

Learning Restart Strategies

Matteo Gagliolo^{1,2} Jürgen Schmidhuber^{1,2,3}

¹*IDSIA*, Lugano, Switzerland

²University of Lugano, Faculty of Informatics, Switzerland

³TU, Munich, Germany

Twentieth International Joint Conference
on Artificial Intelligence
January 6-12, 2007, Hyderabad, India

Roadmap

Restart strategies

Learning restart strategies

Bandit problems

Experiments

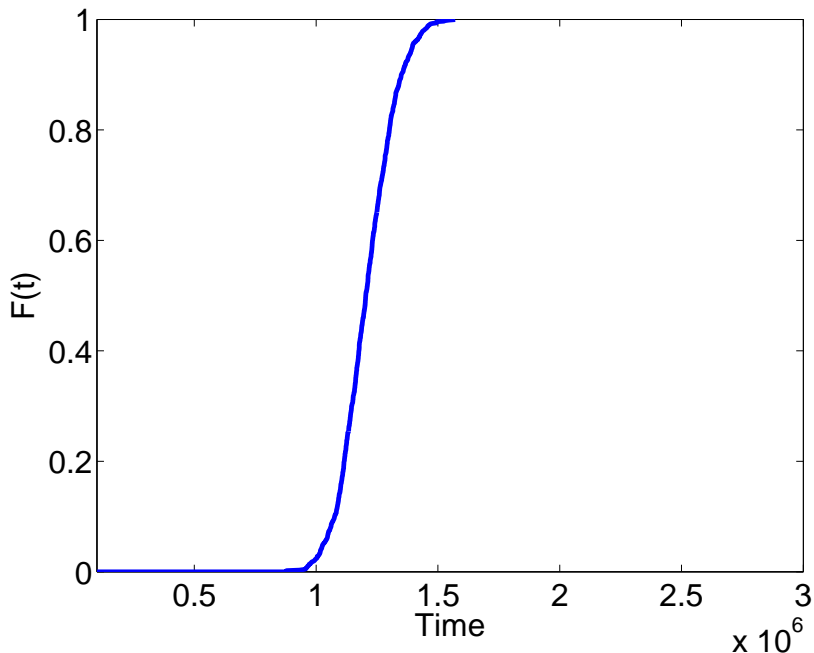
Runtime distribution

Consider a randomized algorithm a solving a *decision* problem (e. g. SAT). The *runtime* spent to obtain a solution is a *random variable*, and can be described by its *cumulative distribution function* (CDF)

$$F(t) = \Pr\{\text{runtime} \leq t\}, \quad F : [0, \infty) \rightarrow [0, 1].$$

F is the *runtime distribution* (RTD) of a .

Runtime distribution



Runtime distribution

- ▶ RTD of a problem **instance**: obtained running ***a*** multiple times on the same instance, with different random seeds.

Runtime distribution

- ▶ RTD of a problem **instance**: obtained running ***a*** multiple times on the same instance, with different random seeds.
- ▶ RTD of a problem **set**: obtained running ***a*** on multiple instances, randomly picked from the set.

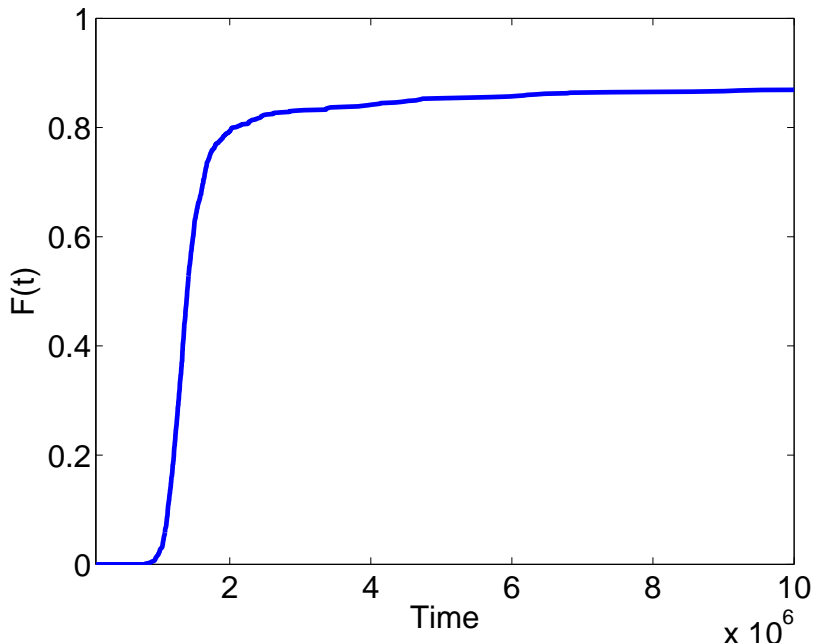
Heavy-tailed RTD

Some RTD present *heavy tails*

$$F(t) \rightarrow_{t \rightarrow \infty} 1 - Ct^{-\alpha}.$$

This means: most runs are short, some take very long...
Example: backtracking SAT/CP solvers on structured underconstrained problems (Gomes et al. 2000)

Heavy-tailed RTD



Restart strategies

A restart strategy consists in executing a sequence of runs of a , in order to solve a same problem instance, stopping each run j after a time $T(j)$ if no solution is found, and restarting the algorithm with a different random seed. (Hoos et al. 2004, Kautz et al. 2002, van Moorsel et al. 2004, Wu 2006)

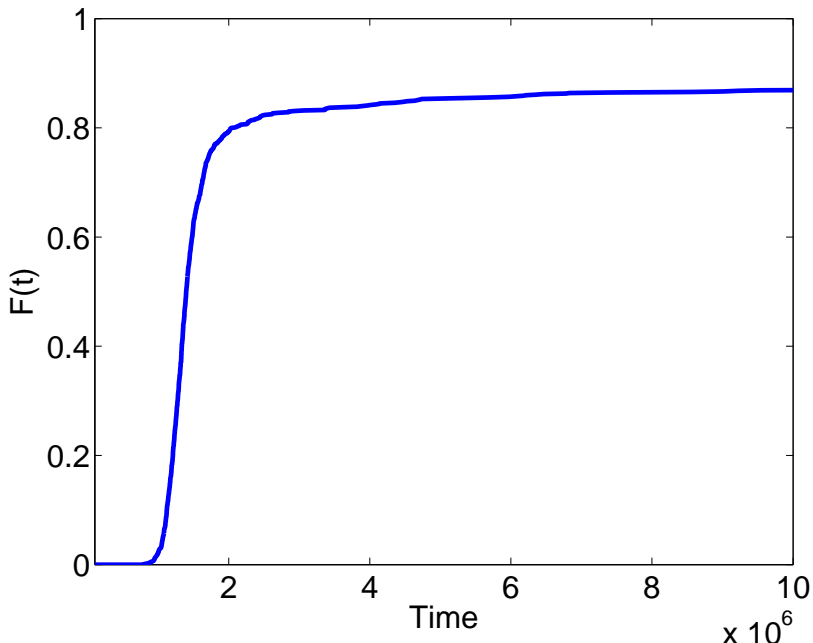
Restart strategies

A restart strategy consists in executing a sequence of runs of a , in order to solve a same problem instance, stopping each run j after a time $T(j)$ if no solution is found, and restarting the algorithm with a different random seed. (Hoos et al. 2004, Kautz et al. 2002, van Moorsel et al. 2004, Wu 2006)

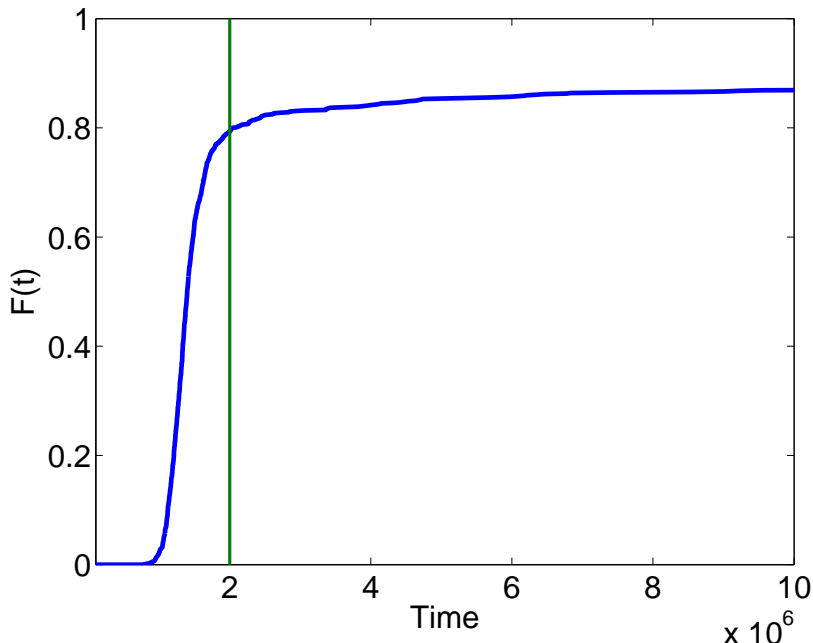
Uniform restart strategy: a constant $T(j) = T$ is used to bound each run. The number of runs R before solution is a Bernoulli process with success probability $F(T)$.

$$p(R) = F(T)(1 - F(T))^{R-1}$$

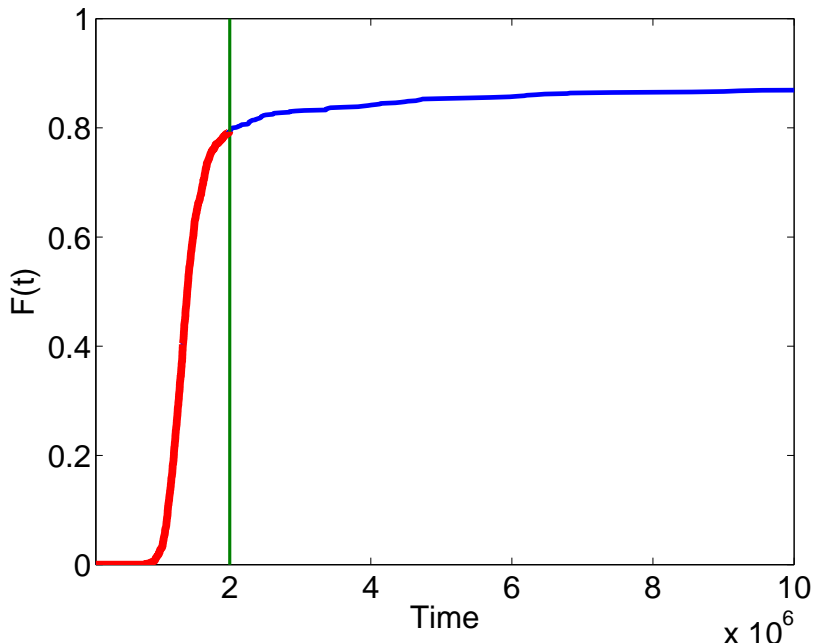
Uniform restart strategy



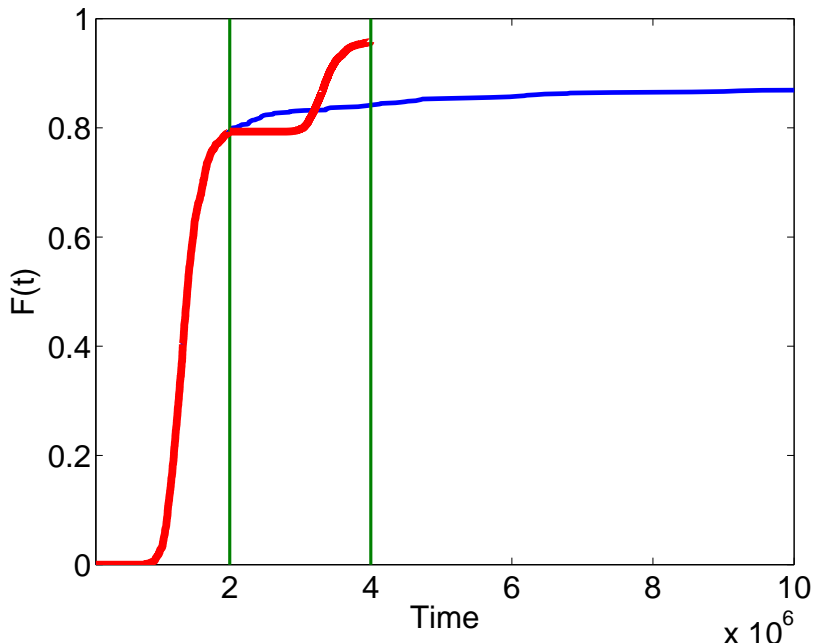
Uniform restart strategy



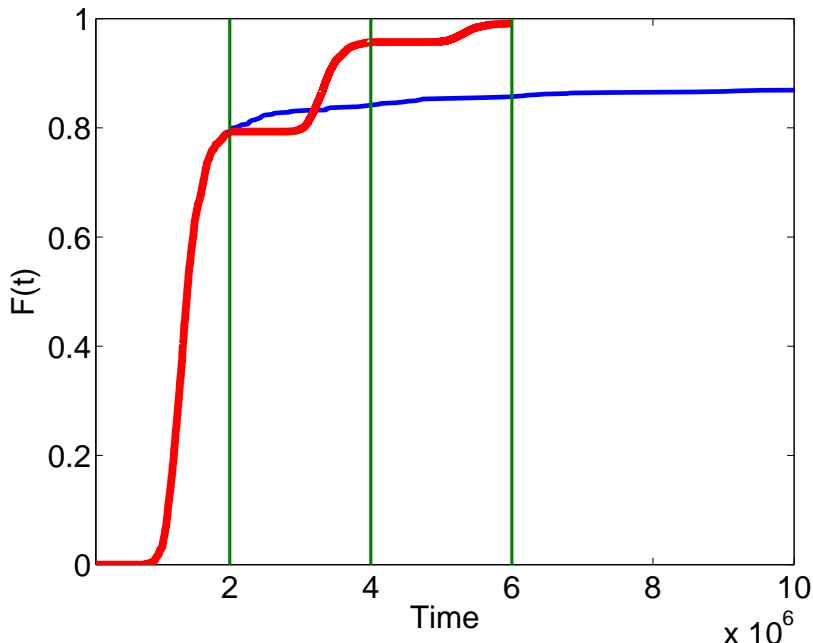
Uniform restart strategy



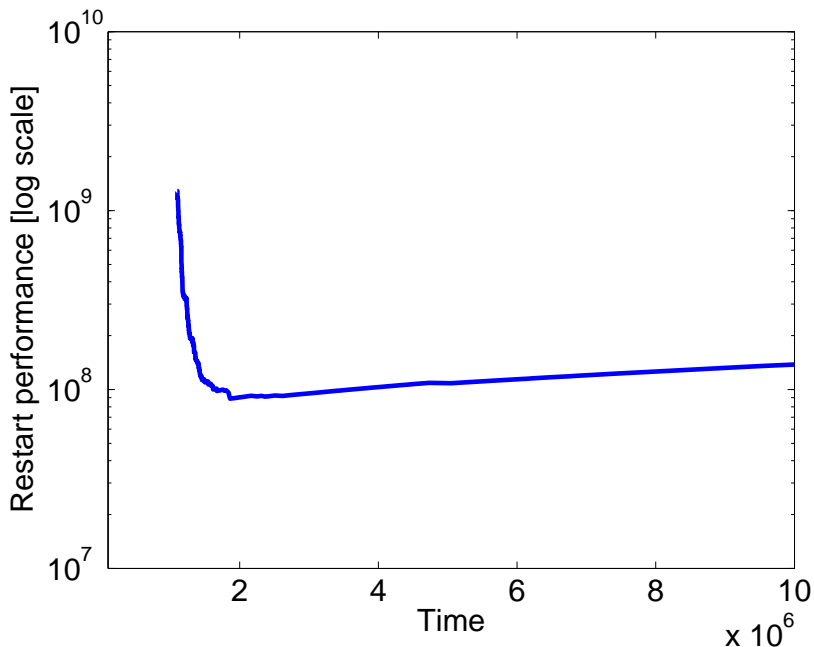
Uniform restart strategy



Uniform restart strategy



Cost of a uniform strategy



Luby's optimal strategy

Luby et. al (1993) proved that the **optimal** restart strategy is uniform. They show that, in this case, the expected value of the total run-time

$$t_T = t^{(R)} + (R - 1)T, \quad R = \min\{r | t^{(r)} \leq T\}$$

can be evaluated as

$$E(t_T) = \frac{T - \int_0^T F(\tau) d\tau}{F(T)}$$

Luby's optimal strategy

Luby et. al (1993) proved that the **optimal** restart strategy is uniform. They show that, in this case, the expected value of the total run-time

$$t_T = t^{(R)} + (R - 1)T, \quad R = \min\{r | t^{(r)} \leq T\}$$

can be evaluated as

$$E(t_T) = \frac{T - \int_0^T F(\tau) d\tau}{F(T)}$$

If F is known I can evaluate the optimal strategy as

$$T(r) = T^* = \arg \min_T E(t_T)$$

Luby's universal strategy

If F is completely unknown I can use Luby's **universal** strategy:

$$T(r) = \{1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, 1, \dots\}$$

whose performance t_U is, with high probability, within a logarithmic factor worse than the *expected* total run-time $E(t_{T^*})$ of the **unknown** optimal strategy.

Learning restart strategies

Couldn't we just **estimate** F ? We could then use the estimated \hat{F} in place of F , to evaluate a **sub**-optimal strategy \hat{T} .

$$\hat{T} = \arg \min_T \frac{T - \int_0^T \hat{F}(\tau) d\tau}{\hat{F}(T)}$$

Learning restart strategies

Couldn't we just **estimate** F ? We could then use the estimated \hat{F} in place of F , to evaluate a **sub-optimal** strategy \hat{T} .

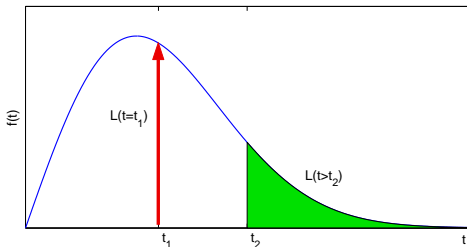
$$\hat{T} = \arg \min_T \frac{T - \int_0^T \hat{F}(\tau) d\tau}{\hat{F}(T)}$$

Issues:

- ▶ F might **differ** on each problem instance.
- ▶ Sampling F might take lot of time... **especially if the algorithm RTD has heavy tails!**

Survival analysis

Training time can be reduced using *censored sampling*



- ▶ Impact of censored sampling on the performance of restart strategies. CP 2006.
- ▶ Learning Dynamic Algorithm Portfolios. AMAI, in press.

Offline learning

Censored sampling simply consists in setting a time threshold, at which a run of a is aborted.

This means that we can use both successful and unsuccessful restarts to learn a model \hat{F} of the RTD.

Offline learning

Censored sampling simply consists in setting a time threshold, at which a run of a is aborted.

This means that we can use both successful and unsuccessful restarts to learn a model \hat{F} of the RTD.

If a set of problems with similar RTD has to be solved, we can set up the following offline learning scheme:

- ▶ solve a part of the problem set with the universal strategy
- ▶ train a model \hat{F} of the RTD of the set using the collected runtime data
- ▶ evaluate Luby's uniform strategy \hat{T} minimizing $E(t_T)$, using \hat{F} in the formula
- ▶ use \hat{T} on the remaining problems

Online learning

Issues:

- ▶ how do I pick the size of the training set?
- ▶ what if the RTD of the single instances differ too much?

Online learning

Issues:

- ▶ how do I pick the size of the training set?
- ▶ what if the RTD of the single instances differ too much?

We are facing an **exploration-exploitation** trade-off.

It would be better to have a principled way of combining the universal and the model-based strategy.

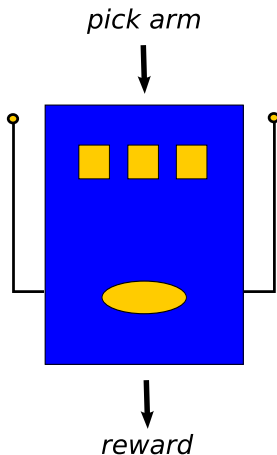
The model \hat{F} and the corresponding \hat{T} could be updated **online**. A *bandit problem solver* could be used to decide which strategy to use.

Bandit problems

A multi-armed bandit (Robbins 1952) has different arms, each generating rewards from different distributions.

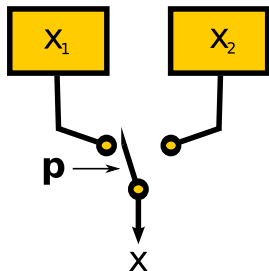
The aim of the game is to maximize the cumulative reward G , or minimize the **re-gret** against the best arm

$$\max_k \sum_j x_k(j) - G$$



Bandit problems

A bandit problem solver (BPS) maps the history of observed rewards to a probability distribution $\mathbf{p} = (p_1, \dots, p_K)$, from which the choice for the successive trial will be picked.

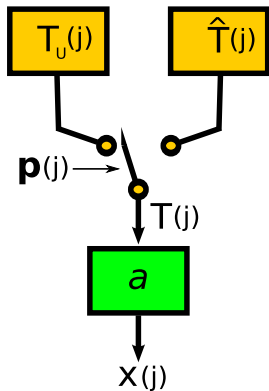


GambleR

We can use a BPS to combine the two strategies.

At every step j

- ▶ pick which strategy to use according to $\mathbf{p}(j)$

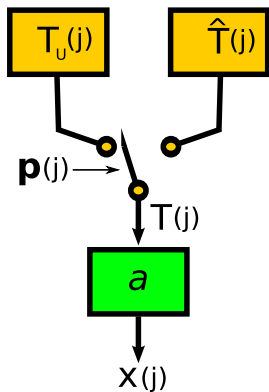


GambleR

We can use a BPS to combine the two strategies.

At every step j

- ▶ pick which strategy to use according to $\mathbf{p}(j)$
- ▶ run a with the chosen restart threshold $T(j)$

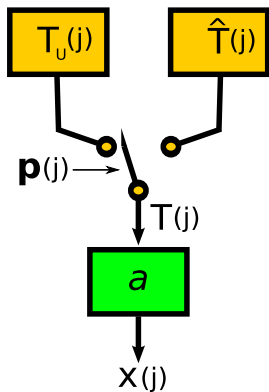


GambleR

We can use a BPS to combine the two strategies.

At every step j

- ▶ pick which strategy to use according to $\mathbf{p}(j)$
- ▶ run a with the chosen restart threshold $T(j)$
- ▶ observe reward $x(j)$

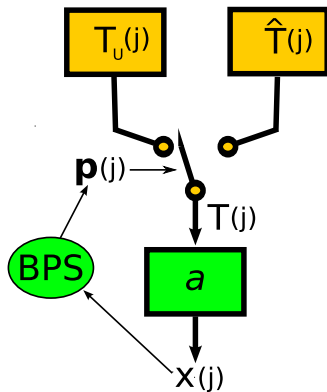


GambleR

We can use a BPS to combine the two strategies.

At every step j

- ▶ pick which strategy to use according to $\mathbf{p}(j)$
- ▶ run a with the chosen restart threshold $T(j)$
- ▶ observe reward $x(j)$
- ▶ update \mathbf{p} using BPS

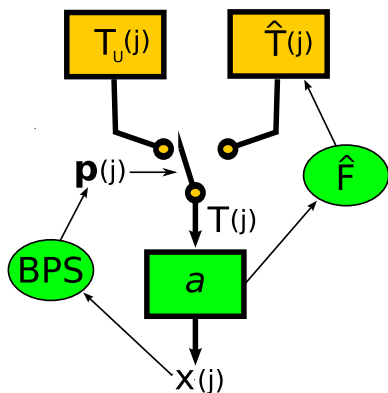


GambleR

We can use a BPS to combine the two strategies.

At every step j

- ▶ pick which strategy to use according to $\mathbf{p}(j)$
- ▶ run a with the chosen restart threshold $T(j)$
- ▶ observe **reward** $x(j)$
- ▶ update \mathbf{p} using BPS
- ▶ update \hat{F} and \hat{T}

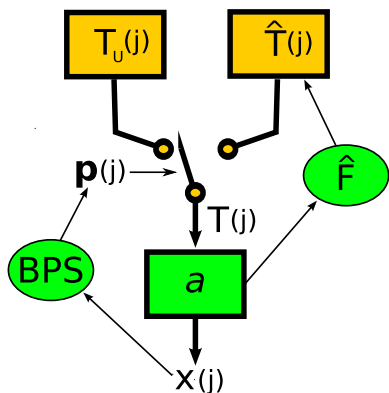


GambleR

We can use a BPS to combine the two strategies.

Related:

- ▶ Max K-armed bandit problem (Cicirello et al. 2005, Streeter et al. 2006)
- ▶ Algorithm selection using RL (Lagoudakis et al. 2000)



Reward attribution

Aim: **minimize** total runtime.

Reward only **successful** strategy.

Reward based on total runtime for a problem instance, accounted separately for each strategy

$$t_T = t^{(R)} + (R - 1)T, \quad R = \min\{r | t^{(r)} \leq T\}$$

Non-stochastic bandit

BPS: Exp3

- ▶ arbitrary *non-oblivious adversarial* reward
- ▶ lower bound on **exploration** probability
- ▶ **parameterless**

Requires reward $x \in [0, 1]$

Auer, Cesa-Bianchi, Freund, Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. 1995.

Reward attribution

Aim: **minimize** total runtime.

Reward only **successful** strategy.

Reward based on total runtime for a problem instance, accounted separately for each strategy

$$t_T = t^{(R)} + (R - 1)T, \quad R = \min\{r | t^{(r)} \leq T\}$$

Simple idea: $x = (t_{max} - t_T) / (t_{max} - t_{min})$

Reward attribution

Aim: **minimize** total runtime.

Reward only **successful** strategy.

Reward based on total runtime for a problem instance, accounted separately for each strategy

$$t_T = t^{(R)} + (R - 1)T, \quad R = \min\{r | t^{(r)} \leq T\}$$

Logarithmic: $x = (\log t_{max} - \log t_T) / (\log t_{max} - \log t_{min})$

Experiments

- ▶ Algorithm: Satz-Rand (Gomes et al. 2000)

Experiments

- ▶ Algorithm: Satz-Rand (Gomes et al. 2000)
- ▶ Benchmark: Morphed GCP (Gent et. al. 1999, SATLIB): 9 groups of 100 instances (group i has structure parameter $p = 2^{-i}$)

Experiments

- ▶ Algorithm: Satz-Rand (Gomes et al. 2000)
- ▶ Benchmark: Morphed GCP (Gent et. al. 1999, SATLIB): 9 groups of 100 instances (group i has structure parameter $p = 2^{-i}$)
- ▶ RTD model \hat{F} : non-parametric product-limit estimator (Kaplan et al., 1958).

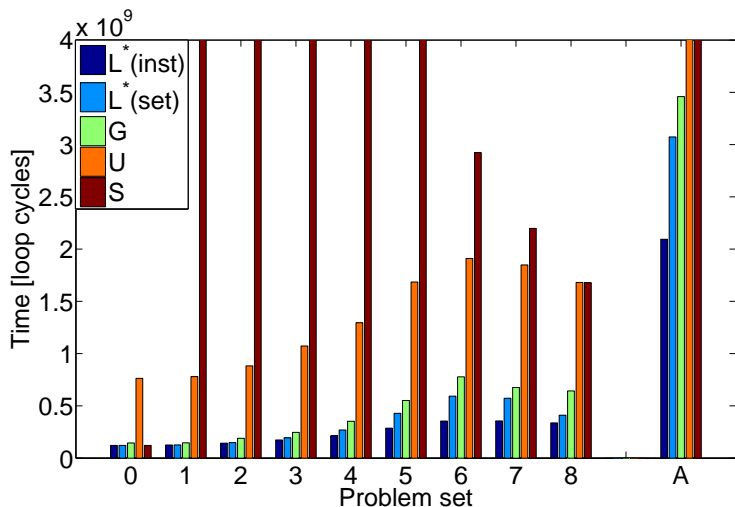
Experiments

- ▶ Algorithm: Satz-Rand (Gomes et al. 2000)
- ▶ Benchmark: Morphed GCP (Gent et. al. 1999, SATLIB): 9 groups of 100 instances (group i has structure parameter $p = 2^{-i}$)
- ▶ RTD model \hat{F} : non-parametric product-limit estimator (Kaplan et al., 1958).
- ▶ BPS: **Exp3** (Auer et al. 1995)

Experiments

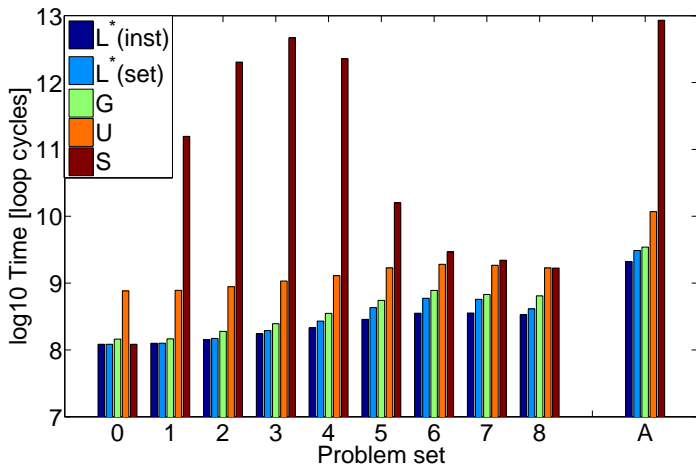
- ▶ Algorithm: Satz-Rand (Gomes et al. 2000)
- ▶ Benchmark: Morphed GCP (Gent et. al. 1999, SATLIB): 9 groups of 100 instances (group i has structure parameter $p = 2^{-i}$)
- ▶ RTD model \hat{F} : non-parametric product-limit estimator (Kaplan et al., 1958).
- ▶ BPS: **Exp3** (Auer et al. 1995)
- ▶ Reward: logarithmic

Experiments



Time to solve each set of problems (20 runs). Per instance ($L^*(inst)$) and per set ($L^*(set)$) optimal restart strategies, GAMBLER (G), universal strategy (U), and Satz-Rand w/out restart (S).

Experiments



Time to solve each set of problems (20 runs). Per instance ($L^*(inst)$) and per set ($L^*(set)$) optimal restart strategies, GAMBLER (G), universal strategy (U), and Satz-Rand w/out restart (S).

Conclusions

Approximations:

- ▶ RTD of the problem **set** (not of the **instance**)

Features:

Issues:

Conclusions

Approximations:

- ▶ RTD of the problem **set** (not of the **instance**)
- ▶ Approximated model of RTD (obtained with heavy *censoring*)

Features:

Issues:

Conclusions

Approximations:

- ▶ RTD of the problem **set** (not of the **instance**)
- ▶ Approximated model of RTD (obtained with heavy *censoring*)

Features:

- ▶ Optimal balance: if \hat{F} is useful, it is exploited, otherwise U is used

Issues:

Conclusions

Approximations:

- ▶ RTD of the problem **set** (not of the **instance**)
- ▶ Approximated model of RTD (obtained with heavy *censoring*)

Features:

- ▶ Optimal balance: if \hat{F} is useful, it is exploited, otherwise U is used
- ▶ Worst case bounds on performance

Issues:

Conclusions

Approximations:

- ▶ RTD of the problem **set** (not of the **instance**)
- ▶ Approximated model of RTD (obtained with heavy *censoring*)

Features:

- ▶ Optimal balance: if \hat{F} is useful, it is exploited, otherwise U is used
- ▶ Worst case bounds on performance
- ▶ **Parameterless** (almost)

Issues:

Conclusions

Approximations:

- ▶ RTD of the problem **set** (not of the **instance**)
- ▶ Approximated model of RTD (obtained with heavy *censoring*)

Features:

- ▶ Optimal balance: if \hat{F} is useful, it is exploited, otherwise U is used
- ▶ Worst case bounds on performance
- ▶ **Parameterless** (almost)
- ▶ **Online**: no preliminary learning required

Issues:

Conclusions

Approximations:

- ▶ RTD of the problem **set** (not of the **instance**)
- ▶ Approximated model of RTD (obtained with heavy *censoring*)

Features:

- ▶ Optimal balance: if \hat{F} is useful, it is exploited, otherwise U is used
- ▶ Worst case bounds on performance
- ▶ **Parameterless** (almost)
- ▶ **Online**: no preliminary learning required

Issues:

- ▶ Worst case performance on a single instance can be very poor

Ongoing and future research

- ▶ Different BPS (Auer et al. 2002), strategies (van Moorsel et al. 2004, Wu 2006), and reward schemes
- ▶ **Conditional** models
- ▶ **Dynamic** features (Kautz et al. 2002)

Further reading (<http://www.idsia.ch/~matteo/>)

[1] Impact of censored sampling on the performance of restart strategies. *CP 2006*

[2] Gambling in a Computationally Expensive Casino: Algorithm Selection as a Bandit Problem. *Online Trading of Exploration and Exploitation - NIPS 2006 Workshop*, Dec. 2006

[3] Learning Dynamic Algorithm Portfolios. *AI & MATH '06 Special Issue of the Annals of Mathematics and Artificial Intelligence*. In press, will be presented at *Lion 2007*, Andalo, Italy, Feb 12-18, 2007

This work was funded by SNF under grant 200020-107590/1