Optimal Parallelization of Las Vegas Algorithms

Michael Luby[†] Wolfgang Ertel[‡]

TR-93-041

September 1993

Abstract

Let A be a Las Vegas algorithm, i.e., A is a randomized algorithm that always produces the correct answer when it stops but whose running time is a random variable. In [1] a method was developed for minimizing the expected time required to obtain an answer from A using sequential strategies which simulate A as follows: run A for a fixed amount of time t_1, then run A independently for a fixed amount of time t_2, etc. The simulation stops if A completes its execution during any of the runs.

In this paper, we consider parallel simulation strategies for this same problem, i.e., strategies where many sequential strategies are executed independently in parallel using a large number of processors. We present a close to optimal parallel strategy for the case when the distribution of A is known. If the number of processors is below a certain threshold, we show that this parallel strategy achieves almost linear speedup over the optimal sequential strategy. For the more realistic case where the distribution of A is not known, we describe a universal parallel strategy whose expected running time is only a logarithmic factor worse than that of an optimal parallel strategy. Finally, the application of the described parallel strategies to a randomized automated theorem prover confirms the theoretical results and shows that in most cases good speedup can be achieved up to hundreds of processors, even on networks of workstations.

[†] Current address: International Computer Science Institute, 1947 Center Street, Berkeley CA 94704, luby@icsi.berkeley.edu. Research supported in part by NSF Grant CCR-9016468 and grant No. 89-00312 from the United States-Israel Binational Science Foundation (BSF), Jerusalem, Israel. ICSI and UC Berkeley

[‡] Current address: International Computer Science Institute, 1947 Center Street, Berkeley CA 94704, ertel@icsi.berkeley.edu. Research supported by an ICSI Postdoctoral Fellowship.

1 Introduction

Let A be a randomized algorithm of the *Las Vegas* type, i.e., an algorithm that uses a source of truly random bits and has the following properties. Let $A(x)$ denote algorithm A with respect to fixed input x and let the random bits used by A be called the *seed*. For any x, and for any setting of the seed, when (and if) $A(x)$ halts it produces a correct answer. However, the exact behavior of $A(x)$, and in particular its running time, $T_A(x)$, depends on the setting of the seed. For example, for some settings of the seed $A(x)$ could halt in one step whereas for other settings $A(x)$ could run forever.

In [1], the following problem was studied in the sequential setting: Find a sequential simulation strategy for $A(x)$ that minimizes the expected time required to get an answer. In this setting, $A(x)$ is viewed as a black box, and the only allowable use of $A(x)$ is the following: choose at random a setting for the seed and run $A(x)$ for t_1 steps. If $A(x)$ halts within time t_1 then we are done, otherwise randomly and independently choose another seed and run $A(x)$ for t_2 steps, etc. Thus, a *sequential strategy* $\mathcal{S}$ consists of a sequence of positive integers $(t_1, t_2, \ldots)$, and $\mathcal{S}(A(x))$ denotes the new Las Vegas algorithm that is obtained by applying $\mathcal{S}$ to $A(x)$. Each element t_i of a strategy defines a time interval $[\tau_1, \tau_2]$ of integers $\tau_1 = \sum_{j<i} t_j$ and $\tau_2 = \tau_1 + t_i$. Depending on the context, we refer to the t_i as either *intervals*, *experiments* or *runs*. For a given strategy $\mathcal{S}$ and an interval t_i in $\mathcal{S}$ we define that t_i *ends before* τ, iff $\sum_{l\leq i} t_l \leq \tau$.

Of special interest are sequential *repeating* strategies, i.e. strategies that consist of repetitions of experiments of the same length. For every positive integer t, we let $\mathcal{S}_t = (t, t, t, \ldots)$ denote the repeating sequential strategy with all experiments of length t. For notational convenience, we let $\mathcal{S}_\infty$ be the sequential strategy which when applied to an algorithm consists of running the algorithm as is, i.e., $\mathcal{S}_\infty(A(x))$ is identical to $A(x)$.

In this paper, we consider parallel simulation strategies for this problem. Throughout, we let k denote the number of processors on which the parallel simulation is to be executed. A *parallel strategy* $\mathcal{S}^k$ is described by k sequential strategies, i.e.,

$$\mathcal{S}^k = \begin{pmatrix} t_1^1, t_2^1, t_3^1, \ldots \\ t_1^2, t_2^2, t_3^2, \ldots \\ \vdots \qquad\quad \vdots \\ t_1^k, t_2^k, t_3^k, \ldots \end{pmatrix},$$

where each of the k rows is a sequential strategy.[1] Strategy $\mathcal{S}^k$ applied to $A(x)$, $\mathcal{S}^k(A(x))$, consists of using the k processors to execute independently and in parallel each of the k sequential strategies applied to $A(x)$. The overall parallel algorithm successfully halts at the first point in time when one of the k sequential simulations halts, i.e., this can be viewed as a *competition* among the k sequential strategies.

The three subclasses of parallel strategies illustrated in Figure 1 will be of special interest for us. A strategy $\mathcal{S}^k$ is *repeating* if each of the k sequential strategies defining $\mathcal{S}^k$ is repeating; such a strategy can be described by a set of k positive integers.

A strategy $\mathcal{S}^k$ is *uniform* if each of the k sequential strategies defining $\mathcal{S}^k$ is identical, and thus when $\mathcal{S}^k$ is applied to any algorithm all k processors restart their experiments within their sequential simulations at exactly the same points in time. Any uniform strategy can be described as a composition of the following strategy $\mathcal{PAR}^k$ with a sequential strategy. $\mathcal{PAR}^k$ is the simple strategy which when applied to any algorithm consists of running the algorithm independently in

[1] In the following, we will use the general term *strategy* for parallel strategies ($k > 1$), but we will always use *sequential strategy* if a strategy is not parallel ($k = 1$).

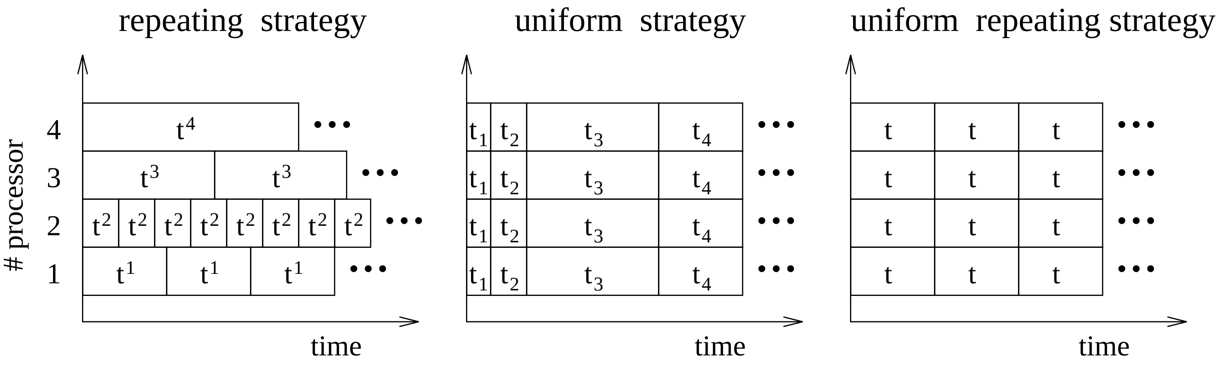

Figure 1: Three important subclasses of strategies.

parallel on k processors and stopping when the first of the k independent executions halts. Thus, $\mathcal{PAR}^k(A(x))$ consists of running $A(x)$ on k processors in parallel using independently chosen seeds. Any uniform strategy can be expressed as the composition of $\mathcal{PAR}^k$ and some sequential strategy $\mathcal{S}$, where the composed strategy is denoted $\mathcal{PAR}^k \circ \mathcal{S}$.[2]

We let $\mathcal{S}_t^k$ denote the *uniform repeating* strategy described by

$$\mathcal{S}_t^k = \begin{pmatrix} t, t, t, \ldots \\ t, t, t, \ldots \\ \vdots \quad\;\; \vdots \\ t, t, t, \ldots \end{pmatrix},$$

i.e., $\mathcal{S}_t^k = \mathcal{PAR}^k \circ S_t$ imposes the same fixed length t on all experiments executed by all processors.

This paper proves results about parallel simulations that are analogous to the sequential simulation results shown in [1]. If full knowledge is available about the distribution of $T_A(x)$, then it is possible to design a uniform repeating strategy $\mathcal{S}_*^k$ that is *close to optimal*, in the sense that for any fixed k it achieves the minimum expected running time within a constant factor amongst all strategies for $A(x)$. In Section 2 we will show that this is true for a carefully chosen value t_*^k that depends on the entire distribution of $T_A(x)$ and the number of processors k used. Let $\ell_A^k(x)$ be the expected running time of this strategy. Similar to the sequential setting [1], $\ell_A^k(x)$ is a natural and easily characterized quantity associated with the distribution of $T_A(x)$.

While the existence of an optimal strategy is an interesting theoretical observation, it is of little value in practice because it requires for its implementation detailed information about the distribution of $T_A(x)$. In practical applications, very little, if any, a priori information is available about this distribution, and its shape may vary wildly with x. Furthermore, since we only want the answer once for any x, there is no point in running experiments to gather information about the distribution: the only information that could be gathered from such a run is that $A(x)$ stops, in which case we also obtain the answer. Thus, the problem we address is that of designing an efficient *universal* strategy, i.e., one that is to be used for all distributions on running times.

In [1], a simple universal sequential strategy $\mathcal{S}_{\text{univ}}$ was introduced. In Section 3 we consider the simple universal parallel strategy $\mathcal{PAR}^k \circ \mathcal{S}_{\text{univ}}$, which we hereafter call $\mathcal{S}_{\text{univ}}^k$. We show that for any $A(x)$ and any k the expected running time of $\mathcal{S}_{\text{univ}}^k(A(x))$ is $\mathrm{O}(\ell_A^k(x)\log(\ell_A^k(x)))$,[3]

[2] The composition of two strategies $\mathcal{S}$ and $\mathcal{R}$ yields a new strategy $\mathcal{R} \circ \mathcal{S}$, and when this new strategy is applied to $A(x)$ it defines a new algorithm $(\mathcal{R} \circ \mathcal{S})(A(x)) \equiv \mathcal{R}(\mathcal{S}(A(x)))$.

[3] All logarithms in this paper are base 2.

which is only a logarithmic factor slower than an optimal strategy that assumes full information about the distribution. For a wide variety of distributions — esp. distributions with infinite or very long running times — even for $k = 1$ this represents a dramatic speedup over the naïve sequential strategy of running the algorithm till termination on one processor. We go on to show that this bound is optimal, i.e., for any universal strategy there is a distribution for which the expected running time of the strategy is slower by a logarithmic factor than that of an optimal strategy.

An important application area for this kind of parallelization of randomized algorithms is combinatorial search. Here, the algorithm $A(x)$ consists of random search of a (often highly unbalanced) tree. A straightforward way to parallelize such a randomized search algorithm is to use $\mathcal{PAR}^k(A(x))$. In [2] this strategy was called *random competition* and applied to randomized combinatorial search algorithms with various examples from automated theorem proving, some of which will be used here again. The distribution on the running time of the randomized theorem prover turned out to be wildly erratic for most of the examples.

We show in Section 2 for large classes of problems that when the distribution is known almost linear speedup can be achieved with an optimal uniform strategy. Even if the distribution is not known, for most examples the speedup of $\mathcal{S}^k_{\text{univ}}$ is close to linear. One of the benefits of $\mathcal{S}^k_{\text{univ}}$ over the straightforward parallelization described in [2] is that there is a guarantee that $\mathcal{S}^k_{\text{univ}}$ has close to optimal speedup, whereas the speedup obtained on many examples using straightforward parallelization was found to be erratic.

As already mentioned, due to the independence of the parallel instances of $A(x)$, no synchronization or communication overhead during the execution of $\mathcal{S}^k_{\text{univ}}$ is required. This makes $\mathcal{S}^k_{\text{univ}}$ ideally scalable up to a very large number of processors, i.e. the speedup results computed in Section 4 are independent of any parallel implementation details. Another important feature of our universal strategy is its simplicity which makes it very easy to implement on almost every parallel computer. It is particularly well suited for implementation on local area networks of high performance workstations.

2 A close to optimal strategy when the distribution is known

In the remainder of this paper, we identify a Las Vegas algorithm A, together with an input x, with the probability distribution p on its running time $T_A(x)$. Thus p is a probability distribution over $\mathbb{Z}^+ \cup \{\infty\}$, and $p(t)$ denotes the probability that $A(x)$ stops after exactly t steps. We will always assume that p is non-trivial in the sense that $p(\infty) < 1$, so that there exists a finite earliest time, $t = t_{\min}$ say, for which $p(t) > 0$. Our main focus of attention is the expected running time of a strategy $\mathcal{S}^k$ on k processors when applied to an algorithm $A(x)$ described by distribution p, which we denote $E(\mathcal{S}^k, p)$. We will always be considering a fixed distribution p, so we abbreviate $E(\mathcal{S}^k, p)$ to $E(\mathcal{S}^k)$.

The first question we ask is the following. Suppose that we have full knowledge of the distribution p; is there some strategy $\mathcal{S}^k$ that is *optimal* for p, in the sense that $E(\mathcal{S}^k) = \inf_{\mathcal{S}^k} E(\mathcal{S}^k)$? In contrast to the sequential case [1], where the optimal sequential strategy is the repeating sequence $\mathcal{S}_* = (t_*, t_*, t_*, \ldots)$ for a carefully chosen value of t_*, the optimal parallel strategy is not easy to describe. Although the best uniform strategy is within a constant factor of optimal as we will see in Theorem 1 it turns out that this strategy is not necessarily optimal.

The uniform repeating strategy $\mathcal{S}^k_t$ can be viewed as a sequential repeating strategy, where each run consists of k parallel independent runs of $A(x)$ with the time bound t. Thus, we can apply the sequential results from [1] for computing the expected value of any uniform repeating

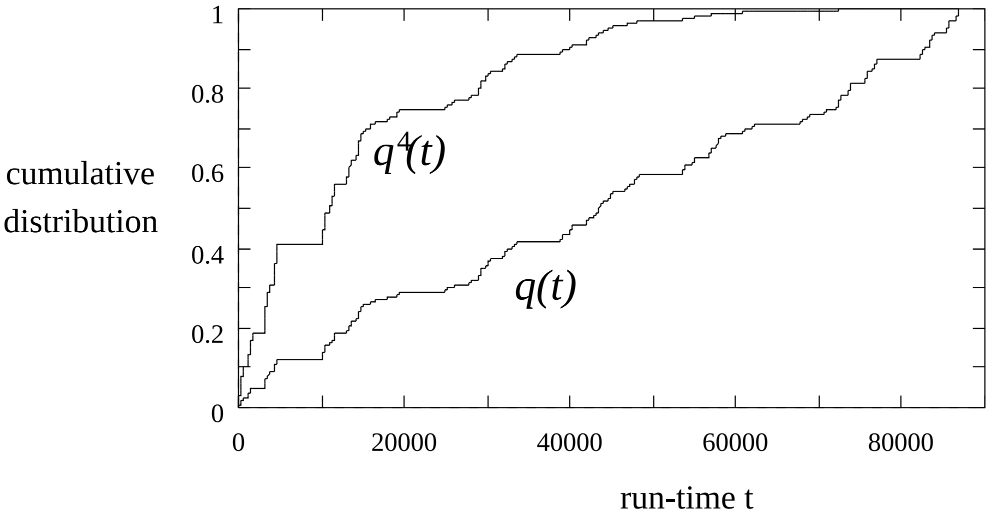

Figure 2: The cumulative probability $q^4(t)$ of the parallel version $\mathcal{PAR}^k(A)(x)$ of $A(x)$ compared to $q(t)$ (the distribution is taken from the example "8-puzzle" in Figure 4).

strategy. However, we have to use a different probability density function $p^k(t)$, which denotes the probability that the shortest of k runs takes t steps.

The parallel density function p^k depends on p and k in the following way. If we define the cumulative distribution q^k (we will use the abbreviations $q = q^1$, $p = p^1$, $r = r^1$ for the sequential case) as

$$q^k(t) \equiv \sum_{t'=0}^{t} p^k(t')$$

and the tail probability as

$$r^k(t) \equiv 1 - q^k(t)$$

it is easy to see that

$$r^k(t) = r(t)^k.$$

An example of how the cumulative probability changes after parallelization is shown in Figure 2, where $q(t)$ and $q^4(t) = 1 - (1 - q(t))^4$ are plotted.

In [1] the expected value $\ell_p(t)$ of a sequential strategy $\mathcal{S}_t = (t, t, t, \ldots)$ was shown to be

$$\ell_p(t) \equiv E(\mathcal{S}_t) = \frac{1}{q(t)}\Big(t - \sum_{t'=0}^{t-1} q(t')\Big) = \frac{\sum_{t'=0}^{t-1} r(t')}{q(t)}. \qquad (1)$$

This can easily be verified as follows. Let $r_t(t')$ be the tail probability of the sequential repeating strategy $\mathcal{S}_t$, which evaluates to

$$r_t(t' + m \cdot t) = r(t)^m \cdot r(t') \qquad 0 \leq t' < t, \quad m \in \mathbb{Z}^+$$

and the expected value to

$$E(\mathcal{S}_t) = \sum_{t'=0}^{\infty} r_t(t') = \sum_{m=0}^{\infty}\sum_{t'=0}^{t-1} r_t(t' + m \cdot t) = \sum_{m=0}^{\infty}\sum_{t'=0}^{t-1} r(t)^m \cdot r(t') = \sum_{m=0}^{\infty} r(t)^m \sum_{t'=0}^{t-1} r(t')$$
$$= \frac{\sum_{t'=0}^{t-1} r(t')}{1 - r(t)} = \frac{\sum_{t'=0}^{t-1} r(t')}{q(t)}$$

In order to specify the optimal sequential strategy, we define

$$\ell_p = \inf_{t<\infty} \ell_p(t). \tag{2}$$

It is easy to see that ℓ_p is finite for any non-trivial distribution p. Let t_* be any finite value of t for which $\ell_p(t) = \ell_p$, if such a value exists, and $t_* = \infty$ otherwise. From [1] we know that for any distribution p, the sequential strategy $\mathcal{S}_* \equiv \mathcal{S}_{t_*} = (t_*, t_*, t_*, \ldots)$ is an optimal sequential strategy for p, and $E(\mathcal{S}_*) = \ell_p$.

To formulate a similar theorem for the parallel case we define t_*^k as that value of t for which $\ell_p^k(t)$ is minimal, in the same way as above, where we now have

$$\ell_p^k = \inf_{t<\infty} \ell_p^k(t) \qquad \text{and} \qquad \ell_p^k(t) = E(\mathcal{S}_t^k) = \frac{\sum_{t'=0}^{t-1} r^k(t')}{q^k(t)}. \tag{3}$$

From [1], it follows that the strategy

$$\mathcal{S}_*^k \equiv \mathcal{S}_{t_*^k}^k$$

is the *optimal uniform strategy*[4], which we will prove to be close to optimal.

Theorem 1 *The expected running time $E(\mathcal{S}_*^k) = \ell_p^k$ of the optimal uniform strategy $\mathcal{S}_*^k$ is within a constant factor of the expected running time of an optimal strategy.*

To prove this theorem we develop a simplified intuitive model of execution which is mathematically easier to analyse.

2.1 The SIMD-model

To simplify the analysis we restrict our general model of execution for uniform repeating strategies and define the *SIMD-model* as follows: a run of any uniform repeating strategy $\mathcal{S}_t^k$ in the SIMD-model is only allowed to stop at times which are integer multiples of the time bound t, i.e. $\mathcal{S}_t^k$ stops at the next multiple of t after the first processor stops, or more formally, $\mathcal{S}_t^k$ stops at mt iff one of the k instances of $A(x)$ terminates at any time t' where $(m-1)t < t' \leq mt$. This makes mathematical analysis much easier since only complete runs of equal length have to be considered. In the following we use $E_{\text{SIMD}}(\mathcal{S}^k)$ to denote that the expected value is computed in the SIMD-model. All other expected values are in the general model.

It is immediately obvious that the expected execution time of any strategy in the *SIMD-model* is at least as big as its expected value in the general model. It will be helpful to introduce one further function associated with a distribution p. For finite values of $t \geq t_{\min}$, define

$$L_p(t) = \frac{t}{q(t)},$$

where q is the cumulative distribution function of p as before, and by analogy with (2) define

$$L_p = \inf_{t<\infty} L(t). \tag{4}$$

and

$$L_p^k = \inf_{t<t(1/k)} L(t), \qquad \text{where} \quad t(x) = \inf\{t : q(t) \geq x\}. \tag{5}$$

[4] Note that *optimal* in this term is relative to uniform strategies.

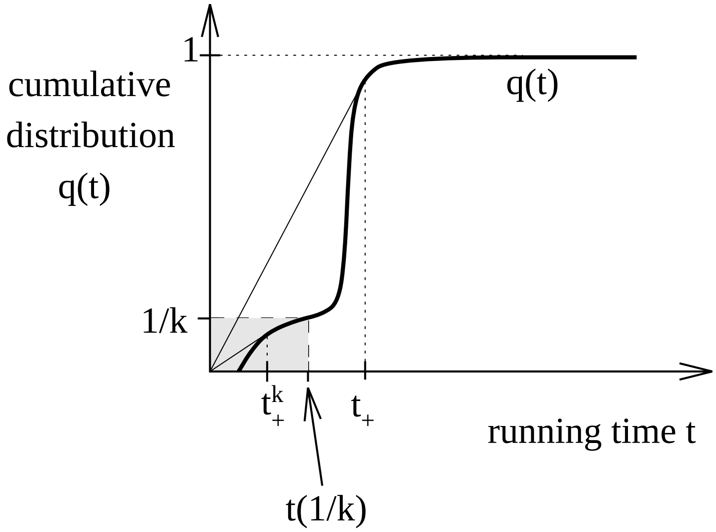

Figure 3: Graphical interpretation of $L_p(t)$ and construction of t_+.

Here $t(x)$ denotes the smallest value of t for which $q(t) \geq x$. Note that $\ell_p \leq L_p \leq 4\ell_p$; the first inequality is obvious, and the second may readily be checked. Furthermore, as can easily be verified, there are always some finite values $t = t_+$ and $t = t^k_+$ such that $L_p(t_+) = L_p$ and $L_p(t^k_+) = L^k_p$.

The quantity $L_p(t)$ is equal to the expected value of the running time of the sequential repeating strategy $\mathcal{S}_t$ in the SIMD-model, since in the average $\frac{1}{q(t)}$ runs of $A(x)$ with length t have to be made. The optimal sequential repeating strategy in the SIMD-model is $\mathcal{S}_+ \equiv \mathcal{S}_{t_+}$ since the minimum of the expected running time $L_p(t)$ is at $t = t_+$. The motivation for the above construction of the parallel time bound t^k_+ is the following. If for any given k the value of $q(t)$ is less than $1/k$, then the amount of redundancy of the k parallel runs of $A(x)$ is negligible and $q^k(t)$ is close to $k\,q(t)$, i.e. the sum of the cumulative probabilities of all processors. To be a little more specific, if $q(t) = 1/k$, then $k\,q(t) = 1$ whereas we have $q^k(t) = 1 - (1 - 1/k)^k$ which is greater than $1 - \frac{1}{\mathrm{e}} \approx 0.632$.[5]

A graphical interpretation of L_p and L^k_p is given in Figure 3 where the cumulative distribution function $q(t)$ is plotted vs. the time t. The point t_+ of minimal expected value of the sequential repeating strategy in the SIMD-model is that value of t for which the slope $(q(t)/t)$ of the straight line from the origin to the graph of $q(t)$ is maximal. In the same way t^k_+ is constructed, but with the constraint that the point of maximal slope must be found within the shaded region $(q(t) \leq 1/k)$.

To prove Theorem 1 we show a lower bound on the expected running time of any strategy.

Lemma 2 $\frac{L^k_p}{4k}$ *is a lower bound on the running time of any strategy* $\mathcal{S}^k$*, i.e.* $E(\mathcal{S}^k) \geq \frac{L^k_p}{4k}$.

Proof: We define $T \equiv 2E(\mathcal{S}^k)$ and write $q_{\mathcal{S}^k}(t)$ for the cumulative distribution of the running time of strategy $\mathcal{S}^k$ applied to $A(x)$ up to time t. Our goal is now to compute T in terms of L^k_p. For this purpose we will first express T as a sum of the individual running times t^j_i. We introduce the abbreviation $u^j_i = \sum_{i'=1}^{i} t^j_{i'}$ for the total amount of time spent on processor j up to the i-th run of $A(x)$ and define

$$I_j \equiv \min\{i : u^j_i \geq T\}.$$

[5] $\mathrm{e} \approx 2.7182818$ is the Euler number.

as the index of the last run of $A(x)$ started on processor j before time T. Since we are only interested in contributions of runs before time T, we chop off the last runs to be of length

$$t^j_{I_j} = T - u_{I_j - 1}.$$

Now we get

$$q_{\mathcal{S}^k}(T) \equiv \Pr[\text{running time of } \mathcal{S}^k \leq T] = 1 - \prod_{j=1}^{k}\prod_{i=1}^{I_j}(1 - q(t^j_i)) \leq \sum_{j=1}^{k}\sum_{i=1}^{I_j} q(t^j_i), \qquad (6)$$

where the final inequality can easily be verified. The lower bound

$$q_{\mathcal{S}^k}(T) \geq \frac{1}{2} \qquad (7)$$

can be proven as follows. If $q_{\mathcal{S}^k(T)}$ would be less than $1/2$, more than half of the probability mass would be located beyond T and therefore the expected value $E(\mathcal{S}^k)$ would be bigger than $T/2$. Now we use this to show that $T \geq \frac{L^k_p}{2k}$, which results in $E(\mathcal{S}^k) = T/2 \geq \frac{L^k_p}{4k}$ and completes the proof. We have do distinguish two cases:

1. $T \geq t(1/k)$:

$$T \geq t(1/k) = \frac{L_p(t(1/k))}{k} \geq \frac{L^k_p}{k}.$$

Here we used the definition $L(t) = \frac{t}{q(t)}$ to get the equality.
2. $T < t(1/k)$:

$$T = \frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{I_j} t^j_i = \frac{1}{k}\sum_{j=1}^{k}\sum_{i=1}^{I_j}\frac{t^j_i}{q(t^j_i)} \cdot q(t^j_i) \geq \frac{L^k_p}{k}\sum_{j=1}^{k}\sum_{i=1}^{I_j} q(t^j_i) \geq \frac{L^k_p}{2k}$$

where the last two inequalities follow from the definition of L^k_p (5) and from equations (6) and (7). □

The next lemma will complete the proof of theorem 1.

Lemma 3 *In the SIMD-model the expected running time of the uniform repeating strategy* $\mathcal{S}^k_+ \equiv \mathcal{S}^k_{t^k_+}$ *has an upper bound of* $\frac{\mathrm{e}L^k_p}{k}$, *i.e.* $E_{\text{SIMD}}(\mathcal{S}^k_+) \leq \frac{\mathrm{e}L^k_p}{k}$.

Proof:

$$E_{\text{SIMD}}(\mathcal{S}^k_+) = \frac{t^k_+}{q^k(t^k_+)} = \frac{t^k_+}{1 - (1 - q(t^k_+))^k} \leq \mathrm{e}\frac{t^k_+}{q(t^k_+)k} = \frac{\mathrm{e}L^k_p}{k}.$$

Here all the equalities are just applications of definitions or easy to see. The inequality is proved as follows:

$$1 - (1 - q(t^k_+))^k \geq 1 - \mathrm{e}^{-q(t^k_+)k} \geq q(t^k_+)k\mathrm{e}^{-q(t^k_+)k} \geq \frac{q(t^k_+)k}{\mathrm{e}}.$$

Here we used the fact that for any ε, if $0 \leq \varepsilon \leq 1$ then

$$1 - \mathrm{e}^{-\varepsilon} \geq \varepsilon\, \mathrm{e}^{-\varepsilon}$$

and that $q(t^k_+) \leq 1/k$. □

We are now able to prove Theorem 1. From Lemmas 2 and 3 we can derive the first and the last of the following inequalities

$$\frac{L_p^k}{4k} \le E(\mathcal{S}_*^k) \le E_{\text{SIMD}}(\mathcal{S}_+^k) \le \frac{\mathrm{e}L_p^k}{k} \tag{8}$$

and the second inequality holds since the optimal uniform strategy is at least as good as any uniform repeating strategy in the SIMD-model. Since $\frac{L_p^k}{4k}$ is a lower bound on the expected running time of every strategy, we have shown that the expected running time of the optimal uniform strategy $\mathcal{S}_*^k$ is within a constant factor of the running time of the optimal strategy with k processors and the proof of Theorem 1 is complete.

Remark: Theorem 1 remains true even for more general strategies, in which the run lengths t_i are themselves random variables and runs may be suspended and then restarted at a later time. A proof for the sequential case can be found in [1].

2.2 The best uniform repeating strategy is not optimal

Unlike in the sequential case, the optimal uniform strategy is not optimal for all distributions $q(t)$. We demonstrate this with a counterexample, i.e. we show that there exists a distribution $q(t)$ for which there is a non-uniform strategy which has smaller expected value than the optimal uniform strategy.

Theorem 4 *The optimal uniform strategy $\mathcal{S}_*^k$ is not the best strategy for all distributions $q(t)$.*

Proof: The distribution $p(t)$ with $p(3) = p(7) = 1/2$ allows the algorithm $A(x)$ to stop only at the times $t = 3$ and $t = 7$. Thus, the only useful time bounds t_i^j in any strategy are 3 and 7. Any other time bound ($t \in \{1, 2, 4, 5, 6\}$) would waste at least part of the running time of $A(x)$ while waiting for the impossible event. If we use two processors, the expected running time of the uniform repeating strategy can be computed by (c.f. eq. (3))

$$E(\mathcal{S}_t^2) = \frac{\sum_{t'=0}^{t-1} r^2(t')}{1 - r^2(t)},$$

The results are shown in the following table

t	1	2	3	4	5	6	7
$E(\mathcal{S}_t^2)$	∞	∞	4	$\frac{13}{3}$	$\frac{14}{3}$	5	4

The two strategies $\mathcal{S}_3^2$ and $\mathcal{S}_7^2$ have the same expected running time of 4. This is not optimal, since the strategy

$$\mathcal{S} = \begin{pmatrix} 3, & 3 \\ 7 & \end{pmatrix},$$

with the expected running time $E(\mathcal{S}) = \frac{3}{4} \cdot 3 + \frac{1}{8} \cdot 6 + \frac{1}{8} \cdot 7 = \frac{31}{8}$ is better. □

Remarks: (a) $\mathcal{S}$ is better than $\mathcal{S}_3^2$ since the guaranteed termination after 7 time-steps reduces the expected running time. $\mathcal{S}$ is better than $\mathcal{S}_7^2$ since a second parallel run of 7 steps can not increase the probability to stop after 7 steps, since this is already 1 with only one processor.

Starting a new run after 3 steps on one processor creates the chance to stop after 6 steps and therefore a shorter expected running time.

This and other similar counter-examples show that finding the optimal parallel strategy for a given distribution $q(t)$ is a complex optimization problem which is similar to bin packing. The task here is to assemble a set of time-bounds from the support of p into a strategy $\mathcal{S}^k$ such that the expected value $E(\mathcal{S}^k)$ is minimal. Like in other optimization problems, a close to optimal solution can be found in time $O(|p|)$ where $|p|$ is the number of different running times with nonzero probability. Since the scheduling strategies in general consist of infinite sequences, it might even be impossible to give a finite description of the optimal strategy. It remains an open question whether an algorithm exists for that problem which is polynomial in the size of the distribution p for the case that full information about the distribution is available.
(b) For the case of more than 2 processors the following strategy is optimal:

$$\mathcal{S} = \begin{pmatrix} 3, & 3 \\ \vdots & \\ 3, & 3 \\ 7 & \end{pmatrix}.$$

2.3 Speedup of the optimal uniform strategy

We have shown that the uniform repeating strategy $\mathcal{S}_*^k$ is close to optimal, i.e. no strategy can do much better. However, we do not know whether the this strategy is efficient, i.e. if for a not too large number of processors k the speedup is linear (i.e. equal to k) or close to linear. We define the Speedup $Sp_{\text{opt}}(k)$ of the strategy $\mathcal{S}_*^k$ by relating its expected running time to the optimal sequential strategy $\mathcal{S}_*$

$$Sp_{\text{opt}}(k) \equiv \frac{E(\mathcal{S}_*)}{E(\mathcal{S}_*^k)} = \frac{\ell_p}{\ell^k}.$$

Since the sequential strategy $\mathcal{S}_*$ has optimal expected running time [1], the speedup $Sp_{\text{opt}}(k)$ can at most be k. Unfortunately, it is not true that the speedup is almost linear for all distributions p. If for example

$$p(t) = \begin{cases} 1 & \text{for } t = t_0 \\ 0 & \text{otherwise} \end{cases}$$

then no strategy — neither parallel nor sequential — can run faster than t_0 and in particular we have $Sp_{\text{opt}}(k) \equiv 1$. However, the next Lemma says that for a large class of distributions, if the number of processors used is not too large, the speedup is close to linear.

Lemma 5 *If $k\,q(t_*) \ll 1$ then $Sp_{\text{opt}}(k) \approx k$.*

Proof: For any k and any t, if $k\,q(t) \ll 1$, we have

$$\ell_p^k(t) = \frac{1}{q^k(t)} \sum_{t'<t} (1 - q^k(t')) = \frac{1}{1-(1-q(t))^k} \sum_{t'<t} (1-q(t'))^k \tag{9}$$

$$\approx \frac{1}{k\,q(t)} \sum_{t'<t} (1 - k\,q(t')) \approx \frac{t}{k\,q(t)} \approx \frac{\ell_p(t)}{k} \tag{10}$$

From (9) and (10) we get $\ell_p(t) \approx k\,\ell_p^k(t)$. Since this holds for all t when $k\,q(t) \ll 1$ it is also true for the infima $\ell_p = \ell_p(t_*)$ and $\ell^k = \ell_p^k(t_*^k)$ of the left-hand and right-hand sides. Here we made use of the fact that $t_*^k \leq t_*$ and therefore $k\, q(t_*^k) \leq k\, q(t_*) \ll 1$. This leads to

$$Sp_{\mathrm{opt}}(k) = \frac{\ell_p}{\ell^k} = \frac{\ell_p(t_*)}{\ell_p^k(t_*^k)} \approx k. \qquad \Box$$

The intuitition behind this is as follows. If $q(t_*)$ is small enough such that $k\, q(t_*)$ is well below 1 then the probability that some processor is successful is close to the sum of the individual success probabilities of the processors and thus their efficiency (speedup) is close to optimal. However, if $k \gg 1/q(t_*)$ then $1/q(t_*)$ processors running for t_* have about the same probability of success (close to probability 1) as k processors running for t_* time, and thus the speedup is clearly sublinear. The experimental speedup figures shown in Section 4 confirm these results and show that for almost all our examples close to linear speedup can be achieved with the optimal uniform strategy.

3 Unknown distributions

The optimal uniform strategy $\mathcal{S}_*^k$ described in the previous section clearly requires detailed knowledge of the distribution p for its implementation. As we have already explained, however, in the applications we have in mind there will be no information available about p. We are therefore led to ask what is the best performance we can achieve in the absence of any a priori knowledge of p. Our next theorem says that, with no knowledge whatsoever about the distribution p, we can always come surprisingly close to the optimum value for the case of full knowledge given in Theorem 1. Moreover, this performance is achieved by a uniform strategy of a very simple form that is easy to implement in practice.

In [1] the universal sequential strategy $\mathcal{S}_{\mathrm{univ}} = (1,1,2,1,1,2,4,1,1,2,1,1,2,4,8,1,\ldots)$ was described and proven to be within a logarithmic factor of any optimal sequential strategy. One way to describe this sequential strategy is to say that for each processor all run lengths are powers of two, and that each time a pair of runs of a given length has been completed, a run of twice that length is immediately executed. For a more formal definition we give a recursive function s that computes any finite prefix of the sequence of running times for one processor up to the first run of length 2^{n+1}

$$\begin{aligned} \forall n \in \mathbb{N} \cup \{0\}: \quad s(n+1) &= (s(n), s(n), 2^{n+1}) \\ s(0) &= (1) \end{aligned}$$

where for all sequences a, b, c we define $((a), (b), c) = (a, b, c)$. Note that the sequential strategy is "balanced" in the sense that the total time spent on runs of each length is roughly equal.

Now, if k processors are available, we run this sequential strategy competitively on all processors and define

$$\mathcal{S}_{\mathrm{univ}}^k \equiv \mathcal{PAR}^k(\mathcal{S}_{\mathrm{univ}}) = \begin{pmatrix} 1,1,2,1,1,2,4,1,1,2,1,1,2,4,8,1,\ldots \\ \vdots \\ 1,1,2,1,1,2,4,1,1,2,1,1,2,4,8,1,\ldots \end{pmatrix}.$$

Theorem 6 *For all distributions p and any strategy $\mathcal{S}^k$*

$$E(\mathcal{S}_{\mathrm{univ}}^k) \leq 16\, E(\mathcal{S}^k)(\log(E(\mathcal{S}^k)) + 6).$$

Proof: For any two strategies $\mathcal{S}$ and $\mathcal{R}$ and any two positive integers r and s we say that the s-prefix of strategy $\mathcal{S}$ *simulates* the r-prefix of strategy $\mathcal{R}$ if there is an injective mapping from intervals in $\mathcal{R}$ that start before time r onto intervals in $\mathcal{S}$ that end before time s with the additional property that an interval of R which has run for time t by time r is mapped to an interval of S with length at least t. The fact we use below is that if an s-prefix of S simulates an r-prefix of R then the probability that S stops by time s is at least as large as the probability that R stops by time r.

The outline of the proof is as follows. Let $\mathcal{S}^k$ be any (e.g. an optimal) parallel strategy with expected running time $E(\mathcal{S}^k)$. Define r to be such that the probability $\mathcal{S}^k$ has not terminated by time 2^{r-1} is at least $1/4$ but the probability that $\mathcal{S}^k$ has not terminated by time 2^r is at most $1/4$. It is not hard to see that

$$E(\mathcal{S}^k) \geq 2^{r-1}/4. \tag{11}$$

Let $\mathcal{S}^k_{\text{chop}}$ be the portion of $\mathcal{S}^k$ chopped off after 2^r time steps. (The intervals of $\mathcal{S}^k$ that are still running at time 2^r are truncated so that all k sequential schedules in the chopped portion finish at exactly time 2^r.) Let $\mathcal{S}^k_{\text{mod}}$ be the parallel strategy that consists of repeating $\mathcal{S}^k_{\text{chop}}$ forever. We obtain an upper bound on $E(\mathcal{S}^k_{\text{univ}})$ by relating the behavior of $\mathcal{S}^k_{\text{univ}}$ to $\mathcal{S}^k_{\text{mod}}$. We do this by determining below, for all integers $j \geq 0$, an integer $t(j)$ such that the $t(j)$-prefix of $\mathcal{S}^k_{\text{univ}}$ simulates the 2^{j+r}-prefix of $\mathcal{S}^k_{\text{mod}}$. For notational simplicity, define $t(-1) = 0$. Since $\mathcal{S}^k$ fails to stop with probability at most $1/4$ by time 2^r, it follows that $\mathcal{S}^k_{\text{mod}}$ fails to stop with probability at most $(1/4)^{2^j} \leq (1/4)^{j+1}$ by time 2^{j+r}, and from this it follows that

$$E(\mathcal{S}^k_{\text{univ}}) \leq \sum_{j \geq 0} (t(j) - t(j-1))(1/4)^j. \tag{12}$$

It is not hard to see (and it follows from the proof of the analogous sequential version of this theorem in [1]) that the $2^{j+r+1}(j+r+1)$-prefix of $\mathcal{S}_{\text{univ}}$ simulates the 2^{j+r} prefix of any sequential strategy. From this it follows that the $2^{j+r+1}(j+r+1)$-prefix of $\mathcal{S}^k_{\text{univ}}$ simulates the 2^{j+r}-prefix of $\mathcal{S}^k_{\text{mod}}$, and thus we can set $t(j) = 2^{j+r+1}(j+r+1)$.

Substituting this into equation (12) and simplifying yields

$$E(\mathcal{S}^k_{\text{univ}}) \leq 2^r \sum_{j \geq 0} (1/2)^j (j+r+2).$$

From this it follows that

$$E(\mathcal{S}^k_{\text{univ}}) \leq 2^{r+1}(r+3).$$

From this and from equation (11) the theorem easily follows. □

Remark: This proof is similar to the proof of the corresponding theorem for $k = 1$ given in [1], but here we don't need to refer to an optimal strategy. Moreover, the constant factor derived here is slightly less than in [1].

Finally we show that the universal strategy $\mathcal{S}^k_{\text{univ}}$ is optimal (within a constant factor) among universal strategies.

Theorem 7 *Let $\mathcal{S}^k$ be any strategy. For all positive integers j, there is a distribution p with $L^k_p/k = 2^j$ such that*

$$E(\mathcal{S}^k, p) \geq \frac{1}{8\,\mathrm{e}} E(\mathcal{S}^k_*, p)\,(\log E(\mathcal{S}^k_*, p) - 0.5).$$

Proof: From (8) we know that $E(\mathcal{S}^k_*, p) \leq \mathrm{e} L^k_p/k$ and thus for any p with $L^k_p/k = 2^j$ we get

$$2^j \geq \frac{E(\mathcal{S}^k_*)}{\mathrm{e}} \tag{13}$$

From Lemma 8 below we know that there is a distribution p with $L^k_p/k = 2^j$ and $E(\mathcal{S}^k, p) \geq (j+1)\, 2^j/8$. Together with (13) we get for this p

$$E(\mathcal{S}^k, p) \geq \frac{1}{8\,\mathrm{e}}\, E(\mathcal{S}^k_*, p)\, (\log E(\mathcal{S}^k_*, p) - 0.5),$$

and the theorem follows immediately. □

Lemma 8 *For any fixed strategy $\mathcal{S}^k$ and all positive integers j, there is a distribution p for which $L^k_p/k = 2^j$ and for which*

$$E(\mathcal{S}^k, p) \geq (j+1)\, 2^j/8.$$

Proof: Fix j. We define a family of $j+1$ distributions $p^{(0)}, \ldots, p^{(j)}$ such that for at least one value of i, $E(\mathcal{S}^k, p^{(i)}) \geq (j+1)2^j/8$. Distribution $p^{(i)}$ is defined as follows.

$$p^{(i)}(t) = \begin{cases} 2^i/(k2^j) & \text{for } t = 2^i; \\ 1 - 2^i/(k2^j) & \text{for } t = \infty; \\ 0 & \text{otherwise}, \end{cases} .$$

Note that $L^k_p = k2^j$ for all i. Let m_i be the number of runs of length at least 2^i required to ensure that the strategy stops on $p^{(i)}$ with probability at least $\frac{1}{2}$. It is easy to see that $m_i \geq k2^j/2^{i+1}$ as follows. Each run of length at least 2^i terminates with probability $2^i/(k2^j)$. Since the probability that at least one of the m_i runs stops is at most the sum of the individual probabilities, we must have $m_i 2^i/(k2^j) \geq 1/2$.

Let τ be the minimum amount of time needed to execute a family of runs where there are m_i runs of length at least 2^i for all $i = 0, \ldots, j$. It is easily seen that

$$\tau \geq \sum_{i=0}^{j} m_i 2^{i-1} \geq (j+1)k2^j/4.$$

Thus, if we look at the schedule $\mathcal{S}^k$, it follows that for at least one value of i there are at most m_i runs of length i that have been executed by time τ/k. For this value of i, $\mathcal{S}^k$ has failed to stop with probability at least $1/2$ by time τ/k, and thus

$$E(\mathcal{S}^k, p^{(i)}) \geq \tau/(2k) \geq (j+1)2^j/8. \qquad \square$$

4 Experimental results

To get an impression of the relevance of the various strategies analysed, we computed the expected value of the running time for a number of different distributions. Each of these distributions is the result of a large number of sample runs of the randomized automated theorem prover SETHEO on a particular mathematical theorem. Thus, here SETHEO acts as the black box algorithm A and any particular theorem as its input x. SETHEO [5] does randomized depth first backtracking search for a first solution in an OR-search-tree. These search-trees usually are highly structured and the solutions are distributed non-uniformly. The resulting running time distributions reflect this complex structure and usually have high variance. As a consequence, in most cases the shortest possible running time $t_{\min}$ of SETHEO is much shorter than the expected running time $E(\mathcal{S}_\infty)$ when SETHEO is run until it stops. Therefore SETHEO is an ideal application of the strategies described above. Since $\mathcal{S}_\infty$ is the default sequential strategy used by SETHEO (and most other combinatorial search algorithms) we will use its expected value as a reference point to compare the sequential repeating strategy $\mathcal{S}_*$ and the sequential universal strategy $\mathcal{S}_{\mathrm{univ}}$.

Two of the example theorems listed in Table 1 (8-puzzle, queens10) are combinatorial puzzle problems which are easy to formalize in logic. All the other theorems have been selected randomly from a set of several hundred mathematical theorems which SETHEO is able to prove.

The offset time t_{off} for $\mathcal{S}^k_{\mathrm{univ}}$ used for computing the figures in Table 1 is 100 inferences, since the shortest running time of SETHEO varies between 10 and 100 inferences in most examples.[6] Since the performance of the sequential universal strategy shows only little dependence on t_{off}, we did no further tuning of this parameter for the sequential measurements. With this setting we computed the expected values of the default sequential strategy $\mathcal{S}_\infty$, the optimal sequential repeating strategy $\mathcal{S}_*$ and the sequential universal strategy $\mathcal{S}_{\mathrm{univ}}$ for a number of theorem proving problems which are shown in Table 1. Hereby we used (3) for $E(\mathcal{S}^k_*)$ and to compute $E(\mathcal{S}^k_{\mathrm{univ}})$ we used the generic formula

$$\begin{aligned} E(\mathcal{S}^k_{\mathrm{univ}}) &= \sum_{t=0}^{t_1} t\,p(t) + r(t_1)\left(t_1 + \sum_{t=0}^{t_2} t\,p(t) + r(t_2)\left(t_2 + \sum_{t=0}^{t_3} t\,p(t) + \ldots\right)\right) \\ &= \sum_{i=1}^{\infty}(t_{i-1} + \sum_{t=0}^{t_i} t\,p(t)) \prod_{j=0}^{i-1} r(t_j) \end{aligned}$$

with $t_0 = 0$. This formula can be applied to any sequential strategy $\mathcal{S} = (t_1, t_2, \ldots)$. For parallel strategies $p(t)$ has to be replaced by $p^k(t)$.

Of special interest are the fourth and the last numeric columns in Table 1, where the time ratios are given. As expected, $\mathcal{S}_*$ is always at least as good as $\mathcal{S}_\infty$ and often much better. The sequential universal strategy however is for two examples significantly worse than $\mathcal{S}_\infty$. The reason is that here $\mathcal{S}_\infty$ is already optimal and therefore the logarithmic overhead is reflected in an almost three times longer running time.

However, it must be noted that the distributions of the above examples are only partial. The real distributions for all these examples are infinite, i.e. due to infinite loops in certain branches of the computation[7] the running time can be infinite with finite probability. This means that the expected value of the default sequential strategy $\mathcal{S}_\infty$ for all real distributions would be infinite, whereas $\mathcal{S}_{\mathrm{univ}}$ and $\mathcal{S}_*$ still have finite expected value for all these examples.[8] A detailed comparison of the sequential universal strategy with other sequential search strategies (e.g. iterative deepening search) that guarantee finite expected values is part of ongoing research and will be published separately.

Here we are mainly interested in speedup results of $\mathcal{S}^k_*$ and $\mathcal{S}^k_{\mathrm{univ}}$ which are shown for four of the above examples in Figure 4. The uppermost diagram for each example shows $n(t)$, the frequency of the observed running times which is approximately proportional to $p(t)$[9] together

[6] The time required for SETHEO to perform one inference step was used as time unit for our measurements.

[7] The infinite loops are essentially due to the undecidability of first order predicate logic.

[8] Note that for all nontrivial distributions p, $\mathcal{S}_{\mathrm{univ}}$ as well as $\mathcal{S}_*$ and $\mathcal{S}_+$ have finite expected running time.

[9] The normalized frequency $\frac{n(t)}{\sum_t n(t)}$ gives an approximation of $p(t)$.

SETHEO- example	$E(\mathcal{S}_\infty)$	t_*	$E(\mathcal{S}_*)$	$\frac{E(\mathcal{S}_*)}{E(\mathcal{S}_\infty)}$ $t_{\text{off}} = 100$	$E(\mathcal{S}_{\text{univ}})$	$\frac{E(\mathcal{S}_{\text{univ}})}{E(\mathcal{S}_\infty)}$
times4	1640888	160	95672	0.06	327462	0.20
mult3	1104297	88	151617	0.137	227442	0.206
s6-i14	4147714	125	591450	0.143	1870577	0.451
8-puzzle-d15	43533	95	12006	0.276	22071	0.507
i1-d7	4376	104	1992	0.455	3656	0.836
s5-i10	6184	2162	4524	0.732	5725	0.926
non-obvious	34567	1457	14742	0.426	36242	1.05
ip1-i19	1532500	12118	758128	0.495	1656073	1.08
queens10	732790	∞	732790	1.0	2092472	2.86
s8t1-i10	4998	∞	4998	1.0	14043	2.81

Table 1: Empirical performance comparison of the two sequential strategies $\mathcal{S}_*^k$ and $\mathcal{S}_{\text{univ}}^k$ with the default sequential strategy $\mathcal{S}_\infty$ on a set of theorem proving examples. The table is ordered by the last column, i.e. by the relative expected time of the sequential universal strategy.

with the mean running times of the different sequential strategies. The two lower graphs which have the same abscissae show the speedup curves Sp_{opt} and Sp_{univ} and the ratio of these two, which gives an estimate of how far $\mathcal{S}_{\text{univ}}^k$ is from optimal.

4.1 Speedup of the optimal uniform strategy

For all examples except the last one in Figure 4 the speedup of the optimal uniform strategy is close to linear if the number of processors is not too large. This is the region where $k\,q(t_*) \ll 1$ and therefore Theorem 5 applies. We also see that at least in this region of few processors $\mathcal{S}_*^k$ is very close to optimal. For the example "mult3", even with 1000 processors a speedup of 870 can be achieved. Close to linear speedup has also been observed in all the other examples listed in Table 1. For the last example "s8t1-i10" the speedup Sp_{opt} of $\mathcal{S}_*^k$ is clearly sublinear, even for a small number of processors. The reason for this is that $q(t_*) = 1$ and therefore $k\,q(t_*) \geq 1$, so the requirement for close to linear speedup is not fulfilled.

4.2 Speedup of the universal strategy

A drawback of the universal strategy is that all runs which are shorter than the shortest possible running time t_{min} of $A(x)$ are superfluous. An obvious way to improve $\mathcal{S}_{\text{univ}}^k$ is therefore to multiply all the time bounds t_i by an offset value t_{off}. The best value for t_{off} is close to t_*^k, but t_*^k is not known in practice. However, it is also helpful to have some knowledge about the shortest possible running time and use this to get a better offset value.

In all the parallel experiments we used $t_{\text{off}} = 10000$. For any fixed input x and fixed number of processors there is a value of t_{off} that minimizes the expected running time $E(\mathcal{S}_{\text{univ}}^k)$. Nevertheless, we always used $t_{\text{off}} = 10000$, since we are interested in a *universal* strategy, that works well for all examples without any knowledge of the distribution. Note that for $t_{\text{off}} = \infty$ the universal strategy is equal to the strategy $\mathcal{PAR}^k = \mathcal{S}_\infty^k$ which does nothing else than competition of k independent runs of $A(x)$.

In all the examples the speedup Sp_{univ} of $\mathcal{S}^k_{\mathrm{univ}}$ is close to that of $\mathcal{S}^k_*$. As we observed, for a fixed probability distribution, if k is fixed large enough, then as t_{off} is getting bigger, the speedup of $\mathcal{S}^k_{\mathrm{univ}}$ is increasing, asymptotically approaching the optimal possible speedup. In other words, for large (but fixed) number of processors the highest speedup can be achieved with $t_{\mathrm{off}} = \infty$, i.e. if on all processors $A(x)$ is being executed without interrupt. If for example $k \gg 1/q(t_{\mathrm{min}})$ and $t_{\mathrm{off}} \geq t_{\mathrm{min}}$ no processor is interrupted before t_{min} and with high probability one of the processors stops within the first t_{min} steps and thus the expected running time is close to t_{min}. On the other hand no strategy can stop before t_{min} steps. Thus it makes no big difference if all processors run with the optimal time bound or without any time bound. If however the offset of $\mathcal{S}^k_{\mathrm{univ}}$ is too small, the logarithmic overhead decreases performance drastically.

These results show that the simplest parallel strategy $\mathcal{PAR}^k = \mathcal{S}^k_\infty$ performs very well if the number of processors is really large. For small k however, in most cases $\mathcal{S}^k_{\mathrm{univ}}$ is better than running $A(x)$ without interrupt. Therefore, at least for small k the universal strategy with a smaller offset value is extremely useful. We found that the performance of the sequential strategy $\mathcal{S}_{\mathrm{univ}}$ does not depend strongly on t_{off} and that as a reasonable heuristic value t_{off} should be somewhat bigger than the shortest possible running time.

5 Implementation

An implementation of $\mathcal{PAR}^k$, which is identical to the parallel default strategy $\mathcal{S}^k_\infty$ for SETHEO has been performed on a network of 110 HP 9000/720 workstations with a total raw performance of more than 6000 MIPS. The only overhead due to parallel implementation is caused by the broadcast used to start and stop the whole system and takes a total time between some milliseconds and 12 seconds, depending on the safety requirements on the communication protocol used. Therefore, for the hard theorem proving problems which require use of a parallel machine these times are negligible.

Since these values do not change if the universal strategy $\mathcal{S}^k_{\mathrm{univ}}$ is used, the speedup figures computed in the previous section will with almost no change transfer to the 110 processor implementation. If a pure broadcast on the Ethernet is used, the communication time for starting and stopping is constant (i.e. it does not depend on the number of workstations used) and therefore, as many workstations as available can be used. For more details on the implementation see [2] and [3].

With this implementation of $\mathcal{S}^k_\infty$ it was possible to prove new theorems which could not be solved by SETHEO. Another significant step forward is expected with a parallel implementation of $\mathcal{S}^k_{\mathrm{univ}}$, since this solves the problem of possible infinite loops as already mentioned.

References

[1] M. Luby, A. Sinclair, and D. Zuckerman. Optimal speedup of las vegas algorithms. *to appear in: Proceedings of the Second Israeli Symposium on Theory of Computing and Systems, 1993* Technical report TR-93-010, International Computer Science Institute, Berkeley, March 1993.

[2] W. Ertel. OR-Parallel theorem proving with random competition. *Proceedings of Logic Programming and Automated Reasoning,* St. Petersburg, July 1992, Springer Lecture Notes in AI Vol. 624, pp. 226–237.

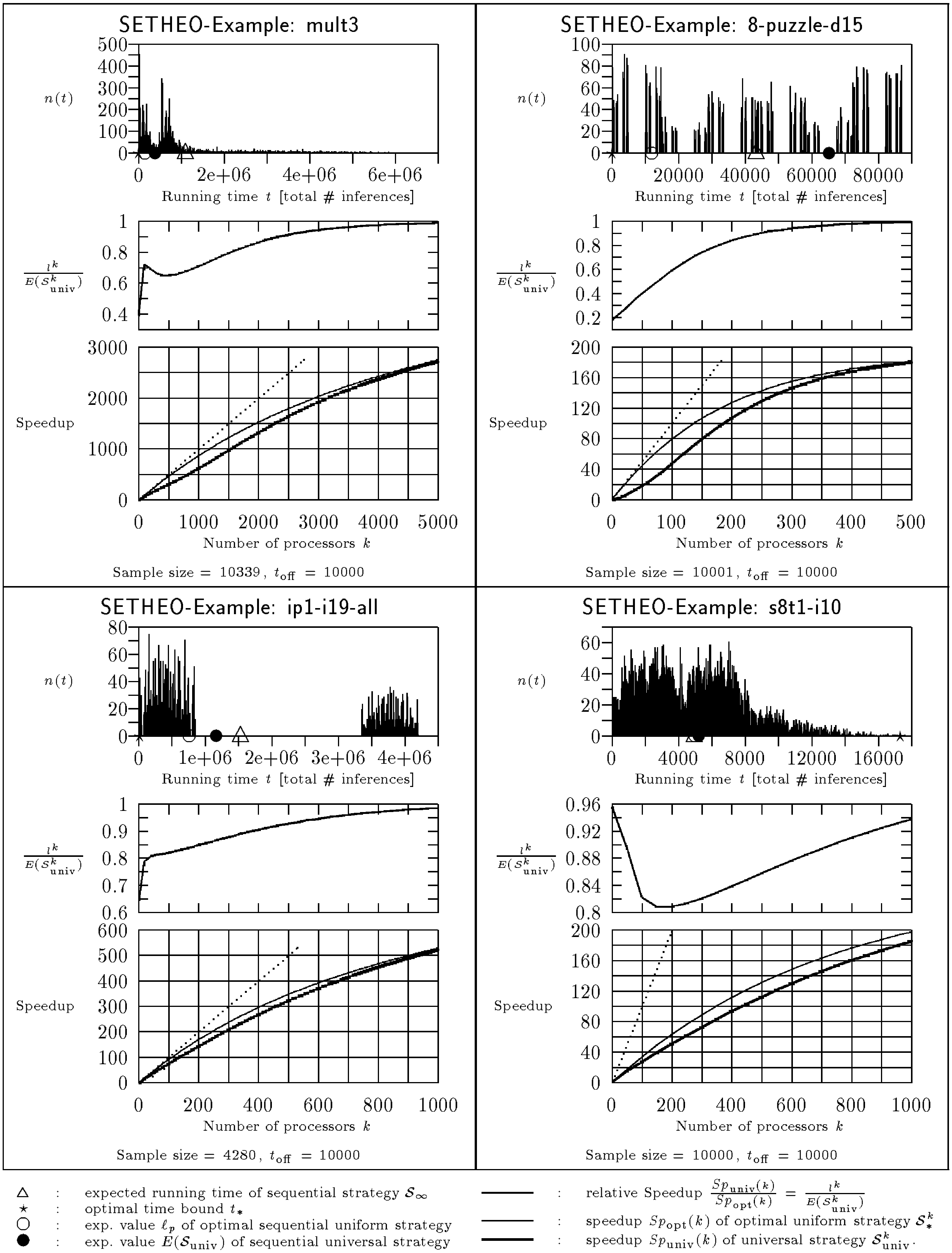

Figure 4: Empirical running time distribution and speedup of optimal repeating strategy and universal strategy computed for four examples.

[3] W. Ertel. *Parallele Suche mit randomisiertem Wettbewerb in Inferenzsystemen.* Infix-Verlag, St. Augustin, Germany, DISKI-series, vol. 25, 1993 PhD-Thesis, Technische Universität München.

[4] V. Kumar, V. N. Rao. Scalable parallel formulations of depth-first search. *in Vipin Kumar, P.S. Gopalakrishnan and Laveen N. Kanal (Hrsg.), Parallel algorithmus for maschine intelligence and vision* Springer Verlag, New York 1990, p. 1–41.

[5] R. Letz, J. Schumann, S. Bayerl and W. Bibel. SETHEO, a high–performance theorem prover. Journal of Automated Reasoning 8(2), 1992, p. 183–212.