Relative Direction: Location Path Providing Method for Allied Intelligent Agent

S. Rayhan Kabir[2], Mirza Mohtashim Alam[1], Shaikh Muhammad Allayear[1,2], Md Tahsir Ahmed Munna[1(✉)], Syeda Sumbul Hossain[2], and Sheikh Shah Mohammad Motiur Rahman[2]

[1] Department of Multimedia and Creative Technology, Daffodil International University, Dhaka, Bangladesh
{mirza.mct, drallayear.swe, tahsir4ll}@diu.edu.bd
[2] Department of Software Engineering, Daffodil International University, Dhaka, Bangladesh
{rayhan56l, syeda.swe, motiur.swe}@diu.edu.bd

Abstract. The most widely recognized relative directions are left, right, forward and backward. This paper has presented a computational technique for tracking location by learning relative directions between two intelligent agents, where two agents communicate with each other by radio signal and one intelligent agent helps another intelligent agent to find location. This proposed method represents an alternative approach to GSM (Global System for Mobile Communications) for the AI (Artificial Intelligence), where no network may not be available. Our research paper has proposed Relative Direction Based Location Tracking (RDBLT) model for understanding how one intelligent agent assists another intelligent agent to find out the location by learning and identifying relative directions. Moreover, three proficient algorithms have been developed for constructing our proposed model.

Keywords: Artificial General Intelligence (AGI) · Multi-agent system
Relative direction · Magnetic Navigation Compass · Radio signal

1 Introduction

Constant or real-time location finding processes are utilized to consequently recognize and track the area. Difficulties in left-right segregation are typically experienced in ordinary every day of real life [1]. Contrasting aspects of relative direction is a regular event in human life, for example, “go forward”, “turn left” or “turn right”. Sometimes these moves or directions are needed for helping find a place or location, for example, “there is a market on your left side”. If we think these aspects from any device’s perspective, where one computational device wants to find a location by using left, right, forward, and backward directions, it is very vital to four relative directions for both perspectives.

M. Singh et al. (Eds.): ICACDS 2018, CCIS 905, pp. 381–391, 2018.
https://doi.org/10.1007/978-981-13-1810-8_38

Relative directions are an auxiliary matter for individuals to locate the cardinal directions. A study displays an investigation regarding the contrasting point of landmark-based guidelines with relative directions over people on foot in genuine city conditions [2] and mentioned relative direction work well than the landmark [3]. A paper tended to an energy proficient routing algorithm, which based on the relative direction [4]. A new research introduces a computation to take in human's relative directions [5], where one intelligent device can take in any human's relative directions. Another new research demonstrates a communication technique for rescuers where data transmit by the signal and estimate the relative localization [6]. In our paper, we displayed a new mapping system, where an intelligent agent wants to help another intelligent agent for finding a particular location by using relative directions; two agents are communicated by radio signal [7, 8]. To keep up with this situation, we have done this research where two intelligent agents help each other for finding a location by using right, left, forward and backward directions.

2 Proposed RDBLT Model

2.1 Handoff–Agent and Tracking–Agent

A recent examination utilized Handoff UAV and Tracking UAV for the hypothesis of the relative attitude between two unmanned flying aircrafts [9]. In our experiment, we utilized two intelligent agents those are Tracking–Agent and Handoff–Agent [10]. Handoff–agent contains the location data. On the other hand, tracking–agent needs to find the location by using relative directions. Figure 1 represents the relative directions and direction points (a, b, c and d) of handoff–agent and also illustrates direction points (a1, b1, c1 and d1) of tracking–agent [11, 12]. In our experiment, handoff– agent's direction points of relative directions are constant but for tracking–agent, direction points depend on several 2D aspects of tracking–agent [13, 14]. In Fig. 1 we have shown the tracking–agent and the various 2D position of tracking–agent.

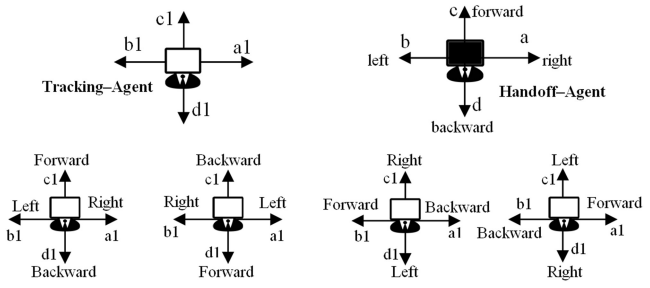

Fig. 1. Relative directions and direction points of two intelligent agents.

2.2 Structure of RDBLT Model

Our research showed Relative Direction Based Location Tracking (RDBLT) model which have demonstrated how one intelligent system assists another intelligent system for finding out the desired location. In this method, there are two intelligent agents are situated in the different area. Imagine a tracking–agent is located in Area-1 and wants to find a specific location. A handoff–agent is located in Area-2 and knows the target location of Area-1. Now tracking-agent communicates with handoff-agent by signal for finding out the desired target place. For that reason, handoff-agent learns and identifies tracking–agent's relative directions by using cardinal directions (North, South, East and West). Then handoff–agent provides a location path to tracking-agent. Afterward, tracking–agent tracks the location. Figure 2 illustrates this scenario.

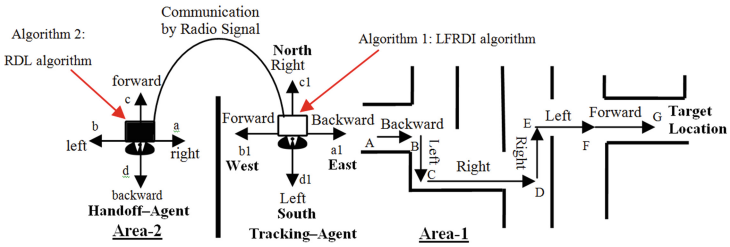

Fig. 2. A scenario of RDBLT model.

In this research, we have used the array structure for learning and identifying relative directions, where directions contain numerical values (See Table 1). The basis of the values depends on specific relative directions of two agents.

Table 1. Numerical value and array index of relative directions

Relative directions of handoff-agent	Relative directions of tracking-agent	Array index and contain values
Right	Right	0
Left	Left	1
Forward	Forward	2
Backward	Backward	3

2.3 Steps of RDBLT Model

Here, we show the structure of Relative Direction Based Location Tracking (RDBLT) model in Artificial General Intelligence (AGI) point of view.

At first, tracking-agent gives a request to handoff–agent by a radio signal for finding target location path. Handoff-agent collects data from the database and learns the location path for tracking–agent's 2D aspects (See Fig. 1 and Fig. 2), where handoff–agent uses the Location Finder's Relative Direction Identification (LFRDI) algorithm,

Relative Direction Learning (RDL) algorithm and RDBLT algorithm (See Algorithms 1, 2 and 3). Then the handoff–agent provides a location path to the tracking–agent. Consequently, tracking–agent can get the relative direction based path of target location. The steps of RDBLT model have been shown in Fig. 3.

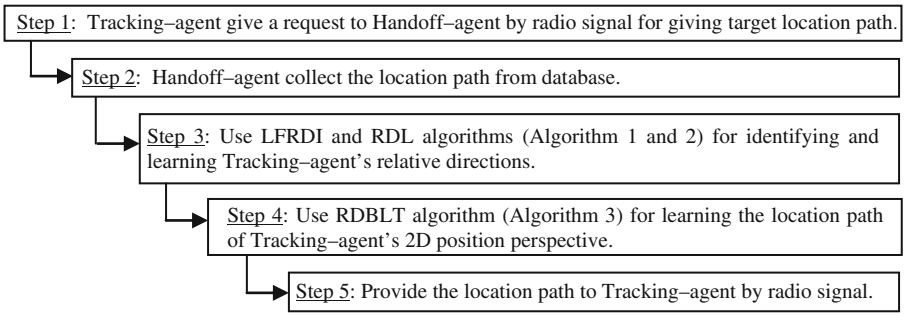

Fig. 3. The steps of RDBLT model.

3 Algorithms of RDBLT Model

3.1 LFRDI Algorithm

Before providing the location path, handoff–agent needs to know the 2D position of tracking–agent. Handoff–agent can identify tracking agent's relative directions by using cardinal directions such as north, south, east, and west. A Magnetic Navigation Compass helps for knowing these directions. The approach of the LFRDI algorithm by the idea of programming perspective is as follows:

- An array *trackingAgent* is created which contains relative directions (*Right, Left, Forward* and *Backward*) of tracking-agent and respectively containing 0, 1, 2 and 3 direction values (See Table 1). We also declared direction points (*a1, b1, c1* and *d1*) of tracking-agent (See Fig. 1).
- Cardinal Directions (*North, South, East* and *West*) also declared which are being obtained by a Magnetic Navigation Compass (See Fig. 2).
- A loop has occurred where variable $k = 0$ to 3. If particular cardinal direction (*North, South, East* and *West*) is coordinated by the specific relative directions (*Right, Left, Forward* and *Backward*). Subsequently, specific direction points (*a1, b1, c1* and *d1*) contain specific relative directions values.
- Upon completion of this loop tracking–agent returns the value of these direction points to handoff–agent by radio signal.

We have exhibited LFRDI algorithm for this issue which is based on Eq. 1. Let North, South, East and West directions are defined as N, S, E and W respectively. Tracking–agent is defined as T.

$$\forall_k \in \{0,1,2,3\} f\,(N,S,E,W) := \begin{cases} \text{a1} \leftarrow \text{k } \textit{if } \text{N} = T_k \\ \text{b1} \leftarrow \text{k } \textit{if } \text{S} = T_k \\ \text{c1} \leftarrow \text{k } \textit{if } \text{E} = T_k \\ \text{d1} \leftarrow \text{k } \textit{if } \text{W} = T_k \end{cases} \tag{1}$$

Here, we are assigning each of the k (ranges from 0 to 3) based on certain inputs of the cardinal directions N, S, E and W. Whatever the Tracking–agent's direction matches with the Handoff–agent's direction, we are assigning the *k*'s value to the direction point (a1, b1, c1, d1) (See Eq. 1). The formation of LFRDI algorithm for the programming point of view is as per the following algorithm 1:

Algorithm 1. LFRDI Algorithm

```
1: function LFRDI():
2:   trackingAgent[] ← [Right, Left, Forward, Backward];
3:   Right ← 0; Left ← 1; Forward ← 2; Backward ← 3;
4:   direction points: a1; b1; c1; d1;
5:   Cardinal direction: North; South; East; West;
6:   /* directions are get by Magnetic Compass */
7:   for k := 0 to 3
8:      if North == trackingAgent[k] then, a1 ← k;
9:      else if South == trackingAgent[k] then, b1 ← k;
10:     else if East == trackingAgent[k] then, c1 ← k;
11:     else if West == trackingAgent[k] then, d1 ← k;
12: end for
13: return a1, b1, c1, d1; /* fixed direction points
14: are returned to caller in Algorithm 2 and 3 */
```

3.2 Relative Direction Learning Algorithm

In this section, we demonstrate Relative Direction Learning (RDL) algorithm for learning tracking–agent's relative directions. Through this algorithm, handoff–agent can give relative direction based location tracking instruction to the tracking-agent. The statement of RDL algorithm by the idea of programming standpoint is as follows:

- Array *handoffAgent* is created which contains handoff–agent's relative directions (*right, left, forward* and *backward*) and respectively containing 0, 1, 2 and 3 direction values (See Table 1). Moreover, direction points (*a, b, c* and *d*) of handoff–agent contains relative directions (See Fig. 1).
- Another array *trackingHuman* is created which also contains relative directions (*Right, Left, Forward* and *Backward*). Their direction point variables (*a1, b1, c1* and *d1*) have been declared accordingly. These points can be identified by calling LFRDI function (See Algorithm 1). These points contain values (See Table 1) which depend on the 2D aspects of tracking–agent (see Fig. 1).
- A loop has been created where variable $i = 0$ to 3. We have declared j, where $j = 0$, 1, 2, 3. When i is equal to direction point (*a1, b1, c1* and *d1*) of tracking-agent then

the value of *j* changes and *j* index of the *handoffAgent* array is assigned to *i* index of the *trackingAgent* array. Through this loop, handoff–agent can learn tracking–agent's relative directions.

- Upon completion of this loop, learning directions are returned to *getLocation (Location)* function of *HandoffAgent* class for routing location which can be easily perceived by seeing Algorithm 3.

Let the direction points be defined as a1, b1, c1 and d1. The Handoff–agent is defined as H and Tracking-agent as T. We are assigning Handoff–agent's indices to certain values (0 to 3) based on function's input values (a1, b1, c1, d1) for each *i* from 0 to 3 (See Eq. 2). Subsequently, each of the indexed values of handoff–agent also have been assigned to tracking-agent's index values (See Eq. 3).

$$\forall_i \in \{0,1,2,3\} f := \begin{cases} H_{j\leftarrow 0} \; if \; i = a1 \\ H_{j\leftarrow 1} \; if \; i = b1 \\ H_{j\leftarrow 2} \; if \; i = c1 \\ H_{j\leftarrow 3} \; if \; i = d1 \end{cases} \tag{2}$$

$$T_i := H_j \tag{3}$$

The formation of RDL algorithm for programming perspective is according to Algorithm 2:

Algorithm 2. RDL Algorithm

```
Function RDL ():
   handoffAgent[] ← [right, left, forward, backward];
   directions points: a; b; c; d;
   right ← 0; left ← 1; forward ← 2; backward ← 3;
   a ← right; b ← left; c ← forward; d ← backward;
   trackingAgent[] ← [Right, Left, Forward, Backward];
   directions points: a1, b1, c1, d1 ← LFRDI();
   /* Call Procedure LFRDI() of Algorithm 1*/
   int j;
   for i:= 0 to 3
      if i == a1 then, j ← 0;
      else if i == b1 then, j ← 1;
      else if i == c1 then, j ← 2;
      else if i == d1 then, j ← 3;
      trackingAgent[i] ← handoffAgent[j];
   end for
   return trackingAgent[];
   /*learning directions are returned to HandoffAgent
   Class of Algorithm 3*/
```

3.3 Structure of RDBLT Algorithm

The procedure of RDBLT algorithm by the idea of the programming perspective is as follows

- At first two class *TrackingAgent* and *HandoffAgent* are created, for representing tracking–agent and handoff–agent
- Tracking–agent wants to get location path from handoff–agent. For that reason, another array *Path* can acquire the target location path by calling *getLocation (Location)* function of *HandoffAgent* class.
- In our experiment handoff–agent contains the location data in the database. An array *LocationPath* of class *HandoffAgent* collects the data of target location.
- An array *Direction* is created which contains handoff–agent's relative directions (*right, left, forward* and *backward*) and respectively contained 0, 1, 2 and 3 direction values (See Table 1).
- For providing the relative direction based mapping, array *LearningDirection* can get of the learning relative directions of tracking–agent by calling RDL() function (See Algorithm 2). For the same reason, handoff–agent need to know the tracking–agent's direction points (*a1, b1, c1* and *d1*) which is called by LFRDI() function (See Algorithm 1).
- A variable *h* has been declared, where $h = 0$. Then declare an array *PathSegment* for understanding the segmentation of *LocationPath* (See Fig. 4).

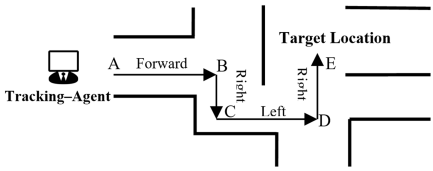

Fig. 4. Figure shows that, a *LocationPath* which is A→E, where have several path segments, which are A→B = Forward = 2, B→C = Right = 0, C→D = Left = 1 and D→E = Right = 0 (See Table 1). In order that, *PathSegment[]* =[2, 0, 1, 0, 1].

- A while loop is finished before the competition of *LocationPath* array length. Another inner loop appears, where variable $i = 0$ to 3.
- If *h* index of *LocationPath* is equal to *i* index of *LearningDirection* then, another inner four conditional statements (if-else if) have occurred where handoff–agent is checking the *LocationPath* for tracking–agent's 2D aspects (See Fig. 1)
- If specific tracking–agent's direction point (*a1, b1, c1* and *d1*) is equal to *i* index of *Direction* array, then the specific direction point is assigned into *PathSegment* array. So handoff–agent can estimate the path for tracking–agent's 2D position.
- Upon completion of these loops, handoff–agent is provided the *PathSegment* to tracking-agent. As a result, tracking-agent gets the location path.

The formation of RDBLT algorithm for programming viewpoint is according to Algorithm 3:

Algorithm 3. RDBLT Algorithm

```
TrackingAgentclass:
  Location ← Tracking or target location;
  HandoffAgent HA = new HandoffAgent();
  Path[] ← HA.getLocation(Location);

HandoffAgentclass:
  getLocation(Location):
     LocationPath[] ← Data(Location);
     Direction[] ← [right, left, forward, backward];
     right ← 0; left ← 1; forward ← 2; backward ← 3;
     LearningDirection[] ← RDL();
     /* Call Procedure RDL() of Algorithm 2 */
     directions points: a1, b1, c1, d1 ← LFRDI();
     /* Call Procedure LFRDI()of Algorithm 1 */
     PathSegment[];
     h ← 0;
     While(LocationPath[].length() not complete)
         for i:= 0 to 3
            if LocationPath[h] == LearningDirection[i]
               if a1 == Direction[i]
                  PathSegment[h] ← a1;
               else if b1 == Direction[i]
                  PathSegment[h] ← b1;
               else if c1 == Direction[i]
                  PathSegment[h] ← c1;
               else if d1== Direction[i]
                  PathSegment[h] ← d1;
            end if
         end for
     h++;
     end while
     return PathSegment[]; /*Return to Tracking-agent*/
```

4 Conceptual Result and Analysis

Figure 5 represents a test case where tracking–agent and handoff–agent are situated in isolated area and their relative direction positions are also different. Now tracking–agent communicates with handoff–agent for getting target location path.

In (A) section of Fig. 5, tracking–agent gives a request to handoff–agent for getting target location path. Handoff–agent contains the location path in the database. Assume that, for handoff–agent's 2D perspective, the Location Path = [right, left, right] = [0, 1, 0] (See Table 1). In the (B) section, by using LFRDI, RDL and RDBLT algorithms,

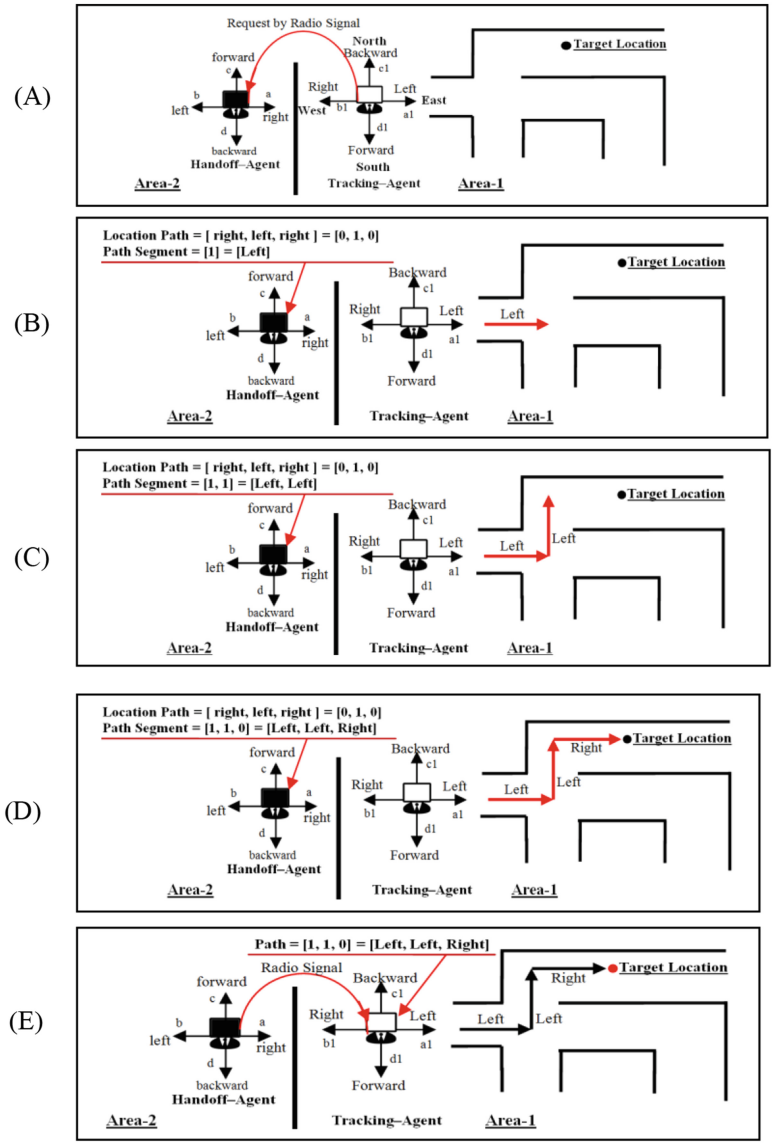

Fig. 5. A simple test case of RDBLT model.

handoff–agent can track the first path which is left. The coordination of tracking agent's Left direction and handoff–agent's right direction is the same. By using these three algorithms full path is achieved, which have shown in (C) and (D) section of Fig. 5. Learning direction values can not follow the Table 1, because these values depend on the distinct 2D position of two agents. After using loop of RDBLT algorithm, the Path Segment = [1, 1, 0] = [Left, Left, Right] is acquired (See Table 1). Finally, handoff–agent

sends this path segment to tracking–agent by radio signal. The inputs and outputs of this test case have been shown in Table 2.

Table 2. Inputs and outputs of purpose test case

Using LFRDI algorithm				Using RDL algorithm				Using RDBLT algorithm			
Tracking-agent				Handoff-agent		Tracking-agent		Handoff-agent		Tracking-agent	
Point	Relative direction	Value	Cardinal Direction	Relative direction	Value	Learning direction	Value	Location path	Value	Path segment	Value
a1	Left	1	East	Right	0	Right	1	Right	0	Left	1
b1	Right	0	West	Left	1	Left	0	Left	1	Left	1
c1	Backward	3	North	Forward	2	Forward	2	Right	0	Right	0
d1	Forward	2	South	Backward	3	Backward	3				

5 Future Works and Conclusion

The research illustrates an introduction to RDBLT model where represents how one intelligent device helps another device for tracking location by learning relative directions. RDBLT model is currently under development in the perspective of Three Dimension (3D), Route Direction and Machine Learning. These are key examination for better comprehension about relative bearing in our future exercises. All the relative direction can be changed into a scholarly model by various learning calculations (Machine Learning, and Neural Networks). In our fourth coming exploration, we are hopeful to complete a real-world test with Machine Learning and Computer Vision with our own deployed bots.

References

1. Hjelmervika, H., Westerhausena, R., Hirnsteina, M., Spechta, K., Hausmann, M.: The neural correlates of sex differences in left–right confusion. NeuroImage **113**, 196–206 (2015)
2. Götze, J., Boye, J.: "Turn left" versus "walk towards the café": when relative directions work better than landmarks. In: Bação, F., Santos, M.Y., Painho, M. (eds.) AGILE 2015. LNGC, pp. 253–267. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16787-9_15
3. Albert, W.S., Rensink, A., Beusmans, J.M.: Learning relative directions between landmarks in a desktop virtual environmen. Spat. Cogn. Comput. **1**, 131–144 (1999)
4. Weng, C.E., Lai, T.W.: An energy-efficient routing algorithm based on relative identification and direction for wireless sensor networks. Wirel. Pers. Commun. **69**, 253–268 (2013)
5. Kabir, S.R., Allayear, S.M., Alam, M.M., Munna, M.T.A.: A computational technique for intelligent computers to learn and identify the human's relative directions. In: International Conference on Intelligent Sustainable Systems, pp. 1036–1039. IEEE Xplore, India (2017)
6. Lurz, F., Mueller, S., Lindner, S., Linz, S., Gardill, M., Weigel, R., Koelpin, A.: Smart communication and relative localization system for firefighters and rescuers. In: 2017 IEEE MTT-S International Microwave Symposium (IMS), pp. 1421–1424. IEEE Xplore, USA (2017)

7. Lowrance, C.J., Lauf, A.P.: Direction of arrival estimation for robots using radio signal strength and mobility. In: 2016 13th Workshop on Positioning, Navigation and Communications (WPNC). IEEE Press (2016)
8. Kolster, A.F., Dunmore, F.W.: The Radio Direction Finder and Its Application to Navigation, Washington (1921)
9. Mahmood, A., Wallace, J.W., Jensen, M.A.: Radio frequency UAV attitude estimation using direction of arrival and polarization. In: 11th European Conference on Antennas and Propagation, pp. 1857–1859. IEEE Press, France (2017)
10. Kabir, S.R.: Computation of multi-agent based relative direction learning specification. B.Sc. thesis, Daffodil International University, Bangladesh (2017)
11. Mossakowski, T., Moratz, R.: Qualitative reasoning about relative direction of oriented points. Artif. Intell. **180–181**, 34–45 (2012)
12. Moratz, R.: Representing relative direction as a binary relation of oriented points. In: 17th European Conference on Artificial Intelligence, pp. 407–411. IOS Press, Netherlands (2006)
13. Hahn, S., Bethge, J., Döllner, J.: Relative direction change - a topology-based metric for layout stability in treemaps. In: 12th International Conference on Information Visualization Theory and Applications, pp. 88–95. Portugal (2017)
14. Lee, J.H., Renz, J., Wolter, D.: StarVars—effective reasoning about relative directions. In: Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013), pp. 976–982. AAAI Press, California (2013)