BEST: International Journal of Management, Information Technology and Engineering (BEST: IJMITE)
ISSN 2348-0513
Vol. 2, Issue 6, Jun 2014, 43-52

LOCATION AWARE RECOMMENDATION SYSTEM

RADHA[1] & MANOJ KUMAR[2]

[1]SGT Institute of Engineering & Technology, Haryana, India

[2]Dissertation Scholar, SGT Institute of Engineering & Technology, Haryana, India

ABSTRACT

Location aware recommendation system is used to find out the location and distance between source and destination using Grid algorithms and Matlab s/w. It is very close to querying user to provide the maximization of the system scalability. LARS exploits the item location using travel penalty, its techniques that favours recommendation candidates closer in travel distance to querying user in a way that avoid the exhaustive access to all items. In this we use the tools of different type in MATLAB, so that we carry out the task. We also calculate the goal distance, time and position of the object.

KEYWORDS: Fuzzy Logic, Membership Function, Position Tracking System of Object, Artificial Intelligence, Tools in Experimental Set up, Simulation & Key Features, Algo to Solve the Problem

INTRODUCTION

It uses location-based ratings to produce recommendations. Traditional recommender systems do not consider spatial properties of users nor items; LARS, on the other hand, supports taxonomy of three novel classes of location based ratings, namely, spatial ratings for non-spatial items, nonspatial ratings for spatial items, and spatial ratings for spatial items. LARS exploits user rating locations through user partitioning, a technique that influences recommendations with ratings spatially close to querying users in a manner that maximizes system scalability while not sacrificing recommendation quality. LARS exploits item locations using travel penalty, a technique that favors recommendation candidates closer in travel distance to querying users in a way that avoids exhaustive access to all spatial items.

FUZZY LOGIC

Fuzzy logic" has become a common buzzword in machine control. However, the term itself inspires certain skepticism, sounding equivalent to "half-baked logic" or "bogus logic". Some other nomenclature might have been preferable, but it’s too late now, and fuzzy logic is actually very straightforward. Fuzzy logic is a way of interfacing inherently analog processes that move through a continuous range of values, to a digital computer, that likes to see things as well-defined discrete numeric values.

Membership Function

The membership function of a fuzzyset is a generalization of the indicator function in classical sets. In fuzzy logic, it represents the degree of truth as an extension of valuation. Degrees of truth are often confused with probabilities, although they are conceptually distinct, because fuzzy truth represents membership in vaguely defined sets, not likelihood of some event or condition.

Fuzzy Logic for Position Tracking System of Object

Fuzzy logic can be effectively used for map matching in urban canyons because of its ability to generate precise output from noisy (error prone) navigation input obtained from GPS. Such robustness is not available with conventional map matching techniques, which uses accurate equations

The basic steps involved in fuzzy logic are:

- Selection of fuzzy inputs and outputs,
- Selection of fuzzy sets and corresponding membership functions
- Definition of fuzzy rules for the FIS,
- Definition of an implication function (which scales the output membership functions). This is not needed in a Sugeno-type FIS, where the rules are of the form:

  If input1 = x and input2 = y, then output z = ax+by+c (where a, b, c are real numbers),
- De fuzzification to get crisp output.

ARTIFICAL INTELLIGENCE

AI research is highly technical and specialized, and is deeply divided into subfields that often fail to communicate with each other. Some of the division is due to social and cultural factors: subfields have grown up around particular institutions and the work of individual researchers. AI research is also divided by several technical issues. Some subfields focus on the solution of specific problems. Others focus on one of several possible approaches or on the use of a particular tool or towards the accomplishment of particular applications. The central problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence is still among the field's long term goals. Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are a large number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. The field was founded on the claim that a central property of humans, intelligence.

How AI Works for Shortest Path (Real Time Analysis)

In our experiment with 30 vertices, the algorithm came up with the optimal solution in real time. While increasing the number of vertices from 5 to 30 with step size 5 and plotting the corresponding algorithm execution time, we have observed that the resulting graph approximately fits the n^2 polynomial curve. We have averaged the results of 20 runs per increase in the number of vertices. This performance was acceptable because the worst case running time for Dijkstra's algorithm is O (n^2). Furthermore, for our suggested application of the algorithm, most of the computation is done for a particular local region of a map, so the achieved performance with 30 nodes should suffice for practical implementation.

Tools of AI

- Search and optimization.
- Logic

- Probabilistic methods for uncertain reasoning
- Classifiers and statistic learning methods

TOOLS FOR EXPERIMENTAL SETUP

- **Symbolic Maths Toolbox:** It provides functions for solving and manipulating symbolic math expressions and performing variable-precision arithmetic. You can analytically perform differentiation, integration, simplification, transforms, and equation solving.
- **Optimization Toolbox:** It provides functions for finding parameters that minimize or maximize objectives while satisfying constraints. The toolbox includes solvers for linear programming, mixed-integer linear programming, quadratic programming, nonlinear optimization, and nonlinear least squares.
- **Fuzzy Logic Toolbox:** It provides functions, apps, and a Simulink block for analyzing, designing, and simulating systems based on fuzzy logic. The product guides you through the steps of designing fuzzy inference systems. Functions are provided for many common methods, including fuzzy clustering and adaptive neurofuzzy learning.
- **Simulation Toolbox:** It provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.

OBJECTIVES OF THE STUDY

To find out the location and distance between source and destination using Grid algorithms and Matlab s/w. It is very close to querying user to provide the maximization of the system scalability.

METHODS

Algo to Solve the Problem

If you have this grid, where a * = obstacle and you can move up, down, left and right, and you start from S and must go to D, and 0 = free position:

Iters - number of iterations

Here is a sample session to find the optimum for the following function:

y = 10 + (X(1) - 2)^2 + (X(2) + 5)^2

The above function resides in file fx1.m. The search for the optimum 2 variables has the search range of [-10 -10] and [10 10] with a divisions vector of [4 5] and a minimum range vector of [1e-5 1e-5]. The search employs a maximum of 10000 iterations and a function tolerance of 1e-7:

[XBest, BestF, Iters]=Grid_Search(2, [-10 -10], [10 10], [4 4], [1e-5 1e-5], 1e-7, 10000, 'fx1')

XBest =2.0001 -5.0000, BestF =10.0000

S 0 0 0

* * 0 *

* 0 0 *

0 0 * *

* 0 0 D

You put S in your queue, then "expand" it:

S 1 0 0

* * 0 *

* 0 0 *

0 0 * *

* 0 0 D

Then expand all of its neighbours:

S 1 2 0

* * 0 *

* 0 0 *

0 0 * *

* 0 0 D

And all of those neighbours' neighbours:

S 1 2 3

* * 3 *

* 0 0 *

0 0 * *

* 0 0 D

And so on, in the end you'll get:

S 1 2 3

* * 3 *

* 5 4 *

7 6 * *

* 7 8 9

So, the distance from S to D is 9. The running time is O(NM), where N = number of lines and M = number of columns. I think this is the easiest algorithm to implement on grids, and it's also very efficient in practice. It should be faster than a classical dijkstra, although dijkstra might win if you implement it using heaps.

MATLAB Program to Find A Function Minimum Using a Grid Search Method by Namir Shammas.

The following program calculates the minimum point of a multi-variable function using the grid search method. This method performs a multi-dimensional grid search. The grid is defined by a multiple dimensions. Each dimension has a range of values. Each range is divided into a set of equal-value intervals.

The multi-dimensional grid has a centroid which locates the optimum point. The search involves multiple passes. In each pass, the method local a node (point of intersection) with the least function value. This node becomes the new centroid and builds a smaller grid around it. Successive passes end up shrinking the multidimensional grid around the optimum.

The function Grid Search has the following input parameters:

N: Number of variables

XLo: Array of lower values

XHi: Array of higher values

NumDiv: Array of number of divisions for each range

MinDeltaX: Array of minimum ranges

Eps_Fx: Tolerance for difference in successive function values

MaxIter: Maximum number of iterations my

Fx: Name of the optimized function

The function generates the following

Output: X - array of optimized variables

BestF: Function Value at optimum

Iters =200

Notice how close the located optimum is to the actual one [-2 5]..

Here is the MATLAB listing:

function y=fx1(X, N)

X.y = 10 + (X(1) - 2)^2 + (X(2) + 5)^2;

End

function [XBest, BestF, Iters]=Grid_Search(N, XLo, XHi, NumDiv, MinDeltaX, Eps_Fx, MaxIter, myFx)

Function performs multivariate optimization using the grid search

Input: N - number of variables

XLo: Array of lower values

XHi: Array of higher values

NumDiv: Array of number of divisions along each dimension

MinDeltaX: Array of minimum search values for each variable

Eps_Fx: Tolerance for difference in successive function value

MaxIter: Maximum number of iterations

myFx: Name of the optimized function

Output: XBest - array of optimized variables

BestF: Function value at optimum

Iters: Number of iterations

```
Xcenter = (XHi + XLo) / 2;
XBest = Xcenter;
DeltaX = (XHi - XLo)./ NumDiv;
BestF = feval(myFx, XBest, N);
if BestF >= 0
LastBestF = BestF + 100;
else
LastBestF = 100 – BestF;
end
X = XLo; % initial search value
Iters = 0;
bGoOn = 1;
while (bGoOn > 0) && (abs(BestF - LastBestF) > Eps_Fx) && (Iters <= MaxIter)
bGoOn2 = 1;
while bGoOn2 > 0
Iters = Iters + 1;
F = feval(myFx, X, N);
if F < BestF
LastBestF = BestF;
BestF = F;
XBest = X;
```

```
end
%*****************************************************
The next For loop implements a programming tricks
that simulated nested loops using just one For loop
%******************************************************
search next grid node
for i = 1:N
if X(i) >= Xhi(i)
if i < N
X(i) = Xlo(i);
Else
bGoOn2 = 0;
break
end
else
X(i) = X(i) + DeltaX(i);
break
End end
end % while bGoOn2 > 0
XCenter = Xbest;
DeltaX = DeltaX./ NumDiv;
XLo = XCenter - DeltaX.* NumDiv / 2;
XHi = XCenter + DeltaX.* NumDiv / 2;
X = XLo;
% set initial X
bGoOn = 0
for i=1:N
if DeltaX(i) > MinDeltaX(i)
bGoOn = 1;
end end end
% while bGoOn > 0 && () && ()
```

RESULTS AND DISCUSSIONS

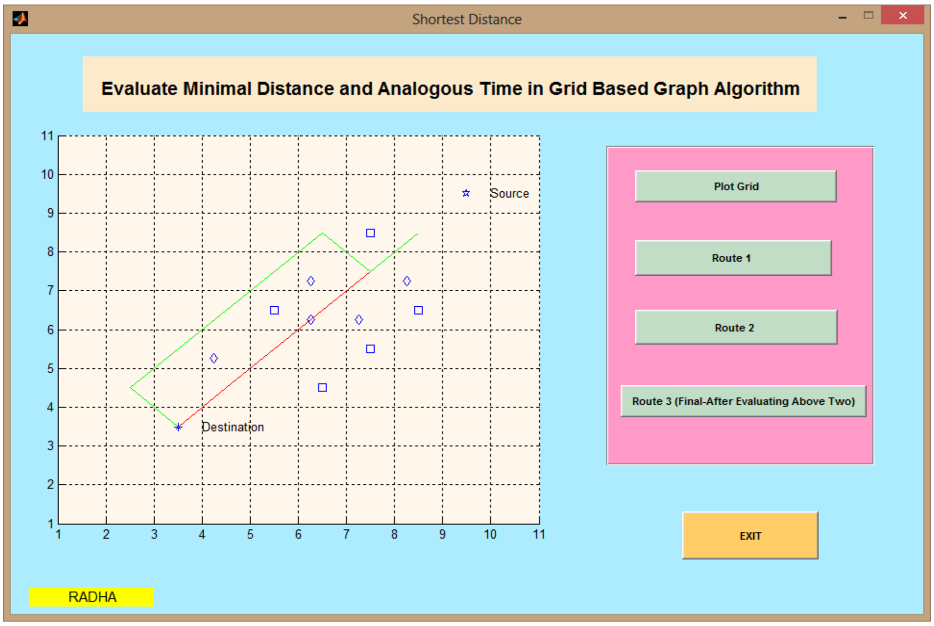

Figure 1

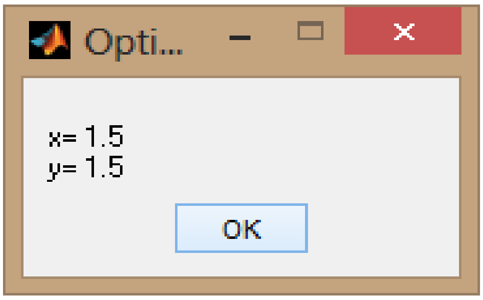

Figure 2

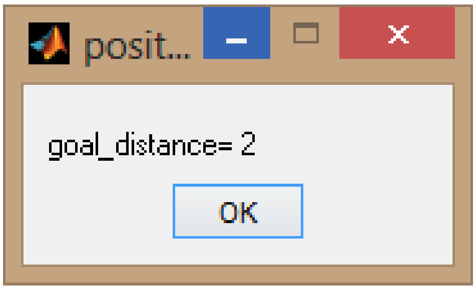

Figure 3

CONCLUSIONS

This shows that the services provide recommendations that satisfy the users. Which service to prefer is a difficult question, since they both have their strengths and weaknesses? Many will probably say that combining the approaches of the two services, would create the ultimate service, something which is not unlikely. It is said that Pandora is considered most promising in becoming the leading music recommender system, than the other way around. However, maintaining the manual work of classifying songs is expensive, and Pandora is probably not delivering proportionally more benefit for that cost.This paper presents a location based recommender system, which provides a user with location recommendations around the specified **geo-position** based on

REFERENCES

1. Pemmaraju, S. and Skiena, S. "All-Pairs Shortest Paths." §8.1.2 in **Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.** Cambridge, England: Cambridge University Press, pp. 330-331, 2003.
2. Skiena, S. "All Pairs Shortest Paths." §6.1.2 in **Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica.** Reading, MA: Addison-Wesley, pp. 228-229, 1990.
3. M. L. Fredman, New bounds on the complexity of the shortest path problem, SIAM Journal on Computing 5(1976) pp. 83-89
4. A. V. Aho, J. Hopcroft and J. B. Ullman, The Design and Analysis of Computer Algorithms, (Addison-Wesley 1974.) pp. 201-206.
5. K. K. Ahuja, K. Mehlhorn, J. B. Orlin and R. E. Tarjan, Faster algorithms for the shortest path problem, Journal of the Association for Computing Machinery 37(1990) pp. 213-223.
6. R. Seidel, On the All-Pairs-Shortest-Path problem, Proc. 24th ACM Annual Symp. on Theory of Computing 1992, to appear.
7. Stuart Russell and Peter Norvig (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall.
8. Zhang, W: Depth-First Branch and Bound versus Local Search: A case study. ISICD, University of Southern California (2000).
9. Garey, M.R., Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman, San Francisco (1979).