->c** is an augmenting path. It allows us to remove **b** and add **a** and **c** to the matching.

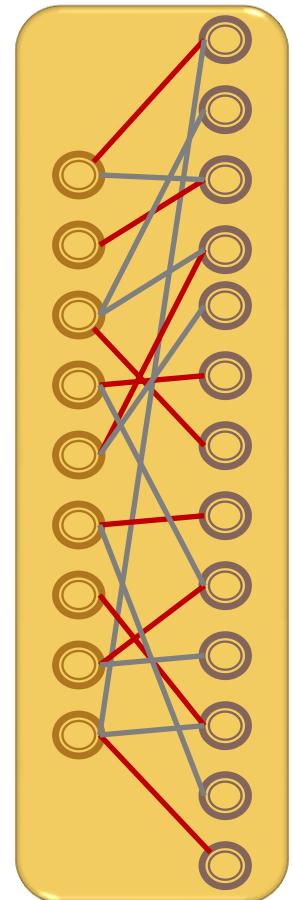
4. Exchange all unmatched and matched edges on the augmenting path.

5. Repeat 3–4 until all unmatched riders have been tried.

This produces a maximal matching:

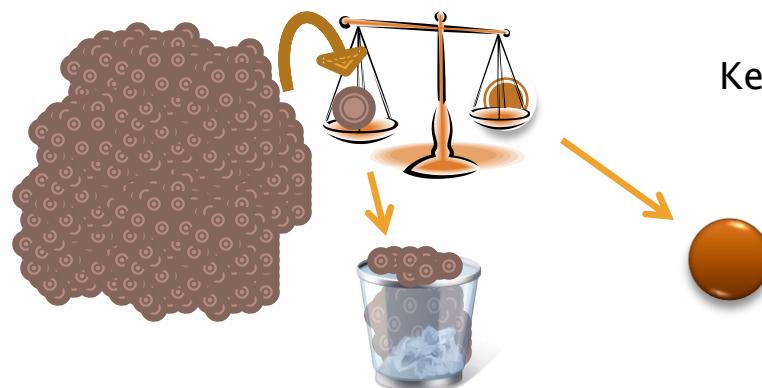
6. Add all matched riders to their rideshares.

7. Repeat 2–6, until no more augmenting paths exist.



Brute Force Optimizer

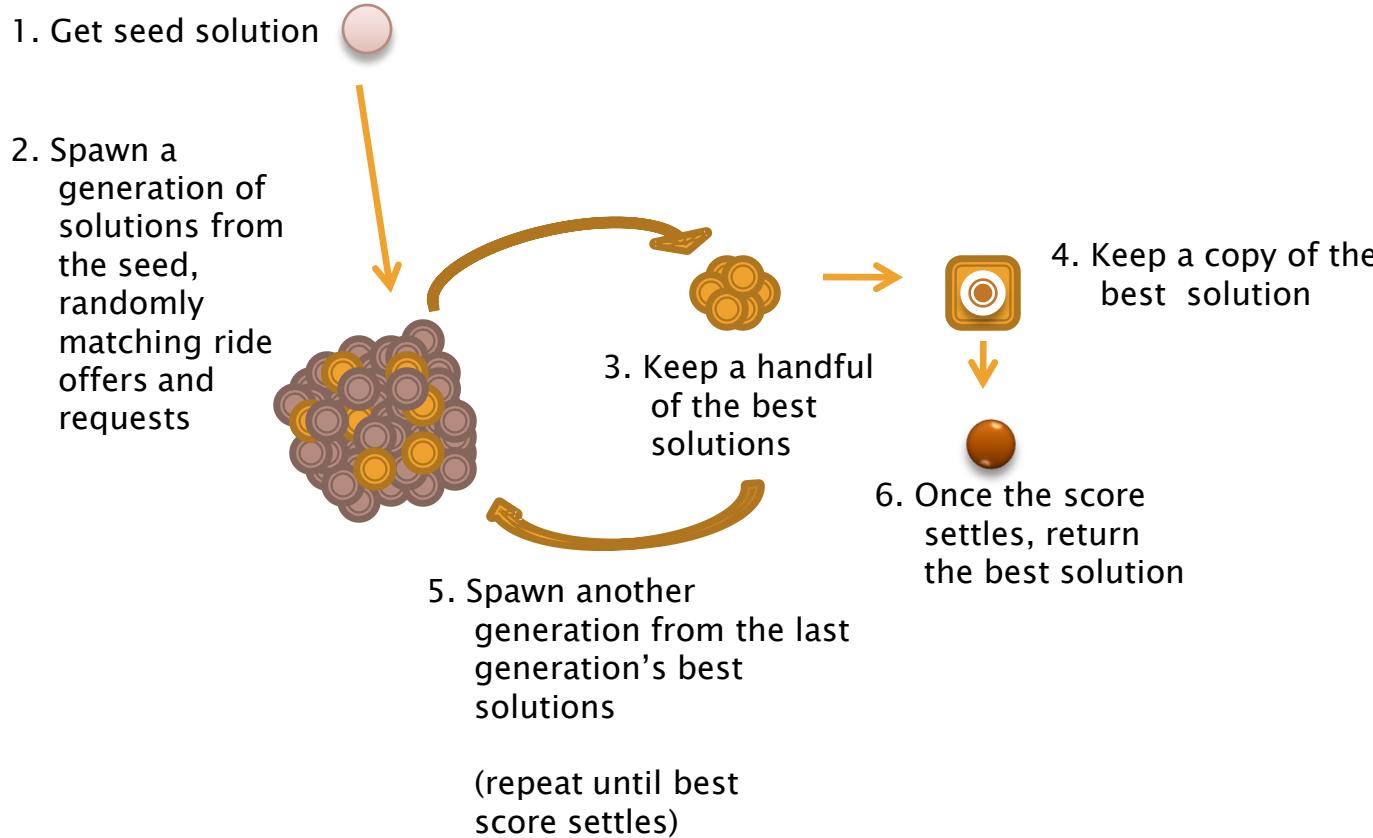
1. Get seed Solution



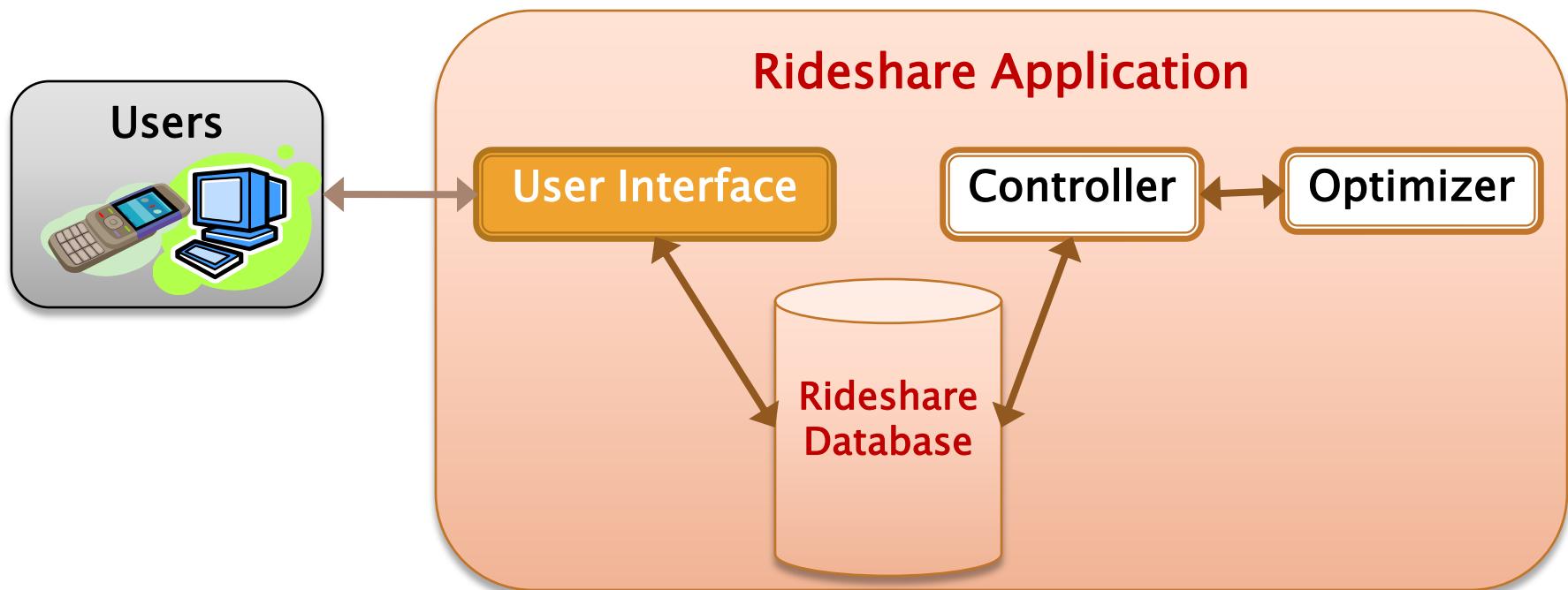
2. For each solution in the space of all possible solutions, compare it with the best one so far.

Keep only the better of the two.

Genetic Optimizer



Architecture (User Interface)



Screen shots

Rideshare

Welcome to Rideshare
Please log in to your account:

Email Address
Password
Submit

© Copyright 2008

About Rideshare Forgot My Password Create Account

New Ride

Rider Driver

1600 Wickersham Ln, Austin, TX 78705
Ut Austin, TX, USA

Look Up

Route Title: Going to school

Date/Time Preferences

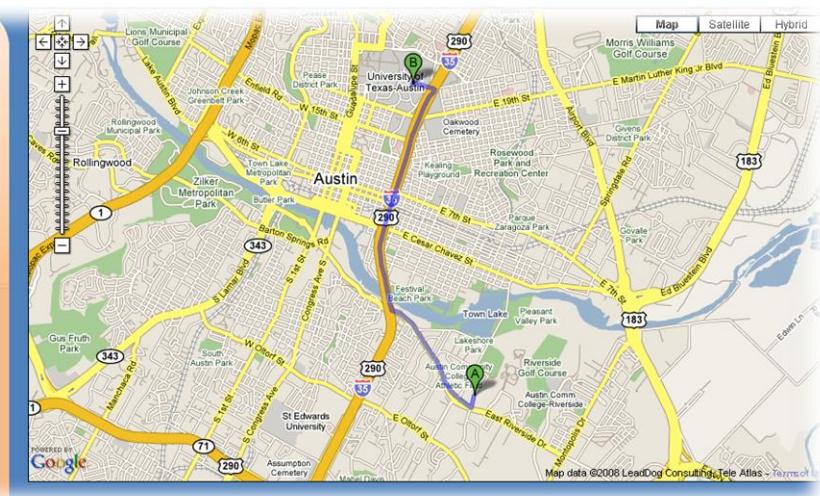
Est. Route Distance: 4.4 mi
Est. Route Duration: 11 mins

I want to leave sometime between:
Earliest Departure: 2008-04-22 14:00:00
Latest Departure: 2008-04-22 15:00:00

I want to arrive sometime between:
Earliest Arrival: 2008-04-22 14:00:00
Latest Arrival: 2008-04-22 15:00:00

Optional Preferences

Confirm Ride



My Rides My Profile New Ride Logout

My Rides (Subscribe)

New Route

Demo Test Case
Departure: 2008-04-17 08:30:00

Your request for a ride has been matched!
OpenHouse
Driver: John Tester
(Other) Rider: Edgar Becerra

OpenHouse
Departure: 2008-04-22 07:00:00

OpenHouse

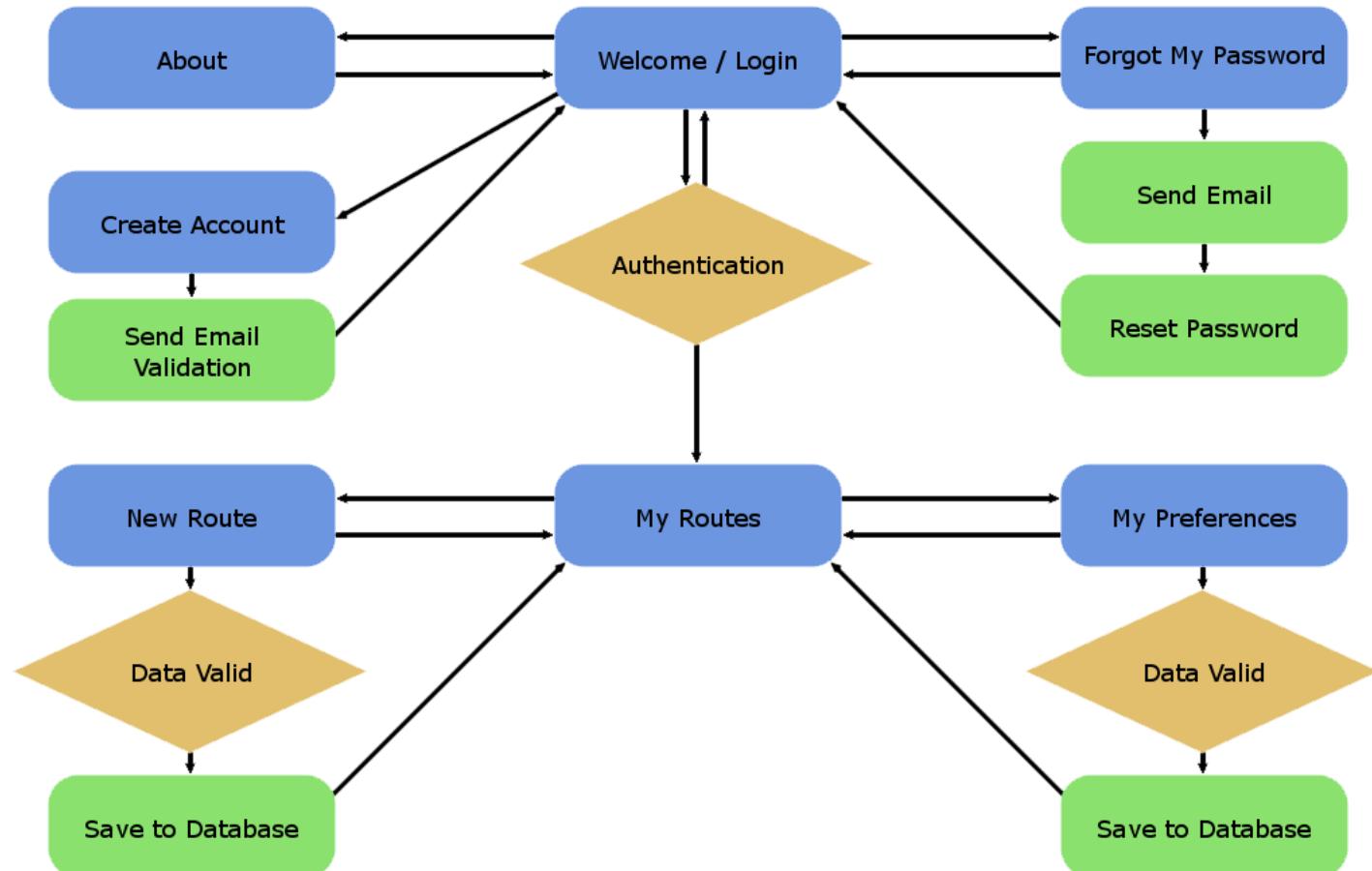
From: E Oltor St, Austin, TX between 2008-04-22 14:00:00 and 2008-04-22 14:57:00
To: 4700 E Riverside Dr, Austin, TX 78741 between 2008-04-22 14:00:00 and 2008-04-22 14:57:00. to pick up **Edgar Becerra**
To: 1600 Wickersham Ln, Austin, TX 78741 between 2008-04-22 14:00:00 and 2008-04-22 14:57:00. to pick up **You**
To: 201 E 21st St, Austin, TX 78705 between 2008-04-22 14:03:00 and 2008-04-22 15:00:00. to drop off **Edgar Becerra**
To: Ut, Austin, TX between 2008-04-22 14:03:00 and 2008-04-22 15:00:00. to drop off **You**
To: Ut, Austin, TX between 2008-04-22 14:03:00 and 2008-04-22 15:00:00.

Distance: 6.6 mi
Duration: 19 mins

Front-End Challenges

- ▶ Multiple Users
- ▶ Extensible
 - Cascading StyleSheets (CSS)
- ▶ Google Maps
 - Offload complex address translation
 - AJAX Problems
- ▶ Input Data Handling
 - Analyze and Interpret Input Data
- ▶ RSS Feeds

Front-End Map



Database Challenges

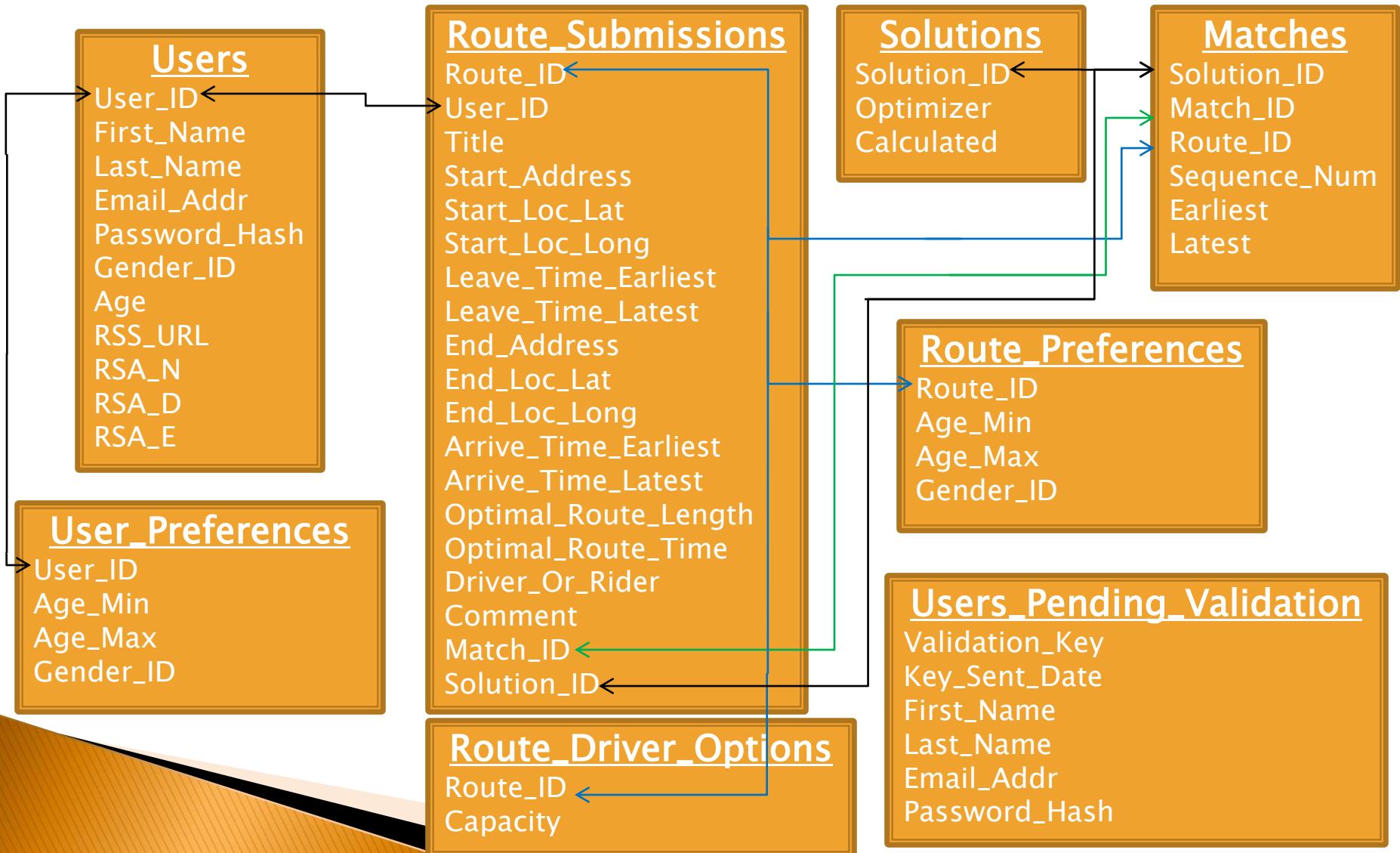
- ▶ Data Formatting and Structure
 - Had to be usable for both the front and back-ends
 - DATETIME, and ENUM data types
- ▶ Data Duplication
 - How to avoid it?
 - More tables
- ▶ How to represent complex data structures?
 - Matches in solutions

Database Implementation

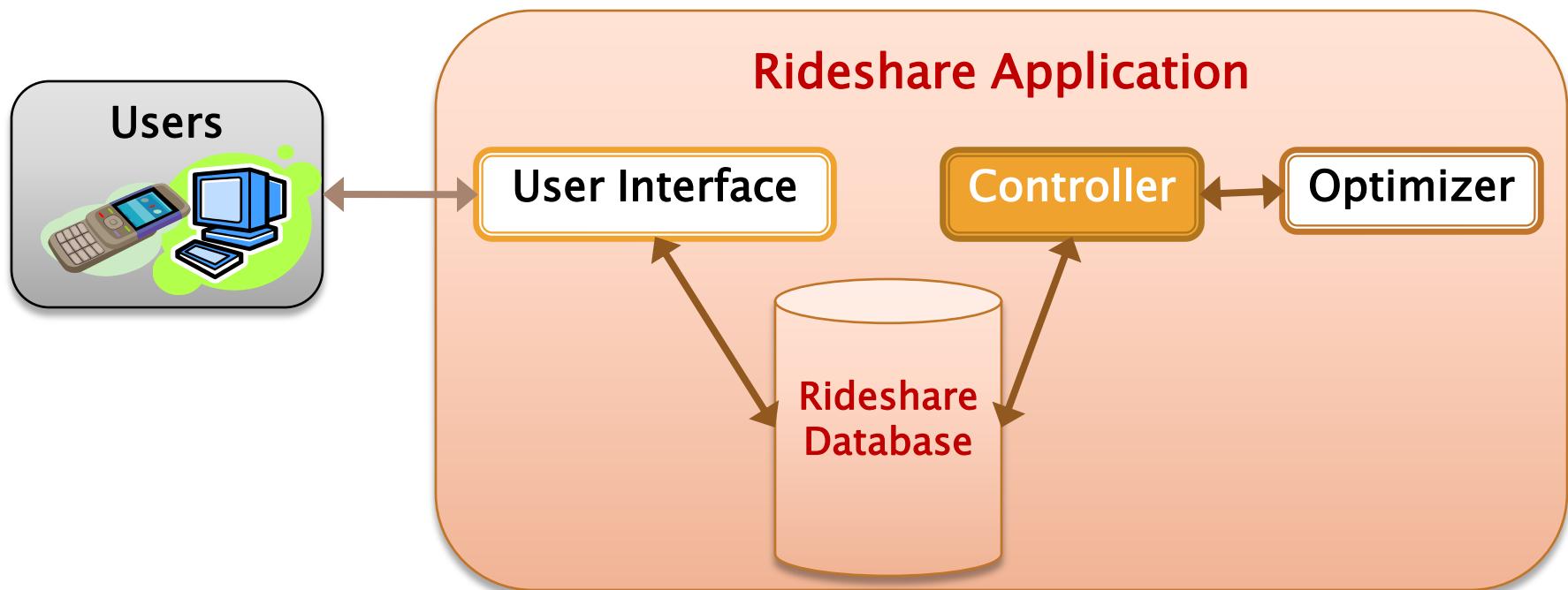
► Relational Database

- Normalization Trade-offs
 - Improves level of relation
 - Able to represent more complex data types
 - Less data duplication
 - More overhead
 - More tables
 - More IDs and joins (links) between tables

Database Structure



Architecture (Controller)



Controller Challenges

- ▶ Database communication
 - MySQL++ Library
- ▶ Limitations of Libraries (MapQuest API)
 - RouteMatrix only supported 25 Locations
 - Solution: RouteMatrixLocal – Local route calculations
- ▶ Data structures
 - STL: Vectors –> Maps
 - Represent Driver and Rider Submissions, Matches, Solutions, etc.
- ▶ Remote Development

Controller Implementation

- ▶ Executable for invoking libController's features
- ▶ Components
 - libController
 - Optimizer Logic
 - Database Access
 - Data structures for optimizer
 - Algorithm Visualizer
 - Notifications

libController

- ▶ 8,000+ Lines of Code
- ▶ 45 C++ Files
 - Optimizer Logic – 13 Files
 - Database Access – 4 Files
 - Data structures – 17 Files
 - Utility – 11 Files
- ▶ Build System
- ▶ MySQL++ and MapQuest Advantage API

Class List



Rideshare /

BruteForceOptimizer
optimize()
getBestScore()
getBestSolution()
iterate()

Optimizer
optimize() = 0
getBestScore() = 0
getBestSolution() = 0
double score()
vector<Solution> solutionHistory

GeneticOptimizer
optimize()
getBestScore()
getBestSolution()
generatePopulation()

BipartiteOptimizer
optimize()
getBestScore()
getBestSolution()
single_optimize()
iterate()
getAugmentingPath()
updateSolution()
updateEdges()
initEdges()
riderIterate()
matchIterate()

Utility:
SolutionUtils
Utils
TesterUtils

Route
startLoc()
endLoc()
getLoc()
numLocs()
get_type() = 0
UserPreference userPref
RoutePreference routePref

DB Structs:
TimeWindow
UserInfo
Solution
Match
Address
Location
PointInfo
Preference
Rideshare

RouteSubmission
setComment()
setUserInfo()
get_type()
uint32_t routeId
uint32_t userId
double optimalDistance
UserInfo userInfo

Clients:
RouteClient
DBAccess

Routes:
Route
RouteSubmission
RiderRouteSubmission
DriverRouteSubmission

Preferences:
UserPreference
RoutePreference

Optimizers:
Optimizer
BruteForceOptimizer
GeneticOptimizer
BipartiteOptimizer

Optimizer Structs:
CompatibilityMatrix
RouteMatrix
RouteMatrixLocal

Testing:
TesterUtil
TestCases

Class List
Address
BipartiteOptimizer
BruteForceOptimizer
CompatibilityMatrix
DBAccess
DriverRouteSubmission
GeneticOptimizer
Location
Match
Optimizer
PointInfo
Preference
RiderRouteSubmission
Rideshare
RouteClient
Route
RouteInfo
RouteMatrix
RouteMatrixLocal
RouteOptimizer
RoutePreference
RouteSubmission
Solution
SolutionUtils
TestCases
TesterUtil
TimeWindow
UserInfo
UserPreference
Utils

RiderRouteSubmission

DriverRouteSubmission
size_t capacity

Algorithm Visualizer

- ▶ Visual display of intermediate solutions generated by optimizers
- ▶ XHTML/JavaScript
- ▶ Animation utilizes MapQuest's image API
- ▶ Metaprogramming
 - C++ generates JavaScript that is executed in browser

Ride Share

Solution Information

Solution: 4

Score: 21.0098

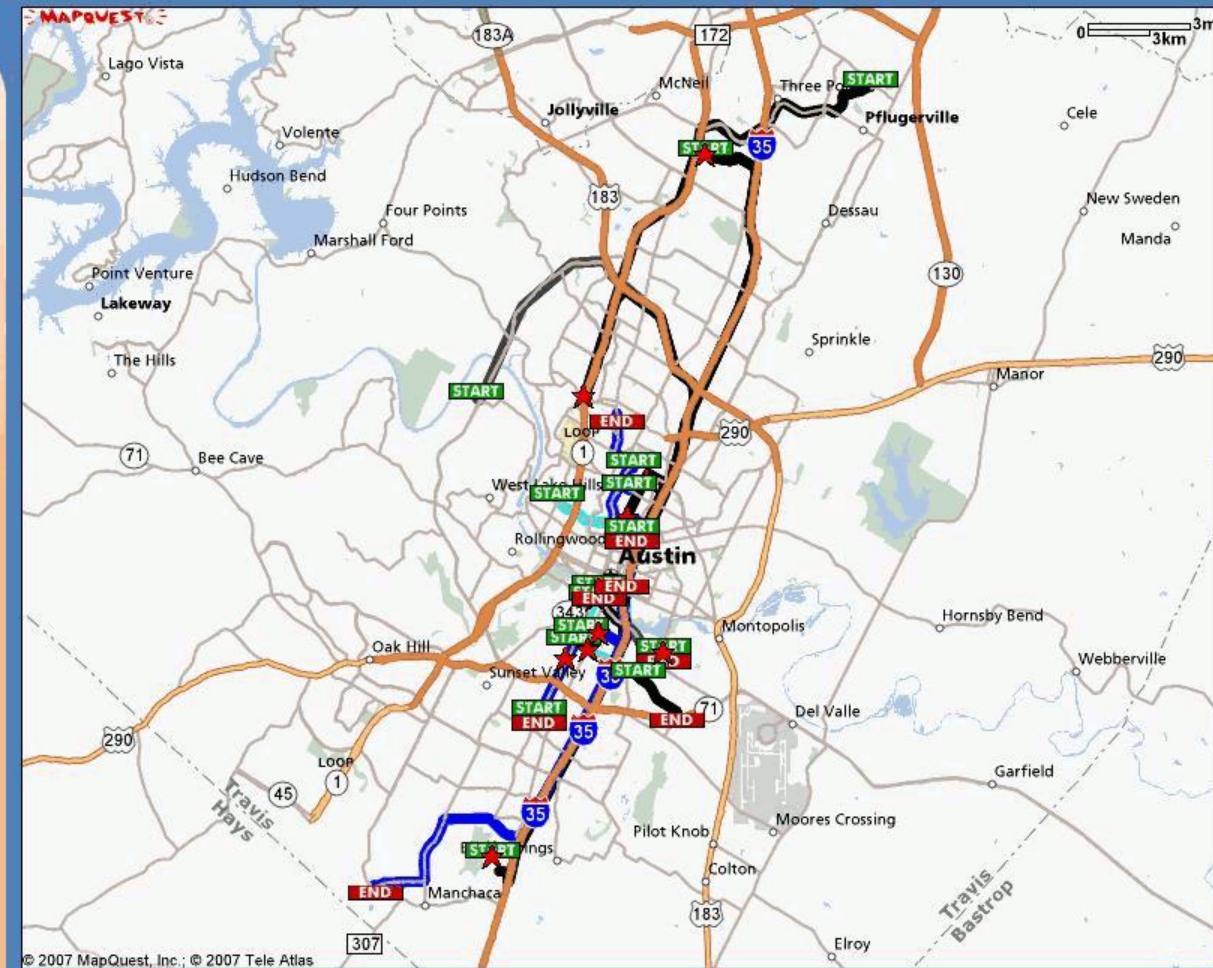
Matches: 18

Riders: 12

Match 1 with Driver: **John Tester**
Rider #1**Rainbow Rider**
Rider #2**Dave Driver**
Match 2 with Driver: **Ronald Rider**
Match 3 with Driver: **Trina Tester**
Rider #1**Ramsey Rider**
Match 4 with Driver: **Ronald Rider**
Rider #1**Rainbow Rider**
Match 5 with Driver: **John Tester**
Rider #1**Ronald Rider**
Match 6 with Driver: **David Tester**
Match 7 with Driver: **Dave Driver**
Rider #1**Trina Tester**
Match 8 with Driver: **Delilah Driver**
Match 9 with Driver: **John Tester**
Rider #1**Delilah Driver**
Rider #2**Dave Driver**
Match 10 with Driver: **John Tester**
Match 11 with Driver: **Garrett Cooper**
Match 12 with Driver: **John Tester**
Rider #1**Dave Driver**
Rider #2**Derrick Huhn**
Match 13 with Driver: **John Tester**
Match 14 with Driver: **Devin Vitone**
Rider #1**Constantine Caramanis**
Match 15 with Driver: **Garrett Cooper**
Match 16 with Driver: **Garrett Cooper**
Match 17 with Driver: **Garrett Cooper**
Rider #1**Patrick Lowry**
Match 18 with Driver: **David Tester**

< >

© Copyright 2008



Notifications (E-mail and RSS Feed)

▶ E-mail

- Users need to be notified when they have been matched in a Rideshare
- C++ Generates Perl which invokes SendMail
- Use libController to retrieve users' emails

▶ RSS Feed

- Users can subscribe to an RSS feed that is dynamically generated using PHP
- Real-time notification for active users

Testing and Optimizer Evaluation

- ▶ Unit Tests
- ▶ uShip Data
- ▶ Corner Cases
- ▶ Optimizer Characterization and Evaluation



Unit Tests

- ▶ Over 100 individual unit tests
 - Each feature/function tested individually
 - Hand-tailored test cases
- ▶ Used for stabilization before each commit

uShip Data

- ▶ Raw uShip Data
- ▶ 2100 Working Locations in Texas
- ▶ uShip Data with Preferences
 - 20 Non-inclusive test cases
 - Size ranges from 7 to 120, with regular increments



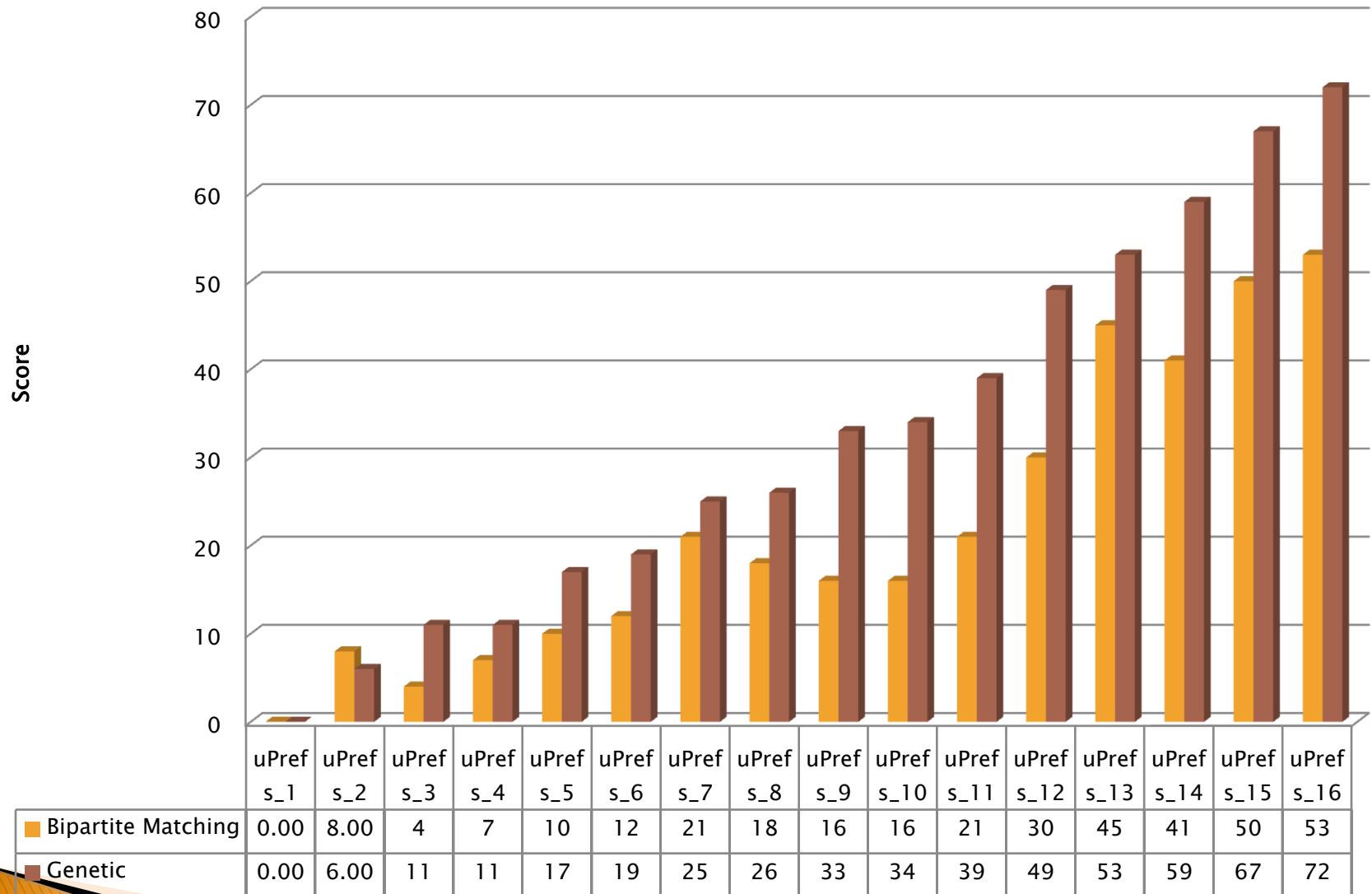
data and screenshot thanks to uShip.com

Corner Cases

- ▶ Designed to test specific functions of the optimizers
 - Capacity
 - Going Out of the Way
 - Moving backwards to pick up a rider
 - Picking up the closer rider
 - Time Windows
 - Maximum Route Deviation
 - Maximizing Riders
 - Single-solution test case for Bipartite

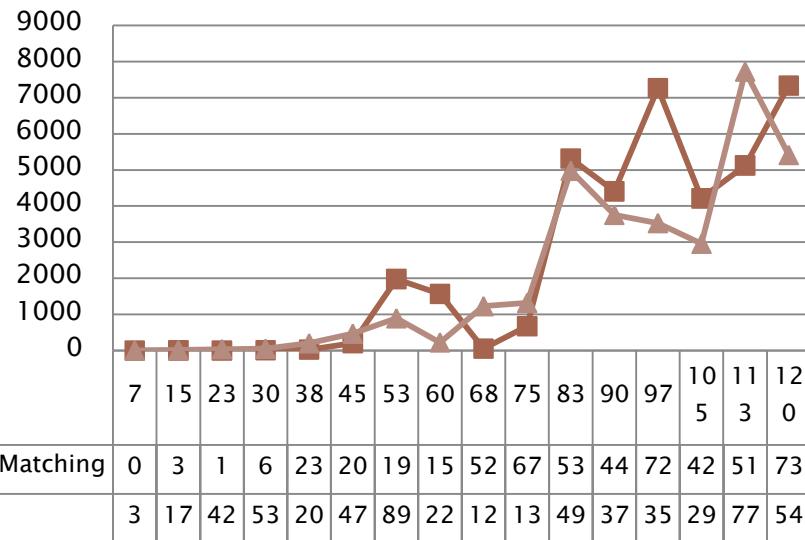
Scores

Genetic vs. Bipartite



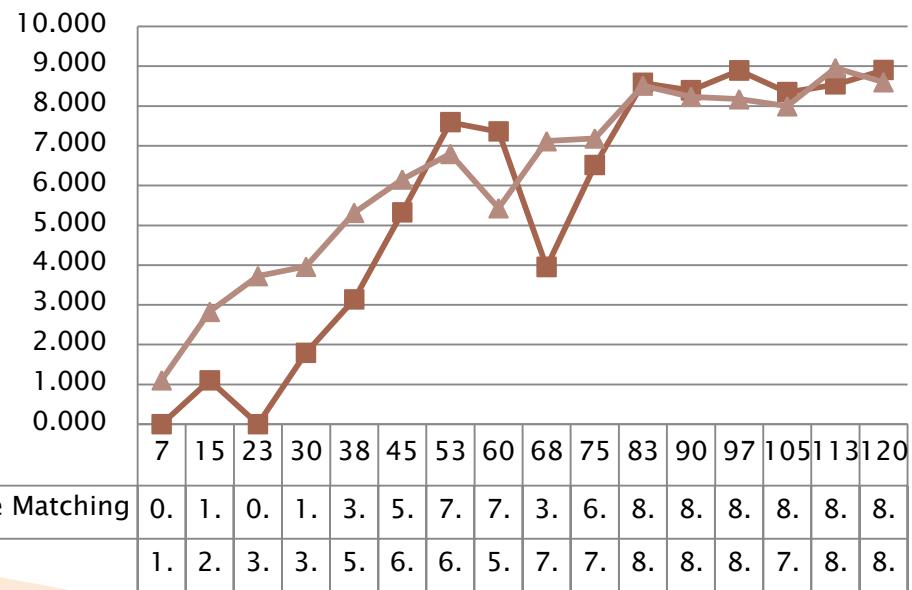
Number of Requests vs. Run Time

Run Time ϵ in seconds



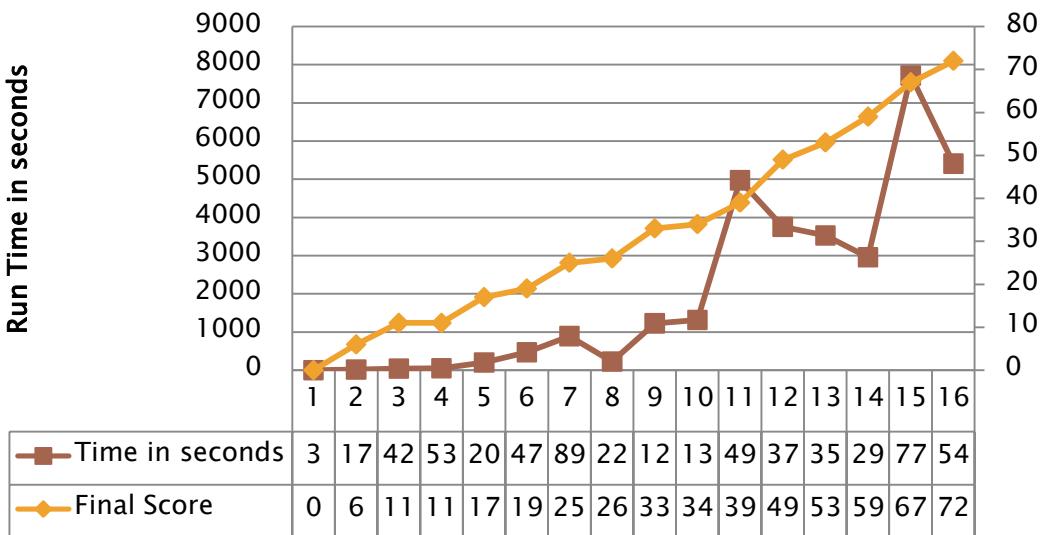
Number of Requests vs. Natural Logarithm of Run Time

Natural Logarithm of the Run Time ϵ
In(run time in seconds)

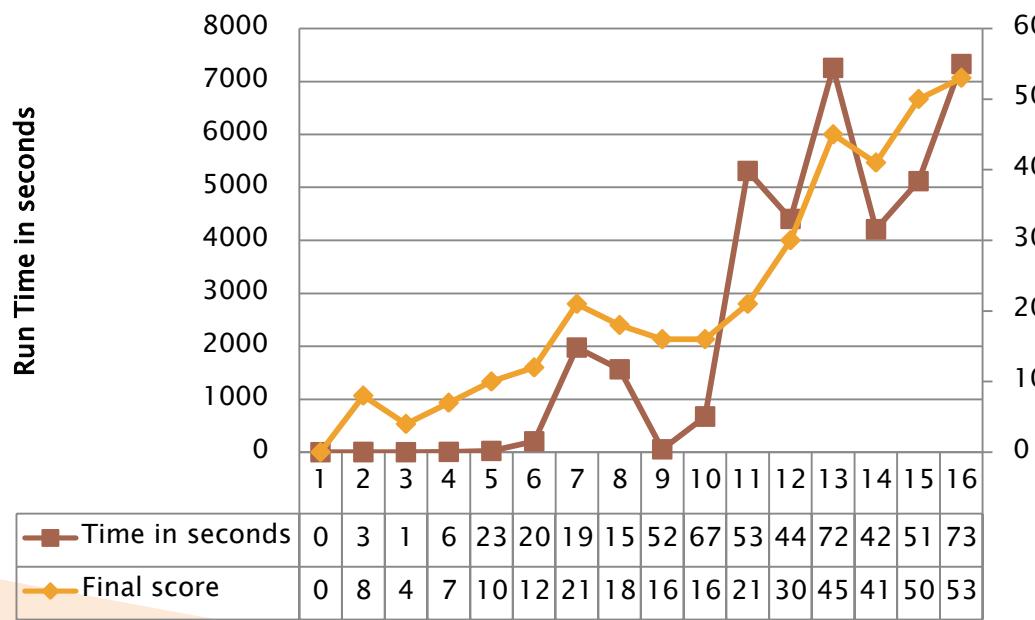


Run Time vs. Final Score

Genetic



Bipartite



Time and Cost Considerations

▶ Cost

- Open Source == Free
- ECE Department provided Development Systems

▶ Time

- Coupled tasks that hindered development
- 15% debugging C++ related problems
- 5% dedicated to code not used
- 20% dedicated to user interface

▶ Improvements

- 20% of tickets involved fixing bugs
- More code review sessions!

Safety and Ethical Aspects

- ▶ Personal safety:
 - Encourage users to take advantage of site features to screen matches
 - Minimum age eighteen
- ▶ Social Innovation
 - Reduce ecological footprint
 - Save time and energy
 - Promote cooperative, sustainable living

Trac

- ▶ Wiki System
- ▶ Ticket System
- ▶ Subversion Integration
- ▶ Timeline/Roadmap

Ticket System

[Wiki] [Timeline] [Roadmap] [Browse Source] [View Tickets] [New Ticket] [Search]

[Previous Ticket] [Back to Q]

Ticket #90 (assigned enhancement)

Tighten up controller program for production		Opened 6 days ago Last modified 6 days ago	
Reported by:	huhn	Assigned to:	huhn (accepted)
Priority:	major	Milestone:	Demo
Component:	component1	Version:	
Keywords:	Cc:	Reply	
Description			
The controller currently works but needs to be tweaked for optimizing submissions from the website. Currently, the controller reads in riders and drivers with certain comments but those test submissions should be marked as matched so the optimizer sees all data.			
In conjunction with this, the Rideshare team should input test data into the site to populate the database with riders and drivers so that students can test Rideshare out on demo day.			

Attachments

[Attach File](#)

Change History

04/21/08 22:26:30 changed by huhn	Reply
■ status changed from <i>new</i> to <i>assigned</i> .	

88	Remove My Friends from menu bar	closed	lowry	defect	minor	Demo
89	Add localization to Route Compatiblity checks in order to reduce run times	closed	bennak	enhancement	minor	

Subversion Integration

← [Previous Change](#) | [Next Change](#) →

 **trac**
Integrated SCM & Project Management

logged in as caramanis | [Logout](#) | [Settings](#) | [Help/Guide](#) | [About Trac](#)

[Wiki](#) | [Timeline](#) | [Roadmap](#) | [Browse Source](#) | [View Tickets](#) | [New Ticket](#) | [Search](#)

[Last Change](#) | [Revision Log](#)

root / lib / branches

View revision:

Name	Size	Rev	Age	Last Change
..				
BUG_40	120	1 month	huhn:	Changed TimeWindow? to use DateTime? rather than string. Added some unit ...
DEF_58	231	1 month	huhn:	Restructured DBAccess class to return vector of Driver and Riders rather ...
DEF_64	343	4 weeks	bmesh:	canAddRider() coded and working, passes all current tests. still need to ...
DEF_70	307	4 weeks	cooper:	Capacity stuff works, and added a new unit test.
DEF_71	304	4 weeks	huhn:	Creating branch DEF_71
DEF_79	357	3 weeks	cooper:	95% sure this works. no test cases failed.
ENH_9	48	2 months	huhn:	Made some progress on RouteClient? Added support for adding and deleteing ...
ENH_23	69	2 months	huhn:	Removed a commented line
ENH_25	89	2 months	bmesh:	updated to latest trunk.
ENH_30	202	1 month	cooper:	Had to add to DBA so I could put all the fields in. This is what I used to ...
ENH_31	80	2 months	huhn:	Added the Address class. Location can now take in an address Added ...
ENH_32	127	1 month	huhn:	Updated copy constructors
ENH_33	116	2 months	huhn:	For some reason i added the trunk in this directory, deleteing it

Conclusions and Recommendations

► Results

- Completed all of the tasks for optimization
- Didn't implement problem reduction
(not a necessity)
- Didn't implement confirmation (time)
- Successful testing of all functionalities

► Recommendations

- Develop user interface for social networking, phones
- Extensible Design