

Anytime Argumentative and Abductive Reasoning*

Rolf Haenni

Computer Science Department, University of California, Los Angeles, CA 90095
e-mail: rolf.haenni@gmx.net, web site: <http://haenni.shorturl.com>

Received: date / Revised version: date

Abstract This article presents a new approximation method for computing arguments or explanations in the context of logic-based argumentative or abductive reasoning. The algorithm can be interrupted at any time returning the solution found so far. The quality of the approximation increases monotonically when more computational resources are available. The method is based on cost functions and returns lower and upper bounds.

1 Introduction

The major drawback of most qualitative approaches to uncertainty management comes from their relatively high time and space consuming algorithms. To overcome this difficulty, appropriate approximation methods are needed. In the domain of argumentative and abductive reasoning, a technique called *cost-bounded approximation* has been developed for probabilistic argumentation systems [16–18]. Instead of computing intractably large sets of minimal arguments for a given query, the idea is that only the most relevant arguments not exceeding a certain *cost bound* are computed. This is extremely useful and has many applications in different fields [1, 3]. In model-based diagnostics, for example, computing arguments corresponds to determining minimal conflict sets and minimal diagnoses. Very often, intractably many conflict sets and diagnoses exist. The method presented in [16] is an elegant solution for such difficult cases. However, the question of

* Research supported by scholarship No. 8220-061232 of the Swiss National Science Foundation.

Correspondence to: Rolf Haenni, Schützenstrasse 10, CH-8280 Kreuzlingen, Switzerland

choosing appropriate cost bounds and the problem of judging the quality of the approximation remain.

The approach presented in this article is based on the same idea of computing only the most relevant arguments. However, instead of choosing the cost bound first and then computing the corresponding arguments, the algorithm starts immediately by computing the most relevant arguments. It terminates as soon as no more computational resources (time or space) are available and returns the cost bound reached during its run. The result is a *lower approximation* that is sound but not complete. Furthermore, the algorithm returns an *upper approximation* that is complete but not sound. The difference between lower and upper approximation allows the user to judge the quality of the approximation. The algorithm is designed such that the cost bound (and therefore the quality of the approximation) increases monotonically when more resources are available. It can therefore be considered as an *anytime algorithm* that provides a result at any time. Note that this is very similar to the natural process of how people collect information from their environment. In legal cases, for example, resources (time, money, etc.) are limited, and the investigation focusses therefore on the search of the most relevant and most obvious evidence and such that the corresponding costs remain reasonable.

Another important property of the algorithm is the fact that the actual query is taken into account during its run. This ensures that those arguments which are of particular importance for the user's actual query are returned first. Such a *query-driven* behavior corresponds to the natural way of how human gathers the relevant information from different sources.

2 Probabilistic Argumentation Systems

The theory of *probabilistic argumentation systems* is based on the idea of combining classical logic with probability theory [17, 15]. It is an alternative approach for non-monotonic reasoning under uncertainty. It allows to judge open questions (hypotheses) about the unknown or future world in the light of the given knowledge. From a qualitative point of view, the problem is to find *arguments* in favor and against the hypothesis of interest. An argument can be seen as a defeasible proof for the hypothesis. It can be defeated by counter-arguments. The strength of an argument is weighted by considering its probability. In this way, the credibility of a hypothesis can be measured by the total probability that it is supported or rejected by such arguments. The resulting *degree of support* and *degree of possibility* correspond to (normalized) belief and plausibility in Dempster-Shafer's theory of evidence [25, 28, 20]. A quantitative judgement is sometimes more useful and can help to decide whether a hypothesis should be accepted, rejected, or whether the available knowledge does not permit to decide.

The technique of probabilistic argumentation systems generalizes de Kleer's and Reiter's original concept of *assumption-based truth maintenance*

systems (ATMS) [6–8, 24, 19, 10] by (1) removing the restriction to Horn clauses and (2) by adding probabilities in a similar way as Provan [23] or Laskey and Lehner [2]. Approximating techniques for intractably large sets of arguments has been proposed for ATMS by Forbus and de Kleer [14, 9], by Collins and de Coste [5], and by Bigham et al. [4].

For the construction of a probabilistic (propositional) argumentation system, consider two disjoint sets $A = \{a_1, \dots, a_m\}$ and $P = \{p_1, \dots, p_n\}$ of propositions. The elements of A are called *assumptions*. \mathcal{L}_{AUP} denotes the corresponding propositional language. If ξ is an arbitrary propositional sentence in \mathcal{L}_{AUP} , then a triple (ξ, P, A) is called (*propositional*) *argumentation system*. ξ is called *knowledge base* and is often specified by a conjunctively interpreted set $\Sigma = \{\xi_1, \dots, \xi_r\}$ of sentences $\xi_i \in \mathcal{L}_{AUP}$ or, more specifically, clauses $\xi_i \in \mathcal{D}_{AUP}$, where \mathcal{D}_{AUP} denotes the set of all (proper) clauses over $A \cup P$. We use $Props(\xi) \subseteq A \cup P$ to denote all the propositions appearing in ξ .

The assumptions play an important role for expressing uncertain information. They are used to represent uncertain events, unknown circumstances and risks, or possible outcomes. Conjunctions of literals of assumptions are of particular interest. They represent possible scenarios or states of the unknown or future world. \mathcal{C}_A denotes the set of all such conjunctions. Furthermore, $N_A = \{0, 1\}^{|A|}$ represents the set of all possible configurations relative to A . The elements $s \in N_A$ are called *scenarios*. The theory is based on the idea that one particular scenario $\hat{s} \in N_A$ is the *true* scenario.

Consider now the case where a second propositional sentence $h \in \mathcal{L}_{AUP}$ called *hypothesis* is given. Hypotheses represent open questions or uncertain statements about some of the propositions in $A \cup P$. What can be inferred from ξ about the possible truth of h with respect to the given set of unknown assumptions? Possibly, if some of the assumptions are set to *true* and others to *false*, then h may be a logical consequence of ξ . In other words, h is *supported* by certain scenarios $s \in N_A$ or corresponding *arguments* $\alpha \in \mathcal{C}_A$. Note that *counter-arguments* refuting h are arguments supporting $\neg h$.

More formally, let $\xi_{\leftarrow s}$ be the formula obtained from ξ by instantiating all the assumptions according to their values in s . We can then decompose the set of scenarios N_A into three disjoint sets

$$I_A(\xi) = \{s \in N_A : \xi_{\leftarrow s} \models \perp\}, \quad (1)$$

$$SP_A(h, \xi) = \{s \in N_A : \xi_{\leftarrow s} \models h, \xi_{\leftarrow s} \not\models \perp\}, \quad (2)$$

$$RF_A(h, \xi) = \{s \in N_A : \xi_{\leftarrow s} \models \neg h, \xi_{\leftarrow s} \not\models \perp\} = SP_A(\neg h, \xi), \quad (3)$$

of *inconsistent*, *supporting*, and *refuting scenarios*, respectively. Furthermore, if $N_A(\alpha) \subseteq N_A$ denotes the set of models of a conjunction $\alpha \in \mathcal{C}_A$, then we can define corresponding sets of *supporting* and *refuting arguments* of h relative to ξ by

$$SP(h, \xi) = \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq SP_A(h, \xi)\}, \quad (4)$$

$$RF(h, \xi) = \{\alpha \in \mathcal{C}_A : N_A(\alpha) \subseteq RF_A(h, \xi)\}, \quad (5)$$

respectively. Often, since $SP(h, \xi)$ and $RF(h, \xi)$ are *upward-closed* sets, only corresponding *minimal* arguments are considered.

So far, hypotheses are only judged qualitatively. A quantitative judgment of the situation becomes possible if every assumption $a_i \in A$ is linked to a corresponding *prior probability* $p(a_i) = \pi_i$. Let $\Pi = \{\pi_1, \dots, \pi_m\}$ denote the set of all prior probabilities. We suppose that the assumptions are mutually independent. This defines a probability distribution $p(s)$ over the set N_A of scenarios¹. Note that independent assumptions are common in many practical applications [1]. A quadruple (ξ, P, A, Π) is then called *probabilistic argumentation system* [17].

In order to judge h quantitatively, consider the conditional probability that the true scenario \hat{s} is in $SP_A(h, \xi)$ but not in $I_A(\xi)$. In the light of this remark,

$$dsp(h, \xi) = p(\hat{s} \in SP_A(h, \xi) \mid \hat{s} \notin I_A(\xi)) \quad (6)$$

is called *degree of support* of h relative to ξ . It is a value between 0 and 1 that represents quantitatively the support that h is true in the light of the given knowledge. Clearly, $dsp(h, \xi) = 1$ means that h is certainly true, while $dsp(h, \xi) = 0$ means that h is not supported (but h may still be true). Note that degree of support is equivalent to the notion of (normalized) *belief* in the Dempster-Shafer theory of evidence [25, 28]. It can also be interpreted as the probability of the provability of h [22, 27].

A second way of judging the hypothesis h is to look at the corresponding conditional probability that the true scenario \hat{s} is not in $RF_A(h, \xi)$. It represents the probability that $\neg h$ can not be inferred from the knowledge base. In such a case, h remains *possible*. Therefore, the conditional probability

$$dps(h, \xi) = p(\hat{s} \notin RF_A(h, \xi) \mid \hat{s} \notin I_A(\xi)) = 1 - dsp(\neg h, \xi) \quad (7)$$

is called *degree of possibility* of h relative to ξ . It is a value between 0 and 1 that represents quantitatively the possibility that h is true in the light of the given knowledge. Clearly, $dps(h, \xi) = 1$ means that h is completely possible (there are no counter-arguments against h), while $dps(h, \xi) = 0$ means that h is false. Degree of possibility is equivalent to the notion of *plausibility* in the Dempster-Shafer theory. We have $dsp(h, \xi) \leq dps(h, \xi)$ for all $h \in \mathcal{L}_{A \cup P}$ and $\xi \in \mathcal{L}_{A \cup P}$. Note that the particular case of $dsp(h, \xi) = 0$ and $dps(h, \xi) = 1$ represents total ignorance relative to h .

An important property of degree of support and degree of possibility is that they behave non-monotonically when new information is added. More precisely, if ξ' represents a new piece of information, then nothing can be said about the new values $dsp(h, \xi \wedge \xi')$ and $dps(h, \xi \wedge \xi')$. Compared to $dsp(h, \xi)$ and $dps(h, \xi)$, the new values may either decrease or increase, both cases are possible. This reflects a natural property of how a human's conviction or belief can change when new information is given. Non-monotonicity is

¹ In cases where no set of independent assumptions exists, the theory may also be defined on an arbitrary probability distribution over N_A .

therefore a fundamental property for any mathematical formalism for reasoning under uncertainty. Probabilistic argumentation systems show that non-monotonicity can be achieved in classical logic by adding probability theory in an appropriate way. This has already been noted by Mary McLeish in [21].

3 Computing Arguments

From a computational point of view, the main problem of dealing with probabilistic argumentation systems is to compute the set $\mu QS(h, \xi)$ of minimal *quasi-supporting* arguments with $QS(h, \xi) = \{\alpha \in \mathcal{C}_A : \alpha \wedge \xi \models h\}$. The term “quasi” expresses the fact that some quasi-supporting arguments of h may be in contradiction with the given knowledge. Knowing the sets $\mu QS(h, \xi)$, $\mu QS(\neg h, \xi)$, and $\mu QS(\perp, \xi)$ allows then to derive supporting and refuting arguments, as well as degree of support and degree of possibility [17]. We use

$$QS_A(h, \xi) = N_A(QS(h, \xi)) = \{s \in N_A : \xi_{\neg s} \models h\} \quad (8)$$

to denote corresponding sets of quasi-supporting scenarios.

Suppose that the knowledge base $\xi \in \mathcal{L}_{AUP}$ is given as a set of clauses $\Sigma = \{\xi_1, \dots, \xi_r\}$ with $\xi_i \in \mathcal{D}_{AUP}$ and $\xi = \xi_1 \wedge \dots \wedge \xi_r$. Furthermore, let $H \subseteq \mathcal{D}_{AUP}$ be another set of clauses such that $\wedge H \equiv \neg h$. $\Sigma_H = \mu(\Sigma \cup H)$ denotes then the corresponding minimal clause representation of $\xi \wedge \neg h$ obtained from $\Sigma \cup H$ by dropping subsumed clauses.

3.1 Exact Computation

The problem of computing minimal quasi-supporting arguments is closely related to the problem of computing prime implicants. Quasi-supporting arguments for h are conjunctions $\alpha \in \mathcal{C}_A$ for which $\alpha \wedge \xi \models h$ holds. This condition can be rewritten as $\xi \vee \neg h \models \neg \alpha$ or $\Sigma_H \models \neg \alpha$, respectively. Quasi-supporting arguments are therefore negations of implicates of Σ_H which are in \mathcal{D}_A . In other words, if $\delta \in \mathcal{D}_A$ is an implicate of Σ_H , then $\neg \delta$ is a quasi-supporting argument for h . We use $PI(\Sigma_H)$ to denote the set of all prime implicates of Σ_H . If $\neg \Psi$ is the set of conjunctions obtained from a set of clauses Ψ by negating the corresponding clauses, then

$$\mu QS(h, \xi) = \neg(PI(\Sigma_H) \cap \mathcal{D}_A). \quad (9)$$

Since computing prime implicates is known to be NP-complete in general, the above approach is only feasible when Σ_H is relatively small. However, when A is small enough, many prime implicates of Σ_H are not in \mathcal{D}_A and are therefore irrelevant for the minimal quasi-support. Such irrelevant prime

implicates can be avoided by the method described in [17]. The procedure is based on two operations

$$\text{Cons}_Q(\Sigma) = \text{Cons}_{x_1} \circ \dots \circ \text{Cons}_{x_q}(\Sigma), \quad (10)$$

$$\text{Elim}_Q(\Sigma) = \text{Elim}_{x_1} \circ \dots \circ \text{Elim}_{x_q}(\Sigma), \quad (11)$$

where Σ is an arbitrary set of clauses and $Q = \{x_1, \dots, x_q\}$ a subset of propositions appearing in Σ . Both operations repeatedly apply more specific operations $\text{Cons}_x(\Sigma)$ and $\text{Elim}_x(\Sigma)$, respectively, where x denotes a proposition in Q .

Let Σ_x denote the clauses of Σ containing x as a positive literal, $\Sigma_{\bar{x}}$ the clauses containing x as a negative literal, and $\Sigma_{\dot{x}}$ the clauses not containing x . Furthermore, if

$$\rho(\Sigma_x, \Sigma_{\bar{x}}) = \{\vartheta_1 \vee \vartheta_2 : x \vee \vartheta_1 \in \Sigma_x, \neg x \vee \vartheta_2 \in \Sigma_{\bar{x}}\} \quad (12)$$

denotes the set of all resolvents of Σ relative to x , then $\text{Cons}_x(\Sigma) = \mu(\Sigma \cup \rho(\Sigma_x, \Sigma_{\bar{x}}))$ and $\text{Elim}_x(\Sigma) = \mu(\Sigma_{\dot{x}} \cup \rho(\Sigma_x, \Sigma_{\bar{x}}))$.

Thus, $\text{Cons}_Q(\Sigma)$ computes all the resolvents (consequences) of Σ relative to the propositions in Q and adds them to Σ . Note that if Q contains all the proposition in Σ , then $\text{Cons}_Q(\Sigma) = \text{PI}(\Sigma)$. In contrast, $\text{Elim}_Q(\Sigma)$ eliminates all the propositions in Q from Σ and returns a new set of clauses whose set of models corresponds to the projection of the original set of models. Note that from a theoretical point of view, the order in which the propositions are eliminated is irrelevant [17], whereas from a practical point of view, it critically influences the efficiency of the procedure. Note that the elimination process is a particular case of Shenoy's fusion algorithm [26] as well as of Dechter's bucket elimination procedure [13].

The set of the minimal quasi-supporting arguments can then be computed in two different ways by

$$\mu QS(h, \xi) = \neg \text{Cons}_A(\text{Elim}_P(\Sigma_H)) = \neg \text{Elim}_P(\text{Cons}_A(\Sigma_H)). \quad (13)$$

Note that in many practical applications, computing the consequences relative to the propositions in A is trivial. In contrast, the elimination of the propositions in P is usually much more difficult and becomes even infeasible as soon as Σ_H has a certain size.

3.2 Cost-Bounded Approximation

A possible approximation technique is based on *cost functions* $c : \mathcal{C}_A \rightarrow \mathbb{R}^+$. Conjunctions α with low costs $c(\alpha)$ are preferred and therefore more relevant. We require that $\alpha \subseteq \alpha'$ implies $c(\alpha) \leq c(\alpha')$. This condition is called *monotonicity criterion*. Examples of common cost functions are:

- the *length* of the conjunction (number of literals): $c(\alpha) = |\alpha|$,
- the *probability* of the negated conjunction: $c(\alpha) = 1 - p(\hat{s} \in N_A(\alpha))$.

The idea of using the length of the conjunctions as cost function is that short conjunctions are usually more weighty arguments. Clearly, if α is a conjunction in \mathcal{C}_A , then an additional literal ℓ is a supplementary condition to be satisfied, and $\alpha \wedge \ell$ is therefore less probable than α . From this point of view, the length of a conjunction expresses somehow its probability. However, if probabilities are assigned to the assumptions, then it is possible to specify the probability of a conjunction more precisely. That's the idea behind the second suggestion.

Let $\beta \in \mathbb{R}^+$ be a fixed bound for a monotone cost function $c(\alpha)$. A conjunction $\alpha \in \mathcal{C}_A$ is called β -*relevant*, if and only if $c(\alpha) < \beta$. Otherwise, α is called β -*irrelevant*. The set of all β -relevant conjunctions for a fixed cost bound β is denoted by

$$\mathcal{C}_A^\beta = \{\alpha \in \mathcal{C}_A : c(\alpha) < \beta\}. \quad (14)$$

Note that \mathcal{C}_A^β is a *downward-closed* set. This means that $\alpha \in \mathcal{C}_A^\beta$ implies that every (shorter) conjunction $\alpha' \subseteq \alpha$ is also in \mathcal{C}_A^β . Evidently, $\mathcal{C}_A^0 = \emptyset$ and $\mathcal{C}_A^\infty = \mathcal{C}_A$. An approximated set of minimal quasi-supporting arguments can then be defined by

$$\mu QS(h, \xi, \beta) = \mu QS(h, \xi) \cap \mathcal{C}_A^\beta. \quad (15)$$

The corresponding set of scenarios is denoted by $QS_A(h, \xi, \beta)$. Note that $\mu QS(h, \xi, \beta)$ is sound but not complete since $\mu QS(h, \xi, \beta) \subseteq \mu QS(h, \xi)$. It can therefore be seen as a lower approximation of the exact set $\mu QS(h, \xi)$.

In order to compute $\mu QS(h, \xi, \beta)$ efficiently, corresponding downward-closed sets are defined over the set of clauses $\mathcal{D}_{A \cup P}$. Obviously, every clause $\xi \in \mathcal{D}_{A \cup P}$ can be split into sub-clauses ξ_A and ξ_P by

$$\xi = \underbrace{\ell_1 \vee \dots \vee \ell_k}_{\in A^\pm} \vee \underbrace{\ell_{k+1} \vee \dots \vee \ell_m}_{\in P^\pm} = \xi_A \vee \xi_P, \quad (16)$$

where A^\pm and P^\pm are the sets of literals of A and P , respectively. Such a clause can also be written as an implication $\neg \xi_A \rightarrow \xi_P$ where $Arg(\xi) = \neg \xi_A$ is a conjunction in \mathcal{C}_A . The set of clauses ξ for which the corresponding conjunction $Arg(\xi)$ is in \mathcal{C}_A^β can then be defined by

$$\mathcal{D}_{A \cup P}^\beta = \{\xi \in \mathcal{D}_{A \cup P} : Arg(\xi) \in \mathcal{C}_A^\beta\}. \quad (17)$$

A new elimination procedure called β -*elimination* can then be defined by

$$Elim_Q^\beta(\Sigma) = Elim_{x_1}^\beta \circ \dots \circ Elim_{x_q}^\beta(\Sigma), \quad (18)$$

where the clauses not belonging to $\mathcal{D}_{A \cup P}^\beta$ are dropped at each step of the process by $Elim_x^\beta(\Sigma) = Elim_x(\Sigma) \cap \mathcal{D}_{A \cup P}^\beta$. In this way, the approximated set of quasi-supporting arguments can be computed by

$$\begin{aligned} \mu QS(h, \xi, \beta) &= \neg Elim_P^\beta(Cons_A(\Sigma_H)) \\ &= \neg Elim_P^\beta(Cons_A(\Sigma_H) \cap \mathcal{D}_{A \cup P}^\beta). \end{aligned} \quad (19)$$

See [17] for a more detailed discussion and the proof of the above formula. Two major problems remain. First, it is difficult to choose a suitable cost bound β in advance (if β is too low, then the result may be unsatisfactory, if β is too high, then the procedure risks to get stuck). Second, there is no mean to judge to quality of the approximation.

4 Anytime Algorithm

The algorithm introduced below helps to overcome the difficulties mentioned at the end of the previous section. Instead of first choosing the cost bound and then computing the corresponding arguments, the algorithm starts immediately by computing the most relevant arguments and terminates as soon as no more computational resources (usually time) are available. Finally, it returns two minimal sets LB and UB of (potential) minimal arguments with

$$N_A(LB) \subseteq QS_A(h, \xi) \subseteq N_A(UB) \quad (20)$$

and a cost bound β with $LB \supseteq \mu QS(h, \xi, \beta)$. Obviously, the sets LB is considered as lower bound (sound but not complete) while UB is considered as upper bound (complete but not sound). From this follows immediately that

$$p(\hat{s} \in N_A(LB)) \leq p(\hat{s} \in QS_A(h, \xi)) \leq p(\hat{s} \in N_A(UB)). \quad (21)$$

Furthermore, if LB' , UB' , and β' are the corresponding results for the same input parameters (knowledge base, hypothesis, cost function) but with more computational resources, then $N_A(LB) \subseteq N_A(LB')$, $N_A(UB) \supseteq N_A(UB')$, and $\beta \leq \beta'$. The quality of the approximation increases thus monotonically during the execution of the algorithm. If the algorithm terminates before all computational resources are used or if unlimited computational resources are available, then the algorithm returns the exact result $\mu QS(h, \xi) = LB = UB$ and $\beta = \infty$.

The idea for the algorithm comes from viewing the procedure described in the previous section from the perspective of Dechter's bucket elimination procedure [13]. From this point of view, the clauses contained in $Cons_A(\Sigma_H)$ are initially distributed among an ordered set of buckets. There is exactly one bucket for every proposition in P . If a clause contains several propositions from P , then the first appropriate bucket of the sequence is selected. In a second step, the elimination procedure takes place among the sequence of buckets.

The idea now is similar. But instead of processing the whole set of clauses at once, the clauses are now iteratively introduced one after another, starting with those having the lowest cost. At each step of the process, possible resolvents are computed and added to the list of remaining clauses. Subsumed clauses are dropped. As soon as a clause containing only proposition from A is detected, it is considered as a possible result. By taking the clauses

with low costs first, it is guaranteed that the most relevant arguments are produced first. This is due to the fact that the cost of a resolvent is always at least as high as the individual costs of the two clauses producing the resolvent. Note that for a given sequence of buckets, exactly the same set of resolvents are produced as in the usual bucket elimination procedure, but in a different order.

The core of the algorithm works with different sets of clauses:

- $\Sigma \Rightarrow$ the remaining set of clauses, initialized to $Cons_A(\Sigma_H) \setminus \mathcal{D}_A$,
- $\Sigma_0 \Rightarrow$ the results, initialized to $Cons_A(\Sigma_H) \cap \mathcal{D}_A$,
- $\Sigma_1, \dots, \Sigma_n \Rightarrow$ the corresponding sequence of buckets for all propositions in $P = \{p_1, \dots, p_n\}$, all initialized to \emptyset .

The details of the whole procedure are described below. The process terminates as soon as $\Sigma = \emptyset$ or when no more resources are available.

```

[01] Function Quasi-Support( $P, A, \Sigma, h, c$ );
[02] Begin
[03]   Select  $H \subseteq \mathcal{D}_{A \cup P}$  such that  $\wedge H \equiv \neg h$ ;
[04]    $\Sigma_H \leftarrow \mu(\Sigma \cup H)$ ;
[05]    $\Sigma \leftarrow Cons_A(\Sigma_H) \setminus \mathcal{D}_A$ ;
[06]    $\Sigma_0 \leftarrow Cons_A(\Sigma_H) \cap \mathcal{D}_A$ ;
[07]    $\Sigma_i \leftarrow \emptyset$  for all  $i = 1, \dots, n$ ;
[08]   Loop While  $\Sigma \neq \emptyset$  And Resources() > 0 Do
[09]     Begin
[10]       Select  $\xi \in \Sigma$  such that  $c(Arg(\xi)) = \min(\{c(Arg(\xi)) : \xi \in \Sigma\})$ ;
[11]        $\Sigma \leftarrow \Sigma \setminus \{\xi\}$ ;
[12]        $k \leftarrow \min(\{i \in \{1, \dots, n\} : p_i \in Props(\xi)\})$ ;
[13]       If  $p_k \in \xi$ 
[14]         Then  $R \leftarrow \rho(\{\xi\}, \{\xi \in \Sigma_k : \neg p_k \in \xi\})$ ;
[15]       Else  $R \leftarrow \rho(\{\xi \in \Sigma_k : p_k \in \xi\}, \{\xi\})$ ;
[16]        $\Sigma \leftarrow \Sigma \cup (R \setminus \mathcal{D}_A)$ ;
[17]        $\Sigma_0 \leftarrow \Sigma_0 \cup (R \cap \mathcal{D}_A)$ ;
[18]        $\Sigma_k \leftarrow \Sigma_k \cup \{\xi\}$ ;
[19]        $S \leftarrow \mu(\Sigma \cup \Sigma_0 \cup \dots \cup \Sigma_n)$ ;
[20]        $\Sigma \leftarrow \Sigma \cap S$ ;
[21]        $\Sigma_i \leftarrow \Sigma_i \cap S$  for all  $i = 1, \dots, n$ ;
[22]     End;
[23]      $LB \leftarrow \{Arg(\xi) : \xi \in \Sigma_0\}$ ;
[24]      $UB \leftarrow \mu\{Arg(\xi) : \xi \in \Sigma_0 \cup \Sigma\}$ ;
[25]      $\beta \leftarrow \min(\{c(Arg(\xi)) : \xi \in \Sigma\} \cup \{\infty\})$ ;
[26]     Return ( $LB, UB, \beta$ );
[27]   End.

```

At each step of the iteration, the following operations take place:

- line [10]: select a clause ξ from Σ such that the corresponding cost $c(Arg(\xi))$ is minimal;

- line [11]: remove ξ from Σ ;
- line [12]: select the first proposition $p_k \in P$ with $p_k \in Props(\xi)$ and
 - lines [13]–[15]: compute all resolvents of ξ with Σ_k ,
 - lines [16] and [17]: add the resolvents either to Σ or Σ_0 ,
 - line [18]: add ξ to Σ_k ,
 - lines [19]–[21]: remove subsumed clauses from $\Sigma, \Sigma_0, \Sigma_1, \dots, \Sigma_n$.

Finally, LB and β are obtained from Σ_0 . Furthermore, UB can be derived from Σ_0 and Σ . Note that the procedure is a true *anytime algorithm* giving progressively better solutions as time goes on and also giving a response however little time has elapsed [12, 11]. In fact, it satisfies most of the basic requirements of anytime algorithms [29]:

- *measurable quality*: the precision of the approximate result is known,
- *monotonicity*: the precision of the result is growing in time,
- *consistency*: the quality of the result is correlated with the computation time and the quality of the inputs,
- *the diminishing returns*: the improvement of the solution is larger at the early stages of computation and it diminishes over time,
- *interruptibility*: the process can be interrupted at any time and provides some answer,
- *preemptability*: the process can be suspended and continued with minimal overhead.

A more detailed discussion of these properties and corresponding proofs of correctness will appear in one of the author’s forthcoming technical reports.

4.1 Example

In order to illustrate the algorithm introduced in the previous subsection, consider a communication network with 6 participants (A, B, C, D, E, F) and 9 connections. The question is whether A is able to communicate with F or not. This is expressed by $h = A \rightarrow F$. It is assumed that connection 1 (between A and B) works properly with probability 0.1, connection 2 with probability 0.2, etc.

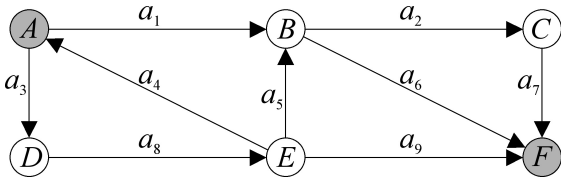


Fig. 1 A simple communication network.

The original knowledge base Σ consists of 9 clauses $\xi_3, \xi_5, \xi_6, \xi_8, \xi_{10}, \xi_{11}, \xi_{13}, \xi_{17}$ and ξ_{18} . Furthermore, $H = \{A, \neg F\} = \{\xi_1, \xi_2\}$ and $\Sigma_H = \mu(\Sigma \cup H) = (\Sigma \cup H) \setminus \{\xi_{11}\}$, since ξ_{11} is subsumed by ξ_1 . Table 1 shows all the clauses produced during the process (ordered according to their probabilities).

Table 1 The clauses produced during the process.

►	ξ_1	A	1.0		ξ_{14}	$D \vee \neg a_3$	0.3
►	ξ_2	$\neg F$	1.0		ξ_{15}	$E \vee \neg a_3 \vee \neg a_8$	0.24
•	ξ_3	$\neg E \vee F \vee \neg a_9$	0.9	★	ξ_{16}	$\neg a_3 \vee \neg a_8 \vee \neg a_9$	0.216
	ξ_4	$\neg E \vee \neg a_9$	0.9	•	ξ_{17}	$\neg B \vee C \vee \neg a_2$	0.2
•	ξ_5	$\neg D \vee E \vee \neg a_8$	0.8	•	ξ_{18}	$\neg A \vee B \vee \neg a_1$	0.1
•	ξ_6	$\neg C \vee F \vee \neg a_7$	0.7		ξ_{19}	$B \vee \neg a_1$	0.1
	ξ_7	$\neg C \vee \neg a_7$	0.7		ξ_{20}	$\neg E \vee C \vee \neg a_2 \vee \neg a_5$	0.1
•	ξ_8	$\neg B \vee F \neg a_6$	0.6	★	ξ_{21}	$\neg a_3 \vee \neg a_5 \vee \neg a_6 \vee \neg a_8$	0.072
	ξ_9	$\neg B \vee \neg a_6$	0.6		ξ_{22}	$\neg E \vee \neg a_2 \vee \neg a_5 \vee \neg a_7$	0.07
•	ξ_{10}	$\neg E \vee B \vee \neg a_5$	0.5	★	ξ_{23}	$\neg a_1 \vee \neg a_6$	0.06
•	ξ_{11}	$\neg E \vee A \vee \neg a_4$	0.4		ξ_{24}	$C \vee \neg a_1 \vee \neg a_2$	0.02
	ξ_{12}	$\neg E \vee \neg a_5 \vee \neg a_6$	0.3	★	ξ_{25}	$\neg a_2 \vee \neg a_3 \vee \neg a_5 \vee \neg a_7 \vee \neg a_8$	0.017
•	ξ_{13}	$\neg A \vee D \vee \neg a_3$	0.3	★	ξ_{26}	$\neg a_1 \vee \neg a_2 \vee \neg a_7$	0.014

The initial clauses of Σ are marked by • and the clauses of H by ►. The minimal quasi-supporting arguments for $h = A \rightarrow F$ are finally obtained from the clauses $\xi_{16}, \xi_{21}, \xi_{23}, \xi_{25}$, and ξ_{26} (marked by ★):

$$\begin{aligned} \mu QS(h, \Sigma) &= \{\neg \xi_{16}, \neg \xi_{21}, \neg \xi_{23}, \neg \xi_{25}, \neg \xi_{26}\} \\ &= \{a_3 \wedge a_8 \wedge a_9, a_3 \wedge a_5 \wedge a_6 \wedge a_8, a_1 \wedge a_6, a_2 \wedge a_3 \wedge a_5 \wedge a_7 \wedge a_8, a_1 \wedge a_2 \wedge a_7\}. \end{aligned}$$

Note that every minimal quasi-supporting argument in $\mu QS(h, \Sigma)$ corresponds to a minimal path from node A to node F in the communication network. If we take A, F, B, D, C, E as elimination sequence (order of the buckets), then the complete run of the algorithm can be described as in Table 2 (where $c(\alpha) = 1 - p(\hat{s} \in N_A(\alpha))$ serves as cost function).

Every row describes a single step. The 2nd column shows the most probable clause in Σ (the one with the lowest cost) that is selected for the next step. The 3rd column contains the remaining clauses in Σ . The 4th column indicates the first proposition in the given sequence that appears in the selected clause. This determines the corresponding bucket Σ_i into which the selected clauses is added (columns 5 to 10). Cross-marked clauses are subsumed by others and can be dropped. Then, column 11 shows the resolvents produced at the actual step. Resolvents containing only proposition from A are added to column 12. Finally, the last column indicates the actual value of β .

At step 1, $\xi_1 = A$ is selected and added to Σ_1 . There are no resolvents and no subsumed clauses. Σ_1 contains then the single clause ξ_1 while Σ_0

Table 2 The complete run of the algorithm.

Step	Σ		p_i	A	F	B	D	C	E	R	Σ_0	$1-\beta$
				Σ_1	Σ_2	Σ_3	Σ_4	Σ_5	Σ_6			
1	ξ_1	$\xi_2, \xi_3, \xi_5, \xi_6, \xi_8, \xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	A	ξ_1								1.00
2	ξ_2	$\xi_3, \xi_5, \xi_6, \xi_8, \xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	F		ξ_2							0.90
3	ξ_3	$\xi_5, \xi_6, \xi_8, \xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	F	ξ_3						ξ_4		0.90
4	ξ_4	$\xi_5, \xi_6, \xi_8, \xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	E						ξ_4			0.80
5	ξ_5	$\xi_6, \xi_8, \xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	D				ξ_5					0.70
6	ξ_6	$\xi_8, \xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	F		ξ_6					ξ_7		0.70
7	ξ_7	$\xi_8, \xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	C					ξ_7				0.60
8	ξ_8	$\xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	F		ξ_8					ξ_9		0.60
9	ξ_9	$\xi_{10}, \xi_{13}, \xi_{17}, \xi_{18}$	B			ξ_9						0.50
10	ξ_{10}	$\xi_{13}, \xi_{17}, \xi_{18}$	B			ξ_{10}				ξ_{12}		0.30
11	ξ_{12}	$\xi_{13}, \xi_{17}, \xi_{18}$	E						ξ_{12}			0.30
12	ξ_{13}	ξ_{17}, ξ_{18}	A	ξ_{13}						ξ_{14}		0.30
13	ξ_{14}	ξ_{17}, ξ_{18}	D				ξ_{14}			ξ_{15}		0.24
14	ξ_{15}	ξ_{17}, ξ_{18}	E						ξ_{15}	ξ_{16}, ξ_{21}	ξ_{16}, ξ_{21}	0.20
15	ξ_{17}	ξ_{18}	B			ξ_{17}				ξ_{20}		0.10
16	ξ_{18}	ξ_{20}	A	ξ_{18}						ξ_{19}		0.10
17	ξ_{19}	ξ_{20}	B			ξ_{19}				ξ_{23}, ξ_{24}	ξ_{23}	0.10
18	ξ_{20}	ξ_{24}	C					ξ_{20}		ξ_{22}		0.07
19	ξ_{22}	ξ_{24}	E						ξ_{22}	ξ_{25}	ξ_{25}	0.02
20	ξ_{24}		C					ξ_{24}		ξ_{26}	ξ_{26}	$-\infty$

is still empty. At step 3, for example, $\xi_3 = \neg E \vee F \vee \neg a_9$ is selected and added to Σ_2 that already contains $\xi_2 = \neg F$ from step 2. A new resolvent $\xi_4 = \neg E \vee \neg a_9$ can then be derived from ξ_2 and ξ_3 . Since ξ_3 is subsumed by ξ_4 , it can be dropped. The new clause ξ_4 is then added to Σ . Σ_0 is still empty. Later, for example at step 14, $\xi_{15} = E \vee \neg a_3 \vee \neg a_8$ is selected and added to Σ_6 that contains two clauses $\xi_4 = \neg E \vee \neg a_9$ and $\xi_{12} = \neg E \vee \neg a_5 \vee \neg a_6$ from previous steps. Two new resolvents $\xi_{16} = \neg a_3 \vee \neg a_8 \vee \neg a_9$ and $\xi_{21} = \neg a_3 \vee \neg a_5 \vee \neg a_6 \vee \neg a_8$ are produced and added to Σ_0 . These are the first two results. After step 20, Σ is empty and the algorithm terminates.

Observe that the clauses representing the query $H = \{A, \neg F\} = \{\xi_1, \xi_2\}$ are processed first. This influences considerably the further run of the algorithm and guarantees that those arguments which are of particular importance for the user's actual query are returned first. Such a *query-driven* behavior is an important property of the algorithm. It corresponds to the natural way of how human gathers the relevant information from a knowledge base.

4.2 Experimental Results

This section discusses the results of testing the algorithm on problems of more realistic size. The discussion focusses on lower and upper bounds and compares them to the exact results.

Communication Network: The knowledge base consists of 26 propositions, 39 assumptions and 74 initial clauses. It describes a communication network like the one used in the previous subsection. The exact solution for a certain query consists of 1,008 minimal arguments (shortest paths). For a given elimination sequence, the complete procedure generates 211,828 resolvents. The corresponding degree of support is 0.284. Figure 2 shows how the approximated solution monotonically approaches the exact value during the process.

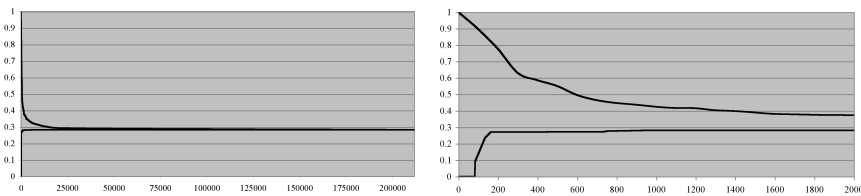


Fig. 2 Left, the complete run of the algorithm; right, the first 2,000 resolvents.

Observe that after generating approximately 1,000 resolvents, the algorithm has found the first 8 arguments and returns a numerical lower bound that corresponds in the first two numbers after the decimal point to the exact solution. The 8 first arguments are found in less than 1 second (instead of approximately 15 minutes for the 211,828 resolutions of the complete solution). The upper bound converges a little bit slower. This is a typical behavior that has been observed for many other examples of different domains [1].

64-Bits Adder: In the second situation, the knowledge base consists of 449 propositions, 320 assumptions and 895 initial clauses. It describes the digital network of an incorrect working 64-bits adder. There are 64 minimal arguments where each argument describes a possible conflict set (group of components of which at least one is broken). For a certain elimination sequence, the exact solution requires the generation of 25'226 resolvents. The corresponding degree of support is 0.292. Figure 3 shows how the approximated solution monotonically increases during the process.

The algorithm behaves like in the previous subsection. After generating ca. 1'500 resolvents (instead of 25'226), the algorithm returns a lower bound that corresponds in the first three numbers behind the comma to the exact solution. Again, the upper bound converges a little bit slower.

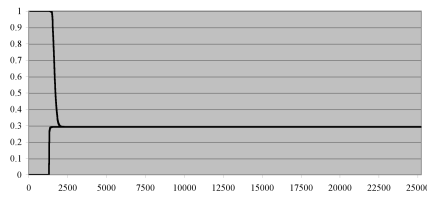


Fig. 3 The complete run of the algorithm.

5 Conclusion

This article introduces a new algorithm for approximated assumption-based reasoning. The advantages to other existing approximation methods are twofold: (1) it is an anytime algorithm that monotonically increases the quality of the result as soon as more computational resources are available; (2) the algorithm produces not only a lower but also an upper approximation without significant additional computational costs. This two improvements are extremely useful and can be considered as an important step towards the practical applicability of logic-based abduction and argumentation in general, and probabilistic argumentation systems in particular.

References

1. B. Anrig, R. Bissig, R. Haenni, J. Kohlas, and N. Lehmann. Probabilistic argumentation systems: Introduction to assumption-based modeling with ABEL. Technical Report 99-1, Institute of Informatics, University of Fribourg, 1999.
2. K. B. Laskey and P. E. Lehner. Assumptions, beliefs and probabilities. *Artificial Intelligence*, 41(1):65–77, 1989.
3. D. Berzati, R. Haenni, and J. Kohlas. Probabilistic argumentation systems and abduction. *Annals of Mathematics and Artificial Intelligence*, 34 (1-3), 177-195. 2002.
4. J. Bigham, Z. Luo, and D. Banerjee. A cost bounded possibilistic ATMS. In Christine Froidevaux and Jürg Kohlas, editors, *Proceedings of the ECSQARU Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 946, pages 52–59, Berlin, 1995. Springer Verlag.
5. J. W. Collins and D. de Coste. CATMS: An ATMS which avoids label explosions. In Kathleen McKeown and Thomas Dean, editors, *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 281–287. MIT Press, 1991.
6. J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.
7. J. de Kleer. Extending the ATMS. *Artificial Intelligence*, 28:163–196, 1986.
8. J. de Kleer. Problem solving with the ATMS. *Artificial Intelligence*, 28:197–224, 1986.
9. J. de Kleer. Focusing on probable diagnoses. In Kathleen Dean, Thomas L.; McKeown, editor, *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 842–848. MIT Press, 1991.

10. J. de Kleer. A perspective on assumption-based truth maintenance. *Artificial Intelligence*, 59(1–2):63–67, 1993.
11. T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 49–54. MIT Press, 1988.
12. T. L. Dean. Intrancibility and time-dependent planning. In M. P. Geofrey and A. L. Lansky, editors, *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans*. Morgan Kaufmann Publishers, 1987.
13. R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.
14. K. D. Forbus and J. de Kleer. Focusing the ATMS. In Tom M. Smith and Reid G. Mitchell, editors, *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 193–198, St. Paul, MN, 1988. Morgan Kaufmann.
15. R. Haenni. Modeling uncertainty with propositional assumption-based systems. In S. Parson and A. Hunter, editors, *Applications of Uncertainty Formalisms*, Lecture Notes in Artificial Intelligence 1455, pages 446–470. Springer-Verlag, 1998.
16. R. Haenni. Cost-bounded argumentation. *International Journal of Approximate Reasoning*, 26(2):101–127, 2001.
17. R. Haenni, J. Kohlas, and N. Lehmann. Probabilistic argumentation systems. In J. Kohlas and S. Moral, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems, Volume 5: Algorithms for Uncertainty and Defeasible Reasoning*. Kluwer, Dordrecht, 2000.
18. R. Haenni, N. Lehmann. Probabilistic argumentation systems: a new perspective on Dempster-Shafer theory. *International Journal of Intelligent Systems: Special Issue on Dempster-Shafer Theory of Evidence* (accepted for publication).
19. K. Inoue. An abductive procedure for the CMS/ATMS. In J. P. Martins and M. Reinfrank, editors, *Truth Maintenance Systems, Lecture Notes in A.I.*, pages 34–53. Springer, 1991.
20. J. Kohlas and P. A. Monney. *A Mathematical Theory of Hints. An Approach to the Dempster-Shafer Theory of Evidence*, volume 425 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 1995.
21. M. McLeish. Nilsson’s probabilistic entailment extended to Dempster-Shafer theory. In L. N. Kanal, T. S. Levitt, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, volume 8, pages 23–34. North-Holland, Amsterdam, 1987.
22. J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
23. G. M. Provan. A logic-based analysis of Dempster-Shafer theory. *International Journal of Approximate Reasoning*, 4:451–495, 1990.
24. R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In Kenneth Forbus and Howard Shrobe, editors, *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 183–188. American Association for Artificial Intelligence, AAAI Press, 1987.
25. G. Shafer. *The Mathematical Theory of Evidence*. Princeton University Press, 1976.
26. P. P. Shenoy. Binary join trees. In Eric Horvitz and Finn Jensen, editors, *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 492–499, San Francisco, 1996. Morgan Kaufmann Publishers.

27. Ph. Smets. Probability of provability and belief functions. In M. Clarke, R. Kruse R., and S. Moral, editors, *Proceedings of the ECSQARU'93 conference*, pages 332–340. Springer-Verlag, 1993.
28. Ph. Smets and R. Kennes. The transferable belief model. *Artificial Intelligence*, 66:191–234, 1994.
29. S. Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, Fall:71–83, 1996.