Bipartite Encoding: A New Binary Encoding for Solving Non-Binary CSPs

Ruiwei Wang & Roland Yap
National University of Singapore
{ruiwei,ryap}@comp.nus.edu.sg

National University of Singapore
School of Computing

Main Results

1. We introduce a new binary encoding called **Bipartite Encoding** (BE);
2. An algorithm and heuristic are used to generate *binary* BE instances from non-binary CSPs;
3. A special Arc Consistency (AC) propagator on BE instances;
4. Experiments show that BE with AC can outperform state-of-the-art generalized AC (GAC) propagators and binary encodings.

Why Binary CSPs with BE?

- Binary & Non-Binary CSPs are NP-complete. Is it better to solve a Non-Binary CSP in original form or by encoding into a Binary CSP?
- BE encoding shows binary encoding of non-binary CSP can be superior: higher consistency & faster propagation

Bipartite Encoding (BE)

The main idea is that every constraint is partitioned into 2 sub-tables and represented as a conjunction between 2 sub-tables, and then

1. sub-tables are encoded as factor variables, where the values of factor variables are mapped to tuples in the sub-tables by using binary constraints called mapping constraints, and
2. conjunctions are encoded as bipartite constraints.

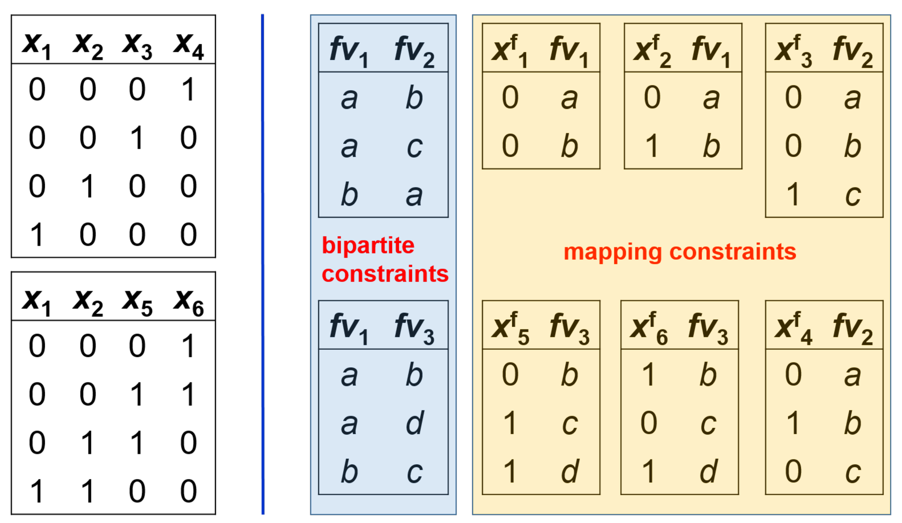


Figure 1: Bipartite encoding (BE) example: The **left** and **right** part is the **original non-binary CSP** and **BE instance** respectively

Example 1. *Figure 1 shows a simple example to illustrate BE encoding. In the example, we encode 2 non-binary constraints into 2 bipartite constraints between the factor variables $\{fv_1, fv_2, fv_3\}$, and each factor variable is mapped to sub-tables by 2 mapping constraints. The factor variable fv_1 is shared, thus, some tuples in the constraint relations can be removed.*

Consistency level of AC on BE instances

Proposition 1. *The image $\mathcal{P}'$ of a binary encoding $BE(\mathcal{P})$ is GAC if $BE(\mathcal{P})$ is AC, where $\mathcal{P}'$ is the same as the original non-binary CSP P except variable domains and constraint relations may be reduced.*

Generation of BE instances

The method used for constructing BE instances is as follows:
(i) The scope of each constraint is split into 2 variable subsets.
(ii) Generating factor variables for the sub-tables over variable subsets.
(iii) Constructing BE instance based on the generated factor variables and partitions of constraint scopes.

Constraint scope partition

Two kinds of methods used to partition a constraint scope $scp(c_1)$:

1. select an edge $\{c_1, c_2\}$ in the dual graph such that the size $|S|$ is greater than 1 where $S = scp(c_1) \cap scp(c_2)$, and partition $scp(c_1)$ and $scp(c_2)$ into subsets S, $scp(c_1) \setminus S$ and $scp(c_1) \setminus S$;
2. select a variable $x \in scp(c_1)$ and partition $scp(c_1)$ into subsets $\{x\}$, $scp(c_1) \setminus \{x\}$ (this is a basic partition).

Every constraint scope can only be partitioned once, meanwhile, there are some partitions such that the BE instances based on the partitions are not compact. Therefore we only select some available edges to do partition, and the rest constraints use the basic partitions. We select the edges with a larger size as soon as possible.

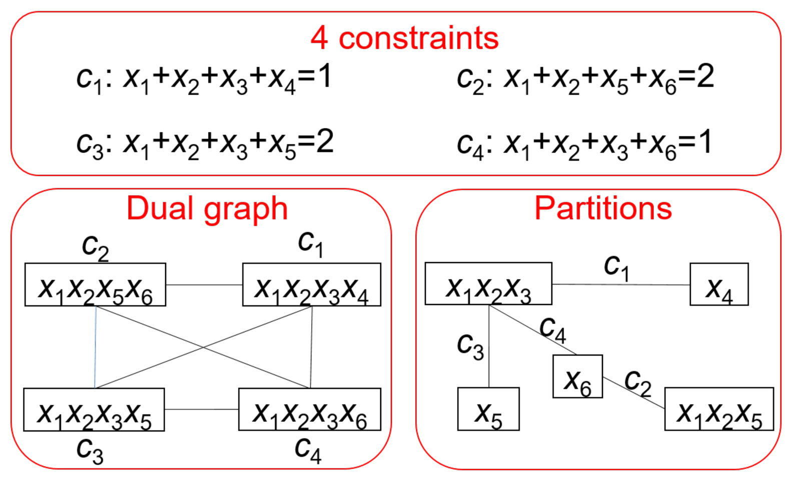


Figure 2: Constraint partition example: The **top** , **bottom left** and **bottom right** part is the **non-binary CSP**, **dual graph** and **partitions** respectively

Example 2. *For the non-binary CSP given in Figure 2, we first select 3 edges from its dual graph $\{c_1, c_3\}$, $\{c_1, c_4\}$, $\{c_3, c_4\}$, and then split the constraint scopes $scp(c_1)$, $scp(c_3)$, $scp(c_4)$ into the subsets $\{x_1, x_2, x_3\}$, $\{x_4\}$, $\{x_5\}$, $\{x_6\}$. There is not any available edge for the last constraint c_2, thus, we use the basic partition.*

AC propagator on BE instances

We follow the algorithm framework used by the HTAC propagator:

1. We first partition the binary constraints $\mathcal{C}$ into a set of connected components in an undirected graph $(\mathcal{C}, E)$ where $E = \{\{c_i, c_j\} \subseteq \mathcal{C} | scp(c_i) \cap scp(c_j)$ includes compound factor variables$\}$.
2. For each connected component com, we partition the variables in com into 2 subsets T and U such that the variables in T are not included by any cycle in the primal graph of com.
3. For variables in T, we update domains from leaves to roots (forward propagation), and then from roots to leaves (backward propagation).
4. For variables in U, we use a normal propagation queue.

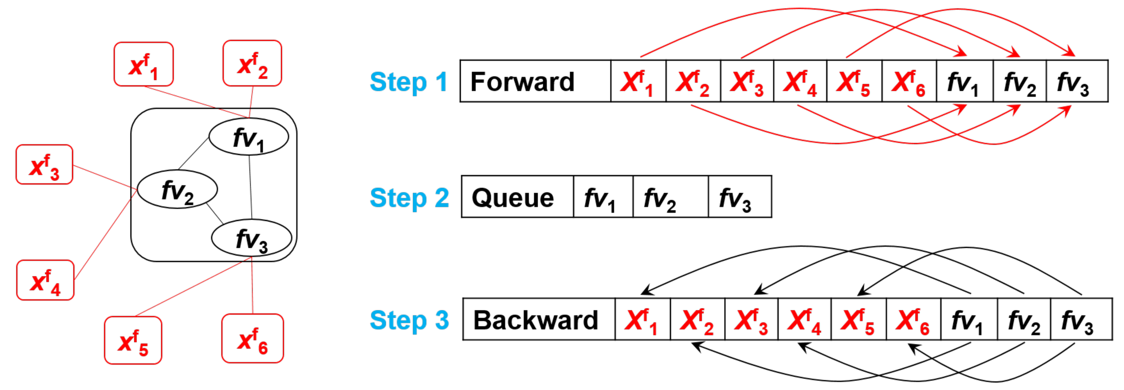


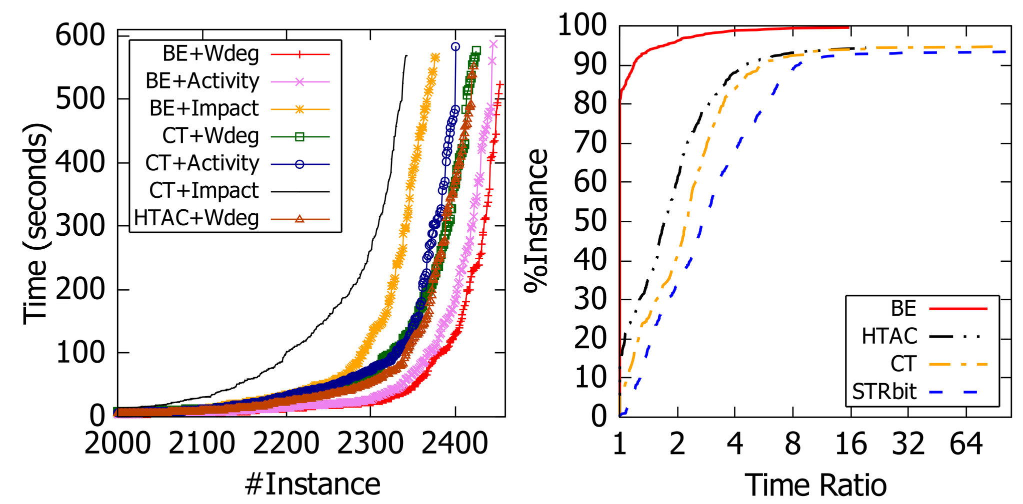
Figure 3: Propagation ordering for a component: The **left** and **right** part is the **component** and **propagation ordering** respectively

Example 3. *Figure 3 shows the propagation ordering used for enforcing AC on 2 subsets T and U of a component, where $T = \{x_1^f, ..., x_6^f\}$ and $U = \{fv_1, fv_2, fv_3\}$. We first do forward propagation on T, and then enforce AC on U, finally do backward propagation on T.*

Experiments

		STRbit	CT	HTAC	BE
Wdeg (682)	AvgR	3.50	2.68	1.99	-
	MaxR	91.46	80.98	19.2	-
	#F	3	37	89	**553**
	#TO	45	36	38	**3**
	#BF	0	0	0	**148**
Activity (655)	AvgR	3.58	2.76	2.80	-
	MaxR	90.54	80.01	77.83	-
	#F	4	87	41	**539**
	#TO	51	47	48	**3**
	#BF	0	0	0	**137**
Impact (691)	AvgR	4.73	3.52	3.64	-
	MaxR	149.78	129.44	138.61	-
	#F	0	89	33	**569**
	#TO	59	44	43	**9**
	#BF	0	0	0	**175**
Initial Time (s)		1.39	1.37	1.38	1.65

Table 1: Relative comparison with Total Times



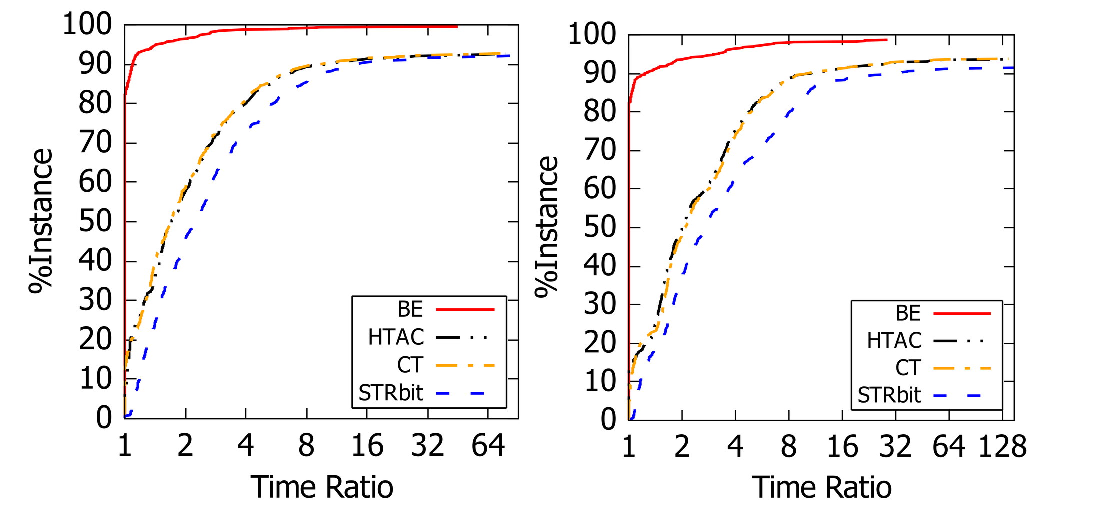


Figure 4: Runtime distribution of total time (top left plot) and Performance profiles comparing algorithms with heuristics: The **top right**, **bottom left** and **bottom right** plot is for the heuristic **WDeg**, **Activity** and **Impact** respectively

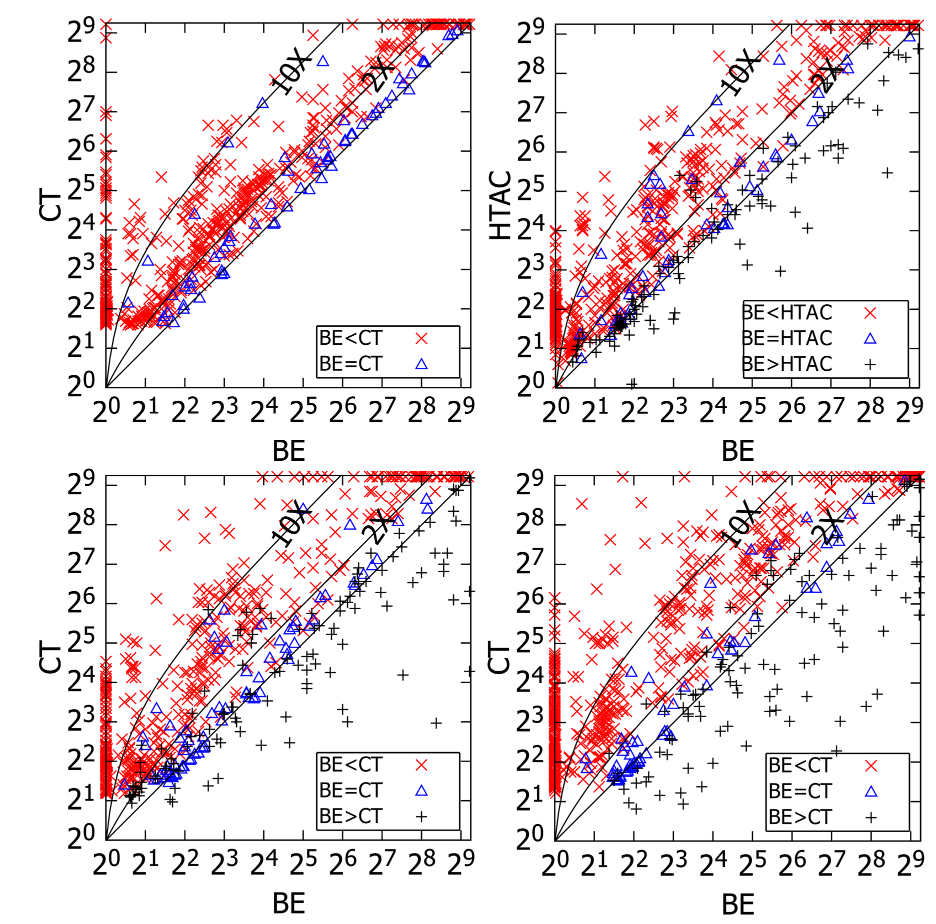


Figure 5: *Solving time* comparison: The **top left**, **top right**, **bottom left** and **bottom right** plot is for the heuristic **DDeg+O**, **WDeg**, **Activity** and **Impact** respectively. Each dot denotes an instance, the time on the x-axis and y-axis is (1 + solving time) to enable logarithmic scales, "A=B" ("A>B" and "A<B") means the number of search nodes of Algorithm A is within 2% (greater than 1.02X and less than 1.02X) than that of B. The 10X (and 2X) line means BE is 10X (and 2X) faster than CT or HTAC

- Fastest against CT (GAC) & HTAC (AC HVE encoding) across Wdeg, Activity, Impact heuristics
- Higher consistency than GAC on original CSP - many instances become Backtrack Free
- Encoding non-binary CSP to binary CSP with BE encoding and solving with AC-BE propagator gives state-of-art results