Improving SAT-Based Weighted MaxSAT Solvers

CP 2012 (Quebec)

Carlos Ansótegui[1] María Luisa Bonet[2]

Joel Gabàs[1] Jordi Levy[3]

Universitat de Lleida (DIEI, UdL)[1].

Universitat Politècnica de Catalunya (LSI, UPC)[2].

Artificial Intelligence Research Institute (IIIA, CSIC)[3].

October, 2012

Weighted Partial MaxSAT Problem

A *weighted clause* is a pair (C, w), where C is a clause and w indicates the penalty for falsifying C.

A clause is called *soft* if $w \in nat$
A clause is called *hard* if $w = \infty$

Weighted Partial MaxSAT (WPM) is the problem of finding an assignment that minimizes the aggregated cost of the falsified clauses.

Partial MaxSAT is the WPM problem when soft clauses have weight 1.

Example: Weighted Partial MaxSAT instance

Consider the pigeon-hole formula PHP_1^5 with 5 pigeons and one hole.

variables: x_i means that pigeon i goes to the only hole

soft clauses: w_i is the cost of pigeon i out of the hole

hard clauses: no two pigeons can go to the same hole

$$\begin{aligned}\varphi = \{\ & (x_1\ , 5),\\ & (x_2\ , 4),\\ & (x_3\ , 3),\\ & (x_4\ , 2),\\ & (x_5\ , 1)\ \} \cup\\ & \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\}\end{aligned}$$

$$cost(\varphi) = 10.$$

Weighted Partial MaxSAT solvers

- Branch and bound based
  - extend DPLL or CDCL SAT algorithms
  - apply incomplete MaxSAT resolution rules
  - use lowerbounding and upperbounding techniques
  - best performance on random instances
- Satisfiability testing based (SAT-based)
  - solve a sequence of SAT formulas
  - exploit unsatisfiable cores and satisfying assignments
  - introduce pseudoBoolean linear constraints
  - best performance on industrial instances

Contributions of this paper

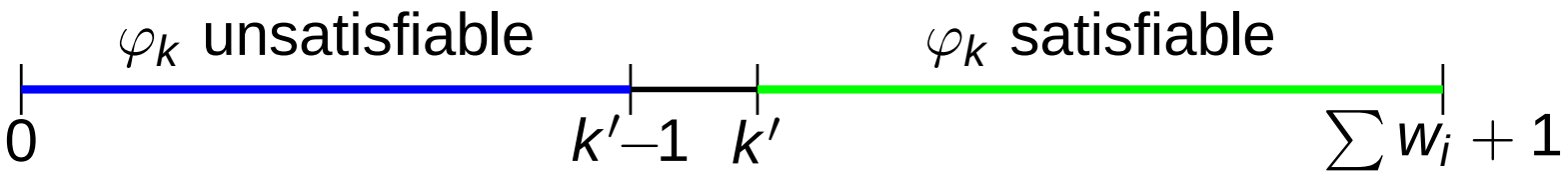
- We improve the Fu and Malik algorithm (SAT-Based) by breaking symmetries
- We improve the Weighted version of the Fu and Malik algorithm by applying an stratified approach
- We provide good experimental results on WPM industrial and crafted instances.

SAT-based MaxSAT algorithms: search scheme

A MaxSAT instance φ can be solved through a sequence of SAT instances φ_k.

φ_k be satisfiable $\leftrightarrow$ $cost(\varphi) \leq k$.

Let $k' = cost(\varphi)$,



$cost(\varphi)$ is the smallest k of satisfiable φ_k.

SAT-based MaxSAT: basic encoding of φ_k

$$
\begin{aligned}
\varphi = \{ & (x_1 \quad , w_1), \\
& (x_2 \quad , w_2), \\
& (x_3 \quad , w_3), \\
& (x_4 \quad , w_4), \\
& (x_5 \quad , w_5) \} \cup \\
& \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\}
\end{aligned}
$$

SAT-based MaxSAT: basic encoding of φ_k

$$
\begin{aligned}
\varphi_k = \{ & (x_1 \vee b_1 \ , w_1), \\
& (x_2 \vee b_2 \ , w_2), \\
& (x_3 \vee b_3 \ , w_3), \\
& (x_4 \vee b_4 \ , w_4), \\
& (x_5 \vee b_5 \ , w_5) \ \} \cup \\
& \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \cup \\
& \mathit{CNF}(\textstyle\sum_{i=1}^{5} w_i \cdot b_i \leq k, \infty)
\end{aligned}
$$

Note: φ_k is sent without weights to the SAT solver

The Fu and Malik algorithm

- Originally designed for solving Partial MaxSAT (soft with $w_i = 1$)
- Starts from $k = 0$, increasing k by 1 until φ_k is satisfiable
- φ_k satisfiable $\leftrightarrow cost(\varphi) = k$
- Exploits the unsatisfiable core returned by the SAT solver

The Fu and Malik algorithm

$$\begin{aligned}\varphi_0 = \{\ & (x_1 \quad , 1),\\ & (x_2 \quad , 1),\\ & (x_3 \quad , 1),\\ & (x_4 \quad , 1),\\ & (x_5 \quad , 1)\ \} \cup\\ & \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\}\end{aligned}$$

The Fu and Malik algorithm

$$\varphi_0 = \{ (x_1 \ , 1), \blacksquare$$
$$(x_2 \ , 1), \blacksquare$$
$$(x_3 \ , 1),$$
$$(x_4 \ , 1),$$
$$(x_5 \ , 1) \} \cup$$
$$\{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \blacksquare$$

SAT(φ_0) returns a core

$cost(\varphi) > 0$

The Fu and Malik algorithm

$$
\begin{aligned}
\varphi_1 = \{\ & (x_1 \vee b_1^1\ , 1), \blacksquare \\
& (x_2 \vee b_2^1\ , 1), \blacksquare \\
& (x_3\ \ \ \ \ \ , 1), \\
& (x_4\ \ \ \ \ \ , 1), \\
& (x_5\ \ \ \ \ \ , 1)\ \} \cup \\
& \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \cup \\
& \mathit{CNF}(b_1^1 + b_2^1 = 1, \infty) \blacksquare
\end{aligned}
$$

Soft clauses in core are relaxed

The Fu and Malik algorithm

$$\varphi_1 = \{\ (x_1 \vee b_1^1\ , 1),$$
$$(x_2 \vee b_2^1\ , 1),$$
$$(x_3\ , 1),$$
$$(x_4\ , 1),$$
$$(x_5\ , 1)\ \} \cup$$
$$\{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \cup$$
$$CNF(b_1^1 + b_2^1 = 1, \infty)$$

SAT(φ_1) returns a core

$cost(\varphi) > 1$

The Fu and Malik algorithm

$$
\begin{aligned}
\varphi_2 = \{\ & (x_1 \vee b_1^1 \quad\quad , 1), \\
& (x_2 \vee b_2^1 \quad\quad , 1), \\
& (x_3 \vee \quad b_3^2 , 1), \\
& (x_4 \vee \quad b_4^2 , 1), \\
& (x_5 \quad\quad\quad , 1)\ \} \cup \\
& \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \cup \\
& \mathit{CNF}(b_1^1 + b_2^1 = 1, \infty) \cup \\
& \mathit{CNF}(b_3^2 + b_4^2 = 1, \infty)
\end{aligned}
$$

φ_2 is unsat, $\mathit{cost}(\varphi) > 2$

The Fu and Malik algorithm

$$
\begin{aligned}
\varphi_3 = \{\ & (x_1 \vee b_1^1 \vee \quad\quad\quad\ b_1^3, 1), \\
& (x_2 \vee b_2^1 \vee \quad\quad\quad\ b_2^3, 1), \\
& (x_3 \vee \quad\ b_3^2 \vee b_3^3, 1), \\
& (x_4 \vee \quad\ b_4^2 \vee b_4^3, 1), \\
& (x_5 \quad\quad\quad\quad\quad\quad\ , 1)\ \} \cup \\
& \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \cup \\
& \mathit{CNF}(b_1^1 + b_2^1 = 1, \infty) \cup \\
& \mathit{CNF}(b_3^2 + b_4^2 = 1, \infty) \cup \\
& \mathit{CNF}(b_1^3 + b_2^3 + b_3^3 + b_4^3 = 1, \infty)
\end{aligned}
$$

φ_3 is unsat, $\mathit{cost}(\varphi) > 3$

The Fu and Malik algorithm

$$
\begin{aligned}
\varphi_4 = \{\ & (x_1 \vee b_1^1 \vee \quad\quad\quad b_1^3 \vee b_1^4\ , 1),\\
& (x_2 \vee b_2^1 \vee \quad\quad\quad b_2^3 \vee b_2^4\ , 1),\\
& (x_3 \vee \quad b_3^2 \vee b_3^3 \vee b_3^4\ , 1),\\
& (x_4 \vee \quad b_4^2 \vee b_4^3 \vee b_4^4\ , 1),\\
& (x_5 \vee \quad\quad\quad\quad\quad b_5^4\ , 1)\ \} \cup\\
& \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \cup\\
& \mathit{CNF}(b_1^1 + b_2^1 = 1, \infty) \cup\\
& \mathit{CNF}(b_3^2 + b_4^2 = 1, \infty) \cup\\
& \mathit{CNF}(b_1^3 + b_2^3 + b_3^3 + b_4^3 = 1, \infty) \cup\\
& \mathit{CNF}(b_1^4 + b_2^4 + b_3^4 + b_4^4 + b_5^4 = 1, \infty)
\end{aligned}
$$

φ_4 is sat!, $\mathit{cost}(\varphi) = 4$

The Fu and Malik algorithm

if $SAT(\{C_i \mid w_i = \infty\}) = (\text{UNSAT}, \_)$ **then return** $(\infty, \emptyset)$;
$cost := 0$; /* Optimal */
$s := 0$; /* Counter of cores */
while *true* **do**
  $(st, \varphi_c) := SAT(\{C_i \mid (C_i, 1) \in \varphi\})$; /* Call without weights */
  if $st = \text{SAT}$ **then return** $(cost, \varphi)$;
  $s := s + 1$;
  $A_s := \emptyset$; /* Indexes of the core */
  foreach $C_i \in \varphi_c$ **do**
    if $w_i \neq \infty$ **then** /* If the clause is soft */
      $b_i^s :=$ new_variable( );
      $\varphi := \varphi \setminus \{(C_i, 1)\} \cup \{(C_i \vee b_i^s, 1)\}$; /* Add blocking variable */
      $A_s := A_s \cup \{i\}$;
    end
  end
  $\varphi := \varphi \cup \{(C, \infty) \mid C \in CNF(\sum_{i \in A_s} b_i^s = 1)\}$; /* Add cardinality */
  $cost := cost + 1$
end

Breaking Symmetries in the Fu and Malik algorithm

Lets look again to φ_3 in our working example without clause $(x_5, 1)$

$$\begin{array}{llll} x_1 \vee & \boxed{b_1^1} \vee & & b_1^3 \\ x_2 \vee & b_2^1 \vee & & \boxed{b_2^3} \\ x_3 \vee & & \boxed{b_3^2} \vee & b_3^3 \\ \boxed{x_4} \vee & & b_4^2 \vee & b_4^3 \end{array}$$

$$\boxed{b_1^1} + b_2^1 = 1$$

$$\boxed{b_3^2} + b_4^2 = 1$$

$$b_1^3 + \boxed{b_2^3} + b_3^3 + b_4^3 = 1$$

Breaking Symmetries in the Fu and Malik algorithm

Lets look again to φ_3 in our working example without clause $(x_5, 1)$

The formula is satisfiable, but it has 8 models instead of 4!

$$\begin{array}{llll} x_1 \vee & \boxed{b_1^1} \vee & & b_1^3 \\ x_2 \vee & b_2^1 \vee & & \boxed{b_2^3} \\ x_3 \vee & & \boxed{b_3^2} \vee & b_3^3 \\ \boxed{x_4} \vee & & b_4^2 \vee & b_4^3 \end{array}$$

$$\boxed{b_1^1} + b_2^1 = 1$$

$$\boxed{b_3^2} + b_4^2 = 1$$

$$b_1^3 + \boxed{b_2^3} + b_3^3 + b_4^3 = 1$$

$$\begin{array}{llll} x_1 \vee & b_1^1 \vee & & \boxed{b_1^3} \\ x_2 \vee & \boxed{b_2^1} \vee & & b_2^3 \\ x_3 \vee & & \boxed{b_3^2} \vee & b_3^3 \\ \boxed{x_4} \vee & & b_4^2 \vee & b_4^3 \end{array}$$

$$b_1^1 + \boxed{b_2^1} = 1$$

$$\boxed{b_3^2} + b_4^2 = 1$$

$$\boxed{b_1^3} + b_2^3 + b_3^3 + b_4^3 = 1$$

Breaking Symmetries in the Fu and Malik algorithm

The two previous models are related by the permutation

$$b_1^1 \leftrightarrow b_2^1, b_1^3 \leftrightarrow b_2^3$$

The existence of so many *partial* models makes the task of showing unsatisfiability of the formula (including x_5) much harder.

After finding the third unsatisfiable core, we break symmetries by adding to φ_3:

$$b_1^3 \rightarrow \neg b_2^1$$
$$b_3^3 \rightarrow \neg b_4^2$$

The WPM1 algorithm

Weighted version of Fu and Malik algorithm [WPM1 and WBO, 2009]

Idea:

Compute the minimum weight, w_{min}, in the unsat core

Replace soft clauses in the unsat core by,

- a copy with with weight w_{min} plus blocking variables and cardinality
- a copy with weight $w_i - w_{min}$

Increase the current cost by w_{min}.

The WPM1 algorithm

$$\begin{aligned}\varphi_0 = \{\ & (x_1 \ , 5),\\ & (x_2 \ , 4),\\ & (x_3 \ , 3),\\ & (x_4 \ , 2),\\ & (x_5 \ , 1)\ \} \cup\\ & \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\}\end{aligned}$$

The WPM1 algorithm

$$\varphi_0 = \{ (x_1 \ , 5), \blacksquare$$
$$(x_2 \ , 4), \blacksquare$$
$$(x_3 \ , 3),$$
$$(x_4 \ , 2),$$
$$(x_5 \ , 1) \} \cup$$
$$\{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \blacksquare$$

SAT(φ_0) returns a core

$cost(\varphi) > 0$

The WPM1 algorithm

$$\varphi_0 = \{\ (x_1\ , 5),\ (x_2\ , 4),\ (x_3\ , 3),\ (x_4\ , 2),\ (x_5\ , 1)\ \} \cup \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\}$$

SAT(φ_0) returns a core

$cost(\varphi) > 0$

$$\varphi_4 = \{\ (x_1 \vee b_1^1\ , 4),\ (x_2 \vee b_2^1\ , 4),\ (x_1\ , 1),\ (x_3\ , 3),\ (x_4\ , 2),\ (x_5\ , 1)\ \} \cup \{(\neg x_i \vee \neg x_j, \infty) \mid i < j\} \cup CNF(b_1^1 + b_2^1 = 1, \infty)$$

copy with w_{min}

copy with $w_i - w_{min}$

$SAT(\varphi_4)$ unsat, $cost(\varphi) > 4$.

The WPM1 algorithm

if $SAT(\{C_i \mid w_i = \infty\}) = (\text{UNSAT}, \_)$ **then return** $(\infty, \emptyset)$;
$cost := 0$; /* Optimal */
$s := 0$; /* Counter of cores */
while *true* **do**
  $(st, \varphi_c) := SAT(\{C_i \mid (C_i, w_i) \in \varphi\})$; /* Call SAT without weights */
  if $st = \text{SAT}$ **then return** $(cost, \varphi)$;
  $s := s + 1$;
  $A_s := \emptyset$; /* Indexes of the core */
  $w_{min} := \min\{w_i \mid C_i \in \varphi_c \wedge w_i \neq \infty\}$; /* Minimum weight */
  foreach $C_i \in \varphi_c$ **do**
    if $w_i \neq \infty$ **then**
      $b_i^s :=$ new_variable()
      $\varphi := \varphi \setminus \{(C_i, w_i)\} \cup \{(C_i, w_i - w_{min}),\ (C_i \vee b_i^s, w_{min})\}$
      $A_s := A_s \cup \{i\}$
    end
  end
  $\varphi := \varphi \cup \{(C, \infty) \mid C \in CNF(\sum_{i \in A_s} b_i^s = 1)\}$; /* Add cardinality */
  $cost := cost + w_{min}$
end

Stratified approach for the WPM1 algorithm

Problem:

- the number of iterations depends on w_{min}.
- SAT solvers have no notion of weights
- SAT solvers can return not minimal unsat cores

Stratified approach:

- idea: force the SAT solver to focus on clauses with higher weights
- only clauses with $w_i > w_{max}$ are sent to the solver
- when SAT solver returns sat, w_{max} is decreased
- copies with $w_i - w_{min}$ are scheduled to their appropiated w_{max}

Stratified approach for the WPM1 algorithm

if $SAT(\{C_i \mid w_i = \infty\}) = (\text{UNSAT}, \_)$ **then return** $(\infty, \emptyset)$;
$cost := 0$; /* Optimal */
$s := 0$; /* Counter of cores */
$w_{max} := \max\{w_i \mid (C_i, w_i) \in \varphi \wedge w_i < \infty\}$;
while *true* **do**
  $(st, \varphi_c) := SAT(\{C_i \mid (C_i, w_i) \in \varphi \wedge w_i \geq w_{max}\})$;
  if $st = \text{SAT}$ **then**
    if $w_{max} = 0$ **then return** $(cost, \varphi)$;
    else $w_{max} := \max\{w_i \mid (C_i, w_i) \in \varphi \wedge w_i < w_{max}\}$;
  end
  else
    $s := s + 1$;
    $A_s := \emptyset$; /* Indexes of the core */
    $w_{min} := \min\{w_i \mid C_i \in \varphi_c \wedge w_i \neq \infty\}$; /* Minimum weight */
    foreach $C_i \in \varphi_c$ **do**
      if $w_i \neq \infty$ **then**
        $b_i^s :=$ new_variable();
        $\varphi := \varphi \setminus \{(C_i, w_i)\} \cup \{(C_i, w_i - w_{min}),\ (C_i \vee b_i^s, w_{min})\}$
        $A_s := A_s \cup \{i\}$
      end
      $\varphi := \varphi \cup \{(C, \infty) \mid C \in CNF(\sum_{i \in A_s} b_i^s = 1)\}$; /* Cardinal. */
      $cost := cost + w_{min}$
    end
  end
end

Experimental investigation

- Experiments conducted within MaxSAT evaluation environment:

  Memory extended from 0.5 to 1GB

  Cutoff extended from 30' to 2h

- Best solvers from MaxSAT evaluation 2011

- Other MaxSAT solvers reported to have good performance:

  bincd: F. Heras, A. Morgado, J. Marques-Silva

  maxhs: J. Davies, F. Bacchus

- WCSP solver:

  toulbar2: Sanchez, Bouveret, Givry, Heras, Jgou, Larrosa, Ndiaye, Rollon, Schiex, Terrioux, Verfaillie, Zytnicki

Experimental investigation

Our solvers:

wpm1	WPM1 algorithm on picoSAT (MaxSAT Eval. 2009/10)
wpm1_s	wpm1 + stratified approach (MaxSAT Eval. 2011)
wpm1_b	wpm1 + breaking symmetries
wpm1_{bs}	wpm1_b + stratified approach
wpm1_{bsd}	wpm1_{bs} + diversity heuristic (MaxSAT Eval. 2012)

Weighted Partial MaxSAT industrial instances

3 families, 226 instances

Solved instances		
1.-	$wpm1_{bsd}$	204
2.-	$wpm1_{bs}$	204
3.-	$wpm1_{s}$	196
4.-	$wpm1_{b}$	188
5.-	wbo1.6	176
6.-	wpm1	172
7.-	maxhs	138
8.-	bincd	105
9.-	sat4j	50
10.-	binc	31
11.-	toulbar2	5

Mean solved family ratio		
1.-	$wpm1_{bsd}$	76.3%
2.-	$wpm1_{bs}$	76.3%
3.-	$wpm1_{s}$	72.6%
4.-	$wpm1_{b}$	71.0%
5.-	wpm1	63.7%
6.-	wbo1.6	62.3%
7.-	maxhs	49.7%
8.-	bincd	40.7%
9.-	sat4j	16.6%
10.-	binc	14.0%
11.-	toulbar2	1.7%

Weighted Partial MaxSAT crafted Instances

8 families, 372 instances

Solved instances		
1.-	wpm1$_{bsd}$	270
2.-	incw_maxsatz	241
3.-	ak_maxsat	226
4.-	toulbar2	225
5.-	wpm1$_{bs}$	224
6.-	wmax_satz09z	222
7.-	maxhs	204
8.-	sat4j	189
9.-	wpm1$_{s}$	184
10.-	bincd	125
11.-	binc	108
12.-	wpm1	84
13.-	wbo1.6	82
14.-	wpm1$_{b}$	67

Mean solved family ratio		
1.-	wpm1$_{bsd}$	66.5%
2.-	incw_maxsatz	56.6%
3.-	wpm1$_{bs}$	53.1%
4.-	toulbar2	43.9%
5.-	ak_maxsat	43.6%
6.-	wmax_satz09z	41.4%
7.-	wpm1$_{s}$	40.5%
8.-	maxhs	39.1%
9.-	sat4j	38.4%
10.-	bincd	35.1%
11.-	wpm1	30.2%
12.-	binc	28.5%
13.-	wpm1$_{b}$	25.5%
14.-	wbo1.6	22.2%

Thanks!

Experimental results

Instance set	#	wpm1$_{bsd}$	wpm1$_{bs}$	wpm1$_{s}$	wpm1$_{b}$	wbo1.6	wpm1
haplotyping	100	**423(95)**	425(95)	378(88)	376(79)	93.4(72)	390(65)
timetabling	26	685.60(9)	**683(9)**	585(8)	992(9)	776(4)	1002(7)
upgradeability	100	**35.8(100)**	37.2(100)	36.5(100)	36.9(100)	63.3(100)	114(100)
Total	226	**204**	204	196	188	176	172

maxhs	bincd	sat4j	binc	toulbar2
1043(34)	196(21)	108(20)	408(27)	383(5)
1238(4)	766(6)	0(0)	1350(4)	0(0)
29.0(100)	637(78)	844(30)	0(0)	0(0)
138	105	50	31	5

Table: Weighted Partial - Industrial.

Experimental results

Instance set	#	maxhs	wbo1.6	wpm1$_b$	wpm1$_{bsd}$	wpm1	wpm1$_{bs}$
Table4 [**?**]	13	**4.41(13)**	5.03(13)	5.54(13)	8.84(13)	18.92(13)	534.13(13)
Total	13	**13**	13	13	13	13	13

wpm1$_s$	bincd	sat4j	binc	toulbar2
559.23(13)	56.69(11)	3485.52(6)	19.50(1)	0.00(0)
13	11	6	1	0

Table: Table 4 from [**?**]. Linux upgradibility family forcing diversity of weights.

Experimental results

Instance set	#	wpm1$_{bsd}$	incw maxsatz	ak maxsat	toulbar2	wpm1$_{bs}$	wmax satz09z
auc-paths	86	274(53)	7.59(86)	**4.6(86)**	28.8(86)	332(53)	570(80)
auc-sched	84	**11.9(84)**	220(84)	123(84)	133(84)	12.2(84)	92(84)
planning	56	27.9(52)	92.2(38)	354(40)	149(41)	**26.3(56)**	220(50)
warehouses	18	44(14)	1184(18)	37(2)	0.03(1)	571(3)	0.32(1)
miplib	12	1187(4)	**1419(5)**	0.47(2)	63.2(3)	1165(4)	266(3)
random-net	74	**241(39)**	1177(1)	1570(2)	0(0)	0(0)	0(0)
spot5dir	21	257(10)	1127(5)	1106(5)	217(5)	383(10)	11.5(2)
spot5log	21	**532(14)**	0.63(4)	200(5)	170(5)	574(14)	15.6(2)
Total	372	**270**	241	226	225	224	222

maxhs	sat4j	wpm1$_{s}$	bincd	binc	wpm1	wbo1.6	wpm1$_{b}$
72.2(86)	994(44)	22(33)	1828(2)	0(0)	0(0)	0(0)	0(0)
1125(69)	716(80)	7.6(80)	130(50)	103(45)	0(0)	0(0)	0(0)
306(29)	3.27(55)	12.5(54)	59.5(47)	51.1(46)	1.46(28)	1.33(30)	3.63(29)
0.37(1)	1.34(1)	1644(3)	7.03(1)	8.67(1)	**4.23(18)**	0.51(4)	0.88(12)
0.07(1)	693(4)	34(3)	618(3)	699(3)	0.21(1)	0(0)	1507(1)
2790(6)	0(0)	0(0)	0(0)	0(0)	615(27)	63(37)	439(12)
199(6)	1.95(2)	1.41(5)	**66.7(11)**	51.7(6)	1.03(4)	2.60(5)	12.8(6)
710(6)	6.04(3)	44.4(6)	124(11)	79.3(7)	21.5(6)	25.5(6)	131(7)
204	189	184	125	108	84	82	67

Table: Weighted Partial - Crafted.

Stratified approach with other algorithms

[Weighted Partial - Industrial]

Instance set	#	bincd-2_{sd}	bincd-2	wpm2_{sd}	wpm2
haplotyping	100	**255(22)**	223(20)	**515(23)**	32.6(22)
timetabling	26	**1243(2)**	1128(1)	257(3)	**246(4)**
upgradeability	100	34.9(100)	**30.8(100)**	**49.0(100)**	55.4(99)
Total	226	**124**	121	**126**	125

[Table 4 from [**?**]. Linux upgradibility family forcing diversity of weights]

Instance set	#	bincd-2_{sd}	bincd-2	wpm2_{sd}	wpm2
Table4 [**?**]	13	27.1(13)	**9.70(13)**	13.9(13)	**9.05(13)**
Total	13	13	**13**	13	**13**

[Weighted Partial - Crafted]

Instance set	#	bincd-2_{sd}	bincd-2	wpm2_{sd}	wpm2
auc-paths	86	**785(47)**	739(42)	**2551(3)**	0(0)
auc-sched	84	**534(81)**	810(70)	**76.0(46)**	0(0)
planning	56	2.60(55)	**3.36(56)**	**89.6(36)**	28.7(23)
warehouses	18	**0.18(1)**	**0.18(1)**	**19.2(1)**	23.9(1)
miplib	12	6.77(2)	**3.61(2)**	**13.7(1)**	16.4(1)
random-net	74	2175(15)	**1918(18)**	0(0)	0(0)
spot5dir	21	529(11)	**380(11)**	**486(10)**	518(9)
spot5log	21	566(11)	**1748(12)**	**572(10)**	0.71(6)
Total	372	**223**	212	**107**	40

Table: Experimental results.