Applying Decomposition Methods to Crossword Puzzle Problems

Yaling Zheng

Constraint Systems Laboratory
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Email: yzheng@cse.unl.edu

Abstract. In this paper, we investigate the performance of several existing structural decomposition methods on fully interlocked Crossword Puzzle Problems (CPPs) and draw directions for future research.

1 Introduction

A Constraint Satisfaction Problem (CSP) is a problem consisting a finite set of variables, each of which is associated with a finite domain, and a set of constraints over a subset of constraints. Each constraint specifies allowable tuples for a subset of all the variables. The task is to find an assignment for every variable so that all constraints are simultaneously satisfied. Although CSPs are in general in **NP**-complete, it is possible and desirable to identify special properties of a problem class that can be efficiently solved.

Structural decomposition methods have been proposed for identifying tractable Constraint Satisfaction Problems (CSPs) [1–5]. The basic principle is to decompose a CSP into sub-problems that are organized in a tree structure. The subproblems are then solved independently, then the original CSP is solved in a backtrack-free manner after the tree structure is made arc-consistent, as described by Dechter and Pearl [1]. The size of the biggest sub-problem is called the width of the decomposition, and is the criterion for judging the tractability of the problem.

To the best of our knowledge, no work in the literature evaluates the performance of structural decomposition techniques on real-world problems, only Harvey and Ghose [6] and our previous work [5] conduct evaluations on randomly generated CSPs. In this paper, we study the application of structural decomposition techniques on fully interlocked Crossword Puzzle Problems. Our goal is two-fold: (1) Evaluate the performance of structural decomposition techniques on a real-world problem, and (2) Initiate the investigation of techniques for solving CPPs.

CPPs, especially those in newspapers, which are usually fully interlocked, are interesting and challenging problems for people. They receive daily wide attention, and many newspapers, such as U.S.A. Today and the New York Times, regularly publish CPPs. Some people make CPPs for newspapers, while some readers solve those CPPs.

We state the definitions as proposed by [7]. A crossword puzzle is defined upon an $m \times n$ grid where most, if not all, of the cells are to be filled in with characters which comprises words along horizontal and vertical axes. An *open cell* is a blank box destined to contain a character in the final solution of the entire puzzle. A *closed cell* appears as a solid box, does not contain any character and is not actually part of the puzzle but indicates an internal border. Contiguous open cells read from left to right or from top to bottom constitute words, and these contiguous cells are referred to as *word slots*. The *degree of interlocking* in a puzzle is the percentage of shared cells. A *shared cell* is an open cell belonging to both a vertical and a horizontal word slot. The cell in which two word slots intersect is called an *orthogonal intercept*. If all open cells in a puzzle are shared, the puzzle is *completely interlocked*. Given a word list and a grid configuration on a crossword compiler, man or machine, should find one or more solutions. A solution in this context is a filling of the grid with words all belonging to the specified word list.

A newspaper CPP is usually a fully interlocked CPP. In [8], Meehan and Gary compared two approaches modeling CPPs as CSPs: letter-by-letter and word-by-word. Also, they discussed the application of arc-consistency to the resulting problem as a pre-processing step before search. In [7], CambonJensen showed several grid-walk heuristics for finding a solution for the CPPs. However, the identification of tractable fully interlocked CPPs has not yet been studied. We are motivated to investigate whether decomposition methods is useful for solving CPPs.

In [5], we introduced four new structural decomposition methods: HINGE$^+$, CUT, TRAVERSE, and CaT. The relationships between the decomposition methods are shown in Figure 1. The solid directed-edge from a decomposition D_1 to another one D_2 indicates that D_2 strongly generalizes D_1. The dotted directed edge from D_1 to D_2 indicates D_2 generalizes D_1. Note that the picture is incomplete in the sense that not all relationships are shown. We tested

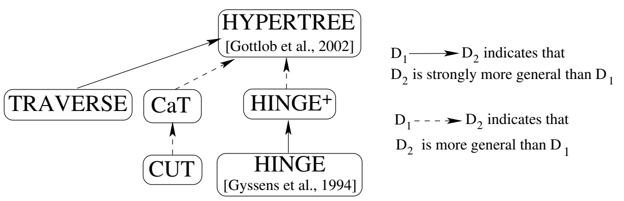

Fig. 1. Illustrating the relationships between the various studied techniques.

these methods on randomly generated CSPs. CaT, which is a hybrid of CUT and TRAVERSE, showed the best trade-off between the width of the computed constraint tree and the time for computing the decomposition In this paper, we apply these decomposition methods to fully interlocked CPPs. The experiment is done on 51 fully interlocked CPPs instances obtained from the Crossword Puzzle Grid Library [9].

This paper is organized as follows. Section 2 shows the decomposition result of CPPs. Section 3 discusses directions for future research.

2 Applying structural decomposition methods to CPPs

We apply the following decomposition methods: HINGE [3], HINGE$^+$, CUT, TRAVERSE, and CaT [5] to the associated constraint hypergraphs of 51 instances of fully interlocked CPPs obtained from Crossword Puzzle Grid Library [9]. A cut k in a decomposition technique is the number of hyperedges of the constraint graph of a connected CSP that, when removed, separate the graph into two or more components. The techniques HINGE$^+$, CUT, and CaT take k as a input parameter, usually starting with k=1, and increasing its value until the CSP can be decomposed. TRAVERSE operates by sweeping through the constraint graph, and it output depends on the starting hyperedge. Tables 1 and 2 show the CPU time for computing the decompositions and the width of those decomposition methods on the 51 instances. The x axis in these figures represents the identifier of an instance. $H+1$ denotes HINGE$^+$ with k=1, $H+2$ denotes HINGE$^+$ with k=2, CUT1 denotes CUT with k=1, CUT2 denotes CUT with k=2, CaT1 denotes CaT with k=1, and CaT2 denotes CaT with k=2. We start TRAVERSE on each hyperedge in the problem and display the minimum value of the width as TMIN, its maximum value as TMAX, and its average value as TAVER.

Let D-width be the width of the join tree computed by decomposition method D. Note that HINGE$^+$-width is always smaller than HINGE-width because HINGE$^+$-width first finds cuts with size 1, then find cuts with size 2, through cuts with size k.

From Table 2, we notice that only for 1 out of 51 instances, the CaT-width is larger than its HINGE-width. This is instance #27 (a 17×17 fully interlocked CPP instance ranked as difficult), as shown in Figure 2. For instance #27, the decomposition with HINGE results into 5 large nodes shown in Figure 3. The largest node connects with the other 4 nodes via 4 different cuts in a star-shape fashion. The value of width HINGE-width for this instance is 24. Figure 4 shows the join tree computed by CaT for instance #27 is 27, which is larger than the one computed by HINGE. This situation arose in relatively rare cases in our experiments on random CSP and crossword puzzle problems.

Moreover, we notice that the width of the join tree computed by TRAVERSE (TMIN) is smaller than the one computed by CaT with $k = 2$ in 35 out of 51 cases. This results is different from the one on randomly generated CSPs, where CaT gives the best tradeoff between the width of the computed join tree and the computation cost.

3 Future work

For newspaper CPPs, the decomposition result *seems* not particularly promising: if the candidate words for every word slot is about 1000 (when the purpose

Table 1. *CPU time of the decompositions of 51 newspaper CPPs by HINGE, HINGE$^+$, CUT, TRAVERSE, and CaT.*

Instance #	ID	#E	HINGE	HINGE$^+$		CUT		TRAVERSE			CaT	
				$k=1$	$k=2$	$k=1$	$k=2$	min	max	aver	$k=1$	$k=2$
1	c13-b1	68	90	680	1370	540	3500	0	10	3.09	510	3560
2	13-m1	66	160	270	12760	270	12520	0	0	3.03	280	12620
3	c13-m2	70	130	910	2940	690	12520	0	0	3.29	680	12210
4	c13-d1	60	130	510	2810	520	2780	10	10	3.33	530	2800
5	c13-d2	60	130	520	2800	490	2780	0	0	3.33	520	2800
6	c15-b1	90	290	1730	8960	1350	31850	10	10	5.44	1360	32030
7	c15-b2	90	270	950	1140	910	1540	20	0	5.33	930	1540
8	c15-b3	90	240	970	1060	920	1550	0	0	5.22	930	1550
9	c15-m1	82	220	1570	4280	1230	9730	0	10	5.12	1220	9760
10	c15-m2	82	290	1010	20200	1020	20110	10	0	5.49	1020	20250
11	c15-m3	80	340	550	44620	550	44810	0	0	5.38	530	45000
12	c15-m4	80	300	1310	10840	1300	10870	10	10	5.38	1320	10920
13	c15-d1	78	320	540	46440	530	39490	10	0	5.26	540	39630
14	c15-d2	76	280	510	42490	520	42340	0	10	5.53	530	42700
15	c15-d3	76	290	520	20110	540	20100	0	10	5.79	550	20250
16	c15-d4	74	290	500	34500	500	34440	10	10	5.40	510	34890
17	c17-b1	120	530	3370	15120	2700	26160	10	0	8.67	2720	26210
18	c17-b2	110	690	2950	49630	2920	47990	10	10	7.91	2930	48490
19	c17-b3	108	590	2250	31440	2220	17600	10	10	8.43	2200	17690
20	c17-m1	100	680	980	95140	970	94740	0	10	8.10	990	95180
21	c17-m2	102	740	1000	63310	990	63250	10	0	7.8	1000	63140
22	c17-m3	100	710	1030	102820	1010	102890	10	10	8.20	1030	103540
23	c17-m4	106	740	2720	62710	2690	32410	10	10	8.21	2690	32250
24	c17-d1	100	720	1040	63800	1050	64010	10	10	8.80	1060	64000
25	c17-d2	96	630	1000	54520	1000	54340	10	10	8.65	1000	54550
26	c17-d3	90	540	1020	67920	1020	68380	10	10	9.22	1020	67720
27	c17-d4	76	230	1610	1980	1450	6570	10	10	7.90	1460	6550
28	c19-b1	136	1590	2090	355080	2050	279960	10	10	12.43	2070	280180
29	c19-b2	134	1110	7450	147520	5070	228990	10	20	12.84	5140	228370
30	c19-m1	128	1420	2070	346770	2030	270600	10	10	12.97	2060	270530
31	c19-m2	118	1140	1890	215510	1880	216590	20	20	13.64	1880	216150
32	c19-d1	122	1310	1990	119000	1970	119050	10	20	13.52	1970	119620
33	c19-d2	116	1250	1760	255840	1720	256480	10	10	13.19	1760	256120
34	c19-d3	114	1070	1830	103330	1830	103260	10	10	13.60	1850	103130
35	c19-d4	117	1240	1900	243980	1910	243260	10	20	14.10	1930	242800
36	c21-b1	140	2090	3280	227780	3280	228440	20	20	19.64	3310	227650
37	c21-b2	138	2180	3280	223530	3260	222890	20	20	19.78	3270	223480
38	c21-m1	148	2470	3250	370160	3270	354340	20	20	19.12	3280	354840
39	c21-m2	140	2180	3020	259190	3040	259090	20	20	18.14	3040	259130
40	c21-d1	140	2090	3290	229310	3290	228350	20	20	19.50	3290	228250
41	c21-d2	138	2160	3250	223780	3290	224020	20	20	19.93	3280	223970
42	c21-d3	136	2020	3150	448190	3230	447650	20	20	19.63	3260	447390
43	c21-d4	126	1910	2760	173280	2780	173030	20	20	20.16	2770	172710
44	c23-b1	186	1850	2220	596300	2210	454790	10	10	10.11	2220	455000
45	c23-b2	194	1900	6470	541550	6420	165070	10	10	11.70	6430	165270
46	c23-m1	202	1450	5330	131090	5330	61870	10	10	11.98	5340	61780
47	c23-m2	180	1760	2310	674830	2310	498670	10	10	11.11	2310	499020
48	c23-d1	162	1430	2280	183060	2280	183110	10	10	13.21	2290	183430
49	c23-d2	184	1940	2580	408940	2590	408600	10	10	12.72	2590	408830
50	c23-d3	162	1440	2240	180880	2240	181120	10	20	13.15	2260	181230
51	c23-d4	160	1460	2070	334010	2060	334050	10	10	11.63	2070	334420

is to make a crossword or we have no clues about the word), in a decomposition of width 8, the biggest sub-problem has a size of 10^{24}. Thus, the required space is huge. However, since a sub-problem of fully interlocked may be highly-connected, there may actually be few solutions to the subproblem, and it may be the realistic to find them using backtrack search with a full look-ahead technique. Therefore, for a fully interlocked CPP, if we are able to use a structural decomposition technique to identify subproblems, then find all the solutions of these subproblems, and store them without significant overhead because there number is not large, then this processing may allow us to greatly reduce the cost of solving the CPP. This would make decomposition methods practically useful.

On the other hand, we propose to investigate the use of local search for solving fully interlocked CPPs given a dictionary and a grid. Local search starts from an initial assignment, which is randomly generated, continuously improving until it

Table 2. *Width of the decompositions of 51 newspaper CPPs by HINGE, HINGE$^+$, CUT, TRAVERSE, and CaT.*

Instance #	ID	#E	HINGE	HINGE$^+$		CUT		TRAVERSE			CaT	
				$k=1$	$k=2$	$k=1$	$k=2$	min	max	aver	$k=1$	$k=2$
1	c13-b1	68	36	36	19	52	27	16	19	17.53	13	8
2	c13-m1	66	66	66	22	66	36	21	28	22.12	21	16
3	c13-m2	70	46	46	18	58	56	12	23	16.46	21	22
4	c13-d1	60	44	44	23	44	23	15	27	18.70	15	9
5	c13-d2	60	44	44	23	44	23	15	27	18.70	15	9
6	c15-b1	90	58	58	26	74	72	12	30	18.53	19	20
7	c15-b2	90	20	20	10	32	14	14	28	15.87	10	10
8	c15-b3	90	20	20	10	32	14	14	28	16.56	10	10
9	c15-m1	82	50	50	50	66	66	19	36	24.98	26	26
10	c15-m2	82	75	75	31	75	31	17	35	25.52	28	18
11	c15-m3	80	80	80	66	80	66	17	32	23.15	29	24
12	c15-m4	80	66	66	66	66	66	22	45	28.83	21	21
13	c15-d1	78	78	78	50	78	62	18	36	22.97	23	19
14	c15-d2	76	76	76	64	76	64	19	52	29.05	19	19
15	c15-d3	76	76	76	76	76	76	21	46	26.36	22	22
16	c15-d4	74	74	74	58	74	58	15	50	25.81	18	15
17	c17-b1	120	64	64	34	90	90	22	42	27.35	26	26
18	c17-b2	110	98	100	32	100	51	18	32	23.24	23	13
19	c17-b3	108	78	78	60	78	78	17	40	23.00	19	19
20	c17-m1	100	100	100	80	100	80	17	37	26.54	17	25
21	c17-m2	102	102	102	38	102	38	14	44	23.27	14	14
22	c17-m3	100	100	100	82	100	82	17	40	24.06	17	21
23	c17-m4	106	90	90	72	90	90	16	60	25.15	20	20
24	c17-d1	100	100	100	47	100	47	17	43	26.08	17	17
25	c17-d2	96	96	96	49	96	49	19	44	30.71	23	18
26	c17-d3	90	90	90	55	90	55	23	40	29.68	31	25
27	c17-d4	76	24	24	24	50	50	31	52	41.26	27	27
28	c19-b1	136	136	136	98	136	110	19	37	25.41	22	24
29	c19-b2	134	104	104	78	118	116	20	38	26.25	26	27
30	c19-m1	128	128	128	94	128	106	18	52	28.25	26	21
31	c19-m2	118	118	118	94	118	94	26	50	33.15	26	40
32	c19-d1	122	122	122	122	122	122	20	54	33.80	24	24
33	c19-d2	116	116	116	108	116	108	18	54	30.69	20	24
34	c19-d3	114	114	114	114	114	114	22	42	29.60	34	34
35	c19-d4	117	117	117	101	117	101	21	55	32.79	32	37
36	c21-b1	140	140	140	140	140	140	21	66	39.34	44	44
37	c21-b2	138	138	138	138	138	138	27	61	39.72	32	32
38	c21-m1	148	148	148	56	148	84	19	33	28.04	31	31
39	c21-m2	140	140	140	65	140	65	25	54	34.3	25	22
40	c21-d1	140	140	140	140	140	140	21	66	39.34	44	44
41	c21-d2	138	138	138	138	138	138	27	61	39.72	32	32
42	c21-d3	136	136	136	110	136	110	33	63	42.62	34	34
43	c21-d4	126	126	126	126	126	126	24	50	32.87	26	26
44	c23-b1	186	186	186	140	186	160	21	44	27.35	21	25
45	c23-b2	194	176	176	144	176	176	20	42	26.92	23	23
46	c23-m1	202	136	136	116	136	136	23	46	31.93	23	23
47	c23-m2	180	180	180	144	180	162	27	62	36.87	31	41
48	c23-d1	162	162	162	162	162	162	33	78	47.93	53	53
49	c23-d2	184	184	184	128	184	128	25	68	39.18	30	33
50	c23-d3	162	162	162	162	162	162	29	78	44.35	48	48
51	c23-d4	160	160	160	128	160	128	28	82	43.78	28	29

finds a legal assignment. The problem lies in finding a good heuristic suitable for improving the assignment from one step to the next. This method, as far as we know, has not yet been studied to applying for fully interlocked CPPs. Therefore, our future work includes:

1. Identifying more structural configurations of constraint graphs where some decomposition techniques yield better results than others although in general the opposite holds, and building hybrid decompositions techniques that exploit this information;
2. Tailoring existing decomposition methods for fully interlocked CPPs, so that every sub-problem, after backtrack search, has few solutions; and
3. Finding a good heuristic to applying local search for fully interlocked CPPs.

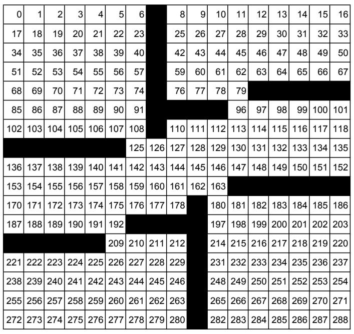

Fig. 2. Instance number #27.

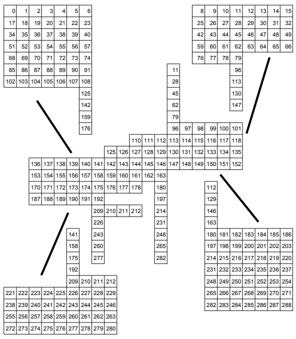

Fig. 3. Applying HINGE to c17-d4.

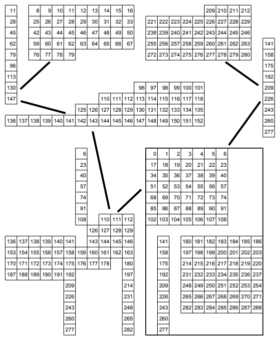

Fig. 4. Applying CaT to c17-d4.

Acknowledgments: This work is supported by CAREER Award #0133568 from the National Science Foundation. The experiments were conducted utilizing the Research Computing Facility of the University of Nebraska-Lincoln.

References

1. Dechter, R., Pearl, J.: Tree Clustering for Constraint Networks. Artificial Intelligence **38** (1989) 353–366
2. Gyssens, M., Jeavons, P.G., Cohen, D.A.: Decomposing Constraint Satisfaction Problems Using Database Techniques. Artificial Intelligence **66** (1994) 57–89
3. Jeavons, P.G., Cohen, D.A., Gyssens, M.: A Structural Decomposition for Hypergraphs. Contemporary Mathematics **178** (1994) 161–177
4. Gottlob, G., Leone, N., Scarcello, F.: A Comparison of Structural CSP Decomposition Methods. Artificial Intelligence **124** (2000) 243–282
5. Zheng, Y., Choueiry, B.Y.: New Structural Decomposition Techniques for Constraint Satisfaction Problems. In et al., B.F., ed.: Recent Advances in Constraints. Volume 3419 of Lecture Notes in Artificial Intelligence. Springer (2005) 113–127
6. Harvey, P., Ghose, A.: Reducing Redundancy in the Hypertree Decomposition Scheme. In: The 15^{th} IEEE International Conference on Tools with Artificial Intelligence (ICTAI 03). (2003) 474–481
7. CambonJensen, S.: Design and Implementation of Crossword Compilation Using Sequential Approaches Programs. Master's thesis, IMADA Odense University (1997)
8. Meehan, G., Gary, P.: Constructing Crossword Grids: Use of Heuristics vs Constraints. In: Proceedings of Expert Systems 97: Research and Development in Expert Systems XIV, SGES Publications. (1997) 159–174
9. CPPLibrary: Crossword Puzzle Grid Library. (http://puzzles.about.com/library)