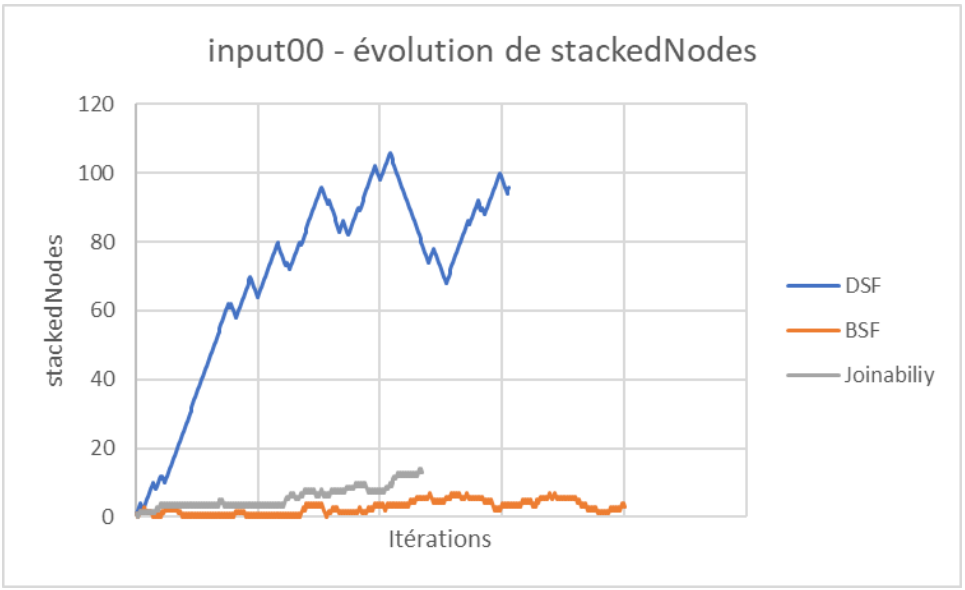
POLYTECHNIQUE
MONTRÉAL

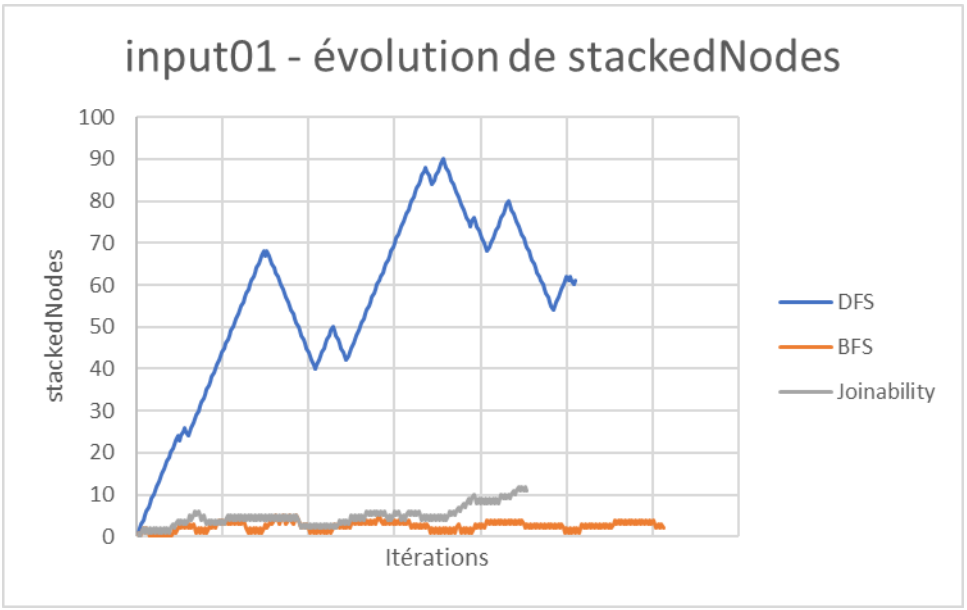
LOG2010
Structures de données et algorithmes

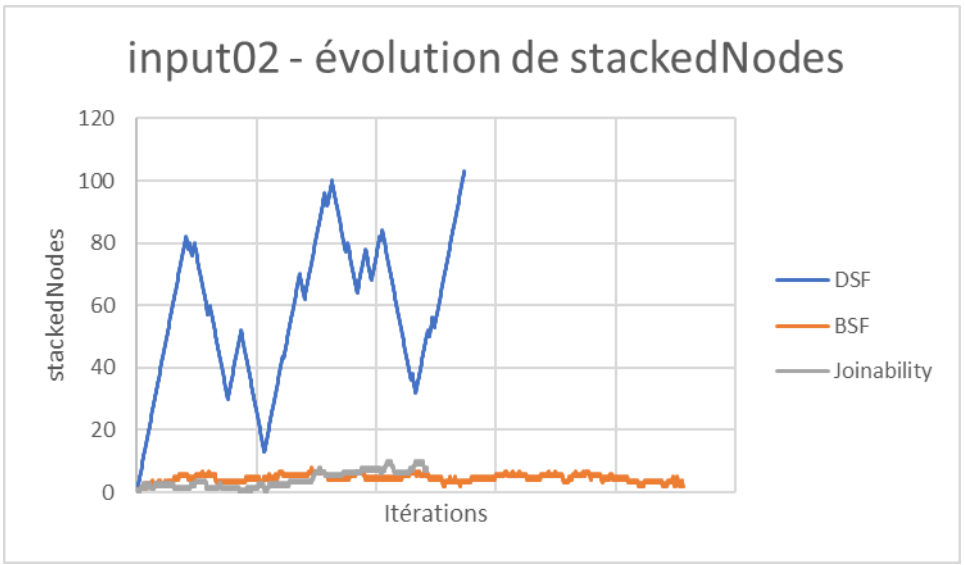
Laboratoire 5

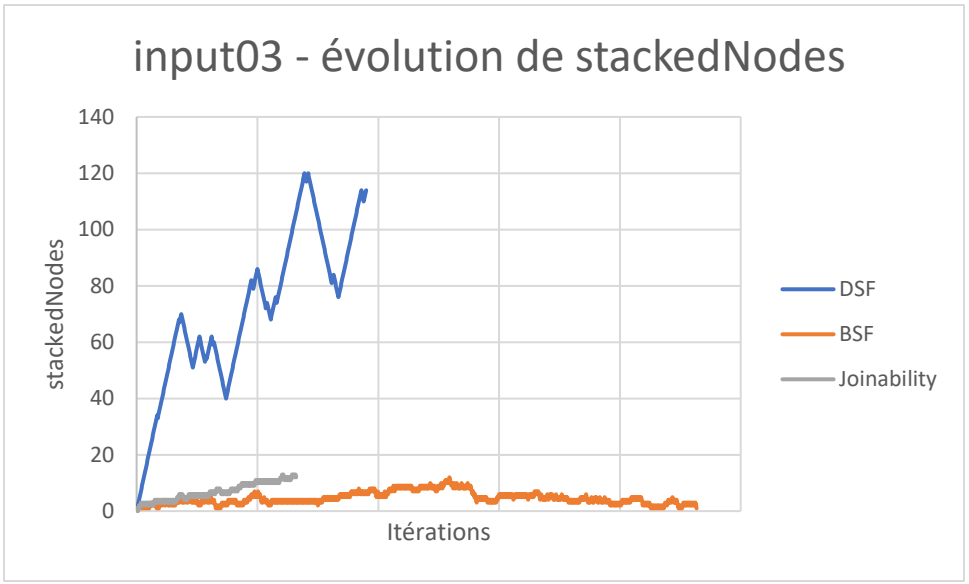
Soumis par:
Marsolais, Edouard - 2154475
Mehdi EL Harami - 2113402

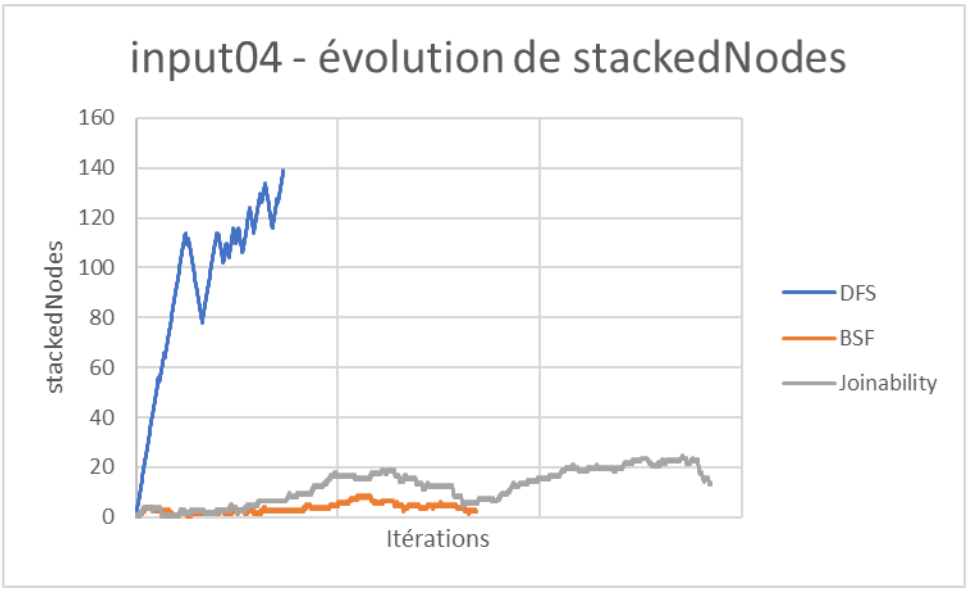
27 novembre 2022

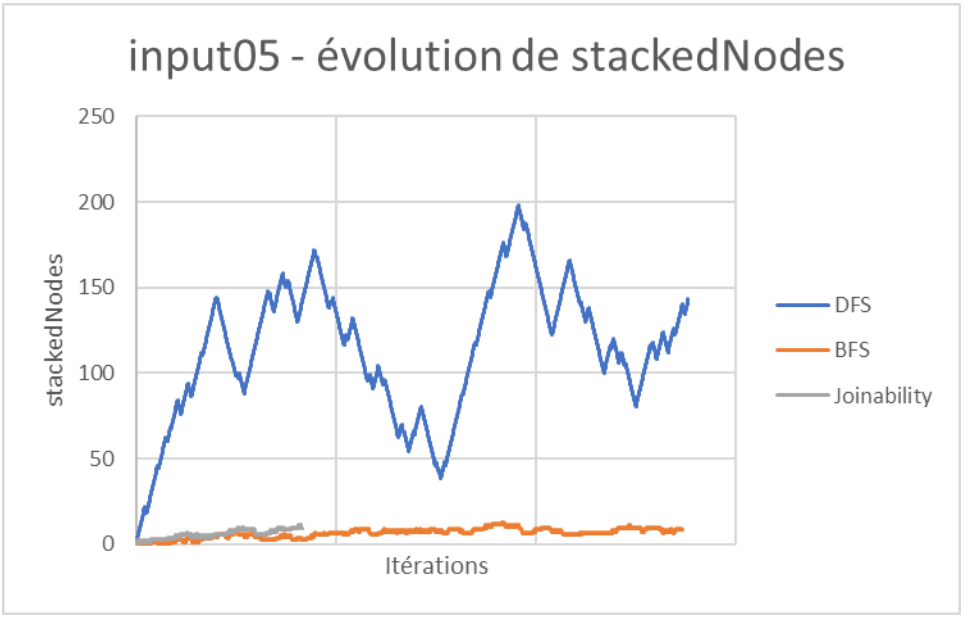

input00 - évolution de stackedNodes
stackedNodes
120
100
80
60
40
20
0
Itérations
DSF
BSF
Joinabiliy

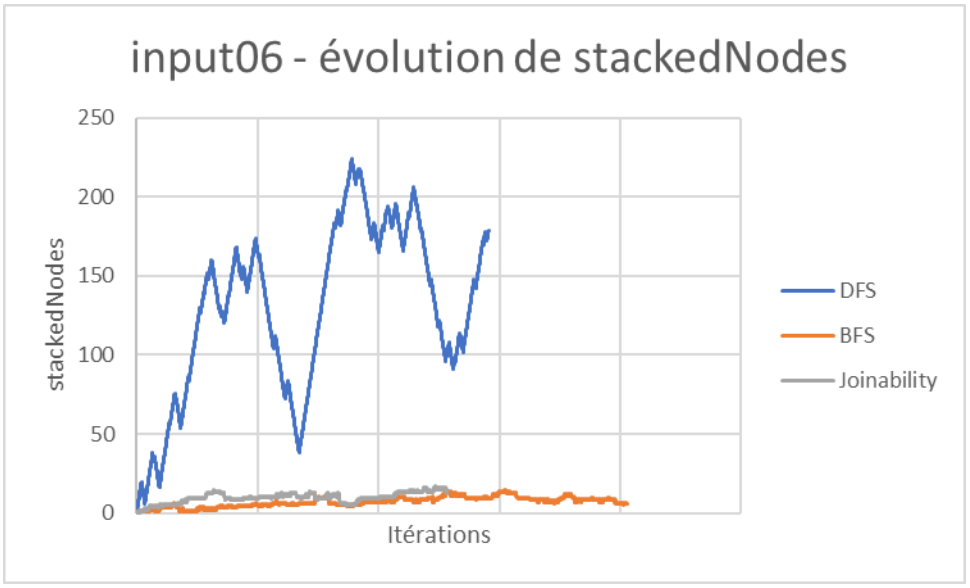

input01 - évolution de stackedNodes
stackedNodes
100
90
80
70
60
50
40
30
20
10
0
Itérations
DFS
BFS
Joinability

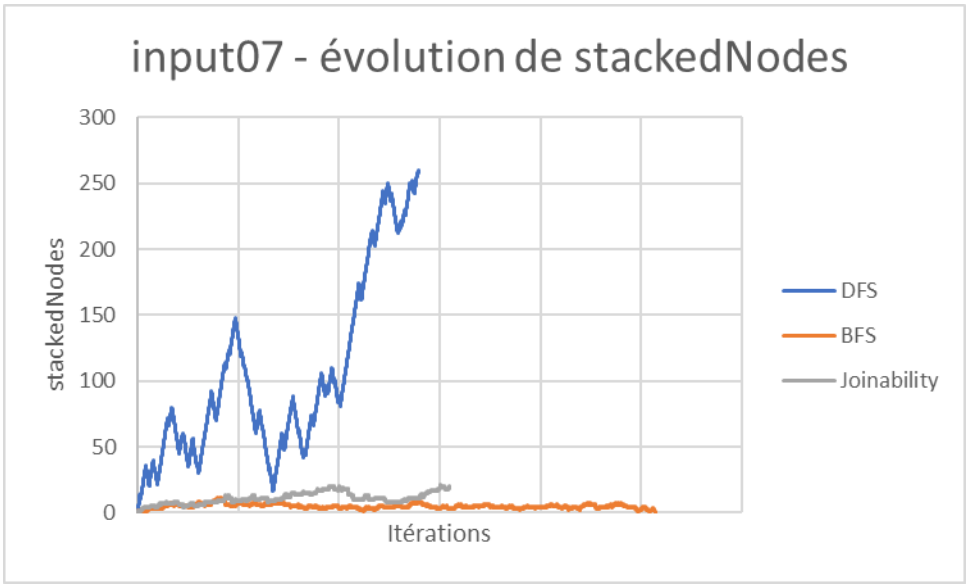

input02 - évolution de stackedNodes
stackedNodes
120
100
80
60
40
20
0
Itérations
DSF
BSF
Joinability

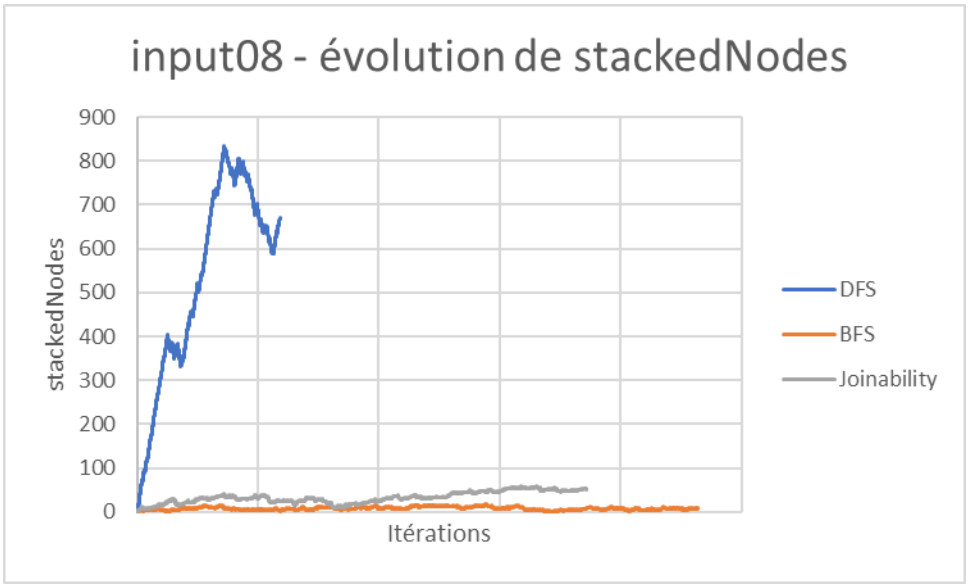

input03 - évolution de stackedNodes
stackedNodes
140
120
100
80
60
40
20
0
Itérations
DSF
BSF
Joinability

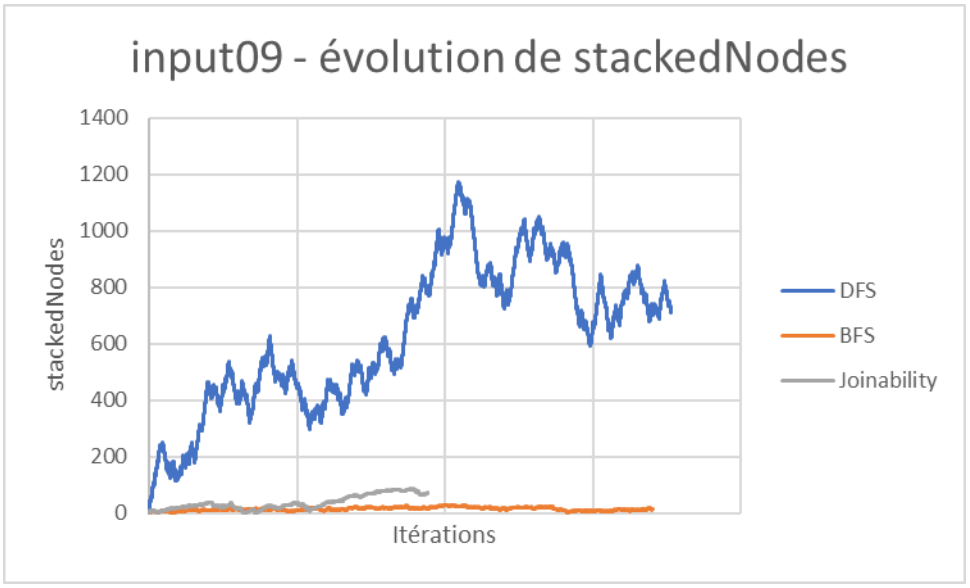

input04 - évolution de stackedNodes
stackedNodes
160
140
120
100
80
60
40
20
0
Itérations
DFS
BSF
Joinability


input05 - évolution de stackedNodes
stackedNodes
250
200
150
100
50
0
Itérations
DFS
BFS
Joinability

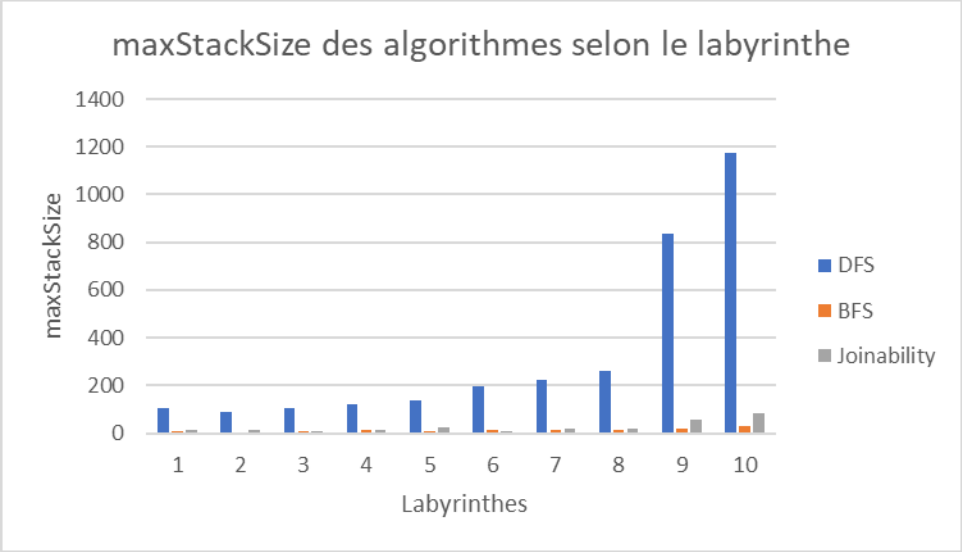

input06 - évolution de stackedNodes
250
200
150
100
50
0
stackedNodes
Itérations
DFS
BFS
Joinability


input07 - évolution de stackedNodes
300
250
200
150
100
50
0
stackedNodes
Itérations
DFS
BFS
Joinability


input08 - évolution de stackedNodes
900
800
700
600
500
400
300
200
100
0
stackedNodes
Itérations
DFS
BFS
Joinability

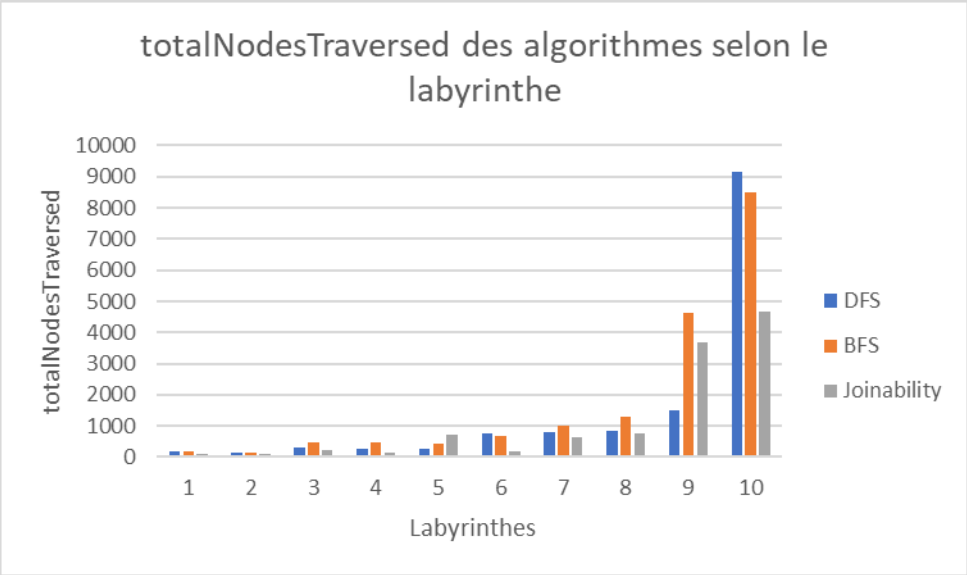

input09 - évolution de stackedNodes
1400
1200
1000
800
600
400
200
0
stackedNodes
Itérations
DFS
BFS
Joinability

Discussion



Le graphique ci-dessus est très représentatif de l'allocation de mémoire des différents algorithmes. **Nous observons qu'en moyenne, DFS alloue le plus de mémoire**. En effet, la récursivité des appels de fonction sera aussi profonde que la plus longue impasse rencontrée, où il faudra revenir bien en arrière.

À l'inverse, BFS alloue le moins de mémoire. Puisque cette approche évalue simultanément toutes les directions dans un labyrinthe, la file se vide régulièrement alors qu'une tuile est complètement visitée. Ainsi, la grandeur de la file plafonne au nombre d'embranchements de la recherche, qui est moindre. Cet avantage de BFS par-dessus DFS n'est toutefois pas sans compromis.



Le graphique ci-dessus illustre qu'**en moyenne, BFS visite le plus de tuiles**. Cet algorithme évalue simultanément toutes les tuiles qui sont à la même distance de la source. Cela explique pourquoi il faut visiter beaucoup de tuiles, même pour aller peu profond dans un labyrinthe. Par contre, cette approche garantit le chemin le plus court.

DFS visite moins de tuile que BFS, en explorant dès le départ des directions jusqu'au bout. Cet algorithme risque ainsi de trouver une sortie plus rapidement, mais il ne garantit pas le chemin le plus court.

Néanmoins, selon nos observations**, l'algorithme de rejoignabilité est en moyenne le plus efficace.**

Nous concluons que le choix de l'algorithme dépend du contexte d'utilisation, s'il y a des contraintes de mémoire, de précision ou de performance. Cependant, l'algorithme de rejoignabilité semble offrir un très bon compromis entre les avantages et les inconvénients de BFS et DFS.