The Bucket Box Intersection (BBI) Algorithm For Fast Approximation Evaluation Of Diagonal Mixture Gaussians

J. Fritsch, I. Rogina

指導教授：陳嘉平

報告人：陳柏含

Introduction

- The BBI algorithm uses a pre-computed binary decision tree to dynamically determine M most significant Gaussians set in the mixture.

- The algorithm is found to be superior in time complexity to both the top-1 and top-all algorithms.

- Word accuracy is still higher than that achieved by restricting the mixture to the top-1 Gaussian.

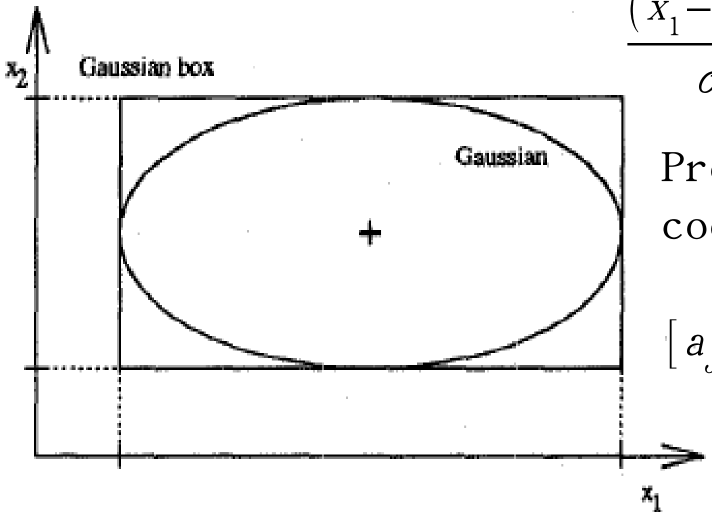
Gaussian Boxing

- Log of a single multivariate Gaussian pdf with diagonal covariance

$$\log N(x,\mu,\sum)=-\frac{1}{2}[\log((2\pi)^K\prod_{j=1}^{K}\sigma_j^2)+\sum_{j=1}^{K}\frac{(x_i-\mu_j)^2}{\sigma_j^2}]>T$$

- This function which give higher logprobs than a absolute threshold *T* is a hyperellipsoid with axes parallel to the coordinate axes.

$$\frac{(x_1-\mu_1)^2}{\sigma_1^2}+\frac{(x_2-\mu_2)^2}{\sigma_2^2}\cdots+\frac{(x_K-\mu_K)^2}{\sigma_K^2}<-2[T+\frac{1}{2}\log((2\pi)^K\prod_{j=1}^{K}\sigma_j^2)]$$



Projection interval of coordinate j is given by :

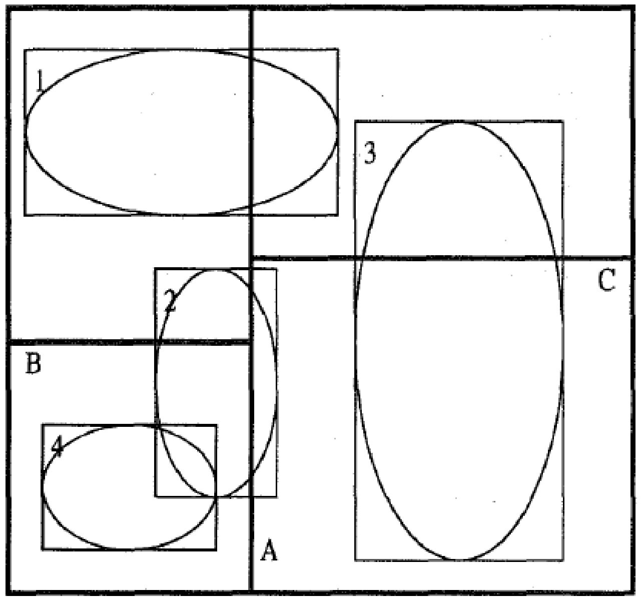
$$[a_j,b_j]=\mu_j\pm\sqrt{-2\sigma_j^2[T+\frac{1}{2}\log((2\pi)^K\prod_{j=1}^{K}\sigma_j^2)]}$$

Bucket Box Intersection Algorithm (1/2)

- Gaussian mixture computation:
  - Replace the computation of all mixtures with computation of the mixture to Gaussians with boxes that contain the current feature vector
- Using *K*-dimensional space partitioning tree (*K*-*d* tree) to find the Gaussian boxes containing a vector *x*
- *K*-*d* tree
  - Depth: *d*
  - Disjoint rectangular regions (bucket): 2^d
  - Non-terminal node: A hyperplane orthogonal to one of the *K* coordinate axes
  - Leaf node: intersection Gaussian boxes

Bucket Box Intersection Algorithm (2/2)

- Localizing a given vector x in one of the buckets by d times scalar comparisons
- Restrict the Gaussian mixture computation to the evaluation of the Gaussians whose boxes are intersecting with the current bucket.



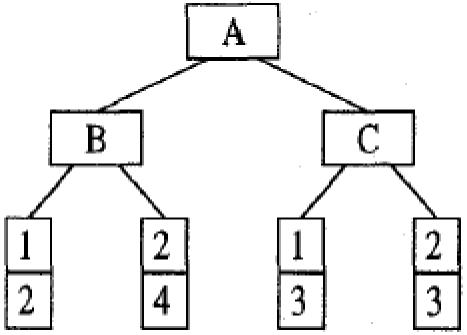
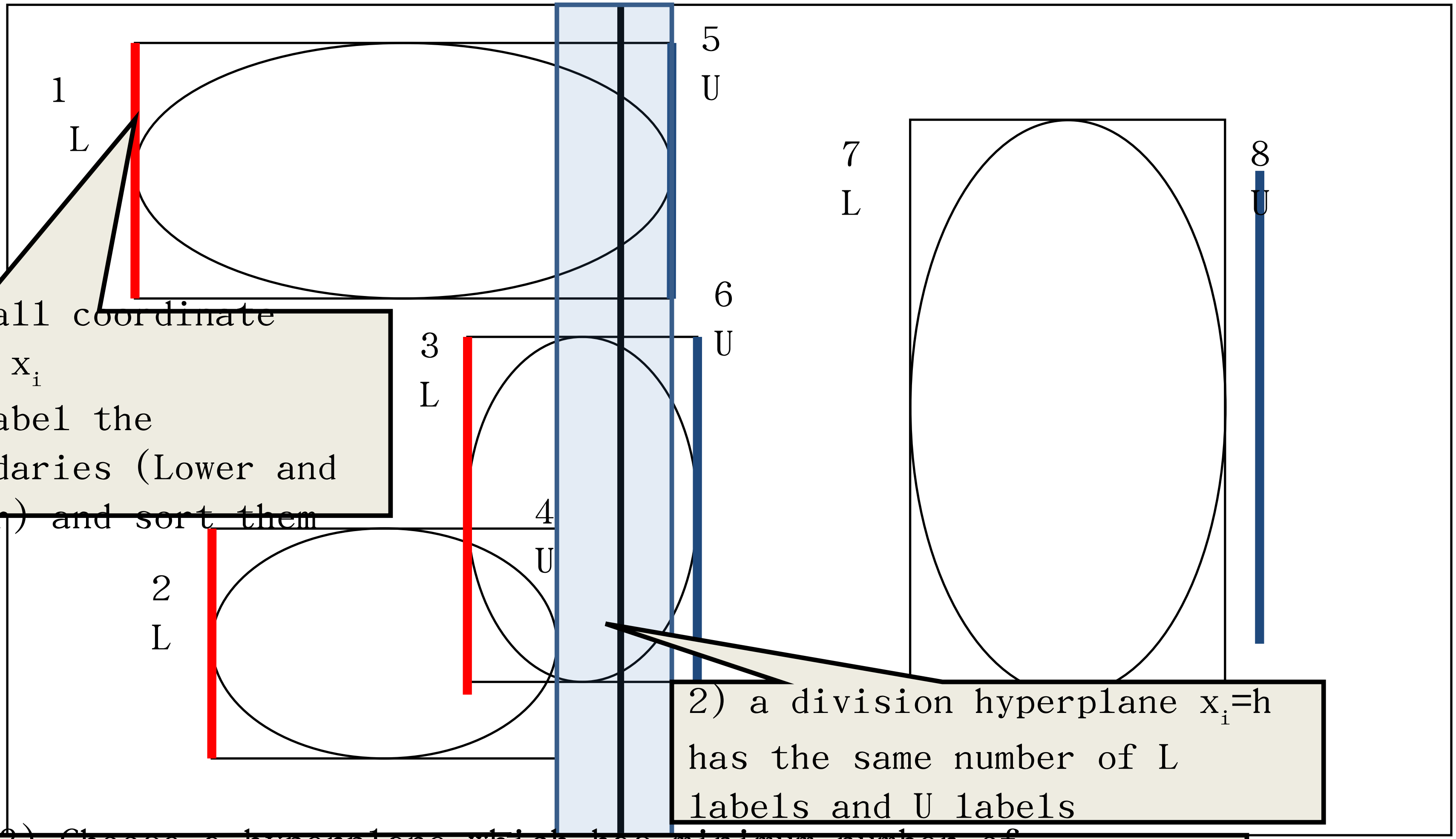


Fig. 2: Bucket Box Intersection (BBI)

Optimizing The Search Tree

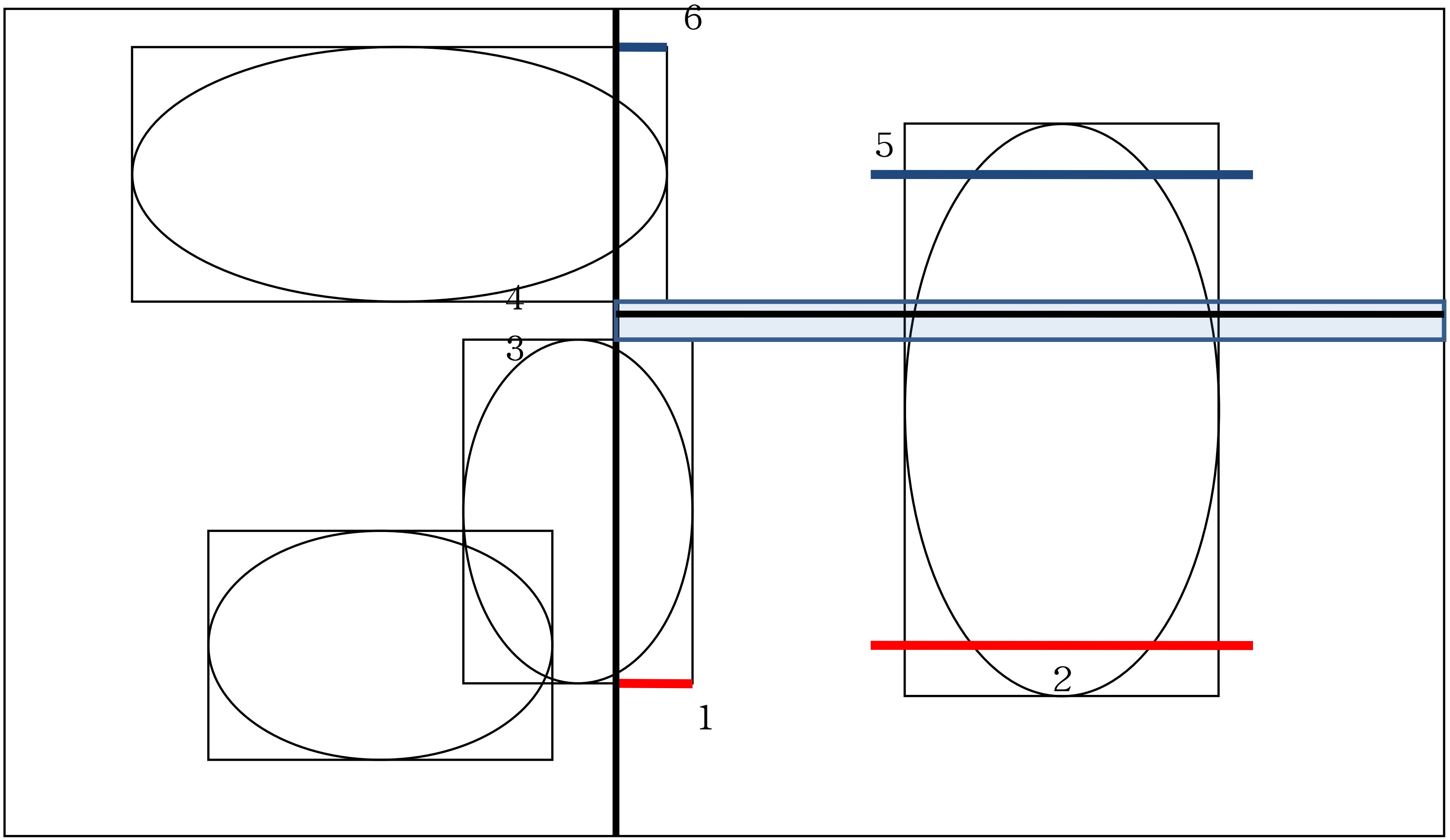


For all coordinate axes x_i

1) Label the boundaries (Lower and Upper) and sort them

2) a division hyperplane $x_i=h$ has the same number of L labels and U labels

3) Choose a hyperplane which has minimum number of C_i(intersection Gaussian boxes that are split by this hyperplane)


6
5
4
3
2
1

Experiment

- German spontaneous scheduling task
- CDHMM system
  - 3300 mixtures
  - 1300 codebooks containing 50 vector
  - 16 LDA-transformed MFCC

Absolute threshold vs. Relative threshold

- By the observation, the maxima of Gaussians in codebooks differ by more than 7 orders of magnitude.

$$[a_j, b_j] = \mu_j \pm \sqrt{-2\sigma_j^2 \log(R)}$$

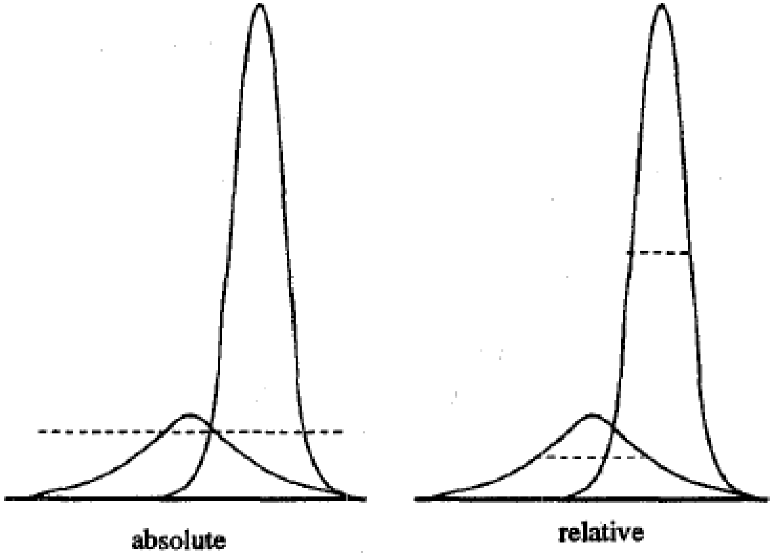
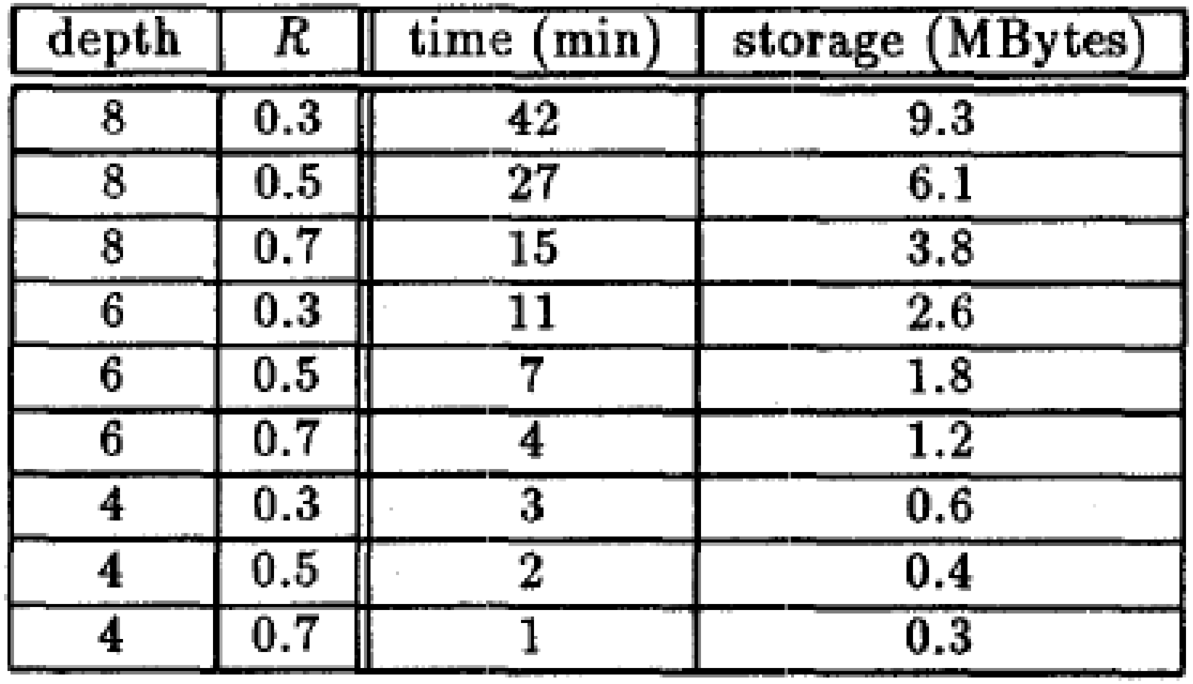


Fig. 3: Absolute vs. Relative Gaussian boxing

- Adopt relative threshold



depth	R	time (min)	storage (MBytes)
8	0.3	42	9.3
8	0.5	27	6.1
8	0.7	15	3.8
6	0.3	11	2.6
6	0.5	7	1.8
6	0.7	4	1.2
4	0.3	3	0.6
4	0.5	2	0.4
4	0.7	1	0.3

Fig. 4: K-d trees computation time and storage requirements

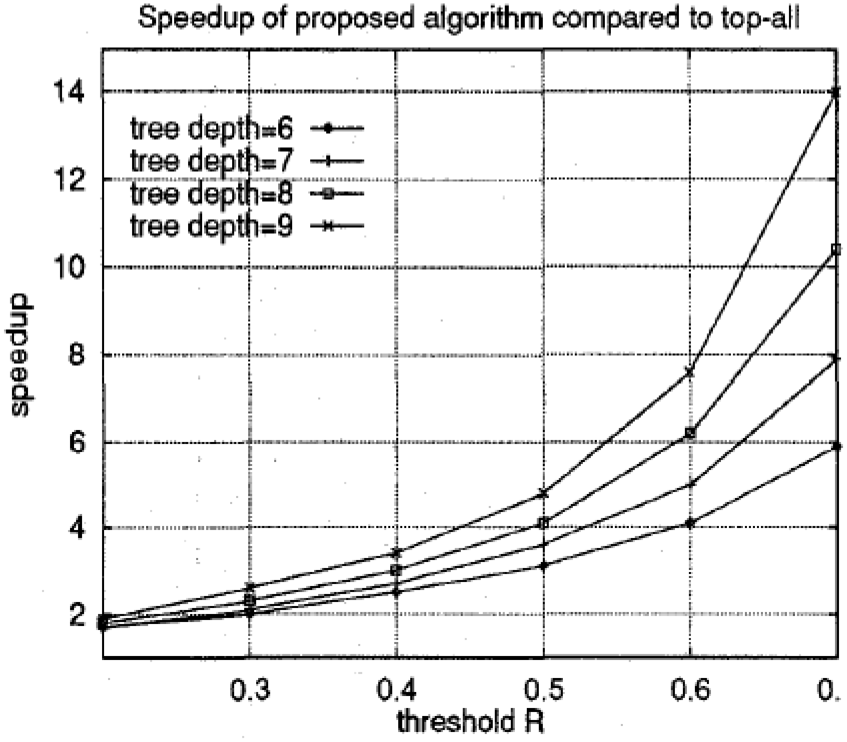


Fig. 5: Speedup of proposed algorithm

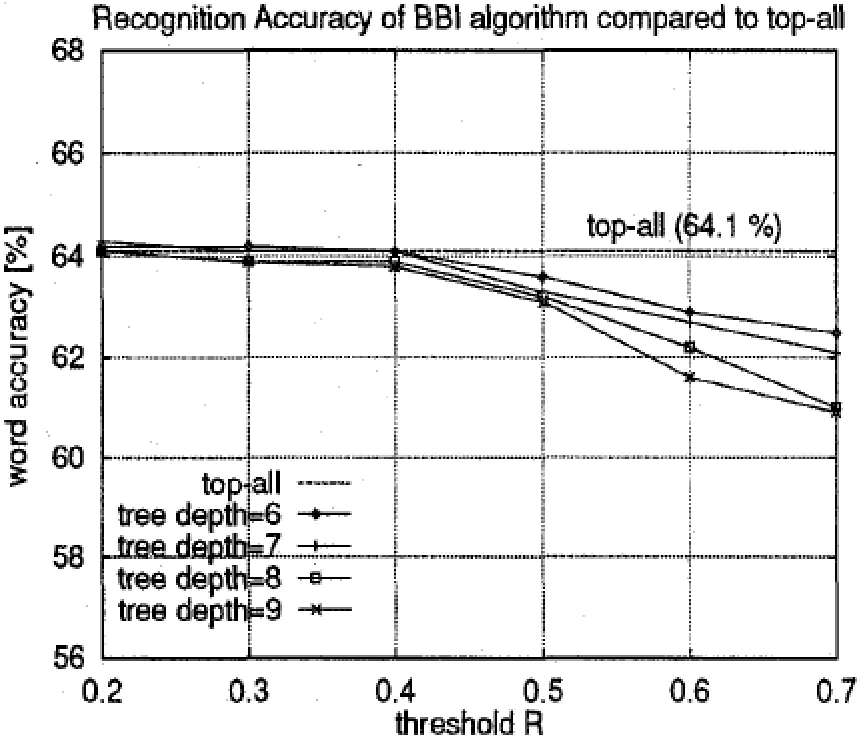


Fig. 6: Word Accuracy on German Spontaneous Scheduling Task