Preliminary Empirical Evaluation of Anytime Weighted AND/OR Best-First Search for MAP

Natalia Flerova
University of California Irvine

Radu Marinescu
IBM Research - Ireland

Rina Dechter
University of California Irvine

Abstract

We explore the potential of anytime best-first search schemes for combinatorial optimization tasks over graphical models (e.g., MAP/MPE). We show that recent advances in extending best-first search into an anytime scheme have a potential for optimization for graphical models. Importantly, these schemes come with upper bound guarantees and are sometime competitive with known effective anytime branch-and-bound schemes.

1 Introduction

The most common search algorithms for combinatorial optimization tasks over graphical models such as MAP/MPE is depth-first branch and bound. Such schemes were extensively studied in recent years for generating exact and approximate solutions [1, 2]. Indeed, best-first search schemes, while known to be more efficient [3], are rarely considered for graphical models because they are inherently memory intensive and lack anytime qualities. Furthermore, one of best-first most attractive qualites, avoiding the exploration of unbounded paths, seems irrelevant to graphical models where all solutions are at the same depth (i.e., the number of variable).

In contrast, for path-finding domains, such as planning, best-first search, and especially its popular variant A* [4], is most attractive and received most of the research focus. Given an admissible heuristic, A* is guaranteed to find the exact solution and is optimally efficient with respect to the number of expanded nodes [3]. In particular, it is guaranteed to explore only nodes whose evaluation function is bounded by the optimal solution cost and thus will never wonder in unbounded directions.

However, A*'s exponential memory requirements and its solution generation only upon termination, are problematic for use in the path-finding domain as well, and therefore proposals extending A* into a more flexible anytime scheme have recently emerged. The most popular approaches are based on *Weighted A** (WA*) [5], which suggest to inflate the heuristic values by a constant factor of $w \geq 1$. This makes the heuristic inadmissible to a degree which is controlled by w, but typically yields faster search and guarantees a solution cost within a factor of w from the optimal one. If the approximate solution is found quickly and if additional time is available, search for a better solution resumes. The simplest anytime scheme is to run A* iteratively with decreasing values of w until a time-bound, returning the best solution found thus far, with the corresponding weight as the bounding factor, or until $w = 1$, returning the optimal solution.

Several anytime weighted best-first search algorithms were proposed ([6, 7, 8, 9]). The two that are perceived to be most effective are the Anytime Repairing A* (ARA*) [7] and the Anytime Nonparametric A* (ANA*) [8]. ARA* runs WA* iteratively while decreasing w but tries to reuse search efforts from previous iterations. Algorithm ANA* implicitly regulates the weights based on the cost of the current best solution.

In light of the above enhancement in anytime best-first search for path-finding benchmarks, we decided to give best-first another chance and explore these weighted best-first search schemes for graphical models. We used as a starting point the AND/OR best-first (AOBF) for graphical models which was investigated and compared with AND/OR branch-and-bound schemes (AOBB) [1]. AOBF searches the AND/OR context minimal graph [10] and is guided by admissible and consistent mini-bucket heuristic [11]. The simplest way to extend AOBF to an anytime scheme is to run AOBF with the weighted heuristic iteratively, decreasing the value of the weight at each iteration. We also extended ARA* to the AND/OR search space, while ANA* was extended to OR search spaces only.

Our empirical evaluation compares the above algorithms and contrasts them with a recent effective anytime branch-and-bound search for graphical models, called Breadth-Rotating AND/OR Branch-and-Bound (BRAOBB) [2] which won the 2011 Probabilistic Inference Challenge[1] in all optimization categories. Our preliminary results are quite promising, showing that our quick implementation of two of the weighted best-first schemes can sometime have a superior anytime performance to BRAOBB, thus providing an alternative that should be further developed. In section 2 and 3 we provide background and discuss our algorithms and in section 4 we provide empirical evaluation.

2 Best First and Weighted Best-First Heuristic Search

Best-first search Best-first search (BFS) always chooses for expansion a node with lowest value of $f(n)$. Its most popular variant is A*, which uses $f(n) = g(n)+h(n)$, where $g(n)$ is minimal cost from the root n_0 to n along the current path, and $h(n)$ is heuristic function that estimates $h^*(n)$, the optimal cost from n to a goal node. For minimization, the $h(n)$ is *admissible* if $\forall n\ h(n) \leq h^*(n)$ The heuristic function h is $consistent$ iff $\forall n'$ successor of n in G, $h(n) \leq c(n, n') + h(n')$.

Weighted A* search (WA*) [5] differs from A* only in using the evaluation function: $f(n) = g(n) + w \cdot h(n)$, weight $w > 1$, $h(n)$ is admissible. Higher values of w typically yield greedier behavior, where a solution is encountered earlier during search. WA* is guaranteed to terminate with a solution cost C such that $C \leq w \cdot C^*$, where C^* is the cost of the optimal solution path.

AND/OR best-first search for graphical models A *graphical model* is a tuple $\mathcal{M} = (\mathbf{X}, \mathbf{D}, \mathbf{F}, \prod)$, **F** is a set of real-valued local functions over subsets of discrete variables **X**, called scopes, with finite domains **D**. The common optimization task is to find $\max_{\mathbf{X}} \prod_i f_i$ called MAP. The scopes of **F** define a *primal graph* G with certain *induced width* and a pseudo tree $\mathcal{T}$ of G that guides an AND/OR search space [10]. An *AND/OR search tree* $\mathcal{S}_{\mathcal{T}}$ associated with $\mathcal{T}$ has alternating levels of OR nodes corresponding to the variables and AND nodes corresponding to the values of the OR parent's variable, with edges weighted according to **F**. The state of the art version of A* that explores the AND/OR search spaces is the AND/OR Best-First (AOBF) algorithm [12] that utilizes the mini-bucket heuristic which is admissible and consistent [11].

Proposing Weighted AOBF The weighted version of the AOBF algorithm can be obtained by inflating the heuristic function with a weight $w \geq 1$ (i.e., substituting $h(n)$ by $w \cdot h(n)$). *Weighted AOBF* is closely related to WAO*, an algorithm introduced previously by Chakrabarti et al. [13] for searching AND/OR search spaces having a restricted structure. It is easy to show that the properties of WA* [5] and WAO* [13] are shared by Weighted AOBF as well. Specifically:

Proposition 1. *If $h(n)$ is admissible, the cost of the solution discovered by Weighted AOBF is guaranteed to be no more than a factor of w worse than the optimal one.*

3 Anytime Heuristic Search

Proposing Anytime Weighted AOBF Since the accuracy of a solution found by Weighted AOBF is bounded by w, it is possible to formulate a simple anytime scheme, called **wAOBF**, that executes Weighted AOBF iteratively, starting with an initial weight, and decreasing the weight at each iteration by a fixed amount. Clearly, this approach, similar to the Restarting Weighted A* by Richter at al. [9], results in a series of solutions, each with a sub-optimality factor equal to the w.

Anytime Repairing AOBF Running each search iteration from scratch, as **wAOBF** does, is wasteful, since the same search subspace might be explored multiple times. To remedy this problem, we propose the Anytime Repairing AOBF scheme, denoted by **wR-AOBF**, which is a simple

[1]http://www.cs.huji.ac.il/project/PASCAL/realBoard.php

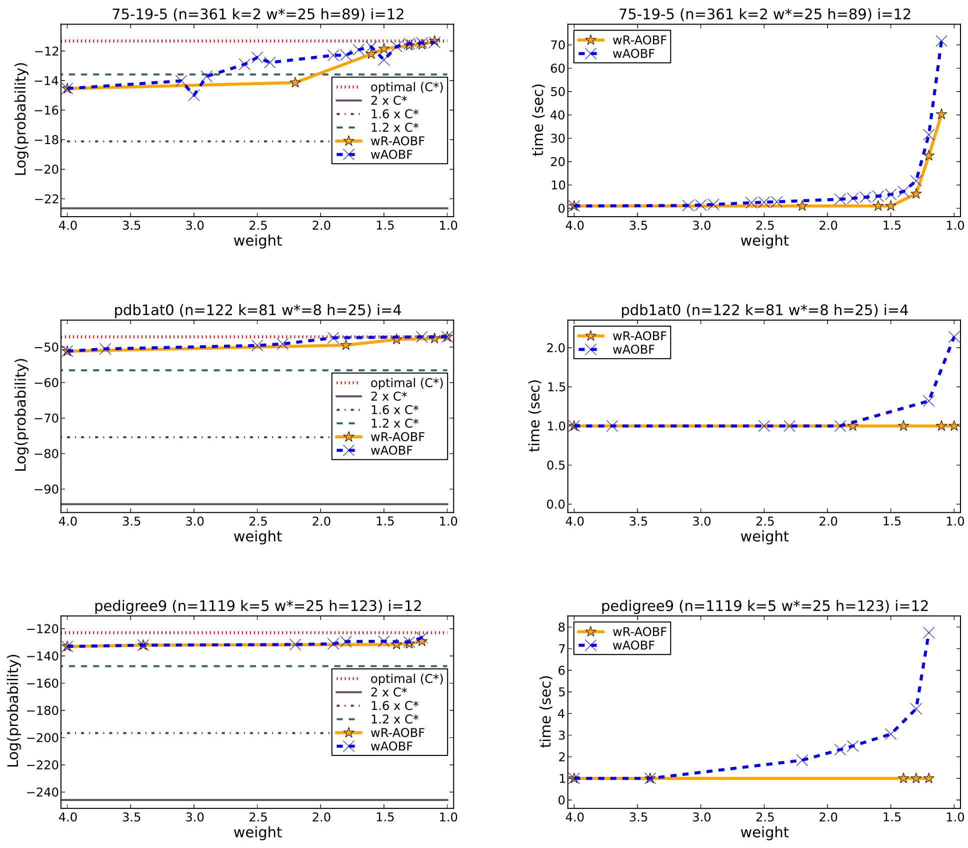

Figure 1: Solution cost and time vs weight (wR-OBF, wAOBF). Starting weight=4. C^* - exact cost, w x C^* - the theoretical bound on the approximate solution for weight w, i - i-bound, n - number of variables, k - domain size, w^* - induced width, h - pseudo-tree height.

extension of the *Anytime Repairing A* (ARA*)* algorithm [7] to AND/OR search spaces. At each iteration, **wR-AOBF** keeps track of the partially explored AND/OR graph and, after decreasing w, it performs a bottom-up update of all node values starting from the leaf nodes (whose h-values are inflated with the new weight) and continuing upwards towards the root node. Then, the search continues with the newly identified best partial solution tree. As well as ARA*, **wR-AOBF** provides the same guarantees with respect to the quality of the suboptimal solutions found. We note that a more recent anytime scheme called *Anytime Nonparametric A* (ANA*)* [8], which regulates automatically the weight based on the cost of the current best solution, also has an extention to AND/OR search spaces for graphical models, wN-AOBF. However, our preliminary evaluation, omitted for space reasons, showed that this scheme does not perform well on our benchmarks and thus wN-AOBF is excluded from further discussion.

Anytime AND/OR branch-and-bound Depth-first AND/OR branch-and-bound (AOBB) [12] is a powerful search scheme for graphical models. The algorithm, however, lacks a proper anytime behavior because at each AND node all but one independent child subproblems have to be solved completely, before the last one is even considered. *Breadth-Rotating AND/OR Branch and Bound* (BRAOBB) [2] remedies this deficiency of AOBB by combining depth-first exploration of the search space with the notion of rotating through different subproblems in a breadth-first manner. Empirically, BRAOBB outputs the first suboptimal solutions significantly faster than plain AOBB [2].

4 Experiments

We compare the anytime behaviour of our schemes using common variable orderings and Mini-Bucket heuristics [11], whose strength is controlled by a parameter *i-bound* (higher i-bounds typically yield more accurate heuristics). We solve the MPE task, the algorithms output lower bounds on

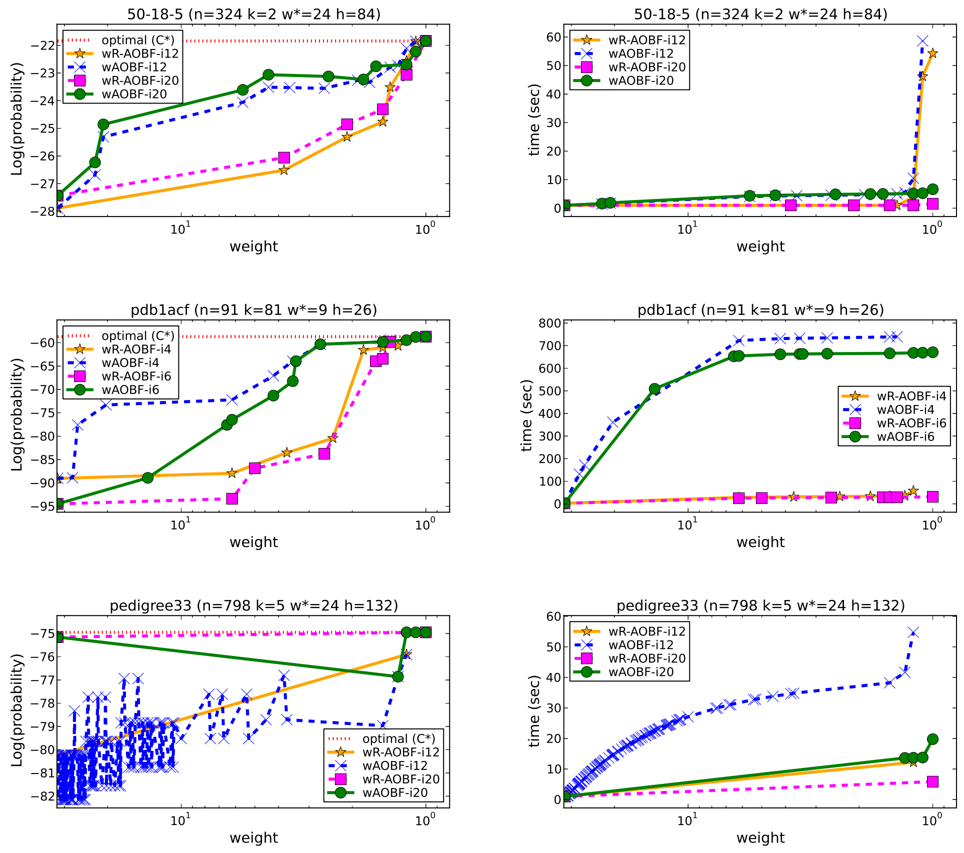

Figure 2: Solution cost, time vs weight (wR-OBF, wAOBF), starting weight = 32. C^* - optimal cost, ”-i” - i-bound, n - number of variables, k - domain size, w^* - induced width, h - pseudo-tree height.

solutions, so higher values are preferable. We use 3 datasets: 16 pedigree networks, 17 binary grids and 20 protein instances. Time limit is 1 hour, memory limit is 2 Gb. Due to lack of space we chose a subset of instances from each dataset to demonstrate the typical performance of the algorithms.

The influence of weight on the performance of AND/OR weighted BF schemes. Fig. 1 shows the solution cost (left) and time to find the corresponding solution (right) as functions of weight for wAOBF and wR-AOBF. The starting weight is 4, decreasing by 0.1 at each iteration.

Theory claims that smaller values of weight yield more accurate solutions, since the cost of the solution C is provably within a factor of w from the exact one C^*. Plots in the left column of Fig. 1 show $C = w \cdot C^*$ for weights $w = \{1.2, 1.6, 2\}$. We see that the actual solutions found by both schemes are much better than theory suggests. Same is true for the all the omitted instances.

Though there are no theoretical guarantees, we expect higher weights to yield faster performances. Experiments show that in practice this intuition is correct for our benchmarks, illustrated by examples in the right column of Fig. 1. While the runtime of wAOBF always noticeably increases as weights get smaller, for some instances (e.g., pdb1at0, pedigree9) the runtime of wR-AOBF remains almost constant.

We observed that for wR-AOBF the solution cost always increases monotonically as weight decreases. But for wAOBF the accuracy of the solutions, though always bounded by $w \cdot C^*$, can either improve or get worse as weight reduces (e.g., 70-19-5).

Such cost fluctuations are even more noticeable on the pedigree33 plot in Fig. 2. The figure displays solution accuracy (left) and runtime (right) for a larger range of w, starting at 32. For pedigrees we observed these cost oscillations for 9 instances out of 16, while for grids and proteins they are rare and cost most often changes monotonically with weight. At this point it is unclear what causes this difference in the behaviour between benchmarks.

As a rule, higher values of the i-bound yield more accurate heuristics and thus better solutions, but for some instances high i-bounds correspond to worse results, e.g. compare wAOBF with i=4 and i=6 for pdb1acf. Although in our experiments the starting weight and the decrease interval were chosen arbitrary, it is obvious that the solution quality is influenced by the relation between the i-bound value and weight, which can potentially be utilized to pick optimal weight parameters.

Anytime profiles of weighted AOBF schemes. Consider the left column of Fig. 3. It shows the cost of the best solution found by wAOBF and wR-AOBF algorithms as a function of time for teo i-bounds, corresponding to medium and high accuracy of the heuristic.

The plots use three problems to illustrate the recurring trends observed across all instances. For a given time wR-AOBF finds solutions of better quality than wAOBF. For both schemes larger values of i-bound normally yield more accurate performance (e.g. 50-20-5, pedigree33), however for a few instances the opposite is true (e.g. for pdb1acf wAOBF performs worse for $i = 6$ than for $i = 4$).

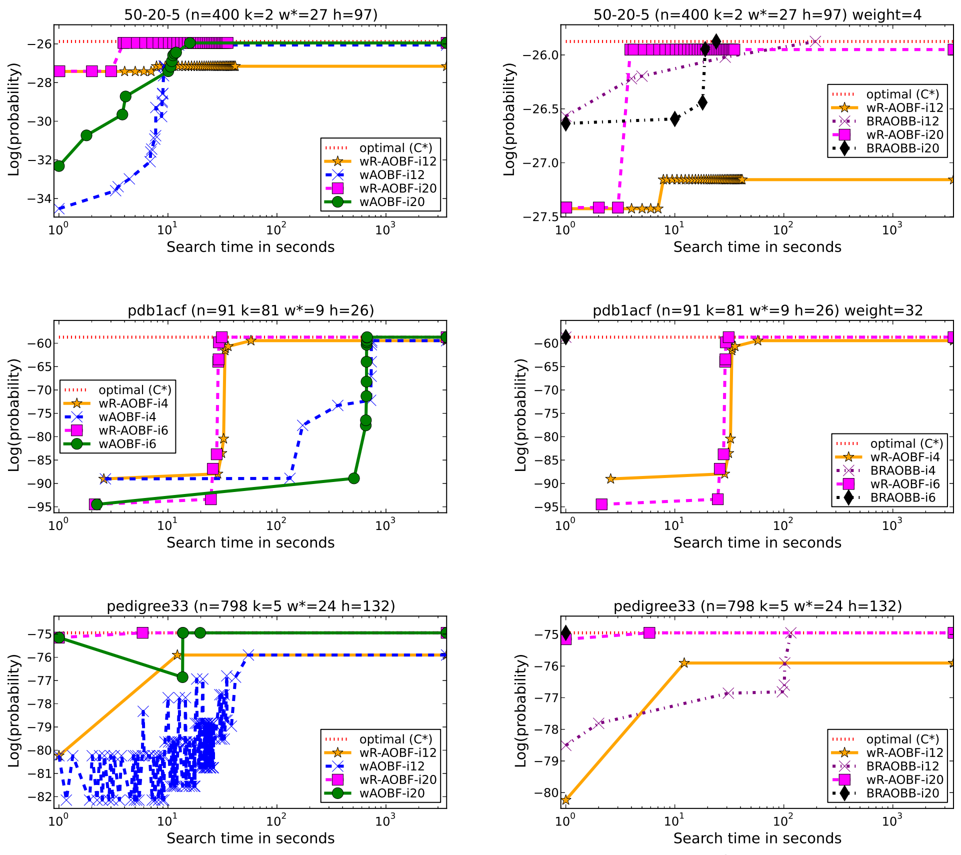

Figure 3: Solution cost vs time (sec); right: wR-OBF & wAOBF; left: wR-OBF & BRAOBB.

Anytime weighted BF vs BB We compare the more efficient one of the BF schemes, wR-AOBF, to the state-of-the art BRAOBB. Consider the right column of Fig. 3 showing the dependence of the solution cost on time. We observed that for certain instances (here represented by pdb1acf) BRAOBB is faster and finds solutions of better quality than wR-AOBF. However, for some problems (such as 50-20-5, $i = 12$) wR-AOBF produces better solutions for some time cut-offs.

Table 1 compares the performance of all three schemes for 2 i-bounds on a subset of instances, showing the best solution found for various time cut-offs. wAOBF is typically dominated by wR-AOBF, with a few exceptions, such as 50-19-5, $i = 12$. BRAOBB is the fastest to find the exact solution on the majority of instances, but fails to produce any results for problems known to be hard, such as protein pdb1a7w, $i = 6$.

Table 2 summarizes the experiments, providing for several time cut-offs the number of problems, for which within the respective time bound any solution was found, the optimal solution was found and

Instance (n, k, w^*, h)	Algorithms	Optimal cost	Time bound							
			10 sec	1 min	10 min	1 hour	10 sec	1 min	10 min	1 hour
Pedigree networks			i-bound=12				i-bound=20			
pedigree7 (1069, 5, 47, 204)	wR-AOBF	-113.889	-118.071	-118.071	-118.071	-118.071	-119.22	-119.22	-119.22	-119.22
	BRAOBB	-113.889	-124.207	-121.344	-118.745	-118.745	-116.683	-116.683	**-113.889**	**-113.889**
	wAOBF	-113.889	-125.438	-117.117	-117.813	-117.813	-120.103	-114.042	-114.042	-114.042
pedigree9 (1119, 5, 25, 123)	wR-AOBF	-122.904	-129.225	-129.225	-129.225	-129.225	-128.037	-128.037	-128.037	-128.037
	BRAOBB	-122.904	-126.13	-124.142	-123.858	-123.858	**-122.904**	**-122.904**	**-122.904**	**-122.904**
	wAOBF	-122.904	-136.471	-126.529	-126.529	-126.529	-133.254	-124.547	-124.547	-124.547
pedigree31 (1184, 5, 29, 131)	wR-AOBF	-130.461	-136.967	-136.967	-136.967	-136.967	fail	fail	fail	fail
	BRAOBB	-130.461	-141.085	-140.622	-133.977	-131.622	fail	fail	fail	fail
	wAOBF	-130.461	-138.969	-135.701	-135.08	-135.08	fail	fail	fail	fail
pedigree41 (1063, 5, 29, 119)	wR-AOBF	-120.735	-124.872	-124.872	-124.872	-124.872	-121.792	-121.792	-121.792	-121.792
	BRAOBB	-120.735	-125.347	-123.276	-122.614	-120.735	fail	fail	fail	fail
	wAOBF	-120.735	-133.549	-133.549	-126.774	-126.774	-130.7	-130.7	-120.916	-120.916
Grid networks			i-bound=12				i-bound=20			
50-19-5 (361, 2, 25, 93)	wR-AOBF	-23.2894	-24.9678	-24.4856	-24.4856	-24.4856	-23.8008	**-23.2894**	**-23.2894**	**-23.2894**
	BRAOBB	-23.2894	-23.659	-23.3556	-23.3556	**-23.2894**	**-23.2894**	**-23.2894**	**-23.2894**	**-23.2894**
	wAOBF	-23.2894	-24.3493	-23.3495	-23.3495	-23.3495	-23.7389	**-23.2894**	**-23.2894**	**-23.2894**
90-21-5 (441, 2, 28, 106)	wR-AOBF	-7.65823	-8.23285	**-7.65823**	**-7.65823**	**-7.65823**	**-7.65823**	**-7.65823**	**-7.65823**	**-7.65823**
	BRAOBB	-7.65823	**-7.65823**	**-7.65823**	**-7.65823**	**-7.65823**	**-7.65823**	**-7.6582**3	**-7.65823**	**-7.65823**
	wAOBF	-7.65823	-8.59967	-8.59967	**-7.65823**	**-7.65823**	-9.47549	-9.47549	**-7.65823**	**-7.65823**
90-22-5 (484, 2, 30, 109)	wR-AOBF	-6.62929	-9.34089	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**
	BRAOBB	-6.62929	-6.94762	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**	**-6.62929**
	wAOBF	-6.62929	-9.34089	-9.34089	**-6.62929**	**-6.62929**	-7.29493	-7.29493	**-6.62929**	**-6.62929**
90-23-5 (529, 2, 31, 116)	wR-AOBF	-8.37526	-9.64825	-8.50184	-8.50184	-8.50184	-9.00014	**-8.37526**	**-8.37526**	**-8.37526**
	BRAOBB	-8.37526	-8.93201	-8.93201	**-8.37526**	**-8.37526**	**-8.37526**	**-8.37526**	**-8.37526**	**-8.37526**
	wAOBF	-8.37526	-9.64825	-9.64825	-8.40246	-8.40246	-10.2926	-10.2926	**-8.37526**	**-8.37526**
Protein networks			i-bound=4				i-bound=6			
pdb1a62 (106, 81, 10, 31)	wR-AOBF	-48.3581	**-48.3581**	**-48.3581**	**-48.3581**	**-48.3581**	**-48.3581**	**-48.3581**	**-48.3581**	**-48.3581**
	BRAOBB	-48.3581	**-48.3581**	**-48.3581**	**-48.358**1	**-48.3581**	fail	fail	fail	fail
	wAOBF	-48.3581	-91.7971	**-48.3581**	**-48.3581**	**-48.3581**	-75.5978	**-48.3581**	**-48.3581**	**-48.3581**
pdb1a7w (53, 81, 6, 25)	wR-AOBF	-14.228	-15.7759	-15.7759	**-14.228**	**-14.228**	fail	fail	fail	fai
	BRAOBB	-14.228	**-14.228**	**-14.228**	**-14.228**	**-14.228**	fail	fail	fail	fai
	wAOBF	-14.228	-15.7759	-15.7759	-15.7759	**-14.228**	fail	fail	fail	fail
pdb1aba (77, 81, 8, 30)	wR-AOBF	-31.6949	**-31.6949**	**-31.6949**	**-31.6949**	**-31.6949**	**-31.6949**	**-31.6949**	**-31.6949**	**-31.6949**
	BRAOBB	-31.6949	**-31.6949**	**-31.6949**	**-31.6949**	**-31.6949**	fail	fail	fail	fail
	wAOBF	-31.6949	-38.9925	**-31.6949**	**-31.6949**	**-31.6949**	-35.586	**-31.6949**	**-31.6949**	**-31.6949**
pdb1b0b (98, 81, 9, 29)	wR-AOBF	-89.3655	-107.495	-107.495	-106.091	-106.091	-103.537	-103.537	-103.537	-103.537
	BRAOBB	-89.3655	-90.5106	-89.5608	**-89.3655**	**-89.3655**	fail	fail	fail	fail
	wAOBF	-89.3655	-107.495	-107.495	-107.495	-95.6883	**-89.3655**	**-89.3655**	**-89.3655**	**-89.3655**
pdb1b0y (61, 81, 8, 18)	wR-AOBF	-35.6712	-41.7016	-35.8095	-35.8095	-35.8095	-61.8207	-45.3506	-45.3506	-45.3506
	BRAOBB	-35.6712	**-35.6712**	**-35.6712**	**-35.6712**	**-35.6712**	fail	fail	fail	fail
	wAOBF	-35.6712	-51.0568	-51.0568	**-35.6712**	**-35.6712**	-61.8207	-61.8207	-35.7726	-35.7726

Table 1: Best solution costs found within the time cut-off, exact solutions are shown in bold.

Algorithms	Time bound							
	10 sec	1 min	10 min	1 hour	10 sec	1 min	10 min	1 hour
Pedigree networks (16 total), $\hat{w} = 21.1$	i-bound=8				i-bound=20			
wAOBF	13 / 1 / 1	13 / 2 / 2	15 / 7 / 5	16 / 9 / 6	10 / 4 / 4	11 / 4 / 4	12 / 7 / 6	12 / 12 / 12
wR-AOBF	11 / 1 / 1	13 / 3 / 3	13 / 6 / 6	16 / 6 / 6	11 / 6 / 6	12 / 6 / 6	12 / 7 / 7	12 / 12 / 12
BRAOBB	13 / 6 / 4	13 / 6 / 5	14 / 9 / 8	14 / 11 / 9	3 / 3 / 3	8 / 6 / 6	10 / 8 / 8	11 / 9 / 9
Grid networks (17 total),$\hat{w} = 22.6$	i-bound=8				i-bound=20			
wAOBF	15 / 2 / 1	17 / 4 / 3	17 / 8 / 6	17 / 10 /10	17 / 9 / 9	17 / 12 / 12	17 / 16 / 16	17 / 16 / 16
wR-AOBF	10 / 3 / 3	13 / 5 / 4	16 / 11 / 11	17 / 11 / 11	17 / 13 / 13	17 / 16 / 15	17 / 16 / 16	17 / 16 / 16
BRAOBB	14 / 8 / 5	15 / 13 / 10	16 / 14 / 14	16 / 15 / 15	11 / 10 / 10	17 / 16 / 16	17 / 17 / 17	17 / 17 / 17
Protein networks (20 total),$\hat{w} = 9.8$	i-bound=2				i-bound=6			
wAOBF	9 / 0 / 0	10 / 1 / 1	11 / 4 / 3	20 / 9 / 20	14 / 6 / 4	14 / 10 / 8	14 / 12 / 11	14 / 11 / 10
wR-AOBF	5 / 3 / 2	9 / 6 / 5	10 / 9 / 7	20 / 10 / 20	11 / 9 / 8	14 / 12 / 11	14 / 13 / 12	14 / 13 / 13
BRAOBB	20 / 10 / 9	20 / 14 / 12	20 / 14 / 13	20 / 14 / 14	6 / 5 / 4	14 / 14 / 11	20 / 19 / 17	20 / 20 / 18

Table 2: Statistics over 53 instances for each scheme for a fixed i-bound: the number of cases for which, within the respective time bound, (1) any solution was found, (2) the optimal solution was found, (3) optimality was proven, $\hat{w}$ - average tree-width for benchmark.

optimality was proven. We see that it is hard to pinpoint a clear winner among the three schemes, also the performance is greately influenced by the i-bounds.

To summarize, we see that the reuse of the explored search space by wR-AOBF yields a more efficient scheme than the simpler wAOBF algorithm, and, with a few exceptions, both BF methods perform worse than the state-of-the-art BRAOBB algorithm.

5 Conclusion

Best First search schemes are often neglected in the context of solving combinatorial optimization problems over graphical models due to their exponential memory requirements. Yet, this paper demonstrates the potential of the anytime Best-First search for finding approximated solutions.

We conducted an extensive empirical evaluation of several anytime best-first schemes, and showed that they have potential for providing an alternative for anytime branch-and-bound schemes for some benchmarks especially for difficult problem instances.

Acknowledgement

This work was supported by NSF grant IIS-1065618.

References

[1] R. Marinescu and R. Dechter. Memory intensive AND/OR search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1492–1524, 2009.

[2] L. Otten and R. Dechter. Anytime AND/OR depth first search for combinatorial optimization. In *SOCS*, 2011.

[3] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *Journal of the ACM*, 32:506–536, 1985.

[4] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[5] I. Pohl. Heuristic search viewed as path finding in a graph. *Artif. Intell.*, 1(3-4):193–204, 1970.

[6] E.A. Hansen and R. Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28(1):267–297, 2007.

[7] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. *NIPS*, 16, 2003.

[8] J. van den Berg, R. Shah, A. Huang, and K. Goldberg. Anytime nonparametric A*. In *AAAI*, 2011.

[9] S. Richter, J.T. Thayer, and W. Ruml. The joy of forgetting: Faster anytime search via restarting. In *ICAPS*, pages 137–144, 2010.

[10] R. Dechter and R. Mateescu. AND/OR search spaces for graphical models. *Artificial Intelligence*, 171(2-3):73–106, 2007.

[11] R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, 50(2):107–153, 2003.

[12] R. Marinescu and R. Dechter. AND/OR Branch-and-Bound search for combinatorial optimization in graphical models. *Artificial Intelligence*, 173(16-17):1457–1491, 2009.

[13] P.P. Chakrabarti, S. Ghose, and SC De Sarkar. Admissibility of AO* when heuristics overestimate. *Artificial Intelligence*, 34(1):97–113, 1987.