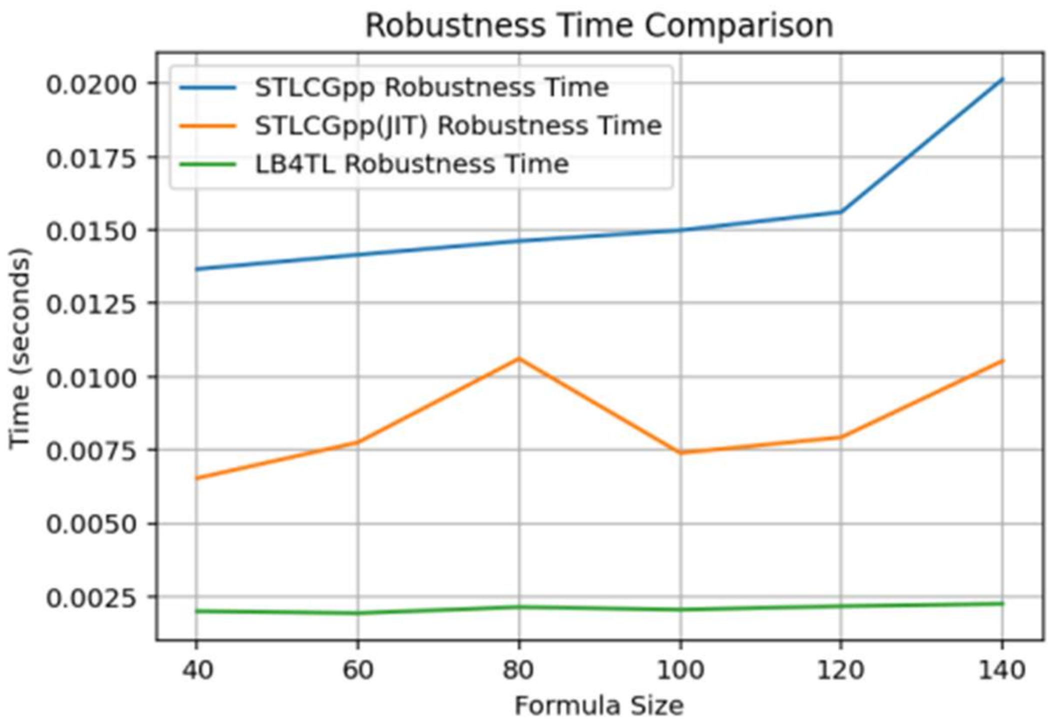
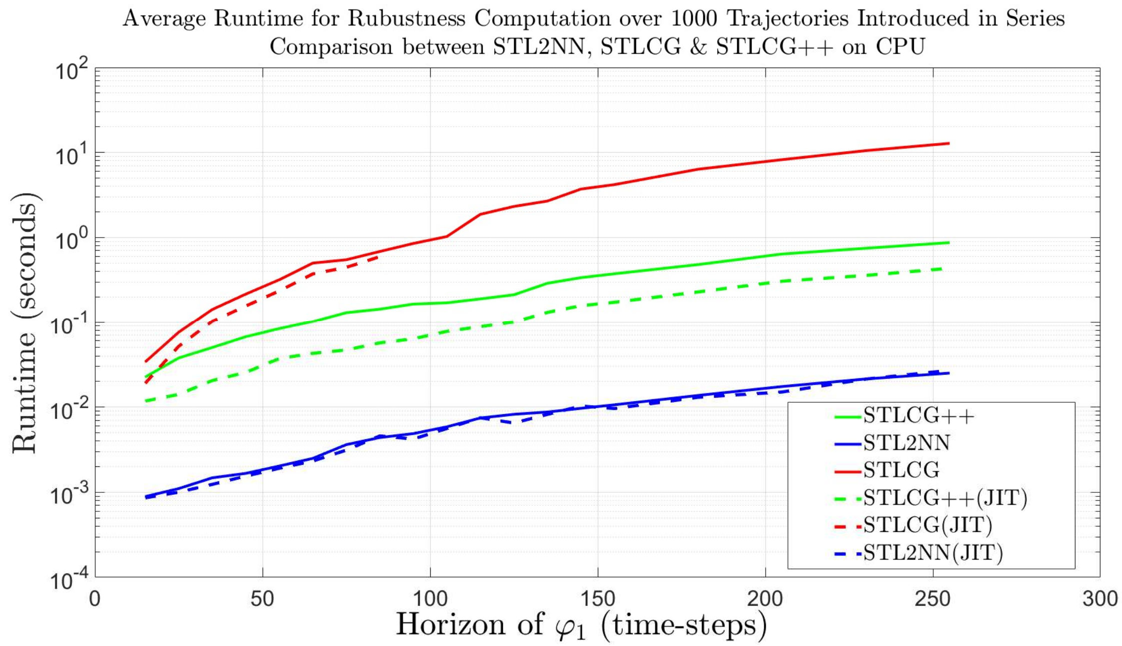
CPU



When the formula does not result in the explosion of the width of the computation graph we outperform STLCG++

$$\varphi_t = \bigwedge_{i=0}^{19} F_{[it\ ,(i+1)t]} \qquad , \qquad t = 1,2,3,4,5,6,7$$

CPU



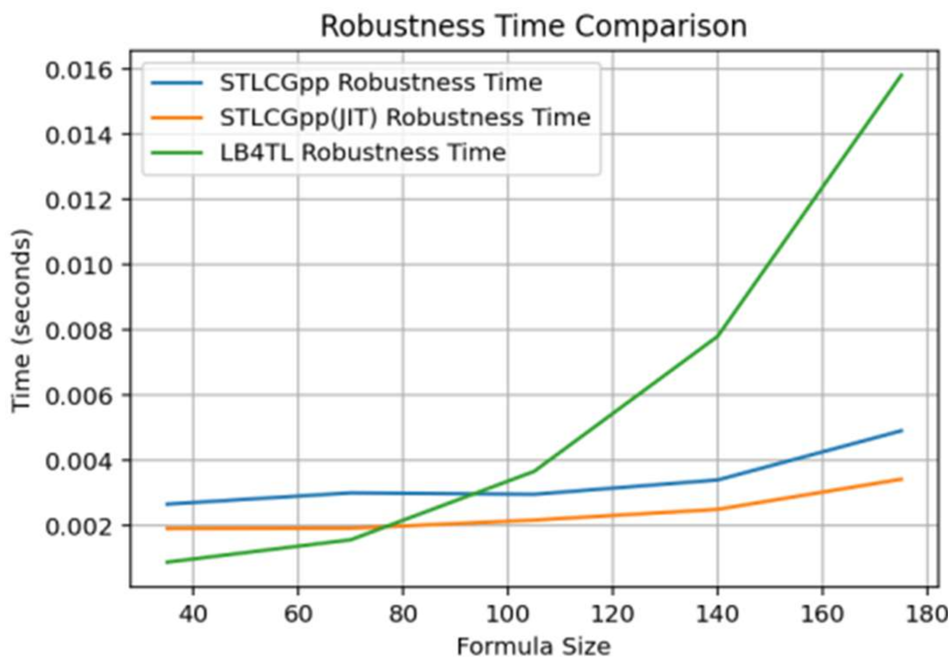
When the formula does not result in the explosion of the width of the computation graph we outperform STLCG++

$$\varphi_1 = F_{[0,T]}\,(\,\text{Goal1 then Goal2}), \qquad T = 15{,}20, \dots, 255$$

However, STL2NN has its own weakness and drawback. Unlike its depth that increases logarithmically, Its width increases linearly with the complexity of the formula. This means, in this case STLCG++ finds a way to take advantage of this drawback and outperforms STL2NN.

Look at an example the results in the explosion of the with of computation graph in the next slide.

CPU



When the formula results in the explosion of the width of the computation graph we face weakness and STLCG++ finds the opportunity to outperform STL2NN

$$\varphi = F_{[5t,8t]}\left(P_1 \wedge F_{[6t,11t]}\left(P_2 \wedge F_{[6t,7t]}\left(P_3 \wedge F_{[8t,9t]}\, P_4\right)\right)\right)$$

$$t = 1,2,3,4,5$$