DM826 – Spring 2014

Modeling and Solving Constrained Optimization Problems

Lecture 11

Search

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Obligatory Assignment 1

- Your model should be clear and comprehensible, such that each of us can understand and implement it without difficulty.
  - Write it in pseudo-code, as in the lecture slides and homeworks.
  - The instance data, the decision variables (even reifying Booleans) and their domains, must be declared.
  - their semantics must be given in English/Danish, and every constraint must be annotated with an English paraphrase.
  - You may use standard mathematical notation and logical notation (but not programming-language-specific notation), such as (but not limited to) the following:
    - $M[i,j]$
    - **sum**$(i \in S)(f(i))$
    - **ForAll**$i \in S : c(i)$ quantified constraint
    - $\wedge$ or & or **and**
    - Avoid using full logic like $\vee$ (logical *or*), $\Longrightarrow$ (logically implies), or $\Longleftrightarrow$ (is logically equivalent to) between two (quantified) constraints
      do not use $\exists i \in S : c(i)$ to express that there must exist at least one i in set S such that the (quantified) constraint $c(i)$ holds
      nor apply $\neg$ (logical negation) to a (quantified) constraint.

- Use the global constraints, as well as any others seen in the course (contrary to full logic they enhance the possibility of propagation!)
  - $\textsc{Distinct}(\{x_1, \ldots, x_n\})$
  - $\textsc{Element}(\textsc{Sequence} a_1, \ldots, a_n, x, y)$, that is $a_x = y$.
  - $\textsc{GlobalCardinality}(\{x_1, \ldots, x_n\}, [v_1, \ldots, v_m], [\ell_1, \ldots, \ell_m], [u_1, \ldots, u_m])$
  - $[x_1, \ldots, x_n] \leq_{\textsc{Lex}} [y_1, \ldots, y_n]$
  - $\textsc{Linear}([a_1, \ldots, a_n], [x_1, \ldots, x_n], R, d)$ that is $\left(\sum_{i=1}^{n} a_i \cdot x_i\right) \; R \; d$.
  - ...
- **Write the linking constraints!!**

- Use different fonts for variables
- Stability was a constraint not a soft objective
- English not necessary
- Comments "?" means not understood but most likely there is an error
- Why no space before parenthesis(there should be one!)?
- Avoid whining for lack of time, if you had other ideas specify them to details, "custom branching" does not say anything.

Search – Resume

- Backtracking
- Branching strategies
- Nogood constraints
- Backjumping
- Restoration service
  Gecode uses a hybrid of copying and batch recomputation, called adaptive recomputation, which remembers a copy in the middle of the path from the root (sec. 40.6)
  more copying when a deadend encountered
  `c-d=8` recomputation commit distance (at most 8 recomputation commits)
  `a-d=2` recomputation adaptation distance (only if path length $n > a_d$ a copy is created)
- Variable-Value heuristics (shared selections: Accumulated Failure Count (sec. 8.5.2), Activity-based. Near to a value)

Van Hentenryck's Videos

- COMET code
- Choose var that leaves more values for other variables
- Value oriented decision (eg, perfect squares)
- Weaker commitment, domain splitting, $>, <$
  (eg, magic squares, car sequencing)
  tends to be a better choice since fixing values less benefit from propagation from other variables (Tip. 8.2)
- Symmetry breaking vs heuristics

Overview

- Random restarts
- Implementation issues
- Search in gecode-python
- Filtering algorithms in Scheduling

Outline

1. Random Restart

Randomization in Search Tree

- Ordering heuristics make mistakes (possibly early) $\rightsquigarrow$ randomization and restarts
- Randomization of choice points in backtracking
  while still maintaining the method complete
  $\rightsquigarrow$ *randomized systematic search*.
- do backtracking until distance from a deadend has exceeded a fixed cutoff number, restart by reordering the variables

Motivations

Definition (Las Vegas algorithms)

Las Vegas algorithms are randomized algorithms that always give the correct answer when they terminate, but running time varies from one run to another and is modeled as a random variable

Algorithm Survival Analysis

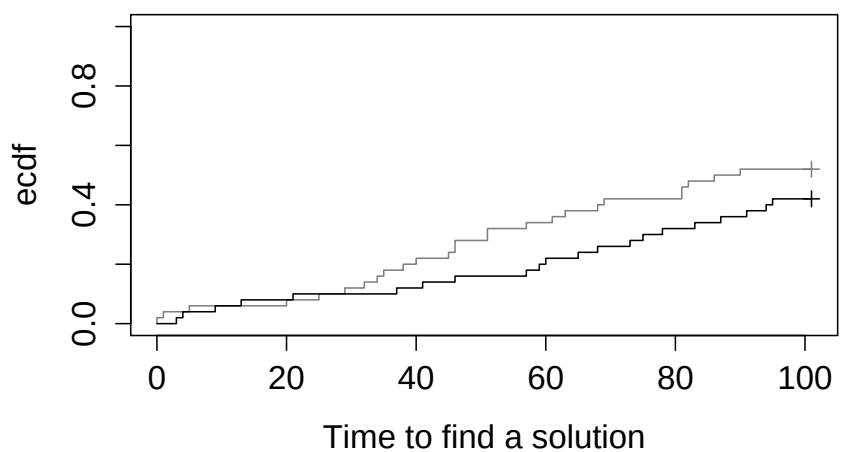
Run time distributions

- $T \in [0, \infty]$ time to find a solution on an instance
- $F(t) = \Pr\{T \leq t\} \quad F : [0, \infty] \mapsto [0, 1]$ cdf/RTD: Run Time Distribution
- $f(t) = \frac{dF(t)}{dt}$ pdf
- $S(t) = \Pr\{T > t\} = 1 - F(t)$ survival function
- $h(t) = \lim_{\Delta t \to 0} \Pr\{t \leq T < t + \Delta t \mid T \geq t\} \Delta t$ hazard function
- $H(t) = \int_0^t h(s)ds \quad h(s)\frac{f(t)}{S(t)} \quad H(t) = -\log S(t)$ cumulative hazrd function
- $E[T] = \int_0^\infty tf(t)dt = \int_0^1 tdF(t) = \int_0^\infty S(t)dt$ expected run time

Empirical Comparisons

```
> load("Data/r37.RData")
> head(R37)
     time iter event case
1 101 185737 0 1
2 57 84850 1 1
3 1 568 1 1
4 51 94974 1 1
5 5 7017 1 1

> require(survival)
> t <- survfit(Surv(time, event) ~ case, data = R37, type = "kaplan-meier",
conf.type = "plain", conf.int = 0.95, se.fit = T)
> plot(t, conf.int = F, xlab = "Time to find a solution", col = c("grey50", "black"), lty = c
      (1, 1), ylab = "ecdf", fun = "event", ylim = c(0,1))
```



Heavy Tails

$$F(t) \to_{t\to\infty} 1 - C^{t^{-\alpha}} \qquad \text{(Pareto like distr.)}$$

In practice, this means that

- most runs are relatively short, but the remaining few can take a very long time.
- Depending on C, α, the mean of a heavy-tailed distribution can be finite or not, while higher moments are always infinite.
- the length of a single run depends on the order with which randomized backtracking assigns values to the variables. [**?**]

  In some runs, backtracking has to search very deep branches in the tree of possible solutions before finding a contradiction.

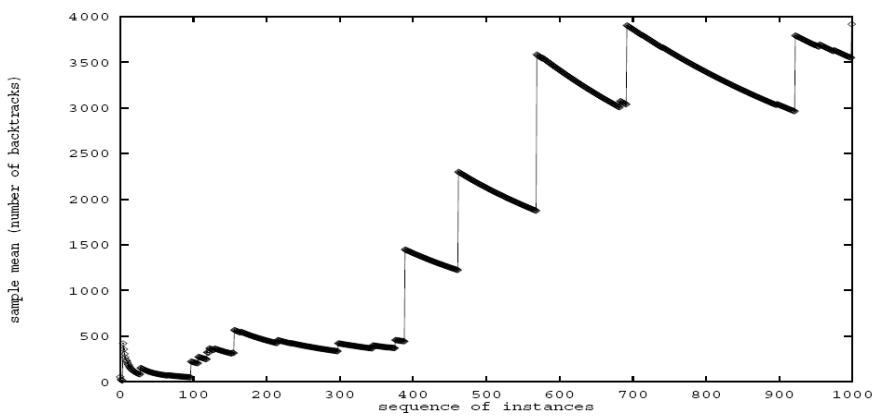
  The same instance may be very easy if solved with a different random reordering of the variables.

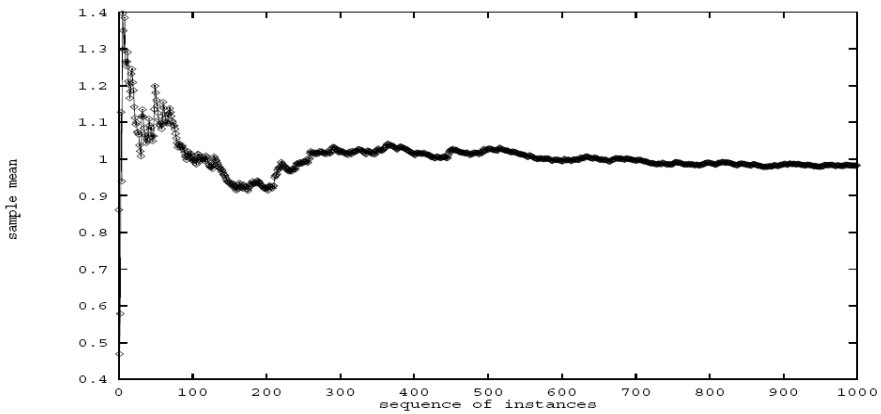
  This is an example phenomenon which is difficult to study based on simple statistics, as mean and variance.

Characterization of Run-time
Heavy Tails

? analyze the mean computational cost to find a solution on a single instance

On the left, the observed behavior calculated over an increasing number of runs.
On the right, the case of data drawn from normal or gamma distributions





- The use of the median instead of the mean is recommended
- The existence of the moments (*e.g.*, mean, variance) is determined by the tails behavior: a case like the left one arises in presence of long tails

Why this happens?
Because heuristics make mistakes which require the backtracking algorithm to explore a large subtree with no solutions.

- Value mistake: a node in the search tree that us a nogood but the parent of the node is not a nogood.

- Backdoor mistake: a selection of a variable that is not in a minimal backdoor, when such a variable is available to be chosen.
  Backdoors are set of variables that if instantiated make the subproblem much easier to solve (polynomially)

Characterization of runtime

Parametric models used in the analysis of run-times to
exploit the properties of the model (eg, the character of tails and completion rate)

Procedure:

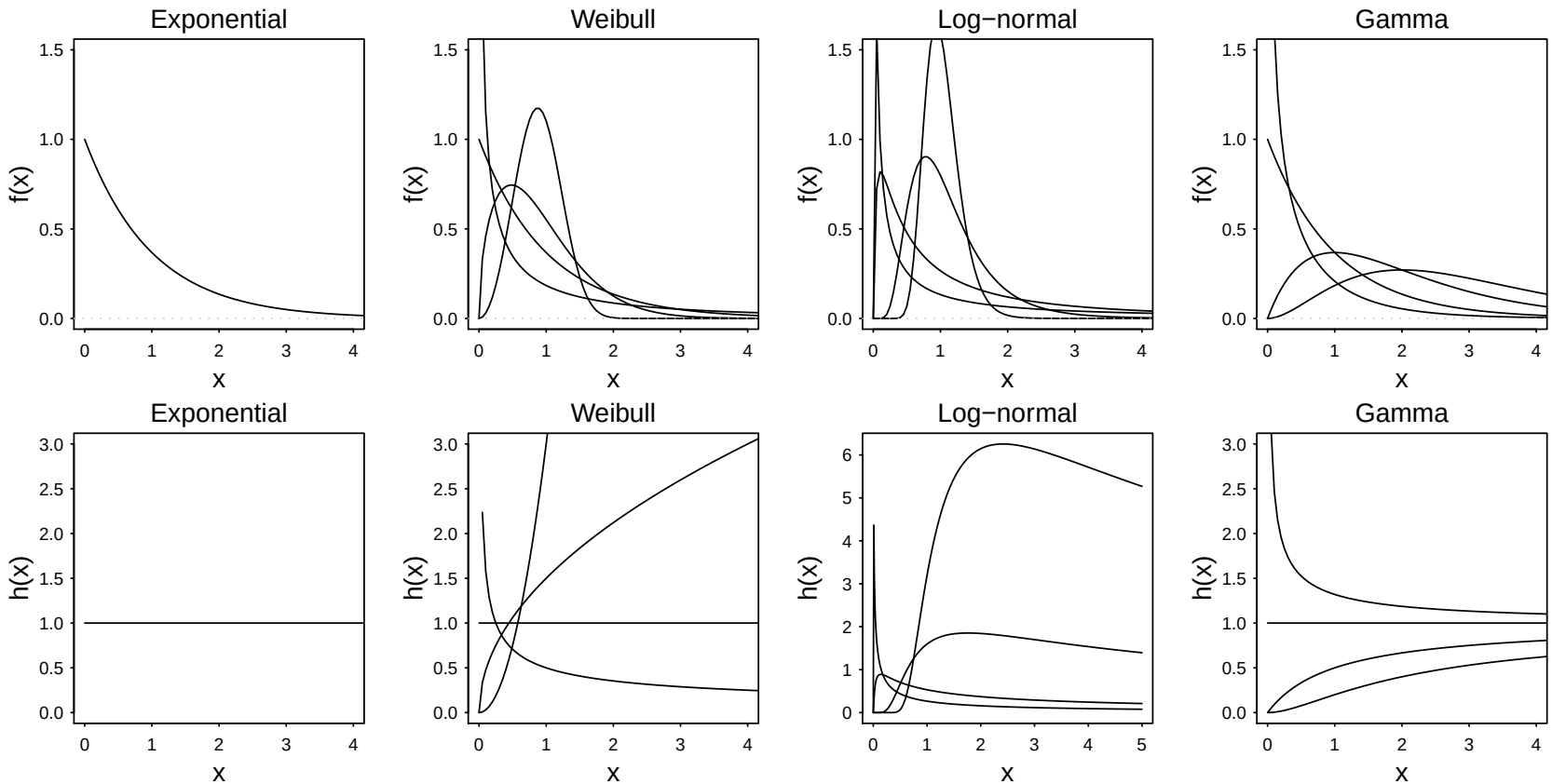
- choose a model
- apply fitting method
  maximum likelihood estimation method:

$$\max_{\theta \in \Theta} \log \prod_{i=1}^{n} p(X_i, \theta)$$

- test the model

Parametric models

The distributions used are [**??**]:



Characterization of Run-time

Motivations for these distributions:

- qualitative information on the completion rate (= hazard function)
- empirical good fitting

To check whether a parametric family of models is reasonable the idea is to make plots that should be linear. Departures from linearity of the data can be easily appreciated by eye.

Example: for an exponential distribution:

$$\log S(t) = -\lambda t \qquad S(t) = 1 - F(t) \text{ is the survivor function}$$

$\rightsquigarrow$ the plot of $\log S(t)$ against t should be linear.

Similarly, for the Weibull the cumulative hazard function is linear on a log-log plot

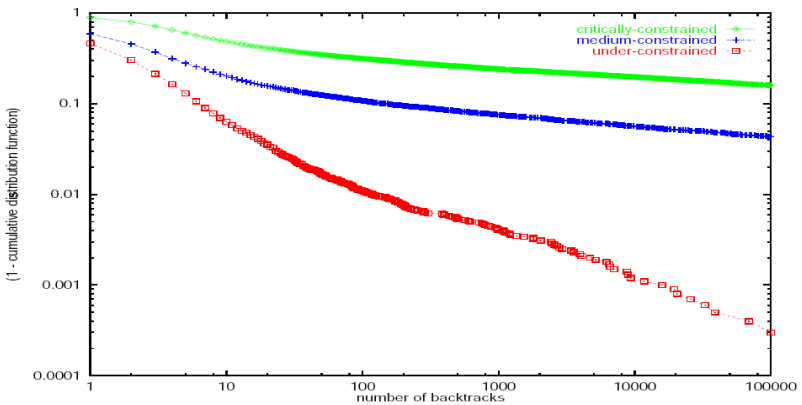
$\rightsquigarrow$ heavy tail if $S(t)$ in log-log plot is linear with slope $-\alpha$

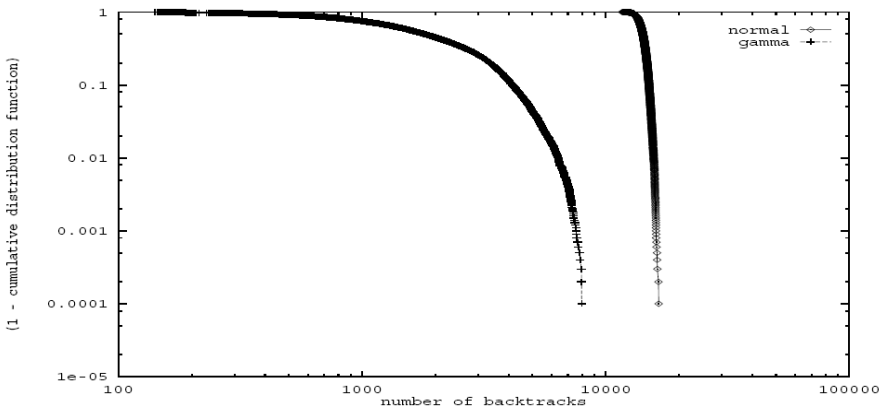
Characterization of Run-time

Heavy Tails

Graphical check using a log-log plot:

- heavy tail distributions approximate linear decay,
- exponentially decreasing tail has faster-than linear decay





Long tails explain the goodness of random restart. Determining the cutoff time is however not trivial.

Extreme Value Statistics

- Extreme value statistics focuses on characteristics related to the tails of a distribution function
  1. extreme quantiles (*e.g.*, minima)
  2. indices describing tail decay

- 'Classical' statistical theory: analysis of means.
  Central limit theorem: $X_1, \ldots, X_n$ i.i.d. with F_X

$$\sqrt{n}\frac{\bar{X} - \mu}{\sqrt{Var(X)}} \xrightarrow{D} N(0,1), \qquad \text{as } n \to \infty$$

  Heavy tailed distributions: mean and/or variance may not be finite!

Extreme Value Statistics

Extreme values theory

- $X_1, X_2, \ldots, X_n$ i.i.d. F_X
  Ascending order statistics $X_n^{(1)} \leq \ldots \leq X_n^{(n)}$
- For the minimum $X_n^{(1)}$ it is $F_{X_n^{(1)}} = 1 - [1 - F_X^{(1)}]^n$ but not very useful in practice as F_X unknown
- Theorem of [Fisher and Tippett, 1928]:
  "almost always" the normalized extreme tends in distribution to a generalized extreme distribution (GEV) as $n \to \infty$.

In practice, the distribution of extremes is approximated by a GEV:

$$F_{X_n^{(1)}}(x) \sim \begin{cases} \exp(-1(1 - \gamma\frac{x-\mu}{\sigma})^{-1/\gamma}, & 1 - \gamma\frac{x-\mu}{\sigma} > 0, \gamma \neq 0 \\ \exp(-\exp(\frac{x-\mu}{\sigma})), & x \in \mathbf{R}, \gamma = 0 \end{cases}$$

Parameters estimated by simulation by repeatedly sampling k values $X_{1n}, \ldots, X_{kn}$, taking the extremes $X_{kn}^{(1)}$, and fitting the distribution.
γ determines the type of distribution: Weibull, Fréchet, Gumbel, ...

Extreme Value Statistics

Tail theory

- Work with data exceeding a high threshold.
- Conditional distribution of exceedances over threshold τ

$$1 - F_\tau(y) = P(X - \tau > y \mid X > \tau) = \frac{P(X > \tau + y)}{P(X > \tau)}$$

- If the distribution of extremes tends to GEV distribution then there exist a Pareto-type function such that for some $\gamma > 0$

$$1 - F_X(x) = x^{-\frac{1}{\gamma}} \ell_F(x), \qquad x > 0,$$

with $\ell_F(x)$ a slowly varying function at infinity.

In practice, fit a function $Cx^{-\frac{1}{\gamma}}$ to the exceedances:
$Y_j = X_i - \tau$, provided $X_i > \tau$, $j = 1, \ldots, N_\tau$.
γ determines the nature of the tail

Characterization of Run-time

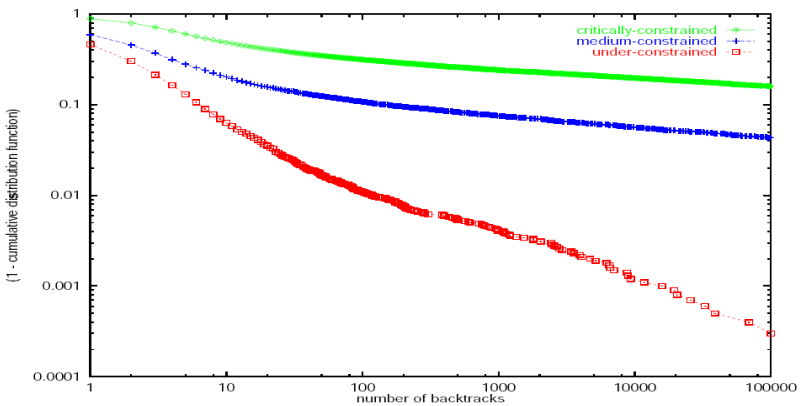
Heavy Tails

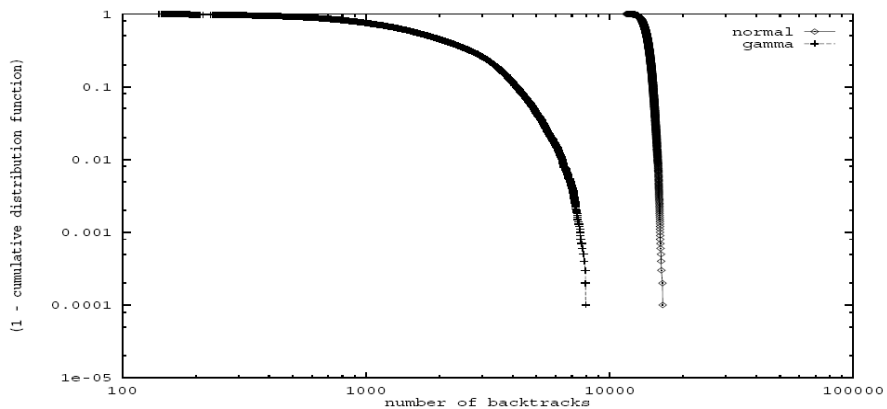
The values estimated for γ give indication on the tails:

- $\gamma > 1$: long tails hyperbolic decay (the completion rate decreases with t) and mean not finite
- $\gamma < 1$: tails exhibit exponential decay

Graphical check using a log-log plot:

- heavy tail distributions approximate linear decay,
- exponentially decreasing tail has faster-than linear decay





Long tails explain the goodness of random restart. Determining the cutoff time is however not trivial.

Randomization

- Randomize the variable ordering
- Randomize tie breaking
- ranking variables within a small factor of the best variable and choosing one at random
- choose a variable with probability proportional to heuristic weight of the variable
- pick one at random from a set of heuristics to use for the selection
- randomize value ordering
- random backwards jump in search space upon backtracking (makes it incomplete)

Wanted: enough different decisions near the top of the search tree

Restart strategies

- Restart strategy: execute a sequence of runs of a randomized algorithm, to solve a single problem instance, stopping the r-th run after a time $\tau(r)$ if no solution is found, and restarting the algorithm with a different random seed
- defined by a function $\tau : \mathbb{N} \to \mathbb{R}^+$ producing the sequence of thresholds $\tau(r)$ employed.
- origins in the field of communication networks
  (Fayolle et al., 1978) derive the optimal timeout for a simple "send and wait" communication protocol, maximizing the transmission rate.
- It can be proved that restart is beneficial under two conditions: if the survival function decreases less fast than an exponential, and if the RTD is improper.

? study Las Vegas algorithms and prove that:

- if $F(t)$ is known:
  the optimal restart strategy is uniform, i.e., $\tau(r) = \tau$, ie,
  $\vec{\tau} = (\tau, \tau, \tau, \tau, \ldots)$.
  Optimal cutoff time $\vec{\tau}^*$ can be evaluated minimizing the expected value of the total run-time T_τ:

$$E\{T_{\vec{\tau}}\} = \frac{\tau - \int_0^\tau F(t)dt}{F(\tau)}$$

  (of course $F(t)$ is not known in practice)

- if $F(t)$ is not known, **?** suggested a universal, non-uniform restart strategy, whose cutoff sequence is composed of powers of 2:

$$\vec{\tau}^{univ} = (1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, 1, \ldots)$$

$$\tau^{univ}(r) := \begin{cases} 2^{j-1} & \text{if } r = 2^j - 1; \\ \tau(r - 2^{j-1} + 1) & \text{if } 2^{j-1} \leq r < 2^j - 1 \end{cases}$$

(everytime a pair of runs of a given length is completed a run of twice that length is execute $\equiv$ when 2^{j-1} is used twice, 2^j is the next)

$\rightsquigarrow$ For all distributions $F(t)$ the performance of $\vec{\tau}^{univ}$ is bounded with high probability with respect to $E_F\{T_{\vec{\tau}^*}\}$:

$$E_F\{T_{\vec{\tau}^{univ}}\} \leq 192 E_F\{T_{\vec{\tau}^*}\}(\log E_F\{T_{\vec{\tau}^*}\} + 5)$$

and the tail decays exponentially. (Note that the result is asymptotic)

$\rightsquigarrow$ It is the best performance it can be achieved by any universal strategy up to a constant factor

Deciding the Restart Strategy in Practice

What counts for primitive operation?

- number of deadends
- distance from a deadend (keep nogoods discovered)
- number of backtracks
- number of nodes visited

For fixed cutoff, which cutoff value?

- instance dependent: hence trial and error
- safer to make larger than too small
- in practice the universal strategy seems slow as it increases too slowly, hence often scaled version: $\vec{\tau}^{univ} = (s, s, 2s, \ldots)$
- Toby Walsh proposes a geometric progression $\vec{\tau}^{g} = (1, s, s^2, \ldots)$ for $1 < s < 2$. Performs well in practice but no guarantees.
- Kautz et al. propose a Bayesian model to predict when run will go long and restart it
- optimization within a given deadline also possible