Bicing

Tomas Barton and ***Mauro Donadeo***

November 1, 2009

1 Introduction

The aim of this project is to provide an program which would help the provider of local bicing service[1] with distribution of bikes to approach to an ideal state, when each user of this service would find a bike when he needs it.

1.1 About Bicing

The Bicing project is quite new in Barcelona (started in 2007[2]), but after few months it became very popular. Owner of a special card, which can be purchased for 30 € per year, can rent a bike from a station for free, if he returns this bike to some station in a half an hour. Otherwise he pays a few cents for each hour until the bike is returned.

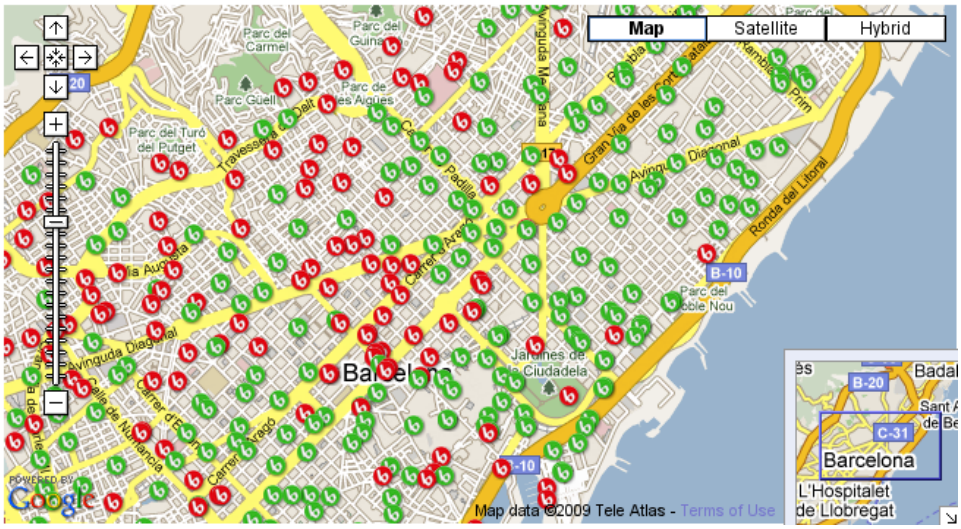

Image 1.1: Map of bicing stations in Barcelona – red stations are without bikes

Nowadays there are more than 400[2] bike stations in Barcelona, the highest concentration of bike stations is of course in centre of city. Usually each station has from 20 up to 30 stands for bikes. In the very center they are placed quite close to each other.

[1] Bicing Barcelona – `http://www.bicing.com`
[2] 418 stations at 12 October 2009

1.2 Our task

With usage of $\boldsymbol{F}$ vans we are supposed to optimize distribution of bikes, so that as many customers as possible will find a disposable bike at time when they need it.

Implementation has been done in Java, we took advantage of AIMA framework[1], which contains most of algorithms commonly used for searching state space.

1.3 Complexity of problem

As long as we have to search just for solution of this problem for upcoming hour, size of state space is significantly reduced. When we display all possible states of this problem as a n-ary tree, we get not really deep tree – this is exactly the number of maximum moves we can do and that is $\boldsymbol{F}$, but this tree can have a huge branching factor.

2 Implementation

2.1 State representation

Each state must contain information about distribution of bikes over all stations. So that we have to keep:

- current number of bikes at the station
- number of bikes that is going to be at station in an hour
- moves how to reach this state (from empty state)

These information are different for each state, moreover we have other information which we need but they are same for every state:

- number of stations
- coordinates of stations
- expected demand for bikes in upcoming hour

2.2 Operators

moveBikes simply moves given number of bikes from one station to another

doubleMove loads $a + b$ bikes at one station and unload a bikes at first station and b bikes at second station

changeMove alter number of bikes or destination of this move

removeMove deletes specified move

Station #	Not used	Next	Demand
1	10	10	0
2	0	0	0
3	0	0	2
4	0	0	8

Table 2.1: Initial state with heristic value $h_1 = 10.0$

2.3 Heuristic functions

Heuristic function is supposed to evaluate state, so that searching algorithm can unambigously determine which state is better.

2.3.1 Version 1

Firstly, we have to take in consideration number of unsatisfied demand of bikes. If there are no bikes needed, or the demand is satisfied, the value of heuristic function would be 0. We are trying to minimize this formula:

$$h_1 = \sum_{i=0}^{numStations} not\_satistifed\_demand_i \tag{2.1}$$

Let us take into consideration situation when we have 4 stations and bike are all bikes are located only on first station, as it is described in table 2.3.1.

3 Bibliography

[1] *Stuart Russell and Peter Norvig:***Artificial Intelligence: A Modern Approach** `http://aima.cs.berkeley.edu/`
Downloaded java implementation of algorithms (AIMA framework) from repository `http://code.google.com/p/aima-java/` by Ravi Mohan.
Date of publication: October 3, 2009.
Date retrieved: October 12, 2009.

[2] *Wikipedia:* **Bicing** `http://en.wikipedia.org/wiki/Bicing`
Date visited: October 27, 2009