1 Introduction: goals of the evaluation

The approximate algorithms providing bounds on the optimal solution for the optimization tasks are widely used to generate heuristics for the search algorithm. In particular, we currently use Mini-bucket Elimination algorithm to generate heuristic functions for the AOBB search. The main goal of this empirical evaluation is to assess the performance of improved approximation schemes and estimate whether they can be used as an alternative for generating heuristics for the Branch and Bound search algorithm. Due to the difference in the implementations, we concentrate on the quality of the bounds produced and pay less attention to the run time results. The comparison of runtimes remains future work.

2 Algorithms

We conducted the experimental comparison of 3 types of algorithms: Mini-bucket with Moment-Matching (MBE-MM), Max Product Linear Programming algorithm (MPLP) [4] and Mini-bucket Elimination with Bucket Propagation (MBEp), aka soft-arc consistency MBE, aka Horizontal Mini Bucket (h-MBE) [6]. In the following we assume the MPE task.

2.1 Mini-Bucket Elimination with Moment Matching

The algorithm denoted MBE-MM has two main parameters that differentiate it from the "standard" Mini-bucket elimination algorithm [3]:

1

Mini-bucket elimination with moment matching

Natalia Flerova

August 25, 2011

Contents

1 Introduction: goals of the evaluation 1

2 Algorithms 1

2.1 Mini-Bucket Elimination with Moment Matching 1

2.2 Max Product Linear Programming 2

2.3 Mini-Bucket Elimination with Horizontal Propagation 3

3 Empirical evaluation 3

3.1 Experimental setting 3

3.2 Experimental results and analysis 3

3.3 Additional experiments with new cpp code 8

3.3.1 Evaluation of pedigree instances 12

3.3.2 Evaluation on Weighted CSP instances 12

4 Future work 18
1. there is a choice of 7 partitioning heuristics that can be used to control the partitioning of the bucket functions into mini-buckets

2. there is an option of performing moment-matching procedure over the “sibling” mini-buckets (i.e. mini-buckets formed by partitioning the same bucket).

Let us elaborate on the options listed above.

Given an integer parameter $i$ and a bucket $B$ with the scope larger than $i + 1$, MBE scheme partitions $B$ into several mini-buckets, whose scopes are bounded by $i + 1$. The most widely used partitioning heuristic is the scope-based heuristic, which does not take into consideration the values of the functions, only their scopes. The functions are added to a mini-bucket as long as the scope of mini-bucket does not exceed $i + 1$. In [5] a new class of content-based partitioning heuristics that consult the functions’ contents in addition to their scopes was proposed. The goal is to find a partitioning that is closest to the original bucket, where closeness can be defined using any distance measure $d$.

The $MBE-MM$ uses the following distance measures:

- L1: integrated absolute error
- L2: integrated squared error
- Linf: maximum absolute error
- KL: Kullback-Leibler divergence
- HPM: Hilbert’s projective metric. $HPM(f_1, f_2) = \max (\log \frac{f_1}{f_2}) - \min (\log \frac{f_1}{f_2})$
- MAS: $MAS(f_1, f_2) = \max \left( \frac{\max (\log f_1)}{\min (\log f_2)}, \frac{1}{\min (\log f_2)} \right) - 1$

The moment-matching uses the idea of sharing information between mini-buckets of the same variable $X$, akin to soft arc-consistency (e.g. [2]).

Consider mini-buckets $\{B_1, \ldots, B_k\}$ generated by partitioning the bucket of variable $X$. Let $\{F_1(S_1), \ldots, F_k(S_k)\}$ be the functions of mini-buckets of variable $X$ $\{B_1, \ldots, B_k\}$ and let $S$ be a set of variables that are the intersection of the scopes of these functions, $S = \{S_1 \cap \cdots \cap S_k\}$. Moment-matching is performed before eliminating the bucket variable $X$ from any of the mini-buckets.

The moment-matching procedure consists of:

1. marginalizing the mini-bucket functions over ”residual” variables in their scopes: $M_i(S) = \max_{S_i \setminus S}(F_i(S_i))$

2. calculate the geometric mean $M$ of $M_i's$

3. the mini-bucket functions are normalized by the geometrical mean $M$

2.2 Max Product Linear Programming

The MPLP algorithm is a message-passing algorithm that closely resembles max-product, but is guaranteed to always converge. The algorithm can be derived via block coordinate descent in a dual of the LP relaxation of MPE problem. The main difference between MPLP and max-product is the formula for the message update (see [4]).

MPLP algorithm can take as an input either original problem factors or a mini-cluster tree. For the latter case we used the trees produced by the MBE algorithm with the scope-based partitioning heuristic and various z-bounds.

MPLP is an iterative algorithm that is guaranteed to improve with more iterations.
2.3 Mini-Bucket Elimination with Horizontal Propagation

Just like MBE-MM, the Mini-Bucket Elimination with Horizontal Propagation $h$-MBE also implements an additional (compared to “standard” MBE) procedure, called movement of costs. It is designed to share information between the mini-buckets of the same variable. Unlike moment-matching this operation is not symmetric and is performed in a particular direction.

Given two functions $f_1(S_1)$ and $f_2(S_2)$ and $S = \{S_1 \cap S_2\}$, movement of costs from $f_1$ to $f_2$ is achieved in 3 steps (assuming MPE task):

1. $M = f_1(S)$
2. $f_1 = f_1/M$
3. $f_2 = f_2 \ast M$

The order and the direction in which the movement of cost is done between the mini-buckets has a considerable impact on the quality of the solution. The $h$-MBE algorithm has 4 propagation options:

- the information is propagated between mini-buckets in the same order as the mini-buckets were originally created
- the mini-buckets are organized into a minimum-spanning tree
- mini-buckets are chosen for cost shifting in random order
- the mini-bucket tree is constructed using the partitioning heuristic described in CP’06 paper. This partitioning heuristic assumes that the buckets whose scopes include the variables that are later in the ordering should be higher in the mini-bucket tree than the buckets, whose scopes have variables later in the orderings, and the buckets that share the maximum variable should be connected.

3 Empirical evaluation

3.1 Experimental setting

Algorithms MBE-MM and MPLP were implemented in Matlab by Alex Ihler, $h$-MBE was implemented in C by Emma Rollon. Because of the implementation differences the time comparisons between the algorithms were not particularly meaningful. However, we still report run-time information for the Matlab-based algorithms for completeness. The algorithms were ran on the pedigree instances from the UAI 2008 competition, the exact solutions for the instances were available (solved by Branch and Bound search). The characteristics of the instances can be seen in Table 1

3.2 Experimental results and analysis

For all instances the MPE task was solved and higher bounds were outputted, thus on all the figures presented here, the lower is the point, the better.

The combination of different partitioning heuristics and the option of using or not using moment-matching gives rise to 14 different variants of the MBE-MM scheme. Each variant was ran with i-bound 5 and 10. MPLP was ran on the original factors and on the mini-cluster trees formed by MBE with i-bounds 1, 5 and 10 and scope-based partitioning heuristics. Each variant was ran for 5, 10, 15, 30, 60, 120 or 240 iterations. $h$-MBE was ran with no cost movement and with all 4 information propagation heuristics. It was ran with i-bound equal to 10.

Summarizing the results, several main conclusions can be made.

Analysing performance of MBE-MM, we can see that the use of moment-matching always improves the bound, even though it introduces time overhead. A Table 2 indicated the percentage of instances for which the version with the particular partitioning heuristic (with or without moment-matching) found the
best bound. The numbers do not sum up to 100% because for some instances more than one scheme found the same bound. There is no clear winner in the partitioning heuristics, though there is a clear looser: the one that uses KL distance measure never produced the best bound with or without moment matching. The scope-based partitioning heuristic fares very well, especially for the i-bound=5.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>% best i-bound=5</th>
<th>% best i-bound=10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope-based partitioning heuristics + no MM</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Scope-based partitioning heuristics + MM</td>
<td>28.6</td>
<td>21.4</td>
</tr>
<tr>
<td>L1 partitioning heuristic + no MM</td>
<td>14.3</td>
<td>14.3</td>
</tr>
<tr>
<td>L1 partitioning heuristic + MM</td>
<td>35.7</td>
<td>28.6</td>
</tr>
<tr>
<td>L2 partitioning heuristic + no MM</td>
<td>14.3</td>
<td>7.1</td>
</tr>
<tr>
<td>L2 partitioning heuristic + MM</td>
<td>28.6</td>
<td>35.7</td>
</tr>
<tr>
<td>Linf partitioning heuristic + no MM</td>
<td>7.1</td>
<td>7.1</td>
</tr>
<tr>
<td>Linf partitioning heuristic + MM</td>
<td>28.6</td>
<td>21.4</td>
</tr>
<tr>
<td>KL partitioning heuristic + no MM</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>KL partitioning heuristic + MM</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>HPM partitioning heuristic + no MM</td>
<td>7.1</td>
<td>0.0</td>
</tr>
<tr>
<td>HPM partitioning heuristic + MM</td>
<td>35.7</td>
<td>7.1</td>
</tr>
<tr>
<td>MAS partitioning heuristic + no MM</td>
<td>7.1</td>
<td>0.0</td>
</tr>
<tr>
<td>MAS partitioning heuristic + MM</td>
<td>21.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 1: The pedigree instances

Figure 2: The percentage of the instances for which each version of MBE-MM found the best bound, for z-bounds 5 and 10

Figures 3, 4, 5 present the bounds outputted by the MBE-MM and MPLP for 3 of the instances. Figures 6, 7, 8 present corresponding runtime results. For MBE-MM for i-bounds equal to 5 and 10 we plot the results for scope-based partitioning heuristic and the best among content-based partitioning heuristics with and without moment matching. For MPLP we plot results of algorithm taking as an input the original factors, and mini-cluster trees created with i-bounds equal to 1, 5 and 10.

Unsurprisingly, we see that the greater is the cluster size, the slower MPLP runs. At the lower number of iterations the MPLP on larger clusters producers better results than on smaller ones. However, as the number of iterations increase, they converge to the similar log(MPE)’s. Compared to MBE-MM, in some
Figure 3: Pedigree 1, upper bounds on the log(MPE) by MBE-MM and MPLP, exact solution in the brackets. For MBE-MM the results are presented for scope-based partitioning heuristic and the best among content-based partitioning heuristics with and without moment matching. For MPLP the results of algorithm taking as an input the original factors, and mini-cluster trees created with i-bounds equal to 1, 5 and 10 are plotted.
Figure 4: Pedigree 23, upper bounds on the log(MPE) by MBE-MM and MPLP, exact solution in the brackets. For MBE-MM the results are presented for scope-based partitioning heuristic and the best among content-based partitioning heuristics with and without moment matching. For MPLP the results of algorithm taking as an input the original factors, and mini-cluster trees created with i-bounds equal to 1, 5 and 10 are plotted.
Figure 5: Pedigree 13, upper bounds on the log(MPE) by *MBE-MM* and *MPLP*, exact solution in the brackets. For *MBE-MM* the results are presented for scope-based partitioning heuristic and the best among content-based partitioning heuristics with and without moment matching. For *MPLP* the results of algorithm taking as an input the original factors, and mini-cluster trees created with i-bounds equal to 1, 5 and 10 are plotted.
Figure 6: Pedigree 1, runtime (sec) by MBE-MM and MPLP. For MBE-MM the results are presented for scope-based partitioning heuristic and the best among content-based partitioning heuristics with and without moment matching. For MPLP the results of algorithm taking as an input the original factors, and mini-cluster trees created with i-bounds equal to 1, 5 and 10 are plotted.

instances (e.g. pedigree 13) MPLP manages to find better solutions after a certain number of iterations, for other instances (e.g. pedigree 23) MPLP seems to be stuck in local minimum and not output as good bounds as the best of MBE-MM variations. MPLP is always slower to produce a bound of the same quality. Such poor performance is not suggested by theory and is attributed to the peculiarities of current implementation, in particular, to the scheduling procedure.

Figure 9 presents the logarithms of Most Probable Explanation found by the h-MBE algorithms with different propagation options, including the "standard" mini-bucket algorithm without any cost shifting. The standard MBE performs well compared to other schemes, considering that it has the best runtime. Figure 10 presents the bounds by both MBE-MM and h-MBE. It can be seen that there is no clear winner or loser across the instances. In Figures 11 and 12 it can be clearly seen that the partitioning heuristics successful for one instance are relatively bad on the other.

3.3 Additional experiments with new cpp code

We performed additional experiments to provide better evaluation of run time performance of new implementation of MBE-MM and MPLP algorithm by Alex Ihler. The algorithms themselves are implemented in cpp, Matlab is used to read the input and provide output.

The implementation of Mini-bucket Elimination algorithm allows to use any of the content-based partitioning heuristics described in Section 2, and the moment-matching is always performed. In this report we show the best results obtained across the different heuristics.

The MPLP algorithm always runs on the original factors. It is implemented in two variants: one uses message-passing updates (i.e. the edges are updated one by one), denoted mp – MPLP, and a fixed update schedule where a "star" of edges is updated simultaneously, denoted MPLP. We expected (and it was proven empirically) the latter version to be considerably faster than the former.
Figure 7: Pedigree 23, runtime (sec) by MBE-MM and MPLP. For MBE-MM the results are presented for scope-based partitioning heuristic and the best among content-based partitioning heuristics with and without moment matching. For MPLP the results of algorithm taking as an input the original factors, and mini-cluster trees created with i-bounds equal to 1, 5 and 10 are plotted.
Figure 8: Pedigree 13, runtime (sec) by MBE-MM and MPLP. For MBE-MM the results are presented for scope-based partitioning heuristic and the best among content-based partitioning heuristics with and without moment matching. For MPLP the results of algorithm taking as an input the original factors, and mini-cluster trees created with i-bounds equal to 1, 5 and 10 are plotted.

Figure 9: Upper bounds on the log(MPE) found by Horizontal Mini Bucket Elimination algorithm ($h$-MBE), i-bound = 10
Figure 10: Upper bounds on the log(MPE) found by $h$-MBE and $MBE-MM$, i-bound=10. For $MBE-MM$ we report the best bounds found.

Figure 11: Pedigree 13, upper bounds on the log(MPE) found by $h$-MBE and $MBE-MM$
Figure 12: Pedigree 19, upper bounds on the log(MPE) found by $h$-$MBE$ and $MBE$-$MM$

### 3.3.1 Evaluation of pedigree instances

For the preliminary evaluation we used a subset of pedigree instances (Table 1), specifically pedigree1, pedigree19, pedigree20, pedigree23, pedigree37, pedigree38, pedigree39 and pedigree50. These particular instances were chosen with the intention to get the results fast. The $MM$-$MBE$ algorithm was ran for $i$-Bound=5.

Figures 13, 14 and 15 represent the upper bounds found by $MBE$-$MM$, $MPLPS$ and $mp$-$MPLP$ on the log(MPE) for 3 chosen problems. The exact solutions are also plotted, in addition to that the value of the optimal solution is outputted in square brackets.

We see, that message-passing $MPLP$ algorithm ($mp$-$MPLP$) performs significantly worse than the fixed-schedule $MPLPS$. Not only it slower, but it outputs worse results for the same number of iterations. Unlike our previous experiments, $MPLPS$ provides better bounds than $MBE$ even for a small number of iterations.

In Figure 16 we seen how the $MPLPS$’s runtime scales with the increase of the number of iterations. For a small number of iterations (up to 30) the runtime of $MPLPS$ is comparable with that of $MM$-$MBE$, though more iterations require longer time.

Figures 17, 18, 19, 20, 21, 22 and 23 present the dependence of the upper bound on the log(MPE) on time for $MPLPS$ for the pedigree instances. The results for $MM$-$MBE$ with different partitioning heuristics are also plotted as a point (since MBE algorithms are not iterative and the results do not change with the time). Note that in this figures we plot the results for $MBE$-$MM$ with $i$-Bound=10.

### 3.3.2 Evaluation on Weighted CSP instances.

We also conducted evaluation of the new implementations of $MM$-$MBE$ and $MPLP$ on the selected Weighted CSP from UAI08 competition [1]. We solve Maximum Probability Explanation task, we use WCSP as a moniker for the dataset, not indication of the task. The $MM$-$MBE$ algorithm was ran for $i$-Bound=10.
Figure 13: Pedigree 1, upper bound on the log(MPE) as a function of number of iterations of the $MBE-MM$, $MPLPS$ and $mp-MPLP$. For $MBE-MM$ the results are presented for the best among content-based partitioning heuristics with moment matching for i-Bound=5. $mp-MPLP$ and $MPLPS$ are ran on the original factors.

Figure 14: Pedigree 20, upper bound on the log(MPE) as a function of number of iterations of the $MBE-MM$, $MPLPS$ and $mp-MPLP$. For $MBE-MM$ the results are presented for the best among content-based partitioning heuristics with moment matching for i-Bound=5. $mp-MPLP$ and $MPLPS$ are ran on the original factors.
Figure 15: Pedigree 23, upper bound on the log(MPE) as a function of number of iterations of the \textit{MBE-MM}, \textit{MPLPS} and \textit{mp−MP LP}. For \textit{MBE-MM} the results are presented for the best among content-based partitioning heuristics with moment matching for i-Bound=5. \textit{mp – MPLP} and \textit{MPLPS} are ran on the original factors.

Figure 16: Pedigrees 1, 20 and 23, runtime (in seconds) as a function of number of iterations of the \textit{MBE-MM}, \textit{MPLPS} and \textit{mp−MP LP}. For \textit{MBE-MM} the results are presented for the best among content-based partitioning heuristics with moment matching for i-Bound=5. \textit{mp – MPLP} and \textit{MPLPS} are ran on the original factors.
Figure 17: Pedigrees 1, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. The results for partitioning heuristics HPM, l2, linf, MAS are the same. MPLPS is ran on the original factors.

Figure 18: Pedigrees 20, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.
Figure 19: Pedigrees 23, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.

Figure 20: Pedigrees 37, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.
Figure 21: Pedigrees 38, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.

Figure 22: Pedigrees 39, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.
Figures 24 and 25 present the upper bounds and runtime as functions of number of iterations. We see that for all three instances the results of MPLP are considerably worse than the bounds found by MM-MBE and for instances 1502 and 29 MPLPS improves the solution very slowly (in fact, it is hard to see any improvement on the graph itself). Increasing the total number of iterations does not improve the results, since the algorithm aborts due to the second stopping condition, namely due to an insignificant change of the results between the iterations, as can be clearly seen on the runtime graph.

Figures 26, 27, 28, 29, 30, 31, 32 and 33 present the dependence of the upper bound on the log(MPE) on time for MPLPS for the WCSP instances. The results for MM-MBE with different partitioning heuristics are also plotted as a point (since MBE algorithms are not iterative and the results do not change with the time). Note that in this figures we plot the results for MBE-MM with i-Bound=10. Some of the results for MBE-MM coincide with each other and overlap on the plot.

4 Future work

Since the main goal of this evaluation was to access the performances of schemes as a possible alternative as a heuristics generator to the currently used Mini-bucket Elimination algorithm, further, more realistic, comparison of the run times is desired. We need to know whether the additional time overhead introduced by new heuristic procedure would be mitigated by better bounds.

Since it was shown that moment-matching always improves the bounds outputted by MBE with little time overhead, it might me desirable to add this option to our existing mbe-based algorithm, such as mbe – m – opt and MBE used as a part of AOBB algorithm. Currently the implementation of mbe-m-opt-MM is work in progress, for the purpose of evaluating of the improvement in the quality of the m bounds.

The MPLP can give rise to a new version of Branch and Bound algorithm with the dynamic ordering, in which the heuristic is recalculated after each new variable instantiations. Unlike our previous attempts at dynamic ordering schemes, where the heuristics were re-calculated from scratch and thus introduced
Figure 24: Weighted CSPs, instances 1502, 29 and 404, upper bound on the log(MPE) as a function of number of iterations of the MBE-MM, MPLPS and $mp - MPLP$. For MBE-MM the results are presented for the best among content-based partitioning heuristics with moment matching for i-Bound=10. $mp - MPLP$ and MPLPS are ran on the original factors.

Figure 25: Weighted CSPs, instances 1502, 29 runtime (sec) as a function of number of iterations of the MBE-MM, MPLPS and $mp - MPLP$. For MBE-MM the results are presented for the best among content-based partitioning heuristics with moment matching for i-Bound=10. $mp - MPLP$ and MPLPS are ran on the original factors. We don’t include the results for instance 404 for scale reasons, but it displays similar behaviour.
Figure 26: 54, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.

Figure 27: 505, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.
Figure 28: 503, log(MPE) as a function of time for the $MPLPS$. For $MBE-MM$ the results are presented for each partitioning heuristic as a point on the graph. $i$-Bound=10. $MPLPS$ is ran on the original factors.

Figure 29: 42, log(MPE) as a function of time for the $MPLPS$. For $MBE-MM$ the results are presented for each partitioning heuristic as a point on the graph. $i$-Bound=10. $MPLPS$ is ran on the original factors.
Figure 30: 408, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.

Figure 31: 404, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors. The increase of the upper bound outputted by MPLP at about 30 seconds mark is curious and can’t be presently explained.
Figure 32: 1502, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.

Figure 33: 29, log(MPE) as a function of time for the MPLPS. For MBE-MM the results are presented for each partitioning heuristic as a point on the graph. i-Bound=10. MPLPS is ran on the original factors.
significant time overhead, *MPLP* can re-use the previously calculated messages to allow for cheap (timewise) updates for instantiations.

**References**


