Constraint Satisfaction Problems

Orfeo Tërkuçi

Problem 1: Backtracking

How many calls does your algorithm need (on average) for n=10? Is there a lot of variation in the number of calls when you try this multiple times?

Run	Calls	Run	Calls	Run	Calls	Run	Calls	Run	Calls
0	128	**20**	16	**40**	60	**60**	24	**80**	77
1	59	**21**	201	**41**	109	**61**	85	**81**	163
2	17	**22**	26	**42**	29	**62**	17	**82**	19
3	74	**23**	74	**43**	144	**63**	168	**83**	93
4	33	**24**	12	**44**	45	**64**	22	**84**	80
5	16	**25**	27	**45**	238	**65**	11	**85**	32
6	183	**26**	306	**46**	150	**66**	48	**86**	62
7	42	**27**	47	**47**	439	**67**	47	**87**	196
8	97	**28**	23	**48**	168	**68**	124	**88**	302
9	23	**29**	656	**49**	244	**69**	16	**89**	46
10	13	**30**	51	**50**	207	**70**	75	**90**	15
11	11	**31**	298	**51**	231	**71**	97	**91**	11
12	135	**32**	117	**52**	11	**72**	31	**92**	34
13	22	**33**	49	**53**	664	**73**	12	**93**	20
14	37	**34**	137	**54**	228	**74**	18	**94**	267
15	52	**35**	45	**55**	39	**75**	62	**95**	296
16	213	**36**	32	**56**	11	**76**	41	**96**	218
17	40	**37**	22	**57**	57	**77**	151	**97**	301
18	104	**38**	69	**58**	23	**78**	124	**98**	29
19	35	**39**	348	**59**	29	**79**	15	**99**	15

This is a report of one hundred runs of the backtracking algorithm.

The mean is 104.8.

The standard deviation is 121.7.

For further insight, the minimum and maximum of the calls number are 11 and 664.

So, on average the algorithm needs 104.8 calls for n = 10, and there is a lot of variation in the number of calls, since the standard deviation is 121.7, but also encouraged by the staggering difference between the minimum number of calls (11) and the maximum (664).

Problem 2: Forward Checking

The following statistics are gathered from 10 runs with n = 50.

Configuration	Mean	Standard Deviation
No MRV or LCV	4724.4	6315.95
Only LCV	2643.1	3172.38
Only MRV	67.0	0.0
Both (MRV and LCV)	73.0	0.0

The functions selectVariable and orderDomain make the algorithm faster and more stable, because they enforce an order to selecting the next value to be assigned (the selectVariable function) and which is the best domain to choose next (the orderDomain function). So now, we no longer leave things to chance. Every run will be the same for the same starting variables list and domains.

Problem 3: AC3

The following statistics are gathered from 10 runs with n = 50.

Configuration	Mean	Standard Deviation
No MRV or LCV	986.4	2264
Only LCV	76.6	24.44
Only MRV	60	0.0
Both (MRV and LCV)	55	0.0

AC3 is more efficient than normal Forward Checking because it enforces arc consistency on every arc in the graph instead of only neighbouring arcs to the assigned variable. Again, enforcing an order to selecting the next variable to be assigned and the next domain to be chosen ensures an efficient route without (much) backtracking.

Problem 4: Sudoku

Can you solve the hard puzzle with any of your algorithms?
The hard sudoku puzzle was solvable with AC3 and Forward Checking
AC3: 3382 calls.
Forward Checking: 5855 calls