Optimal Schedules for Parallelizing Anytime Algorithms: The Case of Independent Processes

Lev Finkelstein, Shaul Markovitch and Ehud Rivlin
Computer Science Department
Technion, Haifa 32000
Israel
{lev,shaulm,ehudr}@cs.technion.ac.il

Abstract

The performance of anytime algorithms having a non-deterministic nature can be improved by solving simultaneously several instances of the algorithm-problem pairs. These pairs may include different instances of a problem (like starting from a different initial state), different algorithms (if several alternatives exist), or several instances of the same algorithm (for non-deterministic algorithms).

In this paper we present a general framework for optimal parallelization of independent processes. We show a mathematical model for this framework, present algorithms for optimal scheduling, and demonstrate its usefulness on a real problem.

1 Introduction

Assume that we try to train an expert system for identifying paintings with a good potential. Assume that we possess a set of example paintings and that we have an access to two human experts in this field for tagging examples. The tagged examples are passed to an inductive algorithm for generating a classifier. Our goal is to obtain a classifier of a given quality (judged, for example, using a validation set). Since we do not assume that the two experts have necessarily similar views of what is considered to be a good painting, we want to train a classifier based on examples tagged by the same expert. On one hand, we would like to finish the task as fast as possible, which could be achieved by parallel training. On the other hand, we would like to minimize the total number of examples sent to tagging, which can be achieved by choosing only one expert. We can also design a more elaborated schedule where we switch between the above modes. Assuming that there is a known cost function that specifies the tradeoff between learning time and tagging cost, what should be the optimal schedule for minimizing this cost?

Another example is a Mars mission which is out of water. There are two known sources of water in two opposite directions. Two robots are sent to the two sources. Our goal is to get water as fast as possible using minimal amount of fuel. Again, given the tradeoff between the cost of fuel and the cost of waiting for water, what should be the optimal schedule for the robots?

What is common to the above examples?
There are potential benefits to be gained from the uncertainty in solving more than one instance of the algorithm-problem pair. In the first example, the probabilistic characteristics of the learning process are determined by the learning algorithm. Learning from the two experts can be considered as two instances of the same problem. In the second example, the probabilistic characteristics of each of the robot processes are determined by the terrain and are different for the two directions.

Each process is executed with the purpose of satisfying a given goal predicate. The task is considered accomplished when one of the instances is solved.

If the goal predicate is satisfied at time $t^*$, then it is also satisfied at any time $t > t^*$. This property is equivalent to quality monotonicity of anytime algorithms [3, 8], while solution quality is restricted to Boolean values.

Our objective is to provide a schedule that minimizes the expected cost, maybe under some constraints. Such problem definition is typical for rational-bounded reasoning [20, 18].

This problem resembles those faced by contract algorithms [19, 23]. There, the task is, given resource allocation, to construct an algorithm providing a solution of the highest quality. In our case, the task is, given quality requirement, to construct an algorithm providing a solution using minimal resources.

There are several research works that are related to the above framework. Simple parallelization, with no information exchange between the processes, may speedup the process due to high diversity in solution times. For example, Knight [11] showed that using many reactive instances of RTA* search [12] is more beneficial than using a single deliberative RTA* instance. Yokoo and Kitamura [22] used several search agents in parallel, with agent rearrangement after pre-given periods of time. Janakiram et. al. [10] showed that for many common distributions of solution time, simple parallelization leads to at most linear speedup. One exception is the family of heavy-tailed distributions [6] for which it is possible to obtain super-linear speedup by simple parallelization.

A superlinear speedup can also be obtained when we have access to the internal structure of the processes involved. For example, Clearwater et. al [2] reported superlinear speedup for cryptarithmetic problems as a result of information exchange between the processes. Another example is the works of Kumar and Rao [17, 13, 16] devoted to parallelizing standard search algorithms where superlinear speedup is obtained by dividing the search space.

The case of non-deterministic algorithms which can be restarted an arbitrary number of times, was analyzed in details in the works of Luby et.al [15] for a single processor case and of Luby and Ertel [14] for a multiprocessor case. In particular, it was proven that for a single processor the optimal strategy is to restart the algorithm after a constant amount of time. This strategy was successfully applied to the case of heavy-tailed distributions [6]. This framework, however, cannot be applied to our settings, since we consider a finite number of heterogeneous processes and do not assume availability of the infinite number of instances of the same process.

An interesting approach in a domain-independent direction is based on “portfolio” construction [9, 7]. This approach provides the processes (agents) with a different amount of resources, which enabled to reduce both the expected resource usage and its variance. The experiments showed the applicability of this approach to many hard computational problems.
In the field of anytime algorithms, similar works were mostly concentrated on scheduling different anytime algorithms or decision procedures in order to maximize overall utility (like in the work of Boddy and Dean [1]). Their settings, however, are different from those presented above.

The goal of this research is to develop algorithms that design an optimal scheduling policy based on the statistical characteristics of the process(es). We present a formal framework for scheduling parallel anytime algorithms. In our previous work [5] we present such framework for the case where the processes share resources (a single processor model). In this work we present similar framework for the case where the processes are independent in the sense that usage of resources by one process does not imply constraints on usage of resources of the others. The framework assumes that we know the probability of the goal condition to be satisfied as a function of time (a performance profile [21, 1] restricted to Boolean quality values). We analyze the properties of optimal schedules for suspend-resume model and present an algorithm for building optimal schedules and demonstrate experimental results for a Latin Square problem.

2 Motivation: a simple example

Before starting the formal discussion, we would like to give a simple example. Assume that two instances of DFS with random tie-breaking are applied to a simple search space shown in Figure 1. We assume that each process uses a separate processor. There is a very large number of paths to the goal, half of them of length 10, and the other half of length 40. When one of the instances finds a solution, the task is considered accomplished. We have

![Figure 1: A simple search task: two instances of DFS search for a path from A to B. Scheduling the processes may reduce cost.](image)

two utility components – the time, which is the elapsed time required for the system to find a solution, and the resources, which is total CPU time consumed by both processes. If the two search processes start together, the expected time usage will be $10 \times \frac{3}{4} + 40 \times \frac{1}{4} = 17.5$ units, while the expected resource usage will be $20 \times \frac{3}{4} + 80 \times \frac{1}{4} = 35$ units. If we apply only one instance, both time and resource usage will be $10 \times \frac{1}{2} + 40 \times \frac{1}{2} = 25$ units. Assume now that the first process is active for the first 10 time units, then it stops and the second process is active for the next 10 time units, and then the second process stops and
the first process continues the execution. Both the expected time and resource usage will be $10 \times 1/2 + 20 \times 1/4 + 50 \times 1/4 = 22.5$ units. Finally, if we consider a schedule, where the processes start together and work in a simultaneous manner for 10 time units, and only one continues the execution in the case of failure, the expected time will be $10 \times 3/4 + 40 \times 1/4 = 17.5$ units, while the expected resource usage will be $20 \times 3/4 + 50 \times 1/4 = 27.5$ units. It is easy to see, that the results for the last two scenarios, with interleaved execution, are better than the other results. The tradeoff between time and resource cost determines the best model between the two – if time and resource cost are equal, the last two models are equivalent.

3 A framework for parallelization scheduling

In this section we formalize the intuitive description of parallelization scheduling. This framework is similar to the frameworks presented in [4, 5].

Let $S$ be a set of states, $t$ be a time variable with non-negative real values, and $A$ be a random process such that each realization (trajectory) $A(t)$ of $A$ represents a mapping from $\mathbb{R}^+$ to $S$. Let $G : S \rightarrow \{0, 1\}$ be a goal predicate, where 0 corresponds to False and 1 corresponds to True. Let $A$ be monotonic over $G$, i.e. for each trajectory $A(t)$ of $A$ the function $\hat{G}_A(t) = G(A(t))$ is a non-decreasing function. Under the above assumptions, $\hat{G}_A(t)$ is a step function with at most one discontinuity point, which we denote by $\hat{t}_{A,G}$ (this is the first point after which the goal predicate is true). If $\hat{G}_A(t)$ is always 0, we say that $\hat{t}_{A,G}$ is not defined. Therefore, we can define a random variable, which for each trajectory $A(t)$ of $A$ with $\hat{t}_{A,G}$ defined, corresponds to $\hat{t}_{A,G}$. The behavior of this variable can be described by its distribution function $F(t)$. At the points where $F(t)$ is differentiable, we use the probability density $f(t) = F'(t)$.

This scheme resembles the one used in anytime algorithms. The goal predicate can be viewed as a special case of the quality measurement used in anytime algorithms, and the requirement for its non-decreasing value is a standard requirement of these algorithms. The trajectories of $A$ correspond to conditional performance profiles [24, 23].

In practice, not every trajectory of $A$ leads to goal predicate satisfaction even after infinitely large time. That is why we define the probability of success $p$ as the probability of selecting a trajectory $A(t)$ which leads to satisfaction of $G$ in a finite time (i.e. with $\hat{t}_{A,G}$ defined) \(^{1}\). Other way to express the possibility that the process will not stop at all, is to use profiles, which approach $1 - p$ when $t \rightarrow \infty$. We prefer to use $p$ explicitly because the distribution function must meet the requirement $\lim_{t \rightarrow \infty} F(t) = 1$.

Assume now that we have a system of $n$ random processes $A_1, \ldots, A_n$ with corresponding distribution functions $F_1, \ldots, F_n$ and goal predicates $G_1, \ldots, G_n$. We define a schedule of the system as a set of binary functions $\{\theta_i\}$, where at each moment $t$, the $i$-th process is active if $\theta_i(t) = 1$ and idle otherwise. We refer to this scheme as suspend/resume scheduling.

A possible generalization of this framework is to extend the suspend/resume control to a more refined mechanism allowing us to determine the intensity with which each process acts. For software processes that means to vary the fraction of CPU usage; for tasks like robot navigation this implies changing the speed of the robots. Mathematically, using intensity

\(^{1}\) Another way to express the possibility that the process will not stop at all, is to use profiles, which approach $1 - p$ when $t \rightarrow \infty$. We prefer to use $p$ explicitly because the distribution function must meet the requirement $\lim_{t \rightarrow \infty} F(t) = 1$. 
control is equivalent to replacing the binary functions \( \theta_i(t) \) with continuous functions with a range between 0 and 1.

Note that scheduling makes the term time ambiguous. On one hand, we have the subjective time for each process which is consumed only when the process is active. This kind of time corresponds to some resource consumed by the process. On the other hand, we have an objective time measured from the point of view of an external observer. The performance profile of each algorithm is defined over its subjective time, while the cost function (see below) may use both kinds of times. Since we are using several processes, all the formulas in this paper are based on the objective time.

Let us denote by \( \sigma_i(t) \) the total time that process \( i \) has been active before \( t \). By definition,

\[
\sigma_i(t) = \int_0^t \theta_i(x) dx.
\]

In practice \( \sigma_i(t) \) provides the mapping from the objective time \( t \) to the subjective time of the \( i \)-th process, and we refer to them as subjective schedule functions. Since \( \theta_i \) can be obtained from \( \sigma_i \) by differentiation, we often describe schedules by \( \{ \sigma_i \} \) instead of \( \{ \theta_i \} \).

The processes \( \{ A_i \} \) with goal predicates \( \{ G_i \} \) running under schedules \( \{ \sigma_i \} \) result in a new process \( A \) with a goal predicate \( G \). \( G \) is the disjunction of \( G_i \) \( G(t) = \bigvee_i G_i(t) \), and therefore \( A \) is monotonic over \( G \). We denote the distribution function of the corresponding random variable by \( F_n(t, \sigma_1, \ldots, \sigma_n) \), and the corresponding distribution density by \( f_n(t, \sigma_1, \ldots, \sigma_n) \).

Assume that we are given a monotonic non-decreasing cost function \( u(t, t_1, \ldots, t_n) \), which depends on the objective time \( t \) and the subjective times per process \( t_i \). Since the subjective times can be represented as \( \sigma_i(t) \), we actually have \( u = u(t, \sigma_i(t), \ldots, \sigma_n(t)) \).

The expected cost of schedule \( \{ \sigma_i \} \) can be, therefore, expressed as

\[
E_u(\sigma_1, \ldots, \sigma_n) = \int_0^{+\infty} u(t, \sigma_1, \ldots, \sigma_n) f_n(t, \sigma_1, \ldots, \sigma_n) dt
\]

(for the sake of readability, we omit \( t \) in \( \sigma_i(t) \)). Under the suspend-resume model assumptions, \( \sigma_i \) must be differentiable (except a countable set of rescheduling points), and have derivatives of 0 or 1 that would ensure correct values for \( \theta_i \). Under intensity control assumptions, the derivatives of \( \sigma_i \) must lie between 0 and 1.

We consider two alternative setups regarding resource sharing between the processes:

1. The processes share resources on a mutual exclusion basis. An example for such framework are several algorithms running on a single processor.

2. The processes are fully independent. An example for such framework are \( n \) algorithms running on \( n \) processors.

The difference between these two alternatives is the additional constraints on \( \sigma_i \): in the case of shared resources the sum of derivatives of \( \sigma_i \) cannot exceed 1, while in the case of independent processes this constraint does not exist. Our goal is to find a schedule that minimizes the expected cost (2) under the corresponding constraints.

The current paper is devoted to the case of independent processes. The case of shared resources was studied in [5].

\footnote{It is possible to show that the generalization to the case where the probabilities of success \( p \) are not 1 is equivalent to replacing \( F_i(t) \) and \( f_i(t) \) by \( p_iF_i(t) \) and \( p_if_i(t) \) respectively.}
4 Suspend-resume based scheduling

In this section we consider the case of suspend-resume based control ($\sigma_i$ are continuous functions with derivatives 0 or 1).

Claim 1 The expressions for the goal-time distribution $F_n(t, \sigma_1, \ldots, \sigma_n)$ and the expected cost $E_u(\sigma_1, \ldots, \sigma_n)$ are as follows:\footnote{u'_i and u'_{\sigma_i} stand for partial derivatives of $u$ by each of its variables.}

\begin{equation}
F_n(t, \sigma_1, \ldots, \sigma_n) = 1 - \prod_{i=1}^{n} (1 - F_i(\sigma_i)),
\end{equation}

\begin{equation}
E_u(\sigma_1, \ldots, \sigma_n) = \int_0^{+\infty} \left( u'_i + \sum_{i=1}^{n} \sigma_i^j u'_{\sigma_j} \right) \prod_{i=1}^{n} (1 - F_i(\sigma_i)) dt,
\end{equation}

where $u'_i$ and $u'_{\sigma_i}$ stand for partial derivatives of $u(t, \sigma_1, \ldots, \sigma_n)$ by each of its variables.

Proof: Let $t_i$ be the time of reaching the goal by the $i$-th process if it would have acted alone (if the process fails to reach the goal, we consider $t_i = \infty$). Let $t^*$ be the time of reaching the goal by the team. In this case, $t^*$ is distributed according to $F_n(t, \sigma_1, \ldots, \sigma_n)$, and $t_i$ are distributed according to $F(t)$. Thus, based on the processes’ independence given a schedule, we obtain

\begin{align*}
F_n(t, \sigma_1, \ldots, \sigma_n) &= P(t^* \leq t) = 1 - P(t^* > t) = 1 - P(t_1 > \sigma_1(t)) \cdots P(t_n > \sigma_n(t)) = \\
&= 1 - (1 - F_1(\sigma_1(t))) \cdots (1 - F_n(\sigma_n(t))) = 1 - \prod_{i=1}^{n} (1 - F_i(\sigma_i(t))),
\end{align*}

which corresponds to (3). Since $F(t)$ is a distribution over time, we assume $F(t) = 0$ for $t \leq 0$.

The average cost function will therefore be

\begin{align*}
E_u(\sigma_1, \ldots, \sigma_n) &= \int_0^{+\infty} u(t, \sigma_1, \ldots, \sigma_n) f_n(t, \sigma_1, \ldots, \sigma_n) dt = \\
&= -\int_0^{+\infty} u(t, \sigma_1, \ldots, \sigma_n) d(1 - F_n(t, \sigma_1, \ldots, \sigma_n)) = \\
&= -u(t, \sigma_1, \ldots, \sigma_n) (1 - F_n(t, \sigma_1, \ldots, \sigma_n)) \big|_{t=0}^{+\infty} + \int_0^{+\infty} \frac{du(t, \sigma_1, \ldots, \sigma_n)}{dt} \prod_{i=1}^{n} (1 - F_i(\sigma_i)) dt = \\
&= \int_0^{+\infty} \left( u'_i + \sum_{i=1}^{n} \sigma_i^j u'_{\sigma_j} \right) \prod_{i=1}^{n} (1 - F_i(\sigma_i)) dt,
\end{align*}

which completes the proof.

Q.E.D.

Note that in the case of $\sigma_i(t) = t$ and $F_i(t) = F(t)$ for all $i$ (parallel application of $n$ homogeneous processes), we obtain the formula presented in [10], i.e., $F_n(t) = 1 - (1 - F(t))^n$.

In this section we assume that the total cost is a linear combination of the objective time cost and the resource cost, and that the resource cost is proportional to the subjective time cost.
spent:
\[ u(t, \sigma_1, \ldots, \sigma_n) = at + b \sum_{i=1}^{n} \sigma_i(t). \]

In this case our minimization problem becomes
\[ E_u(\sigma_1, \ldots, \sigma_n) = \int_0^\infty \left( a + b \sum_{i=1}^{n} \sigma_i^t \right) \prod_{j=1}^{n} (1 - F_j(\sigma_j)) \, dt \to \min, \]
and for the case of suspend-resume scheduling we have the constraints following from the nature of the problem:
\[ \sigma_i^t \in \{0, 1\}. \]

Without loss of generality, in the rest of the paper we consider an equivalent minimization problem normalized by the value \((a+b)\). If we denote \(c = b/(a+b)\), this new problem becomes
\[ E_u(\sigma_1, \ldots, \sigma_n) = \int_0^\infty \left( 1 - c + c \sum_{i=1}^{n} \sigma_i^t \right) \prod_{j=1}^{n} (1 - F_j(\sigma_j)) \, dt \to \min, \]

Obviously, the solutions for (8) and (6) are identical, and \(c\) stands for the normalized resource weight.

4.1 Necessary conditions for optimal solution for two processes

Let \(A_1\) and \(A_2\) be two independent processes. For suspend-resume model, only three states of the system are possible: \(A_1\) is active and \(A_2\) is idle \((S^{01})\); \(A_1\) is idle and \(A_2\) is active \((S^{10})\); and both \(A_1\) and \(A_2\) are active \((S^{11})\). We ignore the case where both processes are idle, since removing such state from the schedule will not increase the cost.

Assume that the system continuously alternates between the three states: \(S^{01} \to S^{10} \to S^{11} \to S^{01} \to \ldots\). This scheme is general if the time spent at each state is allowed to be zero. We call each triplet \(\langle S^{01}, S^{10}, S^{11} \rangle\) a phase and denote phase \(k\) by \(\Phi_k\). We denote the time when state \(S^x\) of \(\Phi_k\) ends by \(t^x_k\) \((x \in \{01, 10, 11\})\). For illustration see Figure 2. Let us denote

\[ \zeta^x_k = \text{the total cumulative time spent in state } S^x \text{ in phases 1 to } k. \]

There is a one-to-one correspondence between the sequence \(\{\zeta^x_k\}\) and the sequence \(\{t^x_k\}\), and
\[ \begin{align*}
\zeta^{10}_{k-1} + \zeta^{11}_{k-1} + \zeta^{01}_k &= t^{01}_k, \\
\zeta^{11}_{k-1} + \zeta^{01}_k + \zeta^{10}_k &= t^{10}_k, \\
\zeta^{01}_k + \zeta^{10}_k + \zeta^{11}_k &= t^{11}_k.
\end{align*} \]

Figure 2: Notations for times, states and phases.
By definition of $t^*_k$, the subjective schedule functions $\sigma_1$ and $\sigma_2$ in time intervals $[t^1_{k-1}, t^0_k]$ (state $S^{01}$) have the form

$$\sigma_1(t) = t - t^1_{k-1} + \xi^{01}_{k-1} + \xi^{11}_{k-1} = t - \xi^{10}_{k-1},$$
$$\sigma_2(t) = \xi^{10}_{k-1} + \xi^{11}_{k-1}. \quad (10)$$

Similarly, in the intervals $[t^0_k, t^{10}_k]$ (state $S^{10}$) the subjective schedule functions are defined as

$$\sigma_1(t) = \xi^{11}_{k-1} + \xi^{01}_k,$$
$$\sigma_2(t) = t - t^{10}_k + \xi^{11}_{k-1} + \xi^{01}_k = t - \xi^{01}_k. \quad (11)$$

Finally, in the intervals $[t^{10}_k, t^{11}_k]$ (state $S^{11}$) the subjective schedule functions can be represented as

$$\sigma_1(t) = t - t^{10}_k + \xi^{11}_{k-1} + \xi^{01}_k = t - \xi^{10}_k,$$
$$\sigma_2(t) = t - t^{10}_k + \xi^{11}_{k-1} + \xi^{01}_k = t - \xi^{01}_k. \quad (12)$$

Substituting the expressions for $\sigma_i$ into (8) and using the constraints for suspend-resume scheduling (7) we obtain

$$F_n(\sigma_1, \ldots, \sigma_n) =$$

$$\sum_{k=0}^{\infty} \left[ \int_{t^{10}_{k-1}}^{t^{10}_k} (1 - F_1(t - \xi^{10}_{k-1}))(1 - F_2(\xi^{01}_k + \xi^{11}_{k-1}))dt + \int_{t^{10}_k}^{t^{01}_k} (1 - F_1(\xi^{11}_{k-1} + \xi^{01}_k))(1 - F_2(t - \xi^{01}_k))dt + (1 + c) \int_{t^{11}_k}^{t^{11}_{k-1}} (1 - F_1(t - \xi^{10}_k))(1 - F_2(t - \xi^{01}_k))dt \right] =$$

$$\sum_{k=0}^{\infty} \left[ (1 - F_2(\xi^{10}_{k-1} + \xi^{11}_{k-1})) \int_{t^{11}_{k-1} - \xi^{01}_{k-1}}^{t^{10}_{k-1} - \xi^{01}_{k-1}} (1 - F_1(x))dx + (1 - F_1(\xi^{11}_{k-1} + \xi^{01}_k)) \int_{\xi^{11}_{k-1} - \xi^{01}_k}^{t^{11}_{k-1} - \xi^{01}_k} (1 - F_2(x))dx + (1 + c) \int_{t^{11}_{k-1} - \xi^{11}_{k-1} - \xi^{01}_k}^{t^{11}_{k-1} - \xi^{01}_k} (1 - F_1(x + \xi^{01}_k))(1 - F_2(x + \xi^{10}_k))dx \right].$$

Using the expressions (9), we can rewrite the last expression as

$$\sum_{k=0}^{\infty} \left[ (1 - F_2(\xi^{10}_{k-1} + \xi^{11}_{k-1})) \int_{\xi^{01}_{k-1} + \xi^{11}_{k-1}}^{\xi^{11}_{k-1} + \xi^{01}_k} (1 - F_1(x))dx + (1 - F_1(\xi^{11}_{k-1} + \xi^{01}_k)) \int_{\xi^{01}_{k-1} + \xi^{11}_{k-1}}^{\xi^{11}_{k-1} + \xi^{01}_k} (1 - F_2(x))dx + (1 + c) \int_{\xi^{11}_{k-1} + \xi^{01}_k}^{\xi^{11}_{k-1} + \xi^{01}_k} (1 - F_1(x + \xi^{01}_k))(1 - F_2(x + \xi^{10}_k))dx \right],$$

8
and finally we can rewrite our minimization problem as follows:

\[
E_u(\zeta^{01}_1, \zeta^{10}_k, \zeta^{11}_k, \ldots) = \\
\sum_{k=0}^{\infty} \left[ (1 - F_2(\zeta^{01}_{k-1} + \zeta^{11}_{k-1})) \int_{\zeta^{01}_{k-1}}^{\zeta^{01}_k} (1 - F_1(x + \zeta^{11}_{k-1})) dx + \\
(1 - F_1(\zeta^{11}_{k-1} + \zeta^{01}_k)) \int_{\zeta^{10}_{k-1}}^{\zeta^{10}_k} (1 - F_2(x + \zeta^{11}_{k-1})) dx + \\
(1 + c) \int_{\zeta^{11}_{k-1}}^{\zeta^{10}_k} (1 - F_1(x + \zeta^{01}_k))(1 - F_2(x + \zeta^{10}_k)) dx \right] \rightarrow \min .
\] (13)

The minimization problem (13) is equivalent to the original problem (8), and the dependency between their solutions is described by (10), (11) and (12). The only constraints are the monotonicity of the sequence \{\zeta_k^x\} for a fixed \(x\), and therefore we obtain

\[
\zeta_0^x = 0 \leq \zeta_1^x \leq \cdots \leq \zeta_k^x \leq \cdots .
\] (14)

Since (13) reaches its optimal values either when

\[
\frac{du}{d\zeta_k^x} = 0 \text{ for } k = 1, \ldots, n, \ldots,
\] (15)

or on the border described by (14), we can prove the following theorem:

**Theorem 1 (The chain theorem for two processes)**

1. The value for \(\zeta^{01}_{k+1}\) may either be \(\zeta^{01}_k\), or can be computed given \(\zeta^{11}_{k-1}, \zeta^{01}_k, \zeta^{10}_k\) and \(\zeta^{11}_k\) using the formula

\[
- f_2(\zeta^{01}_k + \zeta^{11}_k) \int_{\zeta^{01}_k}^{\zeta^{01}_{k+1}} (1 - F_1(x + \zeta^{11}_k)) dx - \\
(1 + c) \int_{\zeta^{11}_{k-1}}^{\zeta^{10}_k} (1 - F_1(x + \zeta^{01}_k)) f_2(x + \zeta^{10}_k) dx + \\
(1 - F_1(\zeta^{11}_{k-1} + \zeta^{01}_k))(1 - F_2(\zeta^{11}_{k-1} + \zeta^{10}_k)) - \\
(1 - F_1(\zeta^{11}_k + \zeta^{01}_{k+1}))(1 - F_2(\zeta^{10}_k + \zeta^{11}_k)) = 0.
\] (16)

2. The value for \(\zeta^{10}_{k+1}\) may either be \(\zeta^{10}_k\), or can be computed given \(\zeta^{01}_k, \zeta^{10}_k, \zeta^{11}_k\) and \(\zeta^{01}_{k+1}\) using the formula

\[
- f_2(\zeta^{01}_k + \zeta^{11}_k) \int_{\zeta^{01}_k}^{\zeta^{01}_{k+1}} (1 - F_1(x + \zeta^{11}_k)) dx - \\
f_1(\zeta^{11}_k + \zeta^{01}_{k+1}) \int_{\zeta^{01}_k}^{\zeta^{01}_{k+1}} (1 - F_2(x + \zeta^{11}_k)) dx + \\
c(1 - F_1(\zeta^{01}_k + \zeta^{11}_k))(1 - F_2(\zeta^{01}_k + \zeta^{11}_k)) - \\
c(1 - F_1(\zeta^{11}_k + \zeta^{01}_{k+1}))(1 - F_2(\zeta^{01}_k + \zeta^{10}_{k+1})) = 0.
\] (17)
3. The value for $\zeta_{k+1}$ may either be $\zeta_k$, or can be computed given $\zeta_k^0$, $\zeta_k^1$, $\zeta_{k+1}^0$ and $\zeta_{k+1}^1$ using the formula

$$
-f_1(\zeta_k^1 + \zeta_{k+1}^0) \int_{\zeta_k^1}^{\zeta_k^0} (1 - F_2(x + \zeta_k^1))dx -
$$

$$(1 + c) \int_{\zeta_k^1}^{\zeta_k^+} f_1(x + \zeta_k^1) (1 - F_2(x + \zeta_k^1))dx +$$

$$(1 - F_1(\zeta_k^1 + \zeta_{k+1}^0))(1 - F_2(\zeta_k^0 + \zeta_{k+1}^1)) -$$

$$(1 - F_1(\zeta_{k+1}^0 + \zeta_{k+1}^1))(1 - F_2(\zeta_{k+1}^0 + \zeta_{k+1}^1)) = 0. $$

The proof of the theorem is given in Appendix A.1. This theorem shows a way to compute the values for $\zeta_k^x$ in a sequential manner, each time using four previously computed values. This leads to the following algorithm for building an optimal schedule.

### 4.2 Optimal solution for two processes: an algorithm

Assume that $S^01$ is the first state which takes a non-zero time ($\zeta_1^0 > 0$). By Theorem 1, given the values of $\zeta_0^1 = 0, \zeta_1^0, \zeta_1^1$ we can determine all the possible values for $\zeta_2^01$ (either $\zeta_1^1$ or one of the roots of (16)). Given the values up to $\zeta_0^01$, we can determine the values for $\zeta_2^01$, and so on.

Therefore, the first three values of $\zeta$ (given $\zeta_1^01 \neq 0$) provide us with a tree of possible values of $\zeta_k^x$. The branching factor of this tree is determined by the number of roots of (16), (17) and (18). Each possible sequence $\zeta_1^0, \zeta_1^01, \zeta_1^1, \ldots$ can be evaluated using (13). The series in that expression must converge, so we stop after a finite number of points. For each triplet $Z_1 = \langle \zeta_1^0, \zeta_1^01, \zeta_1^1 \rangle$ we can find the best sequence using one of the standard search algorithms, such as Branch-and-Bound. Let us denote the value of the best sequence for $Z_1$ by $E_u(Z_1)$. Performing global optimization of $E_u(Z_1)$ by $\zeta_1^0, \zeta_1^01$ and $\zeta_1^1$ provides us with an optimal solution for the case where $S^01$ is the first state of non-zero time.

Note, that the value of $\zeta_1^01$ may also be 0 (if $S^01$ or $S^011$ happen first), so we need to compare the value obtained by optimization of the triplet $\zeta_1^01, \zeta_1^011$ and $\zeta_1^1$ with the value obtained by optimization of the triplet $\zeta_1^0, \zeta_1^01, \zeta_1^1$ given $\zeta_1^01 = 0$, and with the value obtained by optimization of the triplet $\zeta_1^0, \zeta_2^0, \zeta_2^1$ given $\zeta_1^0 = \zeta_1^01 = 0$.

The scheme of the algorithm is presented in Figure 3, and the description of the main routine (realized by DFS Branch and Bound method) in Figure 4. Since we compute $\zeta_k^x$ in a sequential manner, in the algorithm we enumerate them by consecutive numbers. Thus, $\zeta_k^01$ is represented by $\zeta_{3k}$, $\zeta_k^011$ is represented by $\zeta_{3k+1}$, and $\zeta_k^1$ is represented by $\zeta_{3k+2}$.

### 4.3 Optimal solution in the case of additional constraints

Assume now that the problem has additional constraints: the solution time is limited by $T$ (or each one of the processes has an upper limit of $T_i$); and the probability of success of the $i$-th process $p_i$ is not necessarily 1.

Assume first that the system is given an upper limit $T$, and that process $i$ succeeds with a known probability of $p_i$. It is possible to show that the expressions for distribution function and average cost have almost the same form as in the regular framework:
procedure optimize
Input: $F_1(t), F_2(t)$ (performance profiles), $c$ (normalized resource weight).
Output: An optimal sequence and its value.

$[sequence_1, val_1] \leftarrow \text{minimize_sequence_by_first_points}(0)$
$[sequence_2, val_2] \leftarrow \text{minimize_sequence_by_first_points}(1)$
$[sequence_3, val_3] \leftarrow \text{minimize_sequence_by_first_points}(2)$

$i^* = \arg \min_i \{val_i\}$

return $[sequence_{i^*}, val_{i^*}]$
end

procedure minimize_sequence_by_first_points($n_{skips}$)
for $i = -2, \ldots, n_{skips}$
    $zetas[i] \leftarrow 0$
end
$\text{curr\_index} \leftarrow n_{skips} + 1$

Using one of the standard minimization methods, find
$zetas[\text{curr\_index}], zetas[\text{curr\_index} + 1]$ and $zetas[\text{curr\_index} + 2]$,
minimizing the value of the function
$\text{build\_optimal\_sequence}(zetas)$.
end

Figure 3: Procedure optimize builds an optimal sequence for the case when up to $n_{skips} \leq 2$
states in the first phase are omitted, compares the results, and returns the best one. Procedure
minimize_sequence_by_first_points returns an optimal sequence and its value.
Figure 4: Procedure \texttt{build\_optimal\_sequence} given the prefix of time sequence, restores the optimal sequence with this prefix using a DFS Branch and Bound search algorithm, and returns the sequence itself and its value. \texttt{n\_skips} parameter keeps the number of skipped states \( S_x \). Auxiliary functions are shown in Figure 5.
1. calculate\_cost(zetas) computes the cost of the sequence (or its part) in accordance with (13),

2. calculate\_partial\_cost(zetas,ζ) computes the additional cost obtained by adding ζ to the sequence,

3. calculate\_next\_zeta(zetas, last\_value) calculates the value of the next ζ using (16), (17) or (18), which is greater than last\_value. If no such a solution exists, the maximal time value is returned,

4. task\_accomplished(zetas) returns true when the task may be considered to be accomplished (e.g., either maximal possible time is over, or the probability of error is negligible).

Figure 5: Auxiliary functions used in the optimal schedule algorithm.

Claim 2 Let the system stops its work after time T, and let p_i be a probability of success for the i-th process. Then the expressions for the goal-time distribution and expected cost are as follows:

\[ F_n(t, \sigma_1, \ldots, \sigma_n) = 1 - \prod_{i=1}^{n}(1 - p_i F_i(\sigma_i)) \quad (t \leq T), \]  \hspace{1cm} (19)

\[ E_n(\sigma_1, \ldots, \sigma_n) = \int_0^T \left( u'_i + \sum_{i=1}^{n} \sigma'_i u'_{\sigma_i} \right) \prod_{i=1}^{n}(1 - p_i F_i(\sigma_i)) dt. \]  \hspace{1cm} (20)

The proof is similar to the proof of the Claim 1.

This claim shows, that all the formulas used in the previous sections are valid for the current settings, with two differences:

1. We use \(p_j F_j\) instead of \(F_j\) and \(p_j f_j\) instead of \(f_j\).

2. All the integrals are from \(0\) to \(T\) instead of \(0\) to \(\infty\).

The first change may be easily incorporated to all the algorithms. The second condition implies some more changes in the chain theorems and the algorithms. The chain theorem for \(n\) processes will be as follows:

Theorem 2 The value for \(\zeta_j\) can be calculated by one of three methods:

1. It can be equal to \(\zeta_{j-3}\).

2. It can be computed given the previous 3 values of \(\zeta\) using formulas (16), (17) or (18).

3. It can be calculated by the formula

\[ \zeta_j = T - \sum_{i=1}^{n-1} \zeta_{j-i}. \]  \hspace{1cm} (21)
The first two conditions are similar to Theorem 1, while the third one provides one more boundary condition, which is expressed by (9). This third alternative adds one more branch to the DFS Branch and Bound algorithm; the rest of the algorithm remains unchanged.

Similar changes in the algorithms are performed in the case of the maximal allowed time $T_i$ per process. In practice, we always use this limitation setting $T_i$ such that the probability for $A_i$ to reach the goal after $T_i$, $p_i(1 - F_i(T_i))$, becomes negligible.

5 Process scheduling by intensity control

Using intensity control is equivalent to replacing the binary functions $\theta_i(t)$ with continuous functions with a range between 0 and 1. This setup also includes the model described in [9]. In that work, the $i$-th process was allocated $1/f_i$ portions of time, where $f_i$ is a positive constant, and $\sum f_i = 1$. In our terms that means, that the processes work with constant intensities $\theta_i = 1/f_i$ yielding $\sigma_i(t) = t/f_i$.

It is easy to see that all the formulas for the distribution function and the expected cost from Claim 1 are still valid under intensity control settings.

The expression to minimize (8) remains exactly the same as in the suspend-resume model. The constraints, however, are different:

$$0 \leq \sigma_i \leq 1.$$ (22)

For this case, the following important theorem can be proved:

**Theorem 3** If no time cost is taken into account ($c = 1$), the model with shared resources under intensity control settings is equivalent to the model with independent processes under suspend-resume settings. Namely, given a solution for the model with independent processes, we may reconstruct a solution of the same cost for the model with shared resources and vice versa.

This theorem shows a connection between this case and the case of shared resources. The proof of the theorem is given in Appendix A.3.

5.1 Necessary conditions for an optimal solution

We are looking for a set of functions $\{\sigma_1, \sigma_2\}$, providing a solution to the optimization problem (8) under the constraints (22).

First, we show that necessary conditions of Euler-Lagrange which are usually used for solving such type of optimization problems, yield only a false alarm (we want to note also, that even if the conditions above hold, they do not necessarily provide the optimal solution. Moreover, the problems in variation calculus do not necessarily have the minimum, since there is not an analogue for Weierstrass theorem for continuous functions on the closed set). This make us to prove “by hand” the properties of optimal solutions, showing that they should be looked for on a boundary of the region described by (22).

Assume that we want to solve this minimization problem above by traditional methods. We have to minimize a functional $u(\sigma_1, \ldots, \sigma_n)$ expressed by (8). Let us denote the function under the integral sign by $g(t, \sigma_1, \ldots, \sigma_n, \sigma'_1, \ldots, \sigma'_n)$:

$$g(t, \sigma_1, \ldots, \sigma_n, \sigma'_1, \ldots, \sigma'_n) = \left( c + \sum_{i=1}^{n} \sigma_i \right) \prod_{j=1}^{n} (1 - F_j(\sigma_j))$$

$$= \left( c + \sum_{i=1}^{n} \sigma_i \right) \prod_{j=1}^{n} (1 - F_j(\sigma_j))$$

14
A necessary condition of Euler-Lagrange claims, that a set of functions \( \sigma_1, \ldots, \sigma_n \) provide a weak (local) minimum to the functional expressed by (8) only if they satisfy a system of equations of the form
\[
g'_{\sigma_k} - \frac{d}{dt} g'_{\sigma_k} = 0. \tag{23}
\]
In our case,
\[
g'_{\sigma_k} = - \left( c + \sum_{i=1}^{n} \sigma_i \right) f_k(\sigma_k) \prod_{j \neq k} (1 - F_j(\sigma_j)), \tag{24}
\]
and
\[
\frac{d}{dt} g'_{\sigma_k} = \frac{d}{dt} \prod_{j=1}^{n} (1 - F_j(\sigma_j)) = - \sum_{i=1}^{n} \sigma'_i f_i(\sigma_i) \prod_{j \neq i} (1 - F_j(\sigma_j))
\tag{25}
\]
From the last equation we can see that the second term of (23) does not depend on \( k \). Besides, we can ignore the case when one of the terms \( 1 - F_j(\sigma_j) \) is 0, because it is possible only if one of the processes will reach the goal with the probability of 1, and in this case there is no sense in future optimization. Thus, we obtain the following equality:
\[
g'_{\sigma_{k_1}} = g'_{\sigma_{k_2}} \quad \text{for all } k_1, k_2. \tag{26}
\]
Using (24), we can rewrite it as
\[
f_{k_1}(\sigma_{k_1})(1 - F_{k_2}(\sigma_{k_2})) = f_{k_2}(\sigma_{k_2})(1 - F_{k_1}(\sigma_{k_1})), \tag{27}
\]
which leads to
\[
\frac{f_{k_1}(\sigma_{k_1})}{1 - F_{k_1}(\sigma_{k_1})} = \frac{f_{k_2}(\sigma_{k_2})}{1 - F_{k_2}(\sigma_{k_2})}. \tag{28}
\]

The system of equations (28) corresponds to equality of \( n \) elements, and therefore is equivalent to \( n - 1 \) conditions. The last condition can be obtained from (23) for arbitrary \( k \). Substituting (27) to (25), we obtain
\[
\frac{d}{dt} g'_{\sigma_k} = \sum_{i=1}^{n} \sigma'_i f_i(\sigma_i) \prod_{j \neq i} (1 - F_j(\sigma_j)) = - \sum_{i=1}^{n} \sigma'_i f_k(\sigma_k) \prod_{j \neq k} (1 - F_j(\sigma_j)) =
\]
\[
- \left( \sum_{i=1}^{n} \sigma'_i \right) f_k(\sigma_k) \prod_{j \neq k} (1 - F_j(\sigma_j)).
\]
By (23) we get
\[
g'_{\sigma_k} - \frac{d}{dt} g'_{\sigma_k} = - \left( c + \sum_{i=1}^{n} \sigma'_i \right) f_k(\sigma_k) \prod_{j \neq k} (1 - F_j(\sigma_j)) + \left( \sum_{i=1}^{n} \sigma'_i \right) f_k(\sigma_k) \prod_{j \neq k} (1 - F_j(\sigma_j)) =
\]
\[
-c f_k(\sigma_k) \prod_{j \neq k} (1 - F_j(\sigma_j)) = 0.
\]
Since we ignore the case when \( 1 - F_j(\sigma_j) = 0 \), one of the following two conditions hold: either \( c = 0 \), i.e. there is no weight for time and only resources are taken into account, or \( f_k(\sigma_k) \equiv 0 \), meaning that all of the processes work with no outcome for some period of time.

Thus, the necessary conditions for weak minimum are not necessary hold in the inner points described by the boundary conditions, and we should look for the boundary points.

We can prove the following lemma:
Lemma 1 Let the functions $\sigma_1, \ldots, \sigma_n$ provide an optimal solution. Then at each time $t$ there is at least one process working with full intensity, i.e.,

$$\forall t \max_i \sigma_i'(t) = 1.$$  

The proof of the lemma is given in Appendix A.2.

6 Using optimal scheduling for parallelizing the Latin Square problem

The Latin Square problem belongs to the sort of the problems known for a long time. The task is to place $N$ symbols to the $N \times N$ square such that each symbol appears only once at each row and each column. An example is shown in Figure 6.

![Figure 6: An example of a Latin Square of $5 \times 5$.](image)

A more interested problem arises when the square is partially filled. The problem in this case may be solvable (see Figure 7 to the left) or unsolvable (the same figure to the right).

![Figure 7: An example of solvable (to the left) and unsolvable (to the right) prefilled Latin Squares of $5 \times 5$.](image)

We assume that we are allocated two processors and that we attempt to accelerate the time of finding a solution by starting from two different initial configurations in parallel. Each of the processes employs heuristic DFS with the First-Fail heuristic [7]. We also assume that there is a cost associated with the actual CPU time consumed by each of the processors. Note that our goal is not to build the best algorithm for solving this problem but rather to find the best schedule for the two given algorithms.

Our experiments were performed with $N = 20$ and 10% of the square pre-colored. The performance profile was induced based on a run of 50,000 instances. We compare the optimal schedule produced by our algorithm to the schedule which runs both processes in parallel.

Figure 8 shows how the tradeoff between time and CPU cost influences the resulting cost of the schedules. Each point represents an average over 25,000 pairs of problems. The $x$ axis
corresponds to the normalized weight \( c \) of the CPU cost. Thus, \( c = 0 \) means that we consider elapsed time only; \( c = 1 \) means that we consider CPU time only; and \( c = 0.5 \) means that the costs of elapsed and CPU time are equal. The \( y \) axis stands for the average cost measured by the number of generated nodes.

We can see that when the weight of CPU time is high enough, the optimal schedules found by our algorithm outperform the simple parallelization scheme.

7 Conclusions

In this work we present a theoretical framework for optimal scheduling of parallel anytime algorithms for the case of independent processes. We analyze the properties of optimal schedules for the suspend-resume model, and provide an algorithm for designing such schedules. We also present an interesting result stating the redundancy of intensity control for the case of independent processes. Initial experimentation demonstrates the merit of our scheduling algorithm. The advantage of optimal scheduling over simple parallelization becomes more significant when the weight of the resource cost increases.

One potential weakness of the presented algorithm is its high complexity. This complexity can be represented as a multiplication of three factors: 3-variable function minimization, Branch-and-Bound search and solving (16),(17) and (18). The only exponential component is the Branch-and-Bound search. We found, however, that in practice the branching factor, which is roughly the number of roots of the equations above, is rather small while the
depth of the search tree can be controlled by iterative-deepening strategies. The presented framework can be generalized for an arbitrary number of processes although a straightforward generalization will lead to complexity exponential by this number.

Our algorithm assumes the availability of the performance profiles of the involved processes. Such performance profiles can be derived analytically using theoretical models of the processes or empirically from previous experience with solving similar problems. Online learning of performance profiles, which could expand the applicability of the proposed framework, is a subject of ongoing research.

References


A Formal proofs

A.1 Proof of Theorem 1

The theorem claim is as follows:
1. The value for $\zeta_{k+1}^{11}$ may either be $\zeta_k^{11}$, or can be computed given $\zeta_k^{11}, \zeta_k^{10}$, $\zeta_k^0$ and $\zeta_k^1$ using the formula

$$- f_2(\zeta_k^{10} + \zeta_k^{11}) \int_{\zeta_k^{11}}^{\zeta_k^{10+1}} (1 - F_1(x + \zeta_k^{11}))dx -$$

$$(1 + c) \int_{\zeta_k^{11}}^{\zeta_k^{10+1}} (1 - F_1(x + \zeta_k^{10}))f_2(x + \zeta_k^{10})dx +$$

$$\quad (1 - F_1(\zeta_k^{11} + \zeta_k^{10}))(1 - F_2(\zeta_k^{11} + \zeta_k^{10}))-$$

$$\quad (1 - F_1(\zeta_k^{11} + \zeta_k^{10}))(1 - F_2(\zeta_k^{10} + \zeta_k^{11})) = 0. \quad (29)$$

2. The value for $\zeta_{k+1}^{10}$ may either be $\zeta_k^{10}$, or can be computed given $\zeta_k^{11}, \zeta_k^{10}, \zeta_k^1$ and $\zeta_{k+1}^{10}$ using the formula

$$- f_2(\zeta_k^{10} + \zeta_k^{11}) \int_{\zeta_k^{11}}^{\zeta_k^{10+1}} (1 - F_1(x + \zeta_k^{11}))dx -$$

$$f_1(\zeta_k^{11} + \zeta_{k+1}^{10}) \int_{\zeta_k^{11}}^{\zeta_{k+1}^{10+1}} (1 - F_2(x + \zeta_k^{11}))dx +$$

$$c(1 - F_1(\zeta_k^{11} + \zeta_k^{10}))(1 - F_2(\zeta_k^{10} + \zeta_k^{11}))-$$

$$c(1 - F_1(\zeta_k^{11} + \zeta_{k+1}^{10}))(1 - F_2(\zeta_k^{10} + \zeta_{k+1}^{11})) = 0. \quad (30)$$

3. The value for $\zeta_{k+1}^{11}$ may either be $\zeta_k^{11}$, or can be computed given $\zeta_k^{10}, \zeta_k^{11}, \zeta_k^{01}$ and $\zeta_{k+1}^{11}$ using the formula

$$- f_1(\zeta_k^{11} + \zeta_{k+1}^{11}) \int_{\zeta_k^{11}}^{\zeta_k^{11+1}} (1 - F_2(x + \zeta_k^{11}))dx -$$

$$(1 + c) \int_{\zeta_k^{11}}^{\zeta_k^{11+1}} f_1(x + \zeta_{k+1}^{01})(1 - F_2(x + \zeta_{k+1}^{01}))dx +$$

$$\quad (1 - F_1(\zeta_k^{11} + \zeta_k^{01}))(1 - F_2(\zeta_k^{10} + \zeta_k^{11}))-$$

$$\quad (1 - F_1(\zeta_k^{01} + \zeta_{k+1}^{11}))(1 - F_2(\zeta_k^{10} + \zeta_{k+1}^{11})) = 0. \quad (31)$$

**Proof:** Let us consider the points where the derivative of $E_u(\xi_1^{01}, \xi_1^{10}, \xi_1^{11}, \ldots)$ by the points $\xi_1^{01}, \xi_1^{10}$ and $\xi_1^{11}$ becomes zero. Let us start with the derivative by $\xi_1^{01}$. By (13) we can see that $\xi_1^{01}$ is present in four members of the sum representing $E_u(\xi_1^{01}, \xi_1^{10}, \xi_1^{11}, \ldots)$:

$$E_u(\xi_1^{01}, \xi_1^{10}, \xi_1^{11}, \ldots) = \ldots +$$

$$\quad (1 - F_2(\xi_{k-1}^{10} + \xi_{k-1}^{11})) \int_{\xi_{k-1}^{10}}^{\xi_{k-1}^{11}} (1 - F_1(x + \xi_1^{11}))dx +$$

$$\quad (1 - F_1(\xi_{k-1}^{11} + \xi_{k-1}^{01})) \int_{\xi_{k-1}^{11}}^{\xi_{k-1}^{01}} (1 - F_2(x + \xi_1^{11}))dx +$$

$$\quad (1 + c) \int_{\xi_{k-1}^{11}}^{\xi_{k-1}^{10}} (1 - F_1(x + \xi_1^{01}))(1 - F_2(x + \xi_1^{10}))dx +$$

$$\quad (1 - F_2(\xi_{k-1}^{10} + \xi_1^{11})) \int_{\xi_{k-1}^{10}}^{\xi_{k-1}^{11}} (1 - F_1(x + \xi_1^{11}))dx + \ldots. \quad (32)$$
Therefore, differentiating $E_u(\zeta_{10}^{1}, \zeta_{11}^{1}, \ldots)$ by $\zeta_{k}^{10}$ gives us

$$
\frac{dE_u(\zeta_{k}^{10}, \zeta_{k}^{11}, \ldots)}{d\zeta_{k}^{10}} = \left(1 - F_2(\zeta_{k-1}^{11} + \zeta_{k}^{10})\right) \left(1 - F_1(\zeta_{k}^{10} + \zeta_{k}^{11})\right) -
$$

$$
f_1(\zeta_{k-1}^{11} + \zeta_{k}^{10}) \int_{\zeta_{k-1}^{10}}^{\zeta_{k}^{10}} \left(1 - F_2(x + \zeta_{k}^{11})\right) dx -
$$

$$
(1 + c) \int_{\zeta_{k-1}^{11}}^{\zeta_{k}^{11}} f_1(x + \zeta_{k}^{10}) \left(1 - F_2(x + \zeta_{k}^{10})\right) dx -
$$

$$
(1 - F_2(\zeta_{k}^{10} + \zeta_{k}^{11})) \left(1 - F_1(\zeta_{k}^{10} + \zeta_{k}^{11})\right).
$$

Increasing the index $k$ by 1 in the right part of the last expression gives us the left part of (31).

Let us consider now the derivative by $\zeta_{k}^{10}$. As for $\zeta_{k}^{11}$, $\zeta_{k}^{10}$ is present in four members of the sum representing $E_u(\zeta_{10}^{1}, \zeta_{11}^{1}, \ldots)$:

$$
E_u(\zeta_{10}^{1}, \zeta_{11}^{1}, \ldots) = \ldots +
$$

$$
(1 - F_1(\zeta_{k-1}^{11} + \zeta_{k}^{10})) \int_{\zeta_{k-1}^{10}}^{\zeta_{k}^{10}} \left(1 - F_2(x + \zeta_{k}^{11})\right) dx +
$$

$$
(1 + c) \int_{\zeta_{k-1}^{11}}^{\zeta_{k}^{11}} (1 - F_1(x + \zeta_{k}^{10})) \left(1 - F_2(x + \zeta_{k}^{10})\right) dx +
$$

$$
(1 - F_2(\zeta_{k}^{10} + \zeta_{k}^{11})) \int_{\zeta_{k}^{10}}^{\zeta_{k+1}^{10}} \left(1 - F_1(x + \zeta_{k}^{11})\right) dx +
$$

$$
(1 - F_1(\zeta_{k}^{11} + \zeta_{k}^{10})) \int_{\zeta_{k}^{10}}^{\zeta_{k+1}^{11}} \left(1 - F_2(x + \zeta_{k}^{10})\right) dx + \ldots.
$$

Differentiating $E_u(\zeta_{10}^{1}, \zeta_{11}^{1}, \ldots)$ by $\zeta_{k}^{10}$ gives us

$$
\frac{dE_u(\zeta_{10}^{1}, \zeta_{11}^{1}, \ldots)}{d\zeta_{k}^{10}} = 
$$

$$
(1 - F_1(\zeta_{k-1}^{11} + \zeta_{k}^{10})) \left(1 - F_2(\zeta_{k}^{11} + \zeta_{k}^{10})\right) -
$$

$$
(1 + c) \int_{\zeta_{k-1}^{11}}^{\zeta_{k}^{11}} (1 - F_1(x + \zeta_{k}^{10})) f_2(x + \zeta_{k}^{10}) dx -
$$

$$
(1 - F_1(\zeta_{k}^{11} + \zeta_{k}^{10})) \left(1 - F_2(\zeta_{k}^{10} + \zeta_{k}^{11})\right) -
$$

$$
f_2(\zeta_{k}^{10} + \zeta_{k}^{11}) \int_{\zeta_{k}^{10}}^{\zeta_{k+1}^{10}} \left(1 - F_1(x + \zeta_{k}^{11})\right) dx,
$$

which corresponds to the left part of (29).
Finally, $\zeta_k^{11}$ is present in the following four members of $E_n(\zeta_1^{01}, \zeta_1^{10}, \zeta_1^{11}, \ldots)$:

$$E_n(\zeta_1^{01}, \zeta_1^{10}, \zeta_1^{11}, \ldots) = \ldots +$$

$$ (1 - F_2(\zeta_k^{10} + \zeta_k^{11})) \int_{\zeta_k^{10}}^{\zeta_k^{10}+1} (1 - F_1(x + \zeta_k^{11})) \, dx +$$

$$ (1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1)) \int_{\zeta_k^{11}}^{\zeta_k^{10}+1} (1 - F_2(x + \zeta_k^{11})) \, dx +$$

$$ (1 + c) \int_{\zeta_k^{11}}^{\zeta_k^{10}+1} (1 - F_1(x + \zeta_k^{01} + 1))(1 - F_2(x + \zeta_k^{10} + 1)) \, dx +$$

$$ (1 + c) \int_{\zeta_k^{11}}^{\zeta_k^{10}+1} (1 - F_1(x + \zeta_k^{01} + 1))(1 - F_2(x + \zeta_k^{10} + 1)) \, dx,$$

and differentiating this expression by $\zeta_k^{11}$ gives us

$$\frac{dE_n(\zeta_1^{01}, \zeta_1^{10}, \zeta_1^{11}, \ldots)}{d\zeta_k^{11}} =$$

$$ (1 - F_2(\zeta_k^{10} + \zeta_k^{11})) [(1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1)) - (1 - F_1(\zeta_k^{10} + \zeta_k^{11} + 1))] -$$

$$ f_2(\zeta_k^{10} + \zeta_k^{11}) \int_{\zeta_k^{10}}^{\zeta_k^{10}+1} (1 - F_1(x + \zeta_k^{11})) \, dx +$$

$$ (1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1)) [(1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1)) - (1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1))] -$$

$$ f_1(\zeta_k^{11} + \zeta_k^{01} + 1) \int_{\zeta_k^{10}}^{\zeta_k^{10}+1} (1 - F_2(x + \zeta_k^{11})) \, dx +$$

$$ (1 + c)(1 - F_1(\zeta_k^{10} + \zeta_k^{11}))(1 - F_2(\zeta_k^{10} + \zeta_k^{11})) -$$

$$ (1 + c)(1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1))(1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1)),$$

which can be rewritten as

$$\frac{dE_n(\zeta_1^{01}, \zeta_1^{10}, \zeta_1^{11}, \ldots)}{d\zeta_k^{11}} =$$

$$ (1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1))(1 - F_2(\zeta_k^{10} + \zeta_k^{11})) -$$

$$ (1 - F_1(\zeta_k^{11} + \zeta_k^{11} + 1))(1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1)) +$$

$$ (1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1))(1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1)) -$$

$$ (1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1))(1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1)) +$$

$$ (1 + c)(1 - F_1(\zeta_k^{11} + \zeta_k^{11} + 1))(1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1)) -$$

$$ (1 + c)(1 - F_1(\zeta_k^{11} + \zeta_k^{01} + 1))(1 - F_2(\zeta_k^{10} + \zeta_k^{11} + 1)) -$$

$$ f_2(\zeta_k^{10} + \zeta_k^{11}) \int_{\zeta_k^{10}}^{\zeta_k^{10}+1} (1 - F_1(x + \zeta_k^{11})) \, dx -$$

$$ f_1(\zeta_k^{11} + \zeta_k^{01} + 1) \int_{\zeta_k^{10}}^{\zeta_k^{10}+1} (1 - F_2(x + \zeta_k^{11})) \, dx,$$

and, after eliminating the first and the fourth terms, combining the second term with the fifth one, and the third term with the sixth one, we obtain the left part of (30).
Since the function $E_u(\zeta_1^{01}, \zeta_1^{10}, \zeta_1^{11}, \ldots)$ is continuous by its variables (except the rescheduling points), $\zeta_k^x (x \in \{01, 10, 11\})$ should either be a boundary value of $\zeta_k^x$, or satisfy
\[
\frac{dE_u(\zeta_1^{01}, \zeta_1^{10}, \zeta_1^{11}, \ldots)}{d\zeta_k^x} = 0.
\]
The expression (31) depends on $\zeta_k^{10}, \zeta_k^{11}, \zeta_k^{01}, \zeta_{k+1}^{10}, \zeta_{k+1}^{11}$ and $\zeta_{k+1}^{01}$. Thus, given $\zeta_k^{10}, \zeta_k^{11}, \zeta_k^{01}$ and $\zeta_{k+1}^{01}$, the value of $\zeta_k^{11}$ should be either $\zeta_k^{11}$ or satisfy (31), which corresponds to the third item in the theorem. The similar conclusions for $\zeta_{k+1}^{01}$ and $\zeta_{k+1}^{10}$ follow from (29) and (30) respectively.

Q.E.D.

### A.2 Proof of Lemma 1

The lemma claim is as follows:

Let $\sigma_1, \ldots, \sigma_n$ be an optimal solution. Then at each time $t$ there is at least one process working with full intensity, i.e.,

$$\forall t \max_i \sigma_i^x(t) = 1.$$  

**Proof:** \{\sigma_i\} provide minimum for the expression (8)

$$\int_0^\infty \left( 1 - e \sum_{i=1}^n \sigma_i^x \right) \prod_{j=1}^n (1 - F_j(\sigma_j)) dt.$$  

The proof is by contradiction. Let us consider that in some time interval $[t_0, t_1]$, \{\sigma_i\} do not satisfy the lemma’s constrains. We can show, that we can pass the same time interval by less resources. Let us consider a linear time warp $\nu(t) = \alpha t + \beta$ on the time interval $[t_0, t_1]$, satisfying $\nu(t_0) = t_0$. From the last condition follows $\beta = t_0(1 - \alpha)$. Let $t'_1$ be a point, where $\nu(t)$ achieves $t_1$, i.e., $t'_1 = t_0 + (t_1 - t_0)/\alpha$. Let us consider a set of new delay functions $\overline{\sigma}_i(t)$ of the form

$$\overline{\sigma}_i(t) = \begin{cases} 
\sigma_i(t), & t \leq t_0, \\
\sigma_i(\alpha t + \beta), & t_0 \leq t \leq t'_1, \\
\sigma_i(t + t_1 - t'_1), & t > t'_1.
\end{cases}$$

Thus, $\overline{\sigma}_i(t)$ behaves as $\sigma_i(t)$ before $t_0$, as $\sigma_i(t)$ with time shift after $t'_1$, and as a linearly speeded up version of $\sigma_i(t)$ in the interval $[t_0, t'_1]$. Since $\zeta(t_0) = t_0$ and $\zeta(t'_1) = t_1$, $\overline{\sigma}_i(t)$ is continuous in the points $t_0$ and $t'_1$.

Therefore, $\overline{\sigma}_i^x(t)$ is equal to $\alpha \sigma_i^x(t)$ within the interval $[t_0, t_1]$, and to $\sigma_i^x(t)$ without this interval. By the contradiction assumption, $\sigma_i$ do not meet the lemma constrains, and thus we can take $\alpha > 1$, which leads to valid functions $\overline{\sigma}_i^x(t)$. In particular, we can choose

$$\alpha = \frac{1}{\max_{t \in [t_0, t_1]} \max_i \sigma_i(t)}.$$
Substituting $\tilde{\sigma}_i(t)$ to (8), we obtain

$$E_u(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = \int_0^\infty \left( (1 - c) + c \sum_{i=1}^n \tilde{\sigma}_i(t) \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(t))) dt =$$

$$\int_0^{t_0} \left( (1 - c) + c \sum_{i=1}^n \tilde{\sigma}_i(t) \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(t))) dt +$$

$$\int_{t_0}^{t_1} \left( (1 - c) + c \sum_{i=1}^n \tilde{\sigma}_i(t) \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(t))) dt +$$

$$\int_{t_1}^{\infty} \left( (1 - c) + c \sum_{i=1}^n \tilde{\sigma}_i(t) \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(t))) dt =$$

By the substitution $x = \alpha t + \beta$ in the second term of the last sum and $x = t + t_1 - t'_1$ in the third term, we obtain

$$E_u(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = \int_0^{t_0} \left( (1 - c) + c \sum_{i=1}^n \tilde{\sigma}_i(t) \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(t))) dt +$$

$$\int_{t_0}^{t_1} \left( \frac{1 - c}{\alpha} + c \sum_{i=1}^n \tilde{\sigma}_i(x) \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(x))) dx +$$

$$\int_{t_1}^{\infty} \left( (1 - c) + c \sum_{i=1}^n \tilde{\sigma}_i(x) \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(x))) dx =$$

$$E_u(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) - \int_{t_0}^{t_1} \left( 1 - c \right) \left( 1 - \frac{1}{\alpha} \right) \prod_{j=1}^n (1 - F_j(\tilde{\sigma}_j(t))) dt.$$

Since $\alpha > 1$, the last term is positive, and therefore

$$E_u(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) < E_u(\sigma_1, \ldots, \sigma_n),$$

meaning that the set $\{\tilde{\sigma}_i\}$ provide a better solution than $\{\sigma_i\}$, which contradicts to the optimality of $\{\sigma_i\}$.

Q.E.D.

### A.3 Proof of Theorem 3

The claim of the theorem is as follows:

*If no time cost is taken into account ($c = 1$), the model with shared resources under intensity control settings is equivalent to the model with independent processes under suspend-resume settings. Namely, given a solution for the model with independent processes, we may reconstruct a solution of the same cost for the model with shared resources and vice versa.*

**Proof:** Let $E_{\text{shared}}^*$ be the optimal value for the framework with shared resources, and $E_{\text{independent}}^*$ be the optimal value for the framework with independent processes. Since the two problems minimize the same expression

$$E_u(\sigma_1, \ldots, \sigma_n) = \int_0^\infty \left( \sum_{i=1}^n \sigma_i(t) \right) \prod_{j=1}^n (1 - F_j(\sigma_j(t))) dt \to \min,$$  \hspace{1cm} (35)
and each set \( \{ \sigma_i \} \) satisfying the shared resources constraints automatically satisfies the independent processes constraints, we obtain

\[
E_{\text{independent}}^* \leq E_{\text{shared}}^*.
\]

Let us prove that

\[
E_{\text{shared}}^* \leq E_{\text{independent}}^*.
\]

Assume that a set of functions \( \sigma_1, \sigma_2, \ldots, \sigma_n \) is an optimal solution for the problem with independent processes, i.e.,

\[
E_u(\sigma_1, \ldots, \sigma_n) = E_{\text{independent}}^*.
\]

We want to construct a set of functions \( \{ \sigma_i \} \) satisfying the shared resources constrains, such that

\[
E_u(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = E_u(\sigma_1, \ldots, \sigma_n).
\]

Let us consider a set of discontinuity points of \( \sigma_i' \)

\[
T = \{ t \mid \exists i: \sigma_i'(t - \epsilon) \neq \sigma_i'(t + \epsilon) \}.
\]

In our model this set is countable, and we can write it as a sorted sequence \( t_0 = 0 < t_1 < \ldots < t_k < \ldots \) The expected schedule cost in this case will have a form

\[
E_u(\sigma_1, \ldots, \sigma_n) = \sum_{j=0}^{\infty} E_{u_j}(\sigma_1, \ldots, \sigma_n),
\]

where

\[
E_{u_j}(\sigma_1, \ldots, \sigma_n) = \int_{t_j}^{t_{j+1}} \left( \sum_{i=1}^{n} \sigma_i' \right) \prod_{l=1}^{n} (1 - F_i(\sigma_l)) dt.
\]

We want to construct the functions \( \tilde{\sigma}_i \) in the incremental manners For each time interval \( [t_j, t_{j+1}] \) we define a corresponding point \( \tilde{t}_j \) and a set of functions \( \tilde{\sigma}_i \), such that

\[
\tilde{E}_{u_j}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = \int_{\tilde{t}_j}^{\tilde{t}_{j+1}} \left( \sum_{i=1}^{n} \tilde{\sigma}_i' \right) \prod_{l=1}^{n} (1 - F_i(\tilde{\sigma}_l)) dt = E_{u_j}(\sigma_1, \ldots, \sigma_n).
\]

Let us denote \( \sigma_{ij} = \sigma_i(t_j) \) and \( \tilde{\sigma}_{ij} = \tilde{\sigma}_i(t_j) \). At the beginning, \( \tilde{\sigma}_{i0} = 0 \) for each \( i \), and \( \tilde{t}_0 = 0 \). Assume now that we have \( \tilde{t}_{j'} \) defined for \( j' < j \), and \( \tilde{\sigma}_i(t) \) defined on each interval \( [\tilde{t}_{j'}, \tilde{t}_{j'+1}] \). Let us show how to define \( \tilde{t}_j \) and \( \tilde{\sigma}_j \) on \( [\tilde{t}_j, \tilde{t}_{j+1}] \).

By definition of \( t_j \), \( k = \sum_{i=1}^{n} \sigma_i(t) \) is a constant for \( t \in [t_j, t_{j+1}] \). Since \( \{ \sigma_i \} \) satisfy suspend-resume constraints, exactly \( k \geq 1 \) processes are active on this interval, each with full intensity. Without loss of generality, the active processes are \( A_1, A_2, \ldots, A_k \), and

\[
E_{u_j}(\sigma_1, \ldots, \sigma_n) = k \int_{t_j}^{t_{j+1}} \prod_{l=1}^{n} (1 - F_i(\sigma_l)) dt =
\]

\[
k \prod_{l=k+1}^{n} (1 - F_i(\sigma_{lj})) \int_{t_j}^{t_{j+1}} \prod_{l=1}^{k} (1 - F_i(t - t_j + \sigma_{lj})) dt =
\]

\[
k \prod_{l=k+1}^{n} (1 - F_i(\sigma_{lj})) \int_{0}^{t_{j+1}-t_j} \prod_{l=1}^{n} (1 - F_i(x + \sigma_{lj})) dx.
\]
Let \( t_{j+1}^{\sim} = t_j + k(t_{j+1} - t_j) \), and let us define \( \tilde{\sigma}_i(t) \) on the segment \( [t_j, t_{j+1}] \) as follows:

\[
\tilde{\sigma}_i(t) = \begin{cases} 
(t - t_j)/k + \tilde{\sigma}_{ij}, & \text{if } t \in [t_j, t_{j+1}] \\
\sigma_i' > 0, & \text{otherwise.}
\end{cases}
\]

(36)

In this case, on this segment

\[
\sum_{i=1}^{n} \tilde{\sigma}_i'(t) = 1,
\]

which means that the \( \tilde{\sigma}_i \) satisfy the shared resources constraints. By definition,

\[
t_{j+1}^{\sim} - t_j = k(t_{j+1} - t_j),
\]

and therefore for processes active on \([t_j, t_{j+1}]\) we obtain

\[
\tilde{\sigma}_{i,j+1} - \tilde{\sigma}_{ij} = \frac{t_{j+1}^{\sim} - t_j}{k} = t_{j+1} - t_j = \sigma_{i,j+1} - \sigma_{ij}.
\]

For processes idle on \([t_j, t_{j+1}]\) the same equality holds as well:

\[
\tilde{\sigma}_{i,j+1} - \tilde{\sigma}_{ij} = 0 = \sigma_{i,j+1} - \sigma_{ij},
\]

and since \( \tilde{\sigma}_i(t) = 0 \) we obtain the invariant

\[
\tilde{\sigma}_{ij} = \sigma_{ij}.
\]

(38)

The average cost for the new schedules may be represented as

\[
\tilde{E}_{u_j}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = \int_{t_j}^{t_{j+1}} \left( \sum_{i=1}^{n} \tilde{\sigma}_i' \right) \prod_{i=1}^{n} (1 - F_i(\tilde{\sigma}_i)) dt = \\
\prod_{i=k+1}^{n} (1 - F_i(\tilde{\sigma}_{ij})) \int_{t_j}^{t_{j+1}} \prod_{i=1}^{k} (1 - F_i((t - t_j)/k + \tilde{\sigma}_{ij})) dt.
\]

Substituting \( x = (t - t_j)/k \) and using (36), (37) and (38), we obtain

\[
\tilde{E}_{u_j}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = k \prod_{i=k+1}^{n} (1 - F_i(\tilde{\sigma}_{ij})) \int_{0}^{(t_{j+1} - t_j)/k} \prod_{i=1}^{k} (1 - F_i(x + \tilde{\sigma}_{ij})) dx = \\
k \prod_{i=k+1}^{n} (1 - F_i(\tilde{\sigma}_{ij})) dt \int_{0}^{t_{j+1} - t_j} \prod_{i=1}^{k} (1 - F_i(x + \sigma_{ij})) dx = \\
E_{u_j}(\sigma_1, \ldots, \sigma_n).
\]

From the last equation immediately follows, that

\[
E_u(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = \sum_{j=0}^{\infty} \tilde{E}_{u_j}(\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n) = \sum_{j=0}^{\infty} E_{u_j}(\sigma_1, \ldots, \sigma_n) = E_u(\sigma_1, \ldots, \sigma_n),
\]

which completes the proof. 

Q.E.D.

26