



ProBT®

An API for Bayesian Programming

Pierre Bessière

Pierre.Bessiere@probayes.com

probayes.com

bayesian-programming.org

Mastering Uncertainty



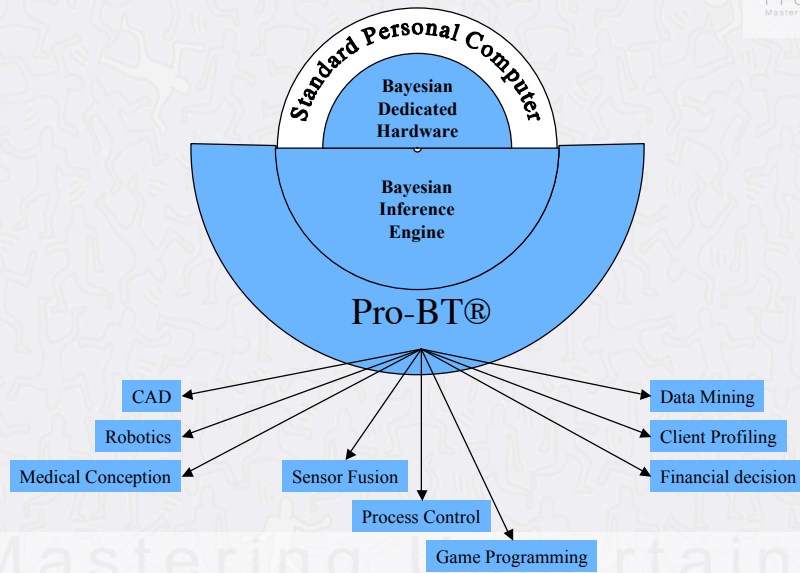
Company Vision

- **Incompleteness**: inescapable difficulty of complex problem
- **Uncertainty**: unavoidable consequence of incompleteness

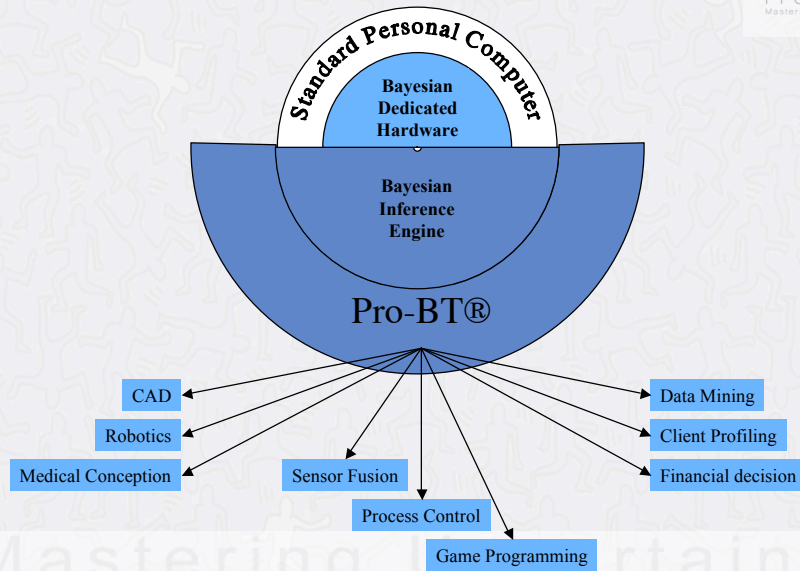
Bayesian Probability:
an alternative to Logic

Mastering Uncertainty

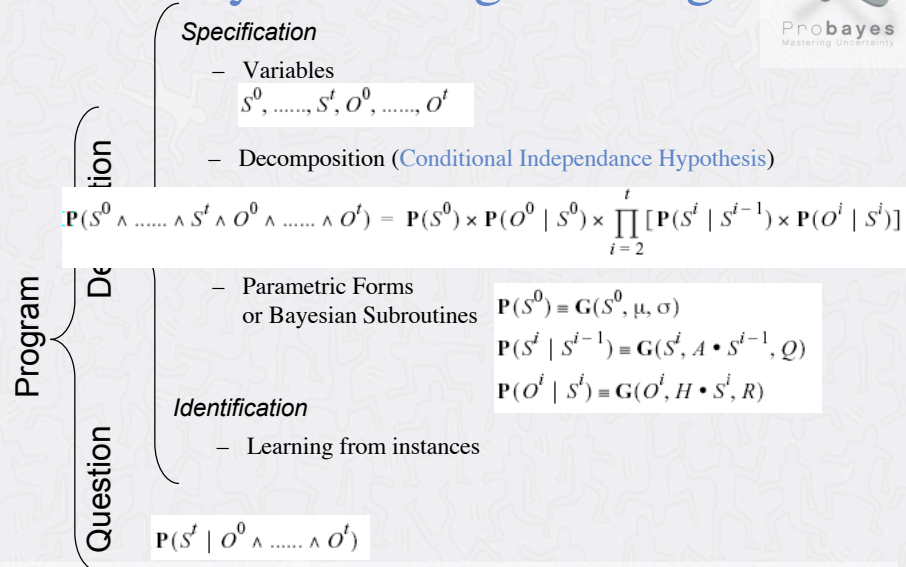
Bayesian Computer



ProBT® (1)



Bayesian Programming



Mastering Uncertainty

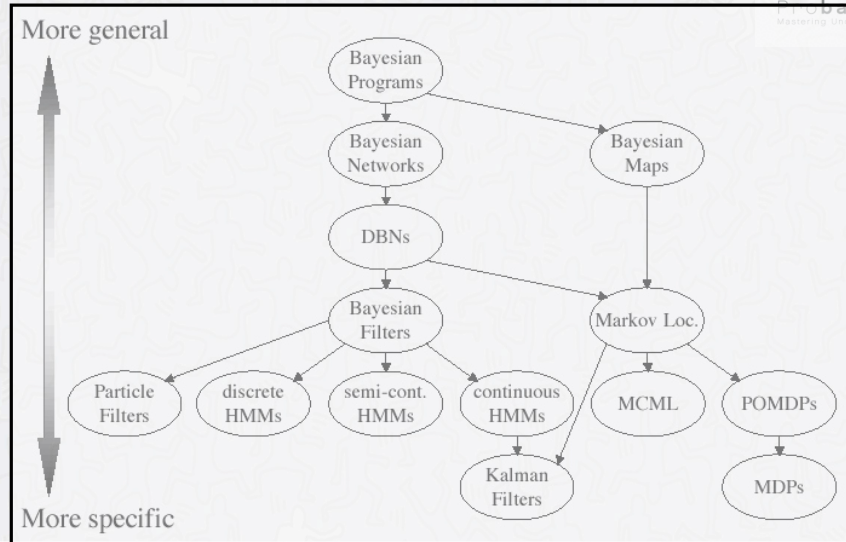
Bayesian Programming (2)



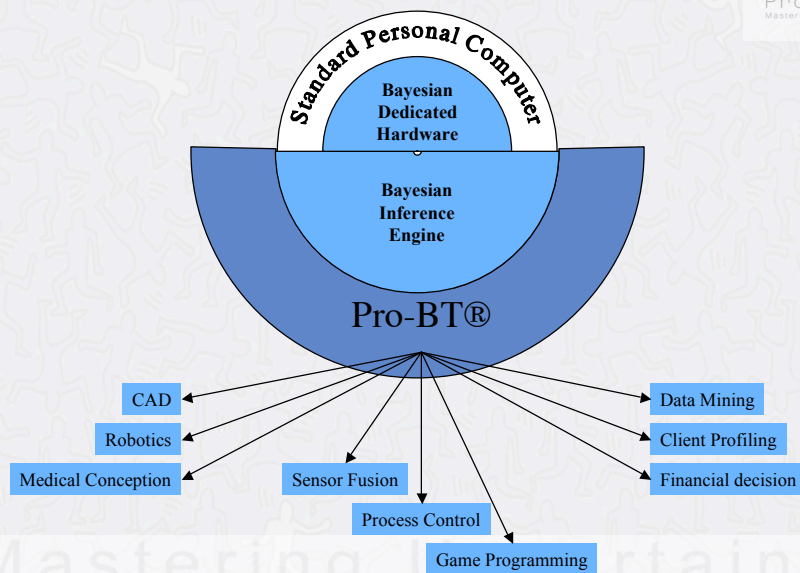
- Calling Bayesian Subroutines
- Bayesian Program if-then-else
- Bayesian Program Sequencing
- Bayesian Program Iteration
- Using Functions

Mastering Uncertainty

Related Approaches



ProBT® (1)



ProBT® (2)



```
main ()
{
    //Variables
    plFloat read_time;
    plIntegerType t_and_id(time,id);
    plFloat times[5] = {1,2,3,5,10};
    plSparseType time_type(5,times);
    plSymbol id("id",id,times);
    plSymbol time("time",time_type);

    //Parametrical forms
    //Construction of P(id)
    plProbValue id_dist[5] = {20,30,10,5,2};
    plProbTable P_id(id,id_dist);

    //Construction of P(t|id = john)
    plProbValue t_john_dist[5] = {20,30,10,5,2};
    plProbTable P_t_john(time,t_john_dist);
    //Construction of P(time | id = bill)
    plProbValue t_bill_dist[5] = {2,6,10,40,20};
    plProbTable P_t_bill(time,t_bill_dist);

    //Construction de P(time | id)
    plKernelTable Pt_id(time,id);
    plValues t_and_id(time^id);
    t_and_id[id] = 0;
    Pt_id.push(P_t_john,t_and_id);
    t_and_id[id] = 1;
    Pt_id.push(P_t_bill,t_and_id);

    //Decomposition
    // P(time id) = P(id) P(time | id)
    plJointDistribution
        jd(time^id,P_id*Pt_id);

    //Question
    plCndKernel Pid_t;
    jd.ask(Pid_t,id,time);

    //Read a time from the key board
    cout<<"P(id,time)= "<<Pid_t<<"\n";
    cout<<"Time? : ";
    cin>>read_time;

    //Getting P(id | time = read_time)
    plKernel Pid_readTime;
```

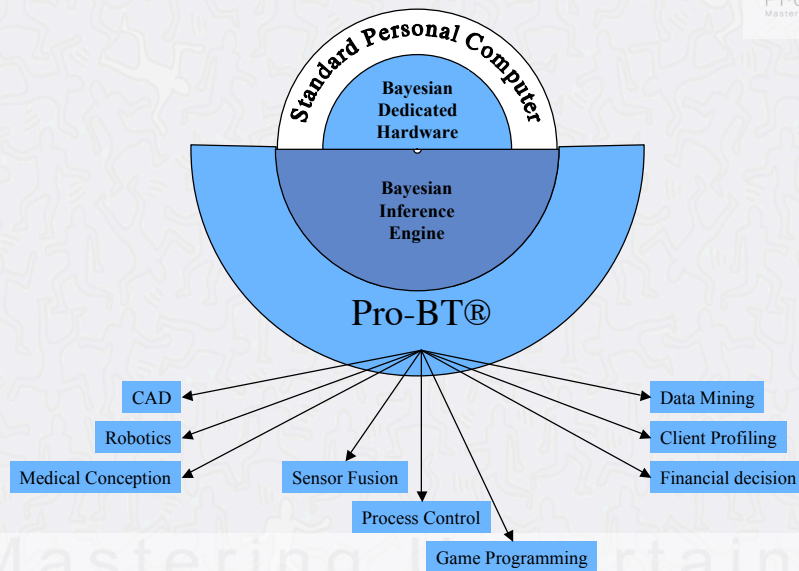
In C++ running on the following systems:

- Windows
- Unix
- Linux

Interpreted versions (under development):

- Java
- ImaLab (Collaboration with PRIMA)
- Python

Bayesian Inference Engine (1)



Inference Rules



$$\begin{aligned} P(X \wedge Y | \pi) &= P(X | \pi) \times P(Y | X \wedge \pi) \\ &= P(Y | \pi) \times P(X | Y \wedge \pi) \end{aligned}$$

$$\sum_X P(X | \pi) = 1$$

$$\sum_X P(X \wedge Y | \pi) = P(Y | \pi)$$

Mastering Uncertainty

Question

$$P(X^1 \wedge X^2 \wedge \dots \wedge X^n) = P(L^1) \times P(L^2 | R^2) \times \dots \times P(L^k | R^k)$$



$$P(\text{Search} | \text{Known})$$

$$= \sum_{Free} P(\text{Search} \wedge Free | \text{Known})$$

$$= \sum_{Free} \frac{P(\text{Search} \wedge Free \wedge \text{Known})}{P(\text{Known})}$$

$$= \frac{1}{Z} \times \sum_{Free} P(\text{Search} \wedge Free \wedge \text{Known})$$

$$= \frac{1}{Z} \times \sum_{Free} P(L^1) \times P(L^2 | R^2) \times \dots \times P(L^k | R^k)$$

- Marginalisation (integration) in high-dimensional space

- Searching the modes in high-dimensional space

2 modes: exact or approximated

- Symbolic simplification

- Numeric computation

Mastering Uncertainty

Symbolic simplification

$$P(Search \mid Known) = \frac{1}{Z} \times \sum_{Free1} \left[\prod_{i=0}^k P(L^i \mid R^i) \right]$$



Factorization:

$$P(Search \mid Known) = \frac{1}{Z} \times \prod_{j \in J} P(L^j \mid R^j) \times \sum_{Free1} \left[\prod_{k \in K} P(L^k \mid R^k) \right]$$

Sum to 1:

[Bessière et al. 2003]

$$P(Search \mid Known) = \frac{1}{Z} \times \prod_{j \in J} P(L^j \mid R^j) \times \sum_{Free2} \left[\prod_{l \in L} P(L^l \mid R^l) \right]$$

Distributivity:

Generalized distributive law [Aji & McEliece2000]

Restrictions successives [Raoul & Smail2003]

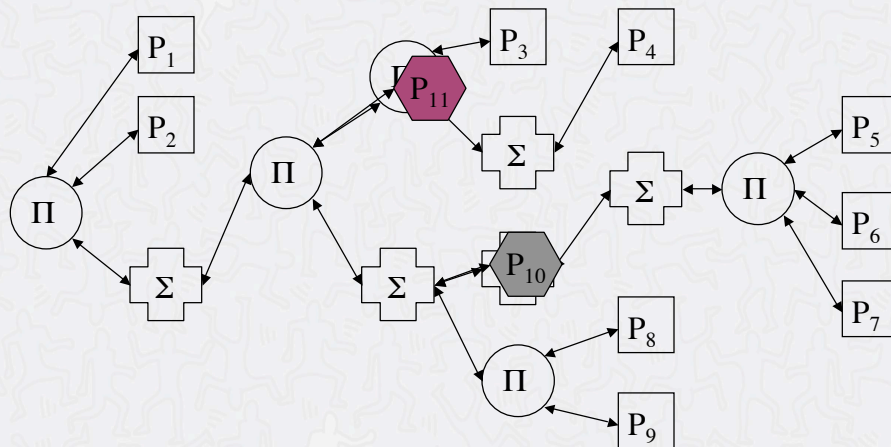
$$P(Search \mid Known) = \frac{1}{Z} \times \prod_{j \in J} P(L^j \mid R^j) \times \sum_{Free3} \left[\prod_{m \in M} P(L^m \mid R^m) \times \sum_{Free4} \left[\prod_{n \in N} P(L^n \mid R^n) \times \dots \times \sum_{FreeX} \left[\prod_{o \in O} P(L^o \mid R^o) \right] \right] \right]$$

Mastering Uncertainty

Numeric computation (exact)



$$P(Search \mid Known) = \frac{1}{Z} \times \prod_{j \in J} P(L^j \mid R^j) \times \sum_{Free3} \left[\prod_{m \in M} P(L^m \mid R^m) \times \sum_{Free4} \left[\prod_{n \in N} P(L^n \mid R^n) \times \dots \times \sum_{FreeX} \left[\prod_{o \in O} P(L^o \mid R^o) \right] \right] \right]$$



Mastering Uncertainty

Arithmetic of very small numbers



- PIFloat a new type of number specific to encode probabilities
- 64 bits representation with 32 bits mantissa
- Very efficient implementation of the corresponding arithmetic

Mastering Uncertainty

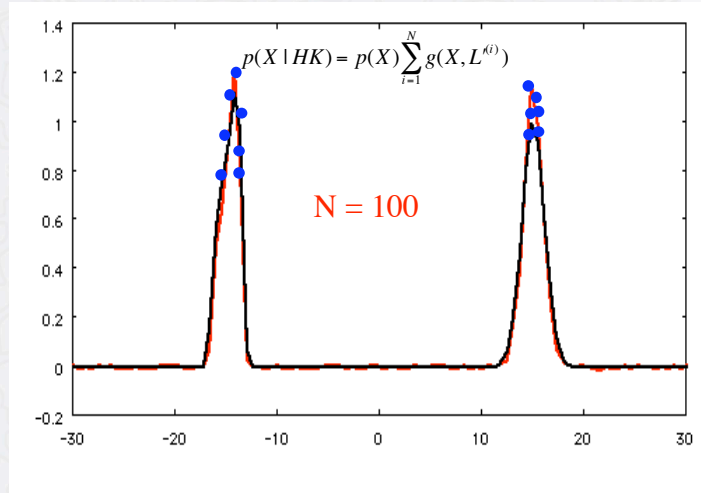
Numeric computation (approximated)



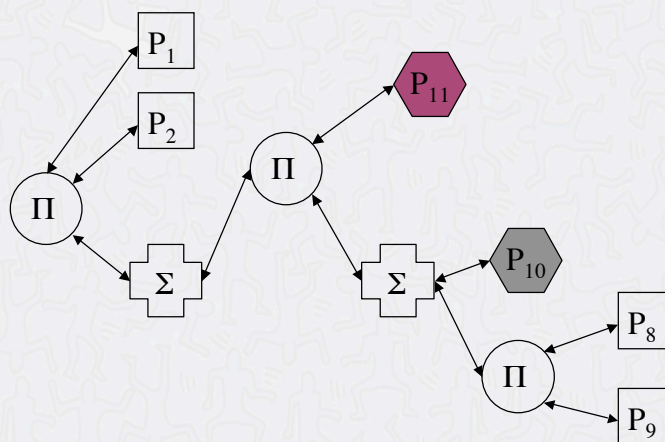
- MCMC for integration
- Genetic Algorithm to search the modes

Mastering Uncertainty

MCMC + Annealing



Multi-Resolution Binary Tree (MRBT)



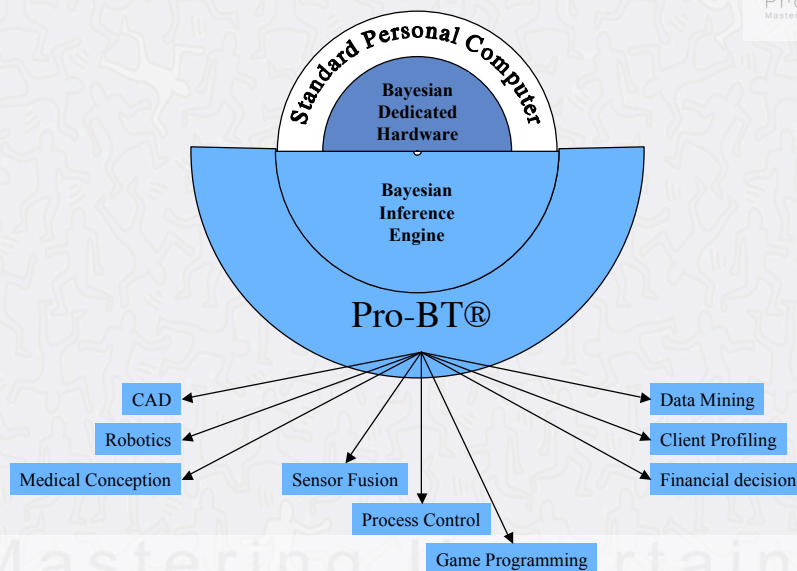
Multi-Resolution Binary Tree (MRBT)



- Very compact representation
- Multi-resolution: more precision in high probability area
- Very efficient draw
- Incremental and anytime representation

Mastering Uncertainty

Bayesian Hardware (1)

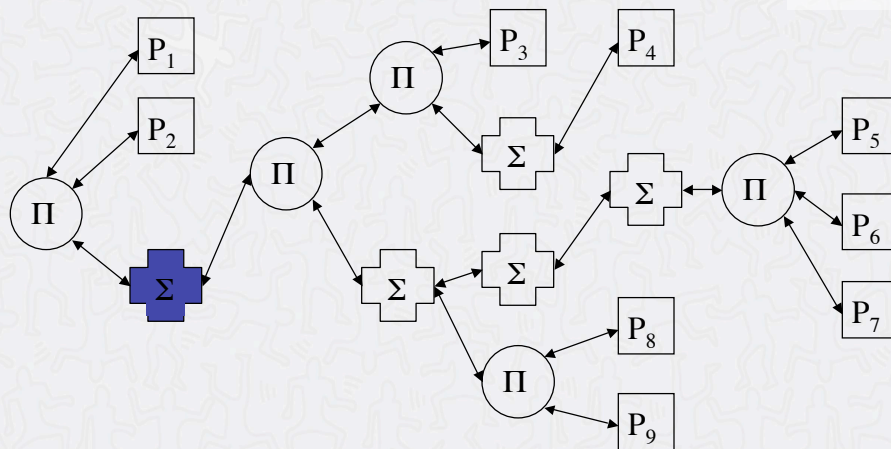


Mastering Uncertainty

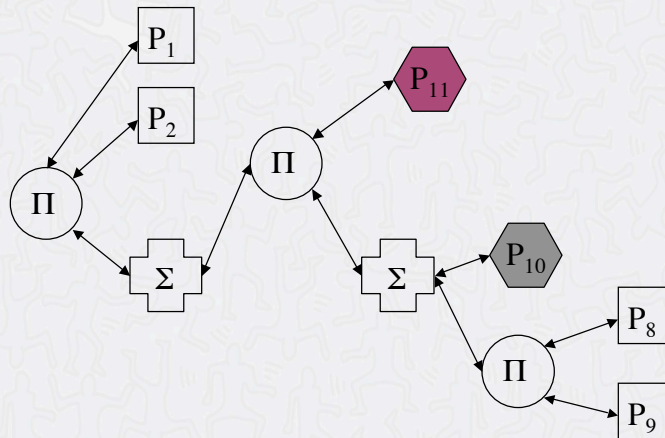
Bayesian Hardware (2)



Parallelization (1)



Parallelization (2)



Mastering Uncertainty

Historic



From fundamental research to industrial applications

- 1989 - Premières idées
- 1995 - Première version du moteur d'inférence (LISP)
- 1995 - Thèse sur l'incomplétude en robotique (E. Dedieu)
- 1997 - Deuxième version du moteur d'inférence (LISP)
- 1999 - Projet IMAG LAPLACE
- 1999 - Thèse sur la programmation bayésienne des robots autonomes (O. Lebeltel)
- 1999 - Thèse sur la CAO bayésienne (K. Mekhnacha)
- 2000 - Troisième version du moteur d'inférence
- 2001 - Transfert technologique - Incubation de la société
- 2001 - Projet Européen BIBA
- 2001 - Regroupement à GRAVIR au sein du projet SHARP
- 2002 - Disponibilité de Pro-BT (alpha)
- 2002 - Contrat SGAM
- 2002 - Projet RNTL télémaintenance
- 2003 - Thèse sur les cartes bayésiennes (J. Diard)
- 2003 - Thèse sur la programmation bayésienne des bras manipulateurs (R. Garcia)
- 2003 - Contrat SAMSE
- 2003 - Création de Pro-Bayes (septembre)
- 2003 - Thèse sur la fusion d'informations capteurs dans les systèmes d'assistance à la conduite automobile (C. Coué)
- 2004 - Livre « Bayesian Programming »
- 2004 - Thèse sur l'acquisition du langage chez le nourrisson (J. Serkhane)
- 2004 - Thèse sur le contrôle de véhicule (C. Pradalier)
- 2005 - Thèse sur la programmation bayésienne des robots autonomes (C. Koike)
- 2005 - Thèse sur la programmation et l'apprentissage bayésien des avatars dans les jeux vidéos (R. Lehy)
- 2005 - Thèse sur la perception de la forme par le mouvement (F. Colas)
- 2006 - Thèse sur la découverte automatisée de connaissances préalables (P. Danghautier)

Mastering Uncertainty

Bayesian-Programming.org



Bayesian Programming		
Hofstad: Monday, January 16, 2006		
Welcome	Welcome	<i>"The Bayesian revolution is just beginning"</i>
About ProBT	Our Approach <p>Computers have brought a new dimension to modeling. A model, once translated into a program, may be used to understand, measure, simulate, mimic, optimize, predict, and control. Science, industry, finance, medicine, entertainment, transport, and communication have been completely transformed by this revolution. However, contemporary programming techniques face great difficulties with models that are either incomplete or uncertain. Unfortunately, in numerous practical problems, some useful information is either missing or approximate.</p> <p>Different attempts have been made in computer science to deal with incomplete and uncertain information. Bayesian computation, which is simple, powerful, and mathematically sound, appears to be the most promising one. Bayesian computation may be extremely time-consuming, but thanks to recent improvements in both computer power and dedicated algorithms it is now possible to solve useful problems. The Bayesian revolution is just beginning. Future computers will require Bayesian capabilities: Bayesian hardware and Bayesian programming languages.</p>	ProBT Project Presentation <p>ProBT is a C++ library for developing efficient Bayesian software.</p> <p>The goal of ProBT is to facilitate the construction of high-performance applications within the Bayesian programming paradigm.</p> <p>The following institutions CHES, INRIA, UIE and INRIA hold all the property rights on ProBT software. The commercial and industrial exploitation of this software was conceded in an exclusive way to the ProBAYES Company.</p> <p>The ProBAYES Company give its agreement to GRAVIR laboratory for the free distribution of the object code of the ProBT library to the scientific community so that it can be used for research and educational purposes. For commercial or industrial use please contact the ProBAYES Company.</p> <p>To date, ProBT is our main Bayesian programming tool and test bed system. ProBT is used in a variety of projects both in the industrial and academic field</p> <p>The ProBT library has two main components: (1) a friendly Application Program Interface (API) for building Bayesian models and (2) a high-performance Bayesian Inference Engine (BIE) allowing execution of all the probability calculus in exact or approximate modes.</p> <p>The ProBT API comprises:</p> <ol style="list-style-type: none">1. A state space module permitting to specify the possible outcomes of a model.2. A Probability distributions module including: the most commonly used statistical distributions, joint distributions construction and n-dimensional inequality constraint specification.3. An experimental data learning module for parameters identification of the commonly used statistical distributions and/or the dependencies between the variables of the model. <p>The ProBT BIE comprises:</p> <ol style="list-style-type: none">1. An exact inference module consisting of a general purpose algorithm for optimal marginalization sequences computation. This algorithm has been developed in collaboration with the university of Harel la Vallée and it is part of the CHES grant project "Mathstat".2. An approximative inference module composed of :<ul style="list-style-type: none">• A set of integration and optimization methods.• Random numbers generator.• A set of sampling methods. <p>Read about more of our recent projects ></p>
Documentation		
Code samples		
FAQs		
Download		

ProBAYES recrute

Mastering Uncertainty