Preliminary Results on Constraint Programming and Branch & Bound Algorithms for the Cyclic Bandwidth Sum Problem

Valentina Narvaez-Teran[1], Eduardo Rodriguez-Tello[1], Frédéric Lardeux[2], and Gabriel Ramírez-Torres[1]

[1] Cinvestav – Tamaulipas.
Km. 5.5 Carretera Victoria-Soto La Marina, 87130 Victoria Tamps., Mexico
{maria.narvaez, ertello, grtorres}@cinvestav.mx
[2] Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France
frederic.lardeux@univ-angers.fr

Abstract. The cyclic bandwidth sum problem (CBSP) consists in embedding a host graph into a cycle graph while minimizing the sum of cyclic distances between guest adjacent vertices embedded in the host. While the problem has been addressed by heuristic and metaheuristic methods, to the best of our knowledge, this is the first effort to apply exact methods. This work presents preliminary results on the use of constraint programming (CP) and a branch & bound (B&B) algorithm to solve the cyclic bandwidth sum problem in small graphs from commonly employed topologies.
We created a CP model of the CBSP and devised two further refined versions by adding new constraints based in problem-specific knowledge. For our proposed B&B algorithm, we designed a custom criterion for search priority employing estimations of potential cost. The results provided an assessment of the pros and cons of both methodologies, with the CP approach offering a more reliable alternative in terms of solved instances, execution time and implementation effort.

Keywords: Cyclic bandwidth sum problem, Exact solution methods, Constraint programming, Branch & bound

1 Introduction

The cyclic bandwidth sum problem (CBSP) is a graph embedding problem (GEP) [2] formally defined as follows. Let $G = (V, E)$ be a simple finite undirected graph (the guest) of order n, and C_n a cycle graph (the host) with vertex set $|V_H| = n$ and edge set E_H. Given an injection $\varphi : V \rightarrow V_H$, the cyclic bandwidth sum (CBS) is defined as:

$$\mathrm{CBS}(G, \varphi) = \sum_{(u,v)\in E} |\varphi(u) - \varphi(v)|_n \ , \tag{1}$$

where $|x|_n = \min\{\, |x|,\ n - |x| \,\}$ (with $1 \leq |x| \leq n-1$) is the *cyclic distance*, and the vertex in V_H associated to vertex $u \in V$ is denoted by the label $\varphi(u)$.

The CBSP consists in finding the optimal embedding φ^*, such that $\mathrm{CBS}(G, \varphi^*)$ is minimum, i.e., $\varphi^* = \arg\min_{\varphi \in \Phi}\{\mathrm{CBS}(G, \varphi)\}$ with Φ denoting the set of all possible embeddings.

The CBSP is an NP-Hard GEP [2] that arises in the simulation of network topologies for parallel computer systems, scheduling in broadcasting based networks, and compressed sensing in sensor networks [5, 6, 8]. It has been tackled with an ad hoc constructive heuristic [3], and metaheuristic algorithms [12, 14]. However, to the best of our knowledge, there is no exact methods reported to solve it.

In this paper we explored the use of constraint programming (CP) [11] and branch & bound (B&B) [9] for solving small instances of the CBSP. We created a CP model and incrementally refined it by adding more problem related information. Then, we compared it with our B&B algorithm, which was also designed with the CBSP in mind. These methods give us a preliminary assessment of the use of exact approaches for our problem and their potential for their improvement.

The rest of this work is organized as follows. Section 2 presents an initial CP model and two refinements. Then, our B&B algorithm is described in Section 3. A performance comparison for these methods is presented in Section 4. Finally, Section 5 summarizes our findings and future work.

2 Constraint programming modeling

CP is a useful paradigm for solving satisfiability and optimization problems by employing a declarative approach, where problems are described by models stating their characteristics. CP models are processed by solver software using efficient filtering algorithms for search space exploration, while the model serves as a guide to discard regions and to recognize optimal solutions. Models use three main types of components to describe problems: variables, domains and constraints. The variables represent the solution to be created by exploring the specified domains, and the constraints are conditions the variables must met, including the problem's objective. A problem is solved through its CP model by producing valid solutions, in the form of assignations of domain values to the variables, such that the constraints are not broken [13].

CP can be particularly powerful for discrete combinatorial problems, because of the finite domain character of their variables [1]. The constraint propagation is a key distinctive aspect in CP. It reduces the search space recursively, by discarding constraint breaking values from the domains of the variables and using information about those values to reduce the domain of other variables involved in the same constraint.

Once a good model is available, the CP approach is accessible and relatively easy to implement, thanks to frameworks of specialized software, providing modeling languages, interpreters, compilers, and solvers to create and process the models. Solvers implement advanced search algorithms based on trees, backtracking, and techniques from various areas, such as mathematical programming,

operational research and artificial intelligence, to efficiently explore within the bounds defined by a CP model. The CP models in this work were created using Minizinc [10], an standardized modeling language that acts as an intermediary between the user and solvers.

2.1 An initial CBSP model

Often, the same constraint can be expressed in different ways, some of which may result more efficient. Therefore, it is key to effectively translate the features of the problem into the CP paradigm. Performance can also benefit from a certain redundancy in the constraints.

Data representation. CBSP instances consist of finite simple undirected graphs. The format representation contains the number of vertices n, the number of edges e, a 2-D array $E(1..e, 1..2)$ listing the edges, where $E(i, 1)$ and $E(i, 2)$ are the endpoints of the i-th edge.

Variables. The decision variables represent the labeling, such that $g(1..n)$ is an array of the labels assigned to vertices, where $g(i)'$ is the label mapping guest vertex $i \in V$ to host vertex $g(i) \in V'$.

Constraints. CBSP embeddings are bijective mappings between guest and host vertices, so the first constraint is that to each unique guest vertex corresponds to one unique host vertex as a label, such that $\forall i, j \in [1..n] \mid i < j$ with $g(i) \neq g(j)$. This constraint would be equal to a series of pairwise conjunctions stating that no pair of vertices can have the same label, in the form $g(1) \neq g(2) \wedge g(1) \neq g(3) \wedge \cdots \wedge g(1) \neq g(n) \wedge \cdots \wedge g(n-1) \neq g(n)$. Large conjunctions can be costly to compute, so many solvers implement instead customized efficient algorithms based on inferences. These algorithms can be accessed via global constraints, which are concisely express relationships among several variables. In terms of representation and reasoning, they provide a higher level of abstraction and better structure to the problem, allowing filtering algorithms to be much more specialized and efficient. Therefore, we used the global constraint $alldifferent$ [7], stating that all elements in an array must be pairwise distinct.

$$alldifferent(g) \tag{2}$$

Objective function. The cost of a solution is the sum of cyclic distances $cbs = \sum_{i=1}^{e} distance(i)$, where $distance(i)$ is the cyclic distance associated to edge i. Each cyclic distance can have a value between 1 and $d_{max} = \lfloor n/2 \rfloor$. A cyclic distance equals the length of the shortest path between two adjacent vertices of the guest graph embedded in the host graph, expressed as $\forall i \in [1..e]\ distance(i) = \min\{n - |g(E(i,1))|, |g(E(i,2))|\}$. The goal of the CBSP is to find the lowest cost embedding, therefore the objective function for the model is to minimize the sum of cyclic distances.

$$minimize(cbs) \tag{3}$$

The first CBSP model is M_0, defined by the previously defined variables, and the conjunction of the constraints and the objective, $M_0 = (2) \wedge (3)$.

2.2 Refined CP models

Breaking cyclic symmetries. Since the host topology is cyclic, different labelings can result in isomorphic embeddings under rotation and mirror symmetries. To remove those solutions from the search space, two constraints were added, ensuring that only the lexicographicaly minor of the isomorphic embeddings is computed. They state that the fist vertex must be associated to the fist label and that the label of the second vertex must be lower than the label of the last one, thus eliminating the rotation and mirror symmetries, respectively. The first refined model, M_1, results from adding these symmetry breaking constraints to the initial model, thus $M_1 = M_0 \wedge (4) \wedge (5)$.

$$g(1) = 1 \tag{4}$$

$$g(2) < g(n) \tag{5}$$

Adding upper and lower bounds. Including cost bounds can improve the performance by discarding solutions with cost outside the bounds. The data representation was modified to include two new input variables, the CBS lower bound lb and upper bound ub. These values vary according to each graph topology[3]. In the case of graph topologies for which there are exact formulas to calculate the value of the optimum, both ub and lb got assigned that value. If this is not the case, the value of ub was calculated according to topology specific upper bound formulas, in the case where those exist, or the topology independent upper bound formula, otherwise. The value of lb was set as $e + 1$. Model M_2 results from adding the upper and lower bound constraint to model M_1, therefore $M_2 = M_1 \wedge (6)$. Notice that in the case the exact value of the optimum is known, then $ub = lb$ and the constraints still holds.

$$lb \leq cbs \leq ub \tag{6}$$

3 A Branch and Bound algorithm for the CBSP

B&B algorithms use a tree to implicitly explore a problem's search space by creating partitions of smaller subproblems. The nodes of the tree contain partially defined solutions to such subproblems. From the root of the tree, the exploration process branches promising nodes into new ones, creating partial solutions of higher order, and pruning branches that can not lead to the optimum. This narrows the search, discarding search space regions that do not contain the optimum [9]. Algorithm 1 shows our B&B. It begins by creating a solution to use its cost as initial upper bound. This solution is created by a greedy labeling algorithm based on a depth first search visit of the vertices of G, starting randomly.

The tree's root is a partially defined solution of order one, with the first label assigned to the first vertex. This solution is inserted in a priority queue to keep track of the exploration, which ends when the queue is empty. When a partial

[3] Lower and upper bounds: https://www.tamps.cinvestav.mx/ertello/cbsp.php

Algorithm 1: Branch and Bound algorithm

```
1: up ← dfs(G)
2: Q ← empty priority queue
3: Create root solution a by assigning a(1) ← 1
4: Q.push(a)
5: while Q is not empty do
6:    b ← Q.pop()
7:    i ← first unassigned node in b
8:    for j ∈ { unassigned labels in b} do
9:       b′ ← b
10:      b′(i) ← j
11:      cost(b′) → f_p(b′) + f_e(b′)
12:      if cost(b′) < f(up) then
13:         if all vertices in b′ are assigned then
14:            up ← b′
15:         else
16:            Q.push(b′)
17:         end if
18:      else
19:         Discard b′
20:      end if
21:   end for
22: end while
23: g ← up
24: return g
```

solution b is extracted from the queue, the branching process creates new nodes by assigning the unused labels to the first unlabeled vertex in b. This produces $n - o(b)$ new partially defined solutions, where n is the number of vertices and $o(b)$ is the order of b. A new solution b' is evaluated to decide if it will be further explored or discarded. Its cost $cost(b')$ is the sum of a partial CBS $f_p(b')$ given by the defined part of the solution, and a potential CBS $f_e(b')$, given by the undefined one. The partial cost is the CBS for the assigned edges, i.e., edges that have labels assigned to both endpoints. The potential CBS is a best-case estimation where all the unassigned edges have cyclic distance equal to one. A partial solution b' is discarded if the sum of partial and estimated CBS is greater than the CBS of the current upper bound solution up. If the sum is instead lower, and b' is fully defined (its order is n), then b' is better than the current upper bound solution up. Therefore b' replaces up. Otherwise, b' can not be discarded, so it enters the queue.

The priority queue sets the exploration order using a combination of partial solution's order, partial cost, and a more elaborated estimation of potential cost. Order is prioritized before potential cost to produce completed solutions as soon as possible. Partial solutions of equal order are untied by the sum of their partial CBS and estimation $f_b(b')$. The later is an heuristic estimation calculating

Table 1. Performance comparison of the original CP model M_0, the refined versions M_1 and M_2, as well as the branch & bound algorithm.

							M_0		M_1		M_2		B&B	
Graph	$\|V\|$	$\|E\|$	den.	Op^*	lb	ub	$Best$	T	$Best$	T	$Best$	T	$Best$	T
path25	25	24	0.08	24			**24**	0.38	**24**	0.50	**24**	0.41	**24**	0.01
path30	30	29	0.07	29			**29**	0.41	**29**	0.51	**29**	0.39	**29**	0.01
path40	40	39	0.05	39			**39**	0.45	**39**	0.55	**39**	0.37	**39**	0.01
cycle25	25	25	0.08	25			**25**	0.35	**25**	0.61	**25**	0.43	**25**	0.01
cycle30	30	30	0.07	30			**30**	0.31	**30**	0.69	**30**	0.41	**30**	0.01
cycle40	40	40	0.05	40			**40**	0.41	**40**	1.81	**40**	0.45	**40**	0.01
wheel25	25	48	0.16	181			181	1hr	181	1hr	**181**	0.39		
wheel30	30	58	0.13	255			255	1hr	255	1hr	**255**	0.40		
wheel40	40	78	0.10	440			440	1hr	440	1hr	**440**	0.44		
cycleP25-2	25	50	0.17	75			**75**	819.80	**75**	15.36	**75**	0.43	**75**	27.63
cycleP30-2	30	60	0.14	90			90	1hr	**90**	231.94	**90**	0.48	**90**	423.72
cycleP40-2	40	80	0.10	120			120	1hr	120	1hr	**120**	0.51		
cycleP25-3	25	75	0.25	150			150	1hr	150	1hr	**150**	0.36	150	1hr
cycleP30-3	30	90	0.21	180			180	1hr	180	1hr	**180**	0.33	180	1hr
cycleP40-3	40	120	0.15	240			240	-	240	1hr	**240**	0.3	240	1hr
c4c3	12	24	0.36		25	52	**52**	298.18	**52**	15.08	**52**	24.31	**52**	9.33
p4p3	12	17	0.26		18	36	**29**	5.94	**29**	1.14	**29**	0.90	**29**	0.13
p4p4	16	24	0.20		25	60	**44**	876.02	**44**	70.92	**44**	51.85	**44**	17.73
p5p3	15	22	0.21		23	48	**42**	267.00	**42**	31.89	**42**	29.24	**42**	9.88
p6c3	18	33	0.22		34	123	69	1hr	**69**	2148.73	**69**	2,180.70	**69**	504.16
p6p3	18	27	0.18		28	60	55	1hr	**55**	3369.98	**55**	1,560.77	**55**	426.58
p3c4	12	20	0.30		21	44	**40**	54.55	**40**	5.89	**40**	2.95	**40**	1.67
p3c5	15	25	0.24		26	55	55	1hr	**55**	936.39	**55**	577.85	**55**	287.31
p4c3	12	21	0.32		22	57	**43**	65.34	**43**	5.00	**43**	4.02	**43**	2.08
p4c4	16	28	0.23		29	76	64	1hr	**64**	2458.22	**64**	2,558.77	**64**	881.27
p5c3	15	27	0.26		28	87	**56**	1,900.63	**56**	156.42	**56**	101.53	**56**	35.63
c3k4	12	30	0.45		31	88	**72**	1,058.78	**72**	76.63	**72**	42.17	**72**	29.69
p3k4	12	26	0.39		27	80	**58**	243.11	**58**	18.12	**58**	16.18	**58**	6.19
rand10-7	10	32	0.71		33	88	**51**	19.81	**51**	1.37	**77**	2.61	**77**	1.39
rand10-9	10	41	0.91		42	113	**90**	78.99	**90**	3.54	**106**	4.88	**106**	1.70

the potential cost of assigning the most suitable available label to one of the endpoints of the first found edge that already has a labeled endpoint.

4 Experimental results

We tested 30 graphs from diverse topologies commonly employed in the CBSP literature. Experiments were ran in a computer with an Intel® Core™ i7-8750H CPU at 2.20GHz and 8GB in RAM and 3,600 seconds (1 hour) as time limit. Our CP models were created and solved using Minizinc [10], while the B&B method was coded C++. Table 1 list the results, comparing the best solution cost and the total execution time for the algorithms. It also includes the order, size and density of the graphs. Instances are considered solved only if the execution finished before the time limit was reached, having produced an optimum (marked in bold). Blank cells mean there was not any solution reported.

Model M_0 solved the smallest number of instances and it took the largest amount of computing time. Adding symmetry breaking constraints in the refined model M_1 was helpful to narrow the search, allowing it to solve five more

instances than to the initial model M_0. It also reduced the execution time for instances previously solved by model M_0. The constraints for upper and lower bounds, added in the the second refined model M_2, helped it achieve further improvements in performance. Model M_2 was successful with all the graphs solved by model M_1, plus seven more. Model M_2 was the only method able to consistently solve wheel graphs, even when compared to the B&B algorithm. The performance of the later was almost comparable to model M_2, being faster in a couple of cases. However, it was not capable of solving all the considered graphs. It did not produced any solution for the wheel graphs of order larger than $n = 15$ or the power of cycle graph *cycleP40-2*. The solutions that the B&B produced for instances *cycleP25-3*, *cycleP30-3* and *cycleP40-3* can be confirmed as optimal by comparing them with the results of model M_2, but the B&B could not demonstrate this by itself, since its execution did not finished before the maximum set time. While the B&B was, in some cases, faster than the CP models to provide an optimal result, its inability to solve several of the instances makes the CP approach with model M_2 overall more successful.

The results allowed us to evaluate the gap between the theoretical upper bound values and the optimums. For instances belonging to the Cartesian product topology, the theoretical upper bounds known were larger than the optimum by an average of 17.25%. For other instances with unknown optimal value, like the last two instances in Table 1, the gap respect to the upper bound formula for any graph was 9.34% in average.

We consider that there is still room for improving the results. CP's performance responded very positively to relatively small changes in the construction of the models that added knowledge of the problem, such as the lower and upper bounds. Therefore, further refining the models by adding more information related to the problem in the form of constraints could further help to solve a broader variety of larger instances in fewer time. It may be possible to improve the B&B performance as well, however, the CP approach offers the advantage of being easier to implement, in the sense that it does not require the micromanagement of the search exploration.

5 Conclusions

This work explored CP and a customized B&B algorithm as means to solve the CBSP. To the best of our knowledge, this work is the first proposal and performance comparison of exact methods for the CBSP.

The CP and B&B approaches were tested on a set of topologically diverse standard instances of order $n \leq 40$, with one hour as the execution time limit. When comparing the results, the CP approach was proven to be more reliable than the B&B algorithm, as shown by model M_2 solving the largest number of instances across all the included topologies. In total, the CP models and the B&B algorithm produced optimal solutions for 30 problem instances. These optimal cost values allowed us to evaluate the gap respect to the theoretical

upper bounds for the Cartesian product topology [4], finding that, the known upper bound values were in average 17.25% larger than the optimal cost.

Considering the results obtained, it is worth exploring the possibility of further refinements of the CP models, specially by adding new constraints using tighter estimations for the cost bounds. It is also desirable to test more graph topologies with unknown upper bound values.

References

1. Bockmayr, A., Hooker, J.N.: Constraint programming. In: Discrete Optimization, *Handbooks in Operations Research and Management Science*, vol. 12, pp. 559–600. Elsevier (2005). DOI 10.1016/S0927-0507(05)12010-6
2. Chung, F.R.K.: Labelings of graphs. In: L.W. Beineke, R.J. Wilson (eds.) Selected Topics in Graph Theory, vol. 3, chap. 7, pp. 151–168. Academic Press (1988)
3. Hamon, R., Borgnat, P., Flandrin, P., Robardet, C.: Relabelling vertices according to the network structure by minimizing the cyclic bandwidth sum. Journal of Complex Networks **4**(4), 534–560 (2016). DOI 10.1093/comnet/cnw006
4. Jianxiu, H.: Cyclic bandwidth sum of graphs. Applied Mathematics - A Journal of Chinese Universities **16**(2), 115–121 (2001). DOI 10.1007/s11766-001-0016-0
5. Li, Y., Liang, Y.: Compressed sensing in multi-hop large-scale wireless sensor networks based on routing topology tomography. IEEE Access (2018). DOI 10.1109/ACCESS.2018.2834550
6. Liberatore, V.: Multicast scheduling for list requests. In: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 2, pp. 1129–1137. IEEE (2002). DOI 10.1109/INFCOM.2002.1019361
7. Mehlhorn, K., Thiel, S.: Faster algorithms for bound-consistency of the sortedness and the alldifferent constraint. In: R. Dechter (ed.) Principles and Practice of Constraint Programming – CP 2000, pp. 306–319. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
8. Monien, B., Sudborough, I.H.: Embedding one interconnection network in another, vol. 7, pp. 257–282. Springer (1990). DOI 10.1007/978-3-7091-9076-0_13
9. Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization **19**, 79 – 102 (2016). DOI 10.1016/j.disopt.2016.01.005
10. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard cp modelling language. In: C. Bessière (ed.) Principles and Practice of Constraint Programming – CP 2007, pp. 529–543. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
11. Pesant, G., Gendreau, M.: A constraint programming framework for local search methods. Journal of Heuristics **5**(3), 255–279 (1999). DOI 10.1023/A:1009694016861
12. Rodriguez-Tello, E., Narvaez-Teran, V., Lardeux, F.: Dynamic multi-armed bandit algorithm for the cyclic bandwidth sum problem. IEEE Access **7**, 40,258–40,270 (2019). DOI 10.1109/ACCESS.2019.2906840
13. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming, 1st edn. Elsevier Science (2006)
14. Satsangi, D., Srivastava, K., Gursaran: General variable neighbourhood search for cyclic bandwidth sum minimization problem. In: Proceedings of the Students Conference on Engineering and Systems, pp. 1–6. IEEE Press (2012). DOI 10.1109/SCES.2012.6199079