Algorithm Survival Analysis

Matteo Gagliolo and Catherine Legrand

Abstract Algorithm selection is typically based on models of algorithm performance, learned during a separate *offline* training sequence, which can be prohibitively expensive. In recent work, we adopted an *online* approach, in which models of the runtime distributions of the available algorithms are iteratively updated and used to guide the allocation of computational resources, while solving a sequence of problem instances. The models are estimated using survival analysis techniques, which allow to reduce computation time, censoring the runtimes of the slower algorithms. Here, we review the statistical aspects of our online selection method, discussing the bias induced in the RTD models by the competition of different algorithms on the same problem instances.

1 Introduction

This chapter is focused on modeling the performance of algorithms for solving *search* or *decision* problems (Hoos and Stützle, 2004). In this broad category of problems, which includes satisfiability (SAT Gent and Walsh, 1999), and constraint satisfaction in general (Tsang, 1993), there is no quality measure associated with a solution candidate, but only a binary criterion for distinguishing a solution: each problem instance may be characterized by zero, one, or more solutions.

In the more general *optimisation* problem, each solution is characterized by one or more scalar values, indicating its quality. While we do not explicitly consider this

Matteo Gagliolo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
University of Lugano, Faculty of Informatics, Lugano, Switzerland
e-mail: mgagliolo@iridia.ulb.ac.be

Catherine Legrand
Institut de statistique, Université catholique de Louvain, Louvain-la-Neuve, Belgium
e-mail: catherine.legrand@uclouvain.be

category here, note that an optimisation problem can be mapped to a decision problem if a target solution quality can be set in advance: in such case, the corresponding decision problem consists in finding a solution with the desired quality level.

From a computer scientist's point of view, an algorithm is nothing but a piece of software, being executed on a particular machine. In this sense, its performance may be related to the use of various resources (memory, bandwidth, etc.), but the most widely used is definitely the CPU time spent, or *runtime*, and performance modeling is usually aimed at describing and predicting this value.

Consider then a randomized algorithm solving a given problem instance, and halting as soon as a feasible solution is found, or the instance is proved unsolvable. The runtime of the algorithm can be viewed as a random variable, which is fully described by its distribution, commonly referred to as the *runtime distribution* (RTD) in literature about algorithm performance modeling (see e.g. Hoos and Stützle, 2004).

Fortunately for us, we do not have to solve this performance modeling task from scratch. There is already a large corpus of research devoted to modeling the random occurrence of events in time: *survival analysis* (Klein and Moeschberger, 2003; Collet, 2003; Machin et al, 2006) is an important field of statistics, with applications in many areas such as medicine, biology, finance, technology, etc. Researchers in medicine will typically be interested in time to death (hence the name), but the analysis of *time-to-event* data in general occurs in other fields, and other terms may be used to describe the same thing. For example, engineers modeling the duration of a device speak of *failure analysis*, or *reliability theory* (Nelson, 1982). Actuaries setting premiums for insurance companies use the term *actuarial science*.

The rest of the chapter is organized as follows. Section 2 introduces the notions of survival analysis that can be of use when modeling runtime distributions. Section 3 describes the statistical aspects of our approach to algorithm selection (GAMBLETA Gagliolo and Schmidhuber, 2006b, 2008a), focusing on the bias introduced in the models by the use of algorithm portfolios to sample the runtime distributions. Section 4 illustrates the presented concepts in a simple experimental setting. Section 5 references related work, and Section 6 concludes the chapter discussing ongoing research.

2 Modeling Runtime Distributions

In this section, we briefly review the basic concepts from survival analysis, and discuss their application to algorithm performance modeling.

2.1 Basic Quantities and Concepts

Let $T \in [0,\infty)$ be a random variable representing the time from a well-defined origin to the occurrence of a specific event: in our case, the *runtime* of an algorithm, defined as the interval between its starting time and its halting time, regardless of whether the algorithm halts because it found a solution, or because it proved that there are none.

An important distinction has to be made between the RTD of a randomized algorithm on a given problem instance, which can be sampled by solving the instance repeatedly, each time initializing the algorithm with a different random seed; and the RTD of a randomized (or deterministic) algorithm on a set of instances, which can be sampled by running the algorithm repeatedly, each time solving a randomly chosen instance. In the first case, the random aspects are inherent to the algorithm. In the second, an additional random aspect is involved when picking the instance; another random process may be responsible of generating the instances in the set. In the following we will refer to these two distributions as the *RTD of the instance* and the *RTD of the set*, respectively; when no distinction is made, the concept exposed are valid in both cases.

A runtime distribution, as any other distribution, can be described by its *cumulative distribution* function (CDF),

$$F(t) = \Pr\{T \leq t\}, \;\; F : [0,\infty) \rightarrow [0,1], \tag{1}$$

which is an increasing function of time, representing the probability that the algorithm halts within a time t. Another commonly used representation is the *probability density* function (pdf), defined as the derivative of the CDF:

$$f(t) = \frac{dF(t)}{dt}. \tag{2}$$

Integrating the pdf obviously gives us back the CDF:

$$F(t) = \int_0^t f(\tau) d\tau. \tag{3}$$

Time-to-event distributions are often described in terms of the *survival* function

$$S(t) = 1 - F(t), \tag{4}$$

which owes its name to the medical application of this branch of statistics, and represents, in our case, the probability that the algorithm will be still running at time t.

Another important concept in survival analysis is the *hazard* function $h(t)$, quantifying the instantaneous probability of the event of interest occurring at time t, given that it was not observed earlier:

$$h(t) = \lim_{\Delta t \to 0} \frac{\Pr\{t \leq T < t + \Delta t \mid T \geq t\}}{\Delta t} = \lim_{\Delta t \to 0} \frac{\Pr\{t \leq T < t + \Delta t\}}{\Delta t \Pr\{T \geq t\}} = \frac{f(t)}{S(t)}, \quad (5)$$

where $f(t)/S(t) = f(t \mid T \geq t)$ is the pdf conditioned on the fact that the algorithm is still running at time t.

The integral of (5) is termed *cumulative* hazard, and has the following relationship with the survival function:

$$H(t) = \int_0^t h(\tau) d\tau = \int_0^t \frac{dF(\tau)}{S(\tau)} = -\ln S(t), \quad (6)$$

or, conversely,

$$S(t) = \exp(-H(t)). \quad (7)$$

The expected value of runtime, or *expected runtime*, can be evaluated as

$$E\{T\} = \int_0^\infty t f(t) dt = \int_0^1 t dF(t). \quad (8)$$

A *quantile* t_α of the RTD is defined as the time at which the probability $F(t)$ of the event reaches a value $\alpha \in [0,1]$, and can be evaluated solving the equation:

$$t_\alpha = F^{-1}(\alpha). \quad (9)$$

A difference between the typical event of interest in survival analysis, death, and problem solution, is that the latter does not necessarily have to happen, as in general there is no guarantee that an algorithm will halt, whether finding a solution, or proving unsolvability of the instance at hand. This situation can characterize the behavior of an algorithm either on a single instance, if not all the runs halt, or on a set of instances, if the algorithm does not always halt on at least one of them. In both cases, the performance of the algorithm can be described by an *improper* RTD, where $F(\infty)$ is strictly smaller than unity, and represents the probability of halting. The integral of an improper pdf over $[0,\infty)$ does not sum to one:

$$\Pr\{T < \infty\} = F(\infty) = \int_0^\infty f(t) dt < 1, \quad (10)$$

the pdf for a successful run is

$$f(t \mid T < \infty) = \frac{f(t)}{F(\infty)}, \quad (11)$$

and the expected runtime is ∞, as:

$$E\{T\} = \Pr\{T < \infty\} E\{T \mid T < \infty\} + \Pr\{T = \infty\} E\{T \mid T = \infty\} \quad (12)$$

$$= F(\infty) \int_0^\infty t f(t \mid T < \infty) dt + [1 - F(\infty)]\infty = \infty, \quad (13)$$

while quantiles t_α are finite if $\alpha < F(\infty)$, infinite otherwise.

2.2 Censoring

The specific feature that makes survival analysis different from classical statistical analysis is *data censoring*, which refers to a situation in which the exact time of the event of interest is not known, but is known to lie in some (possibly open) interval. For example, in a medical research analyzing survival of patients with a particular disease, some patient may still be alive at the time of performing the analysis of the data. The only information available about one of these subjects is that she has survived up to a certain time, which constitutes a lower bound on her survival time. Such incomplete observation of the event time is called *censoring*[1].

Different types of censoring exist. In *type I* censoring, the event is observed only if it occurs before a prespecified censoring time, which can be different for each observation. *Type II* censoring occurs when we keep collecting data until a prespecified number of events has been observed. Quite often, these two types of censoring will result from experimenter's decisions aimed at reducing the duration of an experiment. Coming back to our medical example, one could decide to follow the patients for a predetermined period of time: all patients still alive at the end of this predetermined period will have a censored survival time. In this case, the duration of the experiment is fixed, while the number of observed deaths is a random variable. As an alternative, one could decide to end the experiment as soon as a predetermined number of deaths have been observed, the other patients' lifetimes being censored at the time the experiment ends. In this case, the number of observed events is fixed, while the duration of the experiment is a random variable. In other situations, the censoring mechanism is not controlled by the experimenter. This occurs quite frequently in medical applications, for example when patients are "lost to follow-up" in the course of the study (i.e. they stop participating in it).

In any case, one has to be very careful with discarding incomplete data as this can result in an extremely biased model. Appropriate techniques have therefore been developed to take these censored observations into account, as we will see in the next subsection.

[1] More precisely, *right* censoring. One also speaks about *interval censoring* (we only know that the event occurred in between two time points without knowing when) and *left censoring* (we only know that the event occurred before a certain time). Right censoring is the most common, and is the only one we will consider here.

2.3 Estimation in Survival Analysis

In this section we consider the problem of estimating the survival distribution of an event, and modeling the impact of covariates on survival time, based on experimental data.

A sample of censored survival data is usually represented as a set of realizations $D = \{(t_1, c_1), \ldots, (t_n, c_n)\}$ of a pair of random variables (T, C), generated from another pair of *latent* random variables, the time to the actual event T_e, and the time to censoring T_c. For each realization in the sample, only the minimum of these two variables $T = \min\{T_e, T_c\}$ is observed, along with the *event indicator* $C = I(T_e \leq T_c)$, which is 1 if the event of interest was observed, 0 if it was censored. Most estimation techniques require the time to event T_e and the time to censoring T_c to be statistically independent (*uninformative* censoring).

Methods for estimating a distribution can be broadly classified as parametric or non-parametric. In parametric methods, a parametric function is assumed to represent the time-to-event distribution. In non-parametric methods, no assumption is made on the distribution, and estimates are based solely on the observed data.

2.3.1 Parametric Methods

In this class of methods, a function $f(t; \boldsymbol{\theta})$ is assumed to represent the pdf of the runtime t. Several distributions are used to model survival times data, e.g. exponential, Weibull, lognormal, etc. The choice of a particular distribution is clearly a delicate issue, and should be based on empirical evidence: graphical diagnostic tests are available to evaluate a parametric model (Klein and Moeschberger, 2003, Chapter 12).

The parameter $\boldsymbol{\theta}$ is estimated based on the data. Various approaches can be followed in this sense, usually based on the concept of the *likelihood* of the parameter in light of the data, that is, $L(\boldsymbol{\theta}; D) = \Pr\{D \mid \boldsymbol{\theta}\}$. In a *frequentist* context, *maximum likelihood* methods estimate the parameter to be the one that maximizes the likelihood. In *Bayesian* approaches, the parameter value is assumed to be drawn from a *prior* distribution $\Pr\{\boldsymbol{\theta}\}$, and inference is based on the *posterior* probability $\Pr\{\boldsymbol{\theta} \mid D\} \propto \Pr\{D \mid \boldsymbol{\theta}\} \Pr\{\boldsymbol{\theta}\}$, either considering its mode as point estimate of $\boldsymbol{\theta}$, or evaluating *credible intervals* based on the posterior (Bishop, 1995; Mackay, 2002). As both approaches require computing the likelihood, in the following we illustrate how this can be done in the case of censored survival data (Klein and Moeschberger, 2003).

If $g(.)$ and $G(.)$ are respectively the pdf and the CDF of the censoring times, the contribution of a non-censored observation $(t_i, c_i = 1)$ to the likelihood is

$$L(\boldsymbol{\theta}; t_i, c_i = 1) = (1 - G(t_i)) f(t_i; \boldsymbol{\theta}), \tag{14}$$

while the contribution of a censored observation $(t_i, c_i = 0)$ is

$$L(\boldsymbol{\theta}; t_i, c_i = 0) = (1 - F(t_i; \boldsymbol{\theta}))g(t_i). \tag{15}$$

A sample of size n will be a combination of censored and non-censored observations, and, assuming independence among the n realizations of (T,C), the likelihood will be given by

$$L(\boldsymbol{\theta}; D) = \prod_{i=1}^{n} [(1 - G(t_i))\, f(t_i; \boldsymbol{\theta})]^{c_i} [(1 - F(t_i; \boldsymbol{\theta}))g(t_i)]^{1-c_i}. \tag{16}$$

If we assume independent censoring (Liang et al, 1995; Fleming and Harrington, 1991), the factors $(1 - G(t_i))$ and $g(t_i)$ are not informative for the inference on the parameters and can be removed from the likelihood, which can then be written

$$L(\boldsymbol{\theta}; D) \propto \prod_{i=1}^{n} f(t_i; \boldsymbol{\theta})^{c_i} (1 - F(t_i; \boldsymbol{\theta}))^{1-c_i}. \tag{17}$$

When the proposed parametric distribution for runtimes is appropriate, parametric inference is more efficient than non-parametric (Pintilie, 2006), meaning that the resulting model will converge faster to the underlying distribution as the sample increases. Conversely, the use of a parametric function which does not fit the data well can produce an inefficient model.

2.3.2 Non-parametric Methods

If there is no censored data, a simple non-parametric estimate of the survival function can be obtained based on the empirical estimate of the CDF

$$\hat{F}(t) = \sum_{i:t_i \leq t} \frac{1}{n} \tag{18}$$

where $t_1 < t_2 < \ldots < t_n$ are the ordered survival times observed.

In the presence of censoring, the most commonly used non-parametric estimator of the survival function is the product-limit estimate proposed by Kaplan and Meyer (1958). This method is based on the idea of estimating the hazard at each time t_i in the sample as the portion of patients still alive, or "at risk" (in our case: the algorithms still running), experiencing an event at this time point:

$$\hat{h}(t_i) = \frac{\sum_{i:t_j = t_i, c_j = 1} 1}{\sum_{i:t_j \geq t_i} 1} = \frac{d_i}{n_i}, \tag{19}$$

where c_i is the event indicator, d_i is then the number of events observed at time t_i, and n_i is the number of observations still at risk at time t_i. An estimate of the survival function based on (19) is given by the *product limit* estimator, also known as Kaplan-Meier (KM) estimator:

$$\hat{S}(t) = \prod_{i:t_i \leq t} (1 - \hat{h}(t_i)) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i}), \tag{20}$$

from which one can estimate the CDF as

$$\hat{F}(t) = 1 - \hat{S}(t). \tag{21}$$

The cumulative hazard can be obtained from (20) using the following relation:

$$\hat{H}(t) = -\log(\hat{S}(t)), \tag{22}$$

or based directly on (19), as proposed by Nelson (1972); Aalen (1978):

$$\hat{H}(t) = \sum_{i:t_i < t} \frac{d_i}{n_i}. \tag{23}$$

The two estimators are similar for large sample sizes.

In this and other nonparametric methods, $\hat{F}(t)$, $\hat{S}(t)$, $\hat{H}(t)$ are "stepwise" functions, that change their value only at uncensored observations $\{t_i \mid c_i = 1\}$, and are defined until the largest one; if the largest observation t_n is censored, the resulting estimate $\hat{F}(t_n)$ will be < 1. The hazard estimate $\hat{h}(t)$ (19) is discrete, defined only at the event times $\{t_i\}$ in the sample D. In order to obtain meaningful predictions also for $t \notin \{t_i\}$, hazard estimates can be *smoothed* (Wang, 2005).

A review of nonparametric estimation for censored survival data in a Bayesian framework can be found in (Ibrahim et al, 2001).

2.3.3 Regression Models

If additional information is available about each observation, in the form of some *covariates* or *features* $\mathbf{x} \in \mathbb{R}^d$, it may be of interest to investigate its relationship with the time to event. This can be done estimating a conditional model, or *regression* model, of the survival distribution $S(t \mid \mathbf{x})$ or, as quite often done, of the baseline hazard $h(t \mid \mathbf{x})$. In our case, consider again the RTD of a set of instances, and the RTDs of each instance in the set, which will in general be different from the RTD of the set. When performing algorithm selection, with the aim of reducing runtime, it is clearly not advisable to model the RTD of each instance, as this would require obtaining multiple runtimes for each instance. As we expect similar instances to have similar RTDs, we could instead identify features of the problem instance that can be related to the runtime of the algorithm, in order to recover estimates that are closer to the actual RTD of each instance. In another situation, one might want to study the impact of the various parameters of an algorithm on its runtime distribution: in this case a sample could be formed using different values of the parameters, and the parameters could be used as covariates.

A parametric regression model can be based on a pdf $f(t \mid \boldsymbol{\theta})$ in which the parameter is itself a parametric function of the covariate $\boldsymbol{\theta}(\mathbf{x}; \boldsymbol{\beta})$ (see, e.g. Bishop,

1995, par 6.4): for example, an exponential distribution $f(t;\theta) = \theta \exp(-\theta t)$ can be cascaded with a linear model $\theta(\mathbf{x};\boldsymbol{\beta}) = \boldsymbol{\beta} \cdot \mathbf{x}$, obtaining a parametric conditional model:

$$f(t \mid x;\boldsymbol{\beta}) = \theta(\mathbf{x};\boldsymbol{\beta}) \exp(-\theta(\mathbf{x};\boldsymbol{\beta})t) = (\boldsymbol{\beta} \cdot \mathbf{x}) \exp(-\boldsymbol{\beta} \cdot \mathbf{x}). \tag{24}$$

Various other parametric models have been proposed to study the dependency between covariates and the survival time. Several of them model this dependency via the hazard or the cumulative hazard function, and the most popular one is probably the *proportional hazards* model, or Cox model (Cox, 1972):

$$h(t \mid \mathbf{x}) = h_0(t) \exp(\boldsymbol{\beta} \cdot \mathbf{x}) \tag{25}$$

In this model it is assumed that the covariates act multiplicatively on a baseline hazard $h_0(t)$: if this is left unspecified, (25) is termed a *semi-parametric* model. With such a model, the ratio of hazards of two observations is independent of time, and is determined by the corresponding covariates. Among other (semi-)parametric models, one can cite the additive risk model (Klein and Moeschberger, 2003, Chapter 10), in which the covariates act additively on the hazard function; and the accelerated failure time model (Cox and Oakes, 1984), in which the covariates are related to the logarithm of survival times.

Only limited work has been done on fully non-parametric regression models (see reviews by Keilegom et al, 2001; Spierdijk, 2005). Beran (1981) and later works (e.g. Akritas, 1994; Wichert and Wilke, 2005) are based on the idea of estimating a conditional hazard function, as in (19), but with 1 replaced by a *kernel* function $K(\mathbf{x},\mathbf{y}) : \mathbb{R}^d \times \mathbb{R}^d \to [0,\infty)$ representing a similarity measure in covariate space: a prediction of the hazard at time t for an individual with covariate $\mathbf{x}$ is given by

$$\hat{h}(t \mid \mathbf{x}) = \frac{\sum_{i:t_i=t,c_i=1} K(\mathbf{x},\mathbf{x}_i)}{\sum_{j:t_j \geq t} K(\mathbf{x},\mathbf{x}_j)}. \tag{26}$$

In all these conditional models, the covariates are constant over time. In some situations, it might be interesting to model the effect of covariates which vary over time. In our case, *dynamic* information about the algorithm could be available, for example the state of some internal variables. One possibility is to treat it as a *time-varying* covariate (Li and Doss, 1995; Nielsen and Linton, 1995); another possibility is to consider *longitudinal data* analysis techniques (Fitzmaurice et al, 2008).

2.4 Competing Risks

In some situations, occurrence of another type of event may "compete" with the event of interest, and prevent the observation of the latter for some individuals. For example, in a medical setting, one might want to study the distribution of time to death T_A, due to a particular disease or condition A, and gather a sample of individuals affected by it: at the end of the study, some patients will have died of condition A,

and their lifetime recorded as uncensored, while others will be still alive, and their time to death censored. But some of the patients might have died from a different condition, *B*. If we want to estimate the distribution of time to death of *A*, we may view death of *B* as a censoring event, as it prevented to know what T_A *would* have been if the patient did not die of *B* at a time $T_B < T_A$. Death from *B* is termed a *competing risk* (Pintilie, 2006; Putter et al, 2006).

As death of *B* is an event that we are not interested in modeling, one possibility is to consider it as an additional censoring mechanism. This may pose a problem when the distribution of the event of interest and the one of the competing risk are not independent, for example if both conditions *A* and *B* become more likely with old age. This would obviously result in a censoring mechanism that is not independent from the distribution of time to events, thus not satisfying assumption on which most estimators rely.

When modeling runtime distributions, this situation can generally be avoided, as one can always "kill" the same "patient" (i.e. the same problem instance) multiple times, using different algorithms independently, as different "causes of death". This is not the case of our online algorithm selection method (Sect. 3), where we strive to avoid unnecessary computations, solving each instance only once, and using the fastest algorithm as a censoring mechanism for sampling runtimes of the slower ones; so the topic discussed in this subsection is more relevant to our work, rather than to RTD modeling in general.

Consider a set of *K* competing risks, represented by *K* random variables $T_1, \ldots, T_K$, each with its own distribution, sampled with some censoring mechanism with threshold T_c. For each individual, we only observe a pair of random variables (T, C), where $T = min\{T_1, \ldots, T_K, T_c\}$ is the event time, and the event indicator *C* is 0 if the individual is still alive at T_c, *k* if the individual died of the *k*-th risk, so it is a discrete random variable with value in $\{0, 1, \ldots, K\}$.

Consider the *joint survival function* of the times T_k

$$Z(t_1, \ldots, t_K) = \Pr\{T_1 > t_1, \ldots, T_K > t_k\}. \tag{27}$$

The probability of the event $\{T_k > t_k\}$ is the *marginal* survival function

$$\Pr\{T_k > t_k\} = Z_k(t_k) = Z(0, \ldots, t_k, \ldots, 0). \tag{28}$$

If the event times $\{T_k\}$ are independent, the joint probability (27) equals the product of the probabilities of the single events (28):

$$Z(t_1, \ldots, t_K) = \prod_{j=1}^{K} Z_j(t_j). \tag{29}$$

Unfortunately, if the data is gathered as described above, with at most one uncensored event time per individual, the independence assumption cannot be tested. Neither the joint survival function (27), nor the marginals (28) can be identified, and Kaplan-Meier estimates of the marginals for each risk will be biased (Tsiatis, 1975; Putter et al, 2006).

One can always estimate the overall survival probability for an individual,

$$S(t) = \Pr\{T_1 > t, \ldots, T_K > t\} = Z(t_1 = t, t_2 = t, \ldots, t_K = t), \tag{30}$$

with a product limit estimate (20) from the data, considering all recorded events as they were the same event. Other quantities that can be estimated (Pintilie, 2006; Putter et al, 2006) are the *cause-specific hazard*, which is defined as the hazard of failing for a specific cause k:

$$\lambda_k(t) = \lim_{\Delta t \to 0} \frac{\Pr\{t \leq T < t + \Delta t, C = k \mid T \geq t\}}{\Delta t}, \tag{31}$$

from which one can obtain the cumulative cause-specific hazard,

$$\Lambda_k(t) = \int_0^t \lambda_k(\tau) d\tau, \tag{32}$$

and the *subsurvival* function,

$$R_k(t) = \exp(-\Lambda_k(t)), \tag{33}$$

with

$$S(t) = \prod_k R_k(t) = \exp(-\sum_k \Lambda_k(t)). \tag{34}$$

The *cumulative incidence function* or *subdistribution function* for the cause k is defined as the probability of failing from cause k before time t:

$$I_k(t) = \Pr\{T \leq t, C = k\} = \int_0^t \lambda_k(\tau) S(\tau) d\tau. \tag{35}$$

This is an improper distribution, as $I(\infty) = \Pr\{C = k\} = \Pr\{T_j > T_k \forall j \neq k\}$, which can be smaller than 1.

3 Model-based Algorithm Selection

Attempts to automate *algorithm selection* are not new (Rice, 1976), and are motivated by the fact that alternative algorithms for solving the same problem often display different performance on different problem instances, such that there is no single "best" algorithm. Selection is typically based on predictive models of algorithm performance, which are either assumed to be available, or are learned during an initial training phase, in which a number of problem instances are solved with each of the available algorithms, in order to sample their performances.

As we have seen, the natural performance model for a decision problem solver is its runtime distribution. The advantage of survival analysis techniques is that they allow to reduce the time spent for sampling the RTD, as excessively long runs of

the algorithm can be censored, and still contribute to the model. It is to note that there is an obvious *trade-off* between the computational cost of the sampling experiment, and the portion of uncensored events observed, which in turn will have an impact on the precision of the obtained model: a higher portion of uncensored events will imply a larger time cost, but it will also result in a more precise estimate of model parameters. In the context of algorithm selection techniques, this trade-off should rather be measured between training time and the gain in performance resulting from the use of the learned model: in this sense, the required model precision can be much lower than expected. An example of such a situation is given in (Gagliolo and Schmidhuber, 2006a) where this trade-off is analyzed in the context of restart strategies[2], reporting the training times, and the resulting performance, of a model-based restart strategy, where the RTD model is learned with different censoring thresholds. Inspired by this idea, Gagliolo and Schmidhuber (2007) proposed an *online* method for learning a restart strategy (GAMBLER), in which a model of the RTD of an algorithm is iteratively updated *and* used to evaluate an optimal restart strategy for the algorithm. Gagliolo and Schmidhuber (2006b, 2008a), adopted a similar online approach (GAMBLETA) to allocate computational resources to a set of alternative algorithms. In both cases, the trade-off between exploration and exploitation is represented as a *multi-armed bandit problem* (Auer et al, 2003). This game consists of a sequence of trials against a K-armed slot machine: at each trial, the gambler chooses one of the available arms, whose losses are randomly generated from different unknown distributions, and incurs in the corresponding loss. The aim of the game is to minimize the *regret*, defined as the difference between the cumulative loss of the gambler, and the one of the best arm.

In GAMBLETA (Gagliolo and Schmidhuber, 2006b, 2008a), the different "arms" of the bandit are represented by different *time allocators* (TA), that is, different techniques for allocating computation time to the available algorithms. To understand the rationale of this approach, consider a situation in which we have a way to allocate time based on a model of the RTDs of the algorithms, i.e. a *model-based* TA. In an online setting, in which the models are gradually improved as more instances are solved, and a larger runtime sample is available, the performance of this TA will obviously be very poor at the beginning of the problem sequence. This model-based TA can be combined with a more "exploratory" one, such as a parallel portfolio of all available algorithms, and, at an upper level, the bandit problem solver (BPS) can be used to pick which TA to use for each subsequent problem instance, based on the previous performances of each TA. Intuitively, the BPS will initially penalize the model-based allocator, but only until the model is good enough to outperform the exploratory allocator. Alternative allocation techniques can be easily added as additional "arms" of the bandit.

To introduce some notation, consider a portfolio $\mathscr{A} = \{a_1, a_2, ..., a_K\}$ of K algorithms, which can be different algorithms, different parametrization of the same algorithm, or even copies of the same randomized algorithm differing only in the

[2] A restart strategy consists in executing a sequence of runs of a randomized algorithm, in order to solve a given 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.

random seed, or any combination thereof. The runtime of each a_k on the current problem instance is a random variable, T_k. The algorithms solve the same problem instance in parallel, and share the computational resources of a single machine according to a *share* $\mathbf{s} = \{s_1, .., s_K\}, s_k \geq 0, \sum_{i=1}^{K} s_k = 1$; i.e. for any amount t of machine time, a portion $t_k = s_k t$ is allocated to a_k. Here and in the following we assume an "ideal" machine, with no task switching overhead. Algorithm selection can be represented in this framework by setting a single s_k value to 1, while a uniform algorithm portfolio would have $\mathbf{s} = (1/K, ..., 1/K)$.

A time allocator (TA) can be defined as a device setting the share $\mathbf{s}$ for the algorithms in the portfolio, possibly depending on the problem instance being solved. A *model-based* TA can be implemented mapping the estimated RTDs of the algorithms on the current instance to a share value $\mathbf{s}$. Gagliolo and Schmidhuber (2006b) proposed three different analytic approaches, in which time allocation is posed as a function optimisation problem: the RTD of a portfolio is expressed as a function of the share $\mathbf{s}$, and the value of $\mathbf{s}$ is set optimizing some function of this RTD, for example the expected runtime. The next sections will illustrate these two steps.

3.1 RTD of an Algorithm Portfolio

An a_k that can solve the problem in a time t_k if run alone, will spend a time $t = t_k/s_k$ if run with a share s_k. If the runtime distribution $F_k(t_k)$ of a_k on the current problem is available, one can obtain the distribution $F_{k,s_k}(t)$ of the event "a_k solved the problem after a time t, using a share s_k", by simply substituting $t_k = s_k t$ in F_k:

$$F_{k,s_k}(t) = F_k(s_k t). \tag{36}$$

If the execution of all the algorithms is stopped as soon as one of them solves the problem, the resulting duration of the solution process is a random variable, representing the runtime of the parallel portfolio, whose value is determined by the share $\mathbf{s}$, and by the random runtimes T_k of the algorithms in the set:

$$T_{\mathscr{A}} = \min\{\frac{T_1}{s_1}, \frac{T_2}{s_2}, \ldots, \frac{T_K}{s_K}\}. \tag{37}$$

Its distribution $F_{\mathscr{A},\mathbf{s}}(t)$ can be evaluated based on the share $\mathbf{s}$, and the $\{F_k\}$. The evaluation is more intuitive if we reason in terms of the survival distribution: at a given time t, the probability $S_{\mathscr{A},\mathbf{s}}(t)$ of *not* having obtained a solution is equal to the joint probability that *no single algorithm* a_k has obtained a solution within its time share $s_k t$. Assuming that the solution events are independent for each a_i, this joint probability can be evaluated as the product of the individual survival functions $S_k(s_k t)$

$$S_{\mathscr{A},\mathbf{s}}(t) = \prod_{k=1}^{K} S_k(s_k t). \tag{38}$$

Given (6), equation (38) has an elegant representation in terms of the cumulative hazard function. Apart from the terms s_k, (39) is the method used by engineers to evaluate the failure distribution of a *series* system, which stops working as soon as one of the components fail, based on the failure distribution for each single component (Nelson, 1982):

$$H_{\mathscr{A},\mathbf{s}}(t) = -\ln(S_{\mathscr{A},\mathbf{s}}(t)) = \sum_{k=1}^{K} -\ln(S_k(s_k t)) = \sum_{k=1}^{K} H_k(s_k t). \tag{39}$$

The above formulas rely on the assumption of independence of the runtime values among the different algorithms, which allows the joint probability (38) to be expressed as a product. This assumption is met only if the S_k represent the runtime distributions of the a_k on the particular problem *instance* being solved. If instead the only S_k available capture the behavior of the algorithms on a *set* of instances, which includes the current one, independence cannot be assumed: in this case, the methods presented should be viewed as heuristics. In a less pessimistic scenario, one could have access to models of the S_k conditioned on features, or *covariates*, $\mathbf{x}$ of the current problem. In such a case the *conditional* independence of the runtime values would be sufficient, and the resulting joint survival probability could still be evaluated as a product

$$S_{\mathscr{A},\mathbf{s}}(t \mid \mathbf{x}) = \prod_{i=1}^{K} S_k(s_k t \mid \mathbf{x}). \tag{40}$$

3.2 Model-based Time Allocators

Once the RTD of the portfolio is expressed as a function of the share $\mathbf{s}$, as in (38, 40), the problem of allocating machine time can be formulated as an optimisation problem. In (Gagliolo and Schmidhuber, 2006b), the following alternatives are proposed:

Expected time The expected runtime value $E_{\mathscr{A},\mathbf{s}}(t) = \int_0^\infty t\, f_{\mathscr{A},\mathbf{s}}(t)dt$ can be obtained, and minimized with respect to $\mathbf{s}$:

$$\mathbf{s} = \arg\min_{\mathbf{s}} E_{\mathscr{A},\mathbf{s}}(t). \tag{41}$$

Contract If an upper bound, or *contract*, t_u on runtime is imposed, one can instead use (38) to pick the $\mathbf{s}$ that maximizes the probability of solution within the contract $F_{\mathscr{A},\mathbf{s}}(t_u) = \Pr\{T_{\mathscr{A},\mathbf{s}} \leq t_u\}$ (or, equivalently, minimizes $S_{\mathscr{A},\mathbf{s}}(t_u)$):

$$\mathbf{s} = \arg\min_{\mathbf{s}} S_{\mathscr{A},\mathbf{s}}(t_u). \tag{42}$$

Quantile In other applications, one could want to solve each problem with probability at least α, and minimize the time spent. In this case, a quantile $t_{\mathscr{A},\mathbf{s}}(\alpha) = F^{-1}_{\mathscr{A},\mathbf{s}}(\alpha)$ should be minimized:

$$\mathbf{s} = \arg\min_{\mathbf{s}} F^{-1}_{\mathscr{A},\mathbf{s}}(\alpha). \tag{43}$$

These three methods can be easily adapted to allocate time on multiple CPUs (Gagliolo and Schmidhuber, 2008b). If the F_k are parametric, a gradient of the above quantities can be computed analytically, depending on the particular parametric form: otherwise, the optimisation can be performed numerically.

Note that the shares $\mathbf{s}$ resulting from these three optimisation processes can differ: in the last two cases, they can also depend on the chosen values for t_u and α respectively. In no case there is a guarantee of unimodality, and it may be advisable to repeat the optimisation process multiple times, with different random initial values for $\mathbf{s}$, in case of extreme multimodality.

3.3 Algorithms as Competing Risks

As said, GAMBLETA is a fully online algorithm selection method, in which there is no distinction between learning and using the RTD models. In order to save computation time, for each instance, we only wait until the fastest algorithm solves it: at this point we stop the execution of the remaining algorithms. In medical terms, we are viewing each instance b_j as a patient, with covariates $\mathbf{x}_j$, and the K algorithms as competing risks, one of which should eventually "kill" the patient, and censor the runtime values for the other algorithms.

From a statistical point of view, a clear disadvantage of using an algorithm portfolio to gather runtime data is that, as we saw in Sect. 2.4, the resulting models will be biased. In the next section we will see an illustrative example of this phenomenon.

4 Experiments

In this section we present a simple sampling experiment, with a small set of satisfiability problem solvers, in order to illustrate the concepts introduced in this chapter.

Satisfiability (SAT) problems (Gent and Walsh, 1999) constitute a standard decision problem, with many important practical applications. A conjunctive normal form CNF(k,n,m) problem consists in finding an instantiation of a set of n Boolean variables that simultaneously satisfies a set of m clauses, each being the logical OR of k literals, chosen from the set of variables and their negations. A problem instance is termed *satisfiable* (SAT) if there exists at least one of such instantiations, otherwise it is *unsatisfiable* (UNSAT). An algorithm solving an instance will halt as soon as a single solution is found, or unsatisfiability is proved. With $k = 3$

the problem is NP-complete. Satisfiability of an instance depends in probability on the clauses to variables ratio: a *phase transition* can be observed at $m/n \approx 4.3$ (Mitchell et al, 1992), at which an instance is satisfiable with probability 0.5. This probability quickly goes to 0 for m/n above the threshold, and to 1 below.

SAT solvers can be broadly classified in two categories: *complete* solvers, that execute a backtracking search on the tree of possible variable instantiations, and are guaranteed to determine the satisfiability of a problem in a finite, but possibly unfeasibly high, amount of time; and *local search* (LS) solvers, that cannot prove unsatisfiability, but are usually faster than complete solvers on satisfiable instances. In other words, a local search solver can only be applied to satisfiable instances: at the threshold, there is a 0.5 probability that the instance will be unsat, in which case the solver would run forever. The RTD of a set of instances for a complete solver will have $F(\infty) = 1$ for any value of the ratio m/n; a local search solver would have a $F(\infty) = 0.5$ on a set of instances at the 4.3 threshold, $F(\infty) = 1$ below the threshold, and $F(t) = 0\ \forall t$ above it.

In the following experiments, we used a set of 200 randomly generated instances at the phase transition, 100 of which satisfiable, with $n = 250$ variables and $m = 1065$ clauses: the `uf-250-1065` and `uuf-250-1065` instances from SATLIB (Hoos and Stützle, 2000). The algorithm portfolio consisted of two SAT solvers from the two categories above, the same used in (Gagliolo and Schmidhuber, 2006b). As a complete solver we picked Satz-Rand (Gomes et al, 2000), a randomized version of Satz (Li and Anbulagan, 1997) in which random noise influences the choice of the branching variable. Satz is a modified version of the complete DPLL procedure, in which the choice of the variable on which to branch next follows an heuristic ordering, based on first and second level unit propagation. Satz-Rand differs in that, after the list is formed, the next variable to branch on is randomly picked among the top h fraction of the list. We present results with the heuristic starting from the most constrained variables, as suggested also in (Li and Anbulagan, 1997), and the noise parameter set to 0.4. As a local search solver we used G2-WSAT (Li and Huang, 2005): for this algorithm, we set a high noise parameter (0.5), as advisable for problems at the phase threshold, and the diversification probability at the default 0.05.

This algorithm set/problem set combination is quite interesting, as G2-WSAT almost always dominates the performance of Satz-Rand on satisfiable instances, while the latter is obviously the winner on all unsatisfiable ones, on which the runtime of G2-WSAT is infinite.

This situation is visualized by the continuous lines in Fig. 1, which plot the RTD of the set for the two solvers[3], resulting from a KM estimate of the CDF (21) based on the runtimes collected from 100 runs with different random seeds for each of

[3] As we needed a common measure of time, and the CPU runtime measures are quite inaccurate (see also Hoos and Stützle, 2004, p. 169), we modified the original code of the two algorithms adding a counter, that is incremented at every loop in the code. The resulting time measure was consistent with the number of backtracks, for Satz-Rand, and the number of flips, for G2-WSAT. All runtimes reported for this benchmark are expressed in these loop cycles: on a 2.4 GHz machine, 10^9 cycles take about 1 minute.

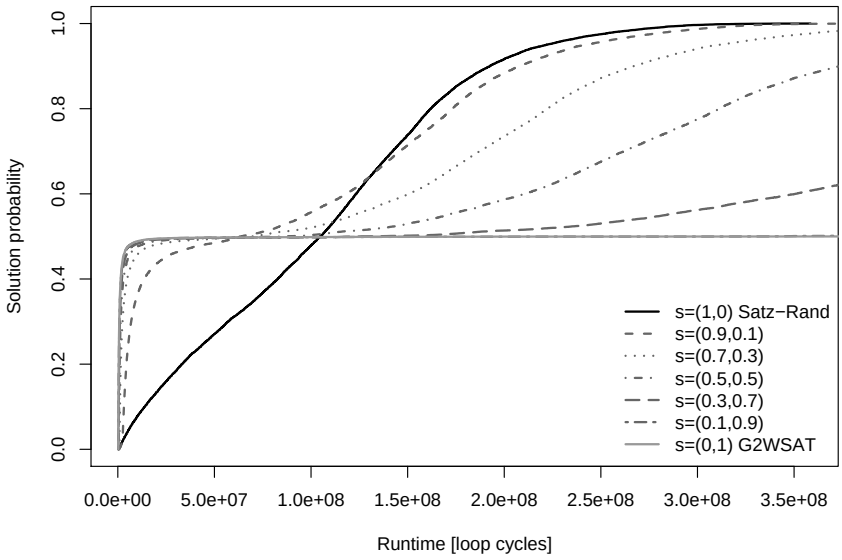
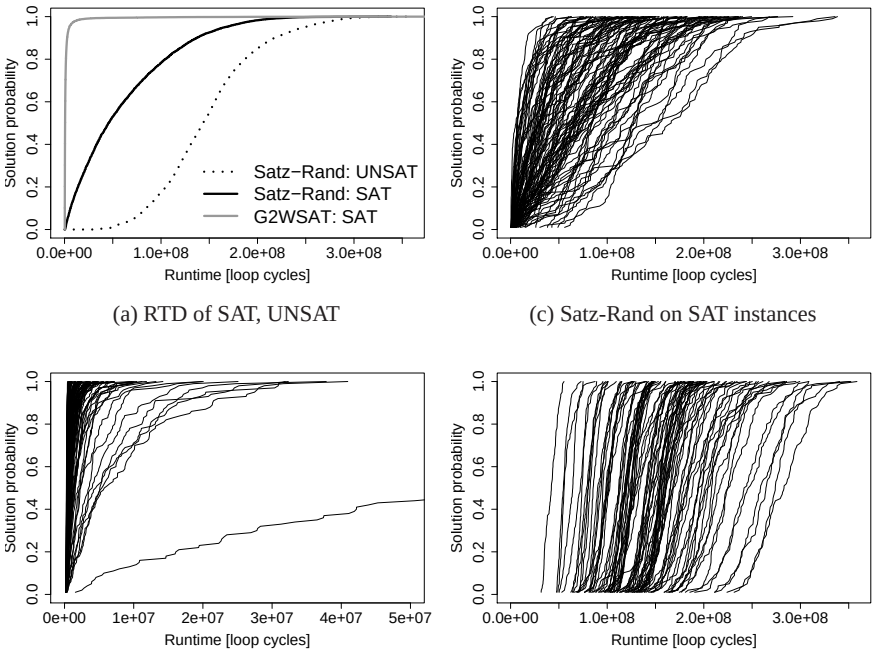


Fig. 1 The RTDs (continuous lines) of Satz-Rand (black) and G2WSAT (gray), along with the RTD of the portfolio (dashed lines) resulting from different values of the share $\mathbf{s} = (s_1, s_2)$, where s_1 is the portion of time allocated to Satz-Rand, and $s_2 = 1 - s_1$ is the portion used by G2WSAT

the 200 instances. No censoring was used, except of course for the runtimes of G2WSAT on the UNSAT instances, which were artificially generated as censored events with threshold 10^9. One can clearly notice the advantage of G2-WSAT on satisfiable instances, represented by the small lower quantiles (below 10^6). From quantile 0.5 on, the RTD remains flat, reflecting the fact that half of the instances are unsatisfiable. Satz-Rand starts solving problems later, and is competitive with G2-WSAT only on a small number of satisfiable instances, but is able to solve also all the unsatisfiable ones, as indicated by the fact that the RTD reaches 1.

On the same plot (dashed lines), we display the RTD of the portfolio (38) for different values of the share, obtained simulating a run of the portfolio (37) for each pair of runtime values. For $\mathbf{s} = (1,0)$ only Satz-Rand is run. Giving a small portion of time to G2-WSAT ($\mathbf{s} = (0.9, 0.1)$) already improves the situation; increasing its share gradually moves the RTD of the portfolio towards the one of G2-WSAT ($\mathbf{s} = (0,1)$).

Let us now look with more detail at the variability of the RTDs within the set. Figure 2 (a) displays the RTDs of the two algorithms on the subsets of SAT and UNSAT instances. Figures 2 (c,d,e) display the RTDs of the instances, again estimated based on 100 runs for each instance, grouped based on the algorithm and on



(a) RTD of SAT, UNSAT

(c) Satz-Rand on SAT instances

(b) G2WSAT on SAT instances

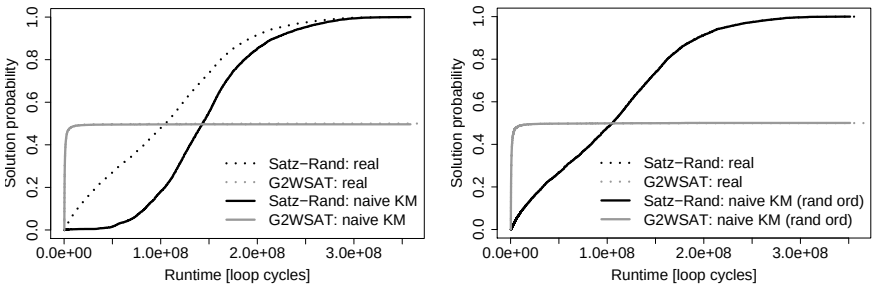
(d) Satz-Rand on UNSAT instances

Fig. 2 (a) RTDs of the two algorithms on the subsets of SAT and UNSAT instances: the line for G2WSAT on UNSAT instances would be constant at 0, and is omitted. (b) RTDs of G2WSAT on each of the 100 satisfiable instances. Note the different time scale. The lower line leaving the plot refers to instance 24, and reaches 1 at time 1.6×10^8. (c,d) RTDs of Satz-Rand on the satisfiable and unsatisfiable instances, respectively.

satisfiability. Note that there still is a huge variability of the RTD of the instances within each subset.

To illustrate the bias induced by a competing risk, in Fig. 3 (a) we display again the unbiased RTDs of the two algorithms (dotted lines) on the whole set of instances, compared with the KM estimates (20) of the RTDs of the two algorithms, this time obtained with the portfolio approach, that is, censoring the runtime of the slowest algorithm for each run and each instance (continuous lines). While the model for G2WSAT remains accurate, as it mostly gets censored on unsatisfiable instances on which it would run forever anyway, one can clearly notice the bias of the product-limit estimator for Satz-Rand: the runtime is overestimated, especially for the satisfiable instances, on which Satz-Rand is slower, so it gets censored.

To reduce the bias, we repeated the sampling, randomly reordering the instances for Satz-Rand: in this way, the two algorithms run in parallel, but each on a different instance. This should reduce the statistical dependence among the runtimes of the two algorithms, allowing to consider the censoring mechanism uninformative, thus



(a) Same instances (b) Random reordering

Fig. 3 (a) The unbiased RTDs (dotted lines) of Satz-Rand (black) and G2WSAT (gray), compared with the biased estimates (continuous lines) obtained censoring, for each instance and each seed, the runtime of the slowest algorithm. Note the bias in the model for Satz-Rand. (b) The same unbiased RTDs (dotted lines), again compared with the estimates (continuous lines) obtained censoring, the runtime of the slowest algorithm, this time after a random reordering of the instances, in order to reduce the bias. Note the improvement in the model for Satz-Rand.

resulting in a more correct estimate: this can indeed be observed in Fig. 3 (b), where we display again the "real" runtime distributions, compared against the estimates obtained after random reordering of the instances. This time the estimator for Satz-Rand is visibly more accurate on the whole range of runtimes observed.

5 Related Work

Literature on RTD modeling aimed at analyzing algorithm performance is relatively recent. The behavior of complete SAT solvers on solvable and unsolvable instances near phase transition have been shown to be approximable by Weibull and lognormal distributions respectively (Frost et al, 1997). Heavy-tailed[4] behavior is observed for backtracking search on structured underconstrained problems in (Hogg and Williams, 1994; Gomes et al, 2000). The performance of local search SAT solvers is analyzed in (Hoos, 1999), and modeled using a mixture of exponential distributions in (Hoos, 2002).

A seminal paper in the field of algorithm selection is (Rice, 1976), in which offline, per instance selection is first proposed, for both decision and optimisation problems. More recently, similar concepts have been proposed, with different terminology (algorithm *recommendation*, *ranking*, *model selection*), in the *Meta-Learning* community (Vilalta and Drissi, 2002). Research in this field usually deals

[4] A *heavy-tailed* runtime distribution $F(t)$ is characterized by a Pareto tail, $F(t) \rightarrow_{t\rightarrow\infty} 1 - Ct^{-\alpha}$. In practice, this means that most runs are relatively short, but the remaining few can take a very long time.

with optimisation problems, and is focused on maximizing solution quality, without taking into account the computational aspect.

Work on *Empirical Hardness Models* (Leyton-Brown et al, 2002; Nudelman et al, 2004) is instead applied to decision problems, and focuses on obtaining accurate models of runtime performance, conditioned on numerous features of the problem instances, as well as on parameters of the solvers. The models are used to perform algorithm selection on a per instance basis, and are learned offline: censored sampling is considered in (Xu et al, 2008).

The foundation papers about algorithm portfolios (Huberman et al, 1997; Gomes et al, 1998; Gomes and Selman, 2001), describe how to evaluate the runtime distribution of a portfolio, based on the runtime distributions of the algorithms, which are assumed to be available. The RTD is used to evaluate mean and variance, and find the (per set optimal) *efficient frontier* of the portfolio, i.e. that subset of all possible allocations in which no element is dominated in both mean and variance.

Another approach based on runtime distributions can be found in (Finkelstein et al, 2002, 2003), for parallel independent processes and shared resources respectively. The RTDs are assumed to be known, and the expected value of a cost function, accounting for both wall-clock time and resources usage, is minimized. A dynamic schedule is evaluated offline, using a branch-and-bound algorithm to find the optimal one in a tree of possible schedules. Examples of allocation to two processes are presented with artificially generated runtimes, and a real Latin square solver. Unfortunately, the computational complexity of the tree search grows exponentially in the number of processes.

Other examples of the application of performance modeling to resource allocation are provided by the literature on restart strategies (Luby et al, 1993; Gomes et al, 1998), and on anytime algorithms (Boddy and Dean, 1994; Hansen and Zilberstein, 2001).

6 Summary and Outlook

In this chapter we illustrated the application of survival analysis methods to model the performance of decision problem solvers, focusing on the application of modeling to algorithm selection. We described in deeper detail the statistical aspects of GAMBLETA (Gagliolo and Schmidhuber, 2006b, 2008a), discussing the bias in the RTD models caused by the *competing risks* censoring scheme adopted for sampling the runtime of the algorithms, and illustrating it with a simple sampling experiment.

Ongoing research is aimed at analyzing the actual impact of this bias on the performance of GAMBLETA, and at finding a computationally cheap way of reducing this bias, possibly inspired by the random reordering trick described in Sect. 4. On a longer term, we are working on transferring other survival analysis methods to model the runtime-quality trade-off of optimisation algorithms, in order to devise an algorithm selection method for this broader class of problems.

Acknowledgements The authors would like to thank an anonymous reviewer, and Faustino Gomez, for useful comments on a draft of this paper. The first author was supported by the Swiss National Science Foundation with a grant for prospective researchers (n. PBTI2–118573).

References

Aalen O (1978) Nonparametric inference for a family of counting processes. Annals of Statistics 6:701–726

Akritas M (1994) Nearest neighbor estimation of a bivariate distribution under random censoring. Annals of Statistics 22:1299–1327

Auer P, Cesa-Bianchi N, Freund Y, Schapire RE (2003) The nonstochastic multi-armed bandit problem. SIAM Journal on Computing 32(1):48–77

Beran R (1981) Nonparametric regression with randomly censored survival data. Tech. rep., University of California, Berkeley, CA

Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press

Boddy M, Dean TL (1994) Deliberation scheduling for problem solving in time-constrained environments. Artificial Intelligence 67(2):245–285

Collet D (2003) Modeling survival data in medical research. Chapman & Hall/CRC, Boca Raton, Florida

Cox D (1972) Regression models and life-tables. Journal of the Royal Statistics Society, Series B 34:187–220

Cox D, Oakes D (1984) Analysis of survival data. Chapman & Hall, London

Finkelstein L, Markovitch S, Rivlin E (2002) Optimal schedules for parallelizing anytime algorithms: The case of independent processes. Tech. rep., CS Department, Technion, Haifa, Israel

Finkelstein L, Markovitch S, Rivlin E (2003) Optimal schedules for parallelizing anytime algorithms: The case of shared resources. Journal of Artificial Intelligence Research 19:73–138

Fitzmaurice G, Davidian M, Verbeke G, Molenberghs G (2008) Longitudinal Data Analysis. Chapman & Hall/CRC Press

Fleming T, Harrington D (1991) Counting processes and survival analysis. John Wiley & Sons, Ltd., New York

Frost D, Rish I, Vila L (1997) Summarizing CSP hardness with continuous probability distributions. In: Kuipers B, et al (eds) Fourteenth National Conference on Artificial Intelligence, pp 327–333

Gagliolo M, Schmidhuber J (2006a) Impact of censored sampling on the performance of restart strategies. In: Benhamou F (ed) Principles and Practice of Constraint Programming, Springer, pp 167–181

Gagliolo M, Schmidhuber J (2006b) Learning dynamic algorithm portfolios. Annals of Mathematics and Artificial Intelligence 47(3-4):295–328

Gagliolo M, Schmidhuber J (2007) Learning restart strategies. In: Veloso MM (ed) Twentieth International Joint Conference on Artificial Intelligence, vol. 1, AAAI Press, pp 792–797

Gagliolo M, Schmidhuber J (2008a) Algorithm selection as a bandit problem with unbounded losses. Tech. Rep. IDSIA - 07 - 08, IDSIA, URL http://arxiv.org/abs/0807.1494

Gagliolo M, Schmidhuber J (2008b) Towards distributed algorithm portfolios. In: Corchado JM, et al (eds) International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2008), Springer, pp 634–643

Gent I, Walsh T (1999) The search for satisfaction. Tech. rep., Dept. of Computer Science, University of Strathclyde

Gomes CP, Selman B (2001) Algorithm portfolios. Artificial Intelligence 126(1-2):43–62

Gomes CP, Selman B, Kautz H (1998) Boosting combinatorial search through randomization. In: Mostow J, et al (eds) Fifteenth National Conference on Artificial Intelligence, pp 431–437

Gomes CP, Selman B, Crato N, Kautz H (2000) Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. Journal of Automated Reasoning 24(1–2):67–100

Hansen EA, Zilberstein S (2001) Monitoring and control of anytime algorithms: A dynamic programming approach. Artificial Intelligence 126(1–2):139–157

Hogg T, Williams CP (1994) The hardest constraint problems: a double phase transition. Artificial Intelligence 69(1–2):359–377

Hoos HH (1999) On the run-time behaviour of stochastic local search algorithms for SAT. In: Hendler J, et al (eds) Sixteenth National Conference on Artificial Intelligence, pp 661–666

Hoos HH (2002) A mixture-model for the behaviour of SLS algorithms for SAT. In: Hendler JA (ed) Eighteenth National Conference on Artificial Intelligence, pp 661–667

Hoos HH, Stützle T (2000) SATLIB: An online resource for research on SAT. In: Gent I, et al (eds) SAT 2000 — Highlights of Satisfiability Research in the Year 2000, IOS press, pp 283–292, URL http://www.satlib.org

Hoos HH, Stützle T (2004) Stochastic Local Search : Foundations & Applications. Morgan Kaufmann

Huberman BA, Lukose RM, Hogg T (1997) An economic approach to hard computational problems. Science 27:51–53

Ibrahim JG, Chen MH, Sinha D (2001) Bayesian Survival Analysis. Springer

Kaplan EL, Meyer P (1958) Nonparametric estimation from incomplete samples. Journal of The American Statistical Association 73:457–481

Keilegom IV, Akritas M, Veraverbeke N (2001) Estimation of the conditional distribution in regression with censored data : a comparative study. Computational Statistics & Data Analysis 35:487–500

Klein JP, Moeschberger ML (2003) Survival Analysis: Techniques for Censored and Truncated Data, 2nd edn. Springer

Leyton-Brown K, Nudelman E, Shoham Y (2002) Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In: Van Hentenryck P (ed) Principles and Practice of Constraint Programming, pp 91–100

Li CM, Anbulagan (1997) Heuristics based on unit propagation for satisfiability problems. In: Georgeff MP, et al (eds) Fifteenth International Joint Conference on Artificial Intelligence, pp 366–371

Li CM, Huang W (2005) Diversification and determinism in local search for satisfiability. In: Bacchus F, et al (eds) Theory and Applications of Satisfiability Testing, Springer, pp 158–172

Li G, Doss H (1995) An approach to nonparametric regression for life history data using local linear fitting. Annals of Statistics 23:787–823

Liang K, Self S, Bandeen-Roche K, Zeger S (1995) Some recent developments for regression analysis of multivariate failure time data. Lifetime Data Analysis 1:403–415

Luby M, Sinclair A, Zuckerman D (1993) Optimal speedup of las vegas algorithms. Information Processing Letters 47(4):173–180

Machin D, Cheung Y, Parmar M (2006) Survival Analysis. A practical approach. John Wiley & Sons, West Sussex, England, second edition

Mackay DC (2002) Information Theory, Inference and Learning Algorithms

Mitchell D, Selman B, Levesque H (1992) Hard and easy distributions of sat problems. In: Swartout W (ed) Tenth National Conference on Artificial Intelligence, pp 459–465

Nelson W (1972) Theory and applications of hazard plotting for censored failure data. Technometrics 14:945–965

Nelson W (1982) Applied Life Data Analysis. John Wiley, New York

Nielsen J, Linton O (1995) Kernel estimation in a nonparametric marker dependent hazard model. Annals of Statistics 23:1735–1748

Nudelman E, Brown KL, Hoos HH, Devkar A, Shoham Y (2004) Understanding random SAT: Beyond the clauses-to-variables ratio. In: Wallace M (ed) Principles and Practice of Constraint Programming, Springer, pp 438–452

Pintilie M (2006) Competing Risks. A practical perspective. John Wiley & Sons, Ltd., New York

Putter H, Fiocco M, Geskus R (2006) Tutorial in biostatistics: Competing risks and multi-state models. Statistics in Medicine 26:2389–2430, DOI 10.1002/sim.2712

Rice JR (1976) The algorithm selection problem. In: Rubinoff M, Yovits MC (eds) Advances in computers, vol 15, Academic Press, New York, pp 65–118

Spierdijk L (2005) Nonparametric conditional hazard rate estimation: a local linear approach. Tech. Rep. TW Memorandum, University of Twente

Tsang E (1993) Foundations of Constraint Satisfaction. Academic Press, London and San Diego

Tsiatis A (1975) A nonidentifiability aspect of the problem of competing risks. Proceedings of the National Academy of Sciences of the USA 72(1):20–22

Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. Artificial Intelligence Review 18(2):77–95

Wang JL (2005) Smoothing hazard rate. In: Armitage P, et al (eds) Encyclopedia of Biostatistics, 2nd Edition, vol 7, Wiley, pp 4986–4997

Wichert L, Wilke RA (2005) Application of a simple nonparametric conditional quantile function estimator in unemployment duration analysis. Tech. Rep. ZEW Discussion Paper No. 05-67, Centre for European Economic Research

Xu L, Hutter F, Hoos HH, Leyton-Brown K (2008) SATzilla: Portfolio-based algorithm selection for SAT. Journal of Articial Intelligence Research 32 (2008) 565-606 32:565–606